Available online at www.sciencedirect.com

ScienceDirect

Neurocomputing 70 (2007) 975–990

NEUROCOMPUTING

www.elsevier.com/locate/neucom

Hierarchical dynamical models of motor function

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

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

Received 29 June 2005; received in revised form 13 January 2006; accepted 21 March 2006
Communicated by K. Fukushima
Available online 23 September 2006

Abstract

Hierarchical models of motor function are described in which the motor system encodes a hierarchy of dynamical motor primitives. The models are based on continuous attractor neural networks, in which the packet of neural activity representing the position (in the state space) is moved by path integration using a motor efference copy after training with a traced associative learning rule. Sequences of movements can be learned because the state space modulates the effects of an intermediate level motor command. A high-level motor command selects a set of intermediate level motor commands, and the whole movement sequence evolves with the correct order because the state space representation effectively selects the next possible intermediate level command. Thus a single high-level motor command can select whole movement trajectories composed of a set of motor primitives. It is further shown that the network can perform completely novel motor sequences, and thus provides one solution to Lashley's 'problem of serial order in behaviour'.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Hierarchical motor function; Continuous attractor neural networks; Self-organisation; Trace learning; Dynamical systems

1. Introduction

Motor systems in humans have a hierarchical structure, with higher levels specifying increasingly complex motor actions [5,4]. This hierarchy may be divided into three distinct brain regions: the motor areas of the 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 [7]. 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 complex sensory information extracted at each level from the spinal cord to the cortex [4]. These sensory representations provide a necessary basis from which motor sequences at each level are regulated.

A key aspect of hierarchical motor control in animals is its dynamical nature. That is, the motor primitives at all levels in the hierarchy take the form of continuous dynamical motions. This is a challenge for computational models of motor control. In current models the motor program seeks to minimise the distance or error between the current state of the agent and a particular target state (or desired trajectory), where the distance is either determined by external visual or proprioceptive feedback, or estimated with internal models [11,26,25]. In this paper we develop an alternative approach which is inspired from a dynamical system perspective, rather than classical control theory. The hierarchical models of motor function developed here build on a single command layer (SCL) model of motor function proposed by Stringer et al. [22]. That model is based on continuous attractor neural

*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.

0925-2312/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2006.03.012

networks, which can maintain a set of neurons firing to represent a position in a continuous space. We showed in the model how the packet of neural activity representing the position can be moved by a movement selector (MS) input [23,21]. The network self-organises during training by using a local associative synaptic modification rule with a short term memory trace between the continuous attractor neurons to associate the trajectory through which the continuous attractor is moving with each MS input. The network enables neural activity to progress dynamically at arbitrary speeds which depend on the magnitude of the MS input, and so allow an agent to control the speed of its movement [22]. The use of 'trace' learning rules which incorporate a form of temporal average of recent cell activity underlies the ability of the network to learn temporal sequences of behaviour. The model [22] is 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 [17,18,2,13,9,10].

The SCL model of motor function developed in [22] did not include a hierarchical pyramid of motor primitives. In this paper, we extend the basic SCL model to develop hierarchical command layer (HCL) network models of motor function. HCL models support hierarchical motor control, in which the motor system encodes a hierarchy of dynamical motor primitives. At the lowest level, the motor system encodes the most simple motor sequences, while at higher levels the motor system encodes more complex motor sequences composed of sequences of the lower level motor primitives.

A key issue addressed is how such hierarchical models of motor function may solve the 'problem of serial order in behaviour' [8]. This is the problem of how animals are able to perform completely novel motor sequences. An example of such a novel motor sequence is how a person may perform a novel reaching movement to grasp an object when there is a unique spatial arrangement of obstacles in the way. We show how the second of the two hierarchical models of motor function described here may give rise to new motor sequences that have not been performed during training.

In Section 2 we present a movement control network with a hierarchy of command layers, and show how it enables hierarchical motor control. In Section 3 the model is extended to allow it to produce new motor sequences that have not been performed during training. In Section 4 we develop these ideas further and show that it is possible to have both the state and motor representations in a single recurrent network, and for all the computations to be performed in a single network. This is of considerable interest, for networks with recurrent connections are a feature of the cerebral cortex, and different neurons which respond to sensory events, or to motor events, are frequently found in the same cortical area [16,3].

2. A model of hierarchical motor control

In this section we present a model of hierarchical motor function, which is an extension of the architecture described by Stringer et al. [22]. The model presented here introduces the concept of how one movement selection signal can be used to drive a whole sequence of motor primitives, which are encoded using a continuous attractor network framework. A motor primitive is an elemental motor sequence that may be recruited by more complex motor programs. Each motor program may be composed of a sequence of motor primitives.

2.1. *SCL architecture for learning to perform low-level motor primitives*

The model has a continuous attractor postural or positional state network with firing rate r_i^{S} for state cell i, labelled as 'Network of state cells' in Fig. 1. (The generic details of recurrently connected continuous attractor networks are described by [1,24,23,21,14].) The continuous attractor properties of the state cells implemented by the associatively modified recurrent collateral connections w^1 maintains the firing of the state cells in the absence of external motor, visual, or proprioceptive inputs. The activation h_i^{S} of state cell i is governed by

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

where r_j^{S} is the firing rate of state cell j, 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 firing rate r_i^{S} of state cell i is determined using the sigmoid activation function

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

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

An implementation detail that was introduced in an earlier paper [23] to help with noise due to irregular training or diluted connectivity is as follows. This implementation detail was not necessary in this paper as there was no noise in the simulations, and though present was shown in further simulations not to affect the results

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

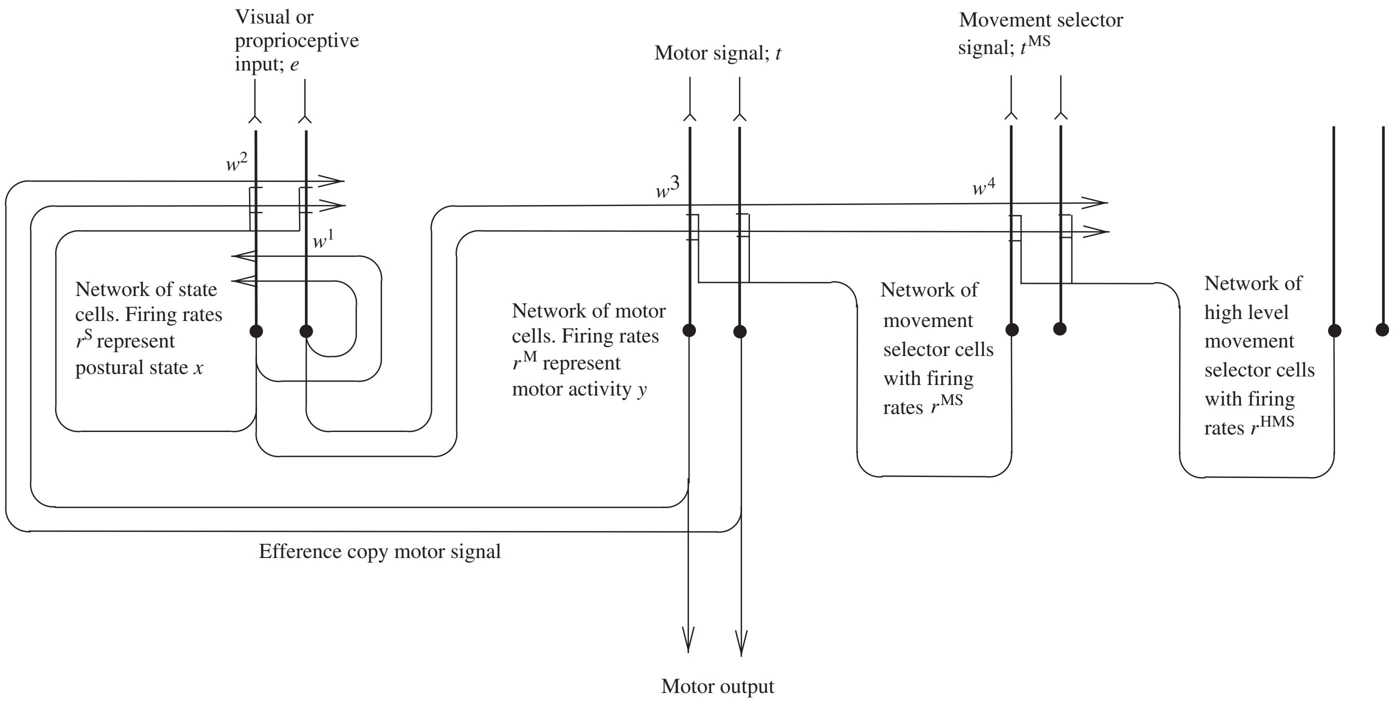
Visual or proprioceptive input; e
Motor signal; t
Movement selector signal; t^{MS}
w^2
w^3
w^4
w^1
Network of state cells. Firing rates r^{S} represent postural state x
Network of motor cells. Firing rates r^{M} represent motor activity y
Network of movement selector cells with firing rates r^{MS}
Network of high level movement selector cells with firing rates r^{HMS}
Efference copy motor signal
Motor output

Fig. 1. Network architecture of the hierarchical network model. There is a network of state cells which represent the postural state of the agent, a network of motor cells which represent the motor activity, and a network of movement selector cells which represent the decision to perform a motor sequence. 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 desired motor task or target state 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 cells (MS) are driven by an additional network of high-level movement selector (HMS) cells through Sigma–Pi synapses w^4. The movement selector cells represent the motor primitives, while the high-level movement selector cells represent the high-level motor programs which are composed of sequences of the motor primitives. The synapses w^4, which drive the low-level movement selector cells, are Sigma–Pi synapses from combinations of state cells and high-level movement selector cells. This allows which movement selector cells are currently activated to be dependent on: (i) the current state of the agent represented by the state cells, and (ii) the current high-level motor program represented by the high-level movement selector cells.

described in this paper. It was shown [23] that the drift of the activity packet within the continuous attractor network of state cells due to imperfect training can be reduced by enhancing the firing of neurons that are already firing. This can be 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

$$\alpha_i = \begin{cases} \alpha^{\mathrm{HIGH}} & \text{if } r_i(t) < \gamma \\ \alpha^{\mathrm{LOW}} & \text{if } r_i(t) \geqslant \gamma, \end{cases} \tag{3}$$

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.

The network of motor cells (labelled as 'Network of motor cells' in Fig. 1 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

$$\tau \frac{\mathrm{d}h_i^{\mathrm{M}}(t)}{\mathrm{d}t} = -h_i^{\mathrm{M}}(t) + t_i + \frac{\phi_2}{C^{\mathrm{S}\times\mathrm{MS}}} \sum_{j,k} w_{ijk}^3 r_j^{\mathrm{S}} r_k^{\mathrm{MS}}, \tag{4}$$

where t_i is the motor signal used during training. (The motor signal could be available to the system described here as a result of the behaviour being generated during the trial and error learning stage using cortical circuitry and feedback signals from the environment that help to guide the movement [6].) The last term in Eq. (4) shows the operation of Sigma–Pi synapses w^3 which allow the postural signals r_j^{S} to be gated by the MS input with firing r_k^{MS} for MS cell k.[2]

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

$$r_i^{\mathrm{M}}(t) = \frac{1}{1+\mathrm{e}^{-2\beta(h_i^{\mathrm{M}}(t)-\alpha)}}. \tag{5}$$

During the learning phase, motor input signals t_i drive each motor cell i to cause the agent to proceed through a set of positional states, and simultaneously each positional state cell i is driven by an external visual or proprioceptive input e_i. While this happens, the recurrent connectivity

[2]The scaling factor $\phi_2/C^{\mathrm{S}\times\mathrm{MS}}$ controls the overall strength of the inputs from couplings of state and MS 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 MS cells.

implemented by w^1_{ij} is learned to allow 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, so that the agent can operate with incomplete sensory input or in the dark. The learning rule used to update the recurrent synapses w^1_{ij} is the Hebb rule

$$\delta w^1_{ij} = k^1 r^S_i r^S_j. \tag{6}$$

Also during training, the positional state continuous attractor neural network (CANN) learns to modify the synapses w^2_{ijk} to utilise an efference copy of the motor signal to update using path integration the positional state CANN, and thus to implement a *forward model* of motor function [11]. A traced Sigma–Pi learning rule operating at synapses w^2_{ijk} 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 is

$$\delta w^2_{ijk} = k^2 r^S_i \bar{r}^S_j \bar{r}^M_k, \tag{7}$$

where $\bar{r}^S_j$ refers to a memory trace of the firing r^S_j, and $\bar{r}^M_k$ refers to a memory trace of the firing of r^M_k. The trace value $\bar{r}$ of the firing rate r of a cell is calculated according to

$$\bar{r}(t+\delta t) = (1-\eta)r(t+\delta t) + \eta\bar{r}(t), \tag{8}$$

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 concepts involved in this implementation of path integration, and the short-term memory traces inherent in these operations using for example the long time constant of NMDA (N-methyl-D-aspartate) receptors, are described by [23,21].

The motor cells are trained to operate without their motor training input t by using their Sigma–Pi synapses w^3_{ijk} using the state cell and movement selector inputs together with the firing rates of the motor cells being set by t during training according to

$$\delta w^3_{ijk} = k^3 r^M_i r^S_j r^{MS}_k. \tag{9}$$

The firing of the MS cells r^{MS}_k represents the instruction to perform a particular motor primitive sequence. In particular, the force, and hence the speed, of the movement may be controlled by varying the firing rates of the MS cells [22]. The synaptic connections w^3_{ijk} thus implement an *inverse model* of motor function: given a desired movement represented by the firing of the MS cells, the synaptic connections w^3_{ijk} drive the motor cells to perform the appropriate motor actions.

During the learning phase, the agent performs the motor primitive sequence to be learned. As the agent performs the motor primitive sequence (such as raising the arm), 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^1_{ij}, w^2_{ijk} and w^3_{ijk} are updated according to the simple learning rules discussed above. During repeated learning cycles of the motor primitive sequence, the synaptic connectivity of the network self-organises such that, after training, the correct motor sequence may be stimulated solely by stimulating the particular set of MS cells.

2.2. Hierarchical command layer (HCL) architecture for learning high-level motor programs

The MS cells (see Fig. 1) are driven by a network of high-level movement selector (HMS) cells through Sigma–Pi synapses w^4 which allow modulation by the firing rates r^S of the postural state cells. A single pattern of HMS cell firing can learn to drive a whole sequence of movement selector cell firing, with the sequence being generated as a result of the interaction between the postural state inputs to the MS cells and the HMS inputs to the MS cells. Let us give an example, related to riding a bicycle after training. The HMS cells have a firing pattern which specifies 'perform cycling', where the cycling is the high-level motor program being specified. The MS cells might then represent different stages of a single revolution (each a separate motor primitive), and would be gated into successive stages by the postural input. The MS cells drive the motor (M) cells which might represent forces being specified for the muscles. The postural state reached, which is reflected in the postural state (S) cell firing, modulates the firing of the motor cells as described for the SHC model to generate the next muscle command.

As the agent performs a high-level motor program, the behaviour of the MS cells must be modelled dynamically to represent the continuously changing firing rates of these cells as they are driven by the HMS cells. Therefore, the equation governing the activation of the MS cells is

$$\tau\frac{\mathrm{d}h^{MS}_i(t)}{\mathrm{d}t} = -h^{MS}_i(t) + t^{MS}_i + \frac{\phi_3}{C^{S\times HMS}}\sum_{j,k} w^4_{ijk} r^S_j r^{HMS}_k, \tag{10}$$

where the activation h^{MS}_i is driven by the following terms. The first term driving the activations of the MS cells in Eq. (10) is the training signal t^{MS}_i for each MS cell i. This term is present only during training with a new high-level motor program. The training signal t^{MS}_i for the MS cells operates in a similar manner to the training signal t_i for the motor cells. The term t^{MS}_i models a mechanism used during learning to stimulate the MS cells associated with the new high-level motor program being learned. After the training phase with a new high-level motor program is completed, the input terms t^{MS}_i are set to zero for the subsequent testing phase with the motor program. The second term driving the activations of the MS cells in Eq. (10) is the input from couplings of the postural state (S) cells and the

HMS cells $\sum_{j,k} w^4_{ijk} r^S_j r^{HMS}_k$, where r^S_j is the firing rate of state cell j, r^{HMS}_k is the firing rate of HMS cell k, and w^4_{ijk} is the corresponding strength of the connection from these cells. The driving term $\sum_{j,k} w^4_{ijk} r^S_j r^{HMS}_k$ initiates activity in the network of MS cells, and then drives the activity through the network.[3] The synapse connecting the state and HMS cells to the MS cells has a Sigma–Pi form in that it computes a weighted sum of the products of the firing rates of the state cells and HMS cells.This ensures that the activity within the network of MS cells is driven by the state cells if and only if the HMS cells are also active.

The firing rate r^{MS}_i of MS cell i is determined from the activation h^{MS}_i and the sigmoid activation function

$$r^{MS}_i(t) = \frac{1}{1 + e^{-2\beta(h^{MS}_i(t) - \alpha)}}, \tag{11}$$

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

As the agent learns to perform the high-level motor program, the synaptic weights w^4_{ijk} are set up by a learning rule which associates the co-firing of the HMS cells r^{HMS}_k and a particular cluster of state cells r^S_j, with the firing of the appropriate cluster of MS cells r^{MS}_i produced during the training by an external motor signal t^{MS}_i (see Fig. 1). The synaptic weights w^4_{ijk} from the state cells and HMS cells to the MS cells are updated during learning according to

$$\delta w^4_{ijk} = k^4 r^{MS}_i r^S_j r^{HMS}_k. \tag{12}$$

During the learning of a new high-level motor program the agent performs the appropriate lower level motor primitives that are needed to compose the new high-level motor program. Specifically, during learning of a new high-level motor program, the agent stimulates the MS cells corresponding to the appropriate (pre-learned) low-level motor primitives in the correct temporal order. This drives the network through the low-level motor primitives. At the same time, however, a new set of cells in the HMS network is activated, and this new set of HMS cells learns the new temporal sequence of activity in the MS network corresponding to the new high-level motor program. After learning, the new set of HMS cells is able to stimulate the new high-level motor program. However, the new set of HMS cells, which encodes the higher level motor program, learns to drive the MS cells associated with the motor primitives, rather than driving the motor cells directly.

2.3. Simulation results

In Experiment 1 we demonstrate the ability of the model to learn two high-level motor programs each of which consists of three particular motor primitives (chosen from a set of six) which must be executed in the correct temporal order.[4] The network architecture is as shown in Fig. 1.

In this experiment, we assumed that the state space x of the agent was a finite one-dimensional space from $x = 0$ to $x = 1$. That is, the state of the agent was defined by the parameter $x \in [0, 1]$. There were 200 state cells, which were mapped onto a regular grid of different states, where for each state cell i there was a unique preferred state x_i of the agent for which the cell was stimulated maximally by the visual and proprioceptive cues. The current motor activity of the agent was defined by the current state x of the agent and which direction the agent was moving in. The motor cells were mapped onto a regular grid of different instantaneous motor activities, where for each motor cell i there was a unique preferred instantaneous motor activity defined by the current state and direction of movement, for which the cell was stimulated maximally. The motor network contained 400 cells, with cells 1–200 representing movement in the positive x-direction, and cells 201–400 representing movement in the negative x-direction. Finally, the MS network and HMS network each contained 200 cells.

Experiment 1 took place in two stages. During the first stage of the experiment the network was taught six low-level motor primitives. Then, during the second stage the network learned to perform two new high-level motor programs, each of which was composed of a temporal sequence of three particular low-level motor primitives. The incorporation of two high-level motor programs in the simulation was to test whether a single network can learn multiple programs each selected by a different HMS command.

Stage 1: *learning to perform the low-level motor primitives*. The numerical simulation began with the learning phase for the six low-level motor primitives in which the synaptic weights, w^1_{ij}, w^2_{ijk} and w^3_{ijk}, were self-organised.[5] At the start of the learning, the synaptic weights are initialized to zero. Then learning proceeds with the agent running through the motor training sequence for each motor primitive in turn.

[3]The scaling factor $\phi_3/C^{S\times HMS}$ controls the overall strength of the inputs from couplings of state and HMS cells, where ϕ_3 is a constant, and $C^{S\times HMS}$ is the number of connections received by each MS cell from couplings of the state and HMS cells.

[4]For Experiment 1 we used the following parameter values. 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 equations (1) and (4) were: $\tau = 1$, $\phi_0 = 3 \times 10^5$, $\phi_1 = 1.75 \times 10^7$, $\phi_2 = 1.25 \times 10^6$, and $w^{INH} = 0.011$. The parameters governing the sigmoid activation functions were as follows. For the state cells we used: $\alpha^{HIGH} = 0.0$, $\alpha^{LOW} = -20.0$, $\gamma = 0.5$, and $\beta = 0.1$. For the motor cells we used: $\alpha^{HIGH} = 10.0$, $\alpha^{LOW} = 10.0$, $\gamma = 0.5$, and $\beta = 0.3$. Since we set $\alpha^{HIGH} = \alpha^{LOW}$ for the motor cells, there was no enhancement of the firing rates of motor neurons with already high firing rates, and the parameter γ was redundant. Finally, for the numerical simulations of the leaky-integrator dynamical equations (1), (4) we employed a Forward Euler finite difference method with the timestep set to 0.2.

[5]In the model, each of the state and motor cells was set to have a Gaussian tuning profile with a standard deviation of 0.02 where the state space $x \in [0, 1]$, and the cells were evenly distributed through the space, and the weights between the neurons were set according to the learning rules (6), (7) and (9).

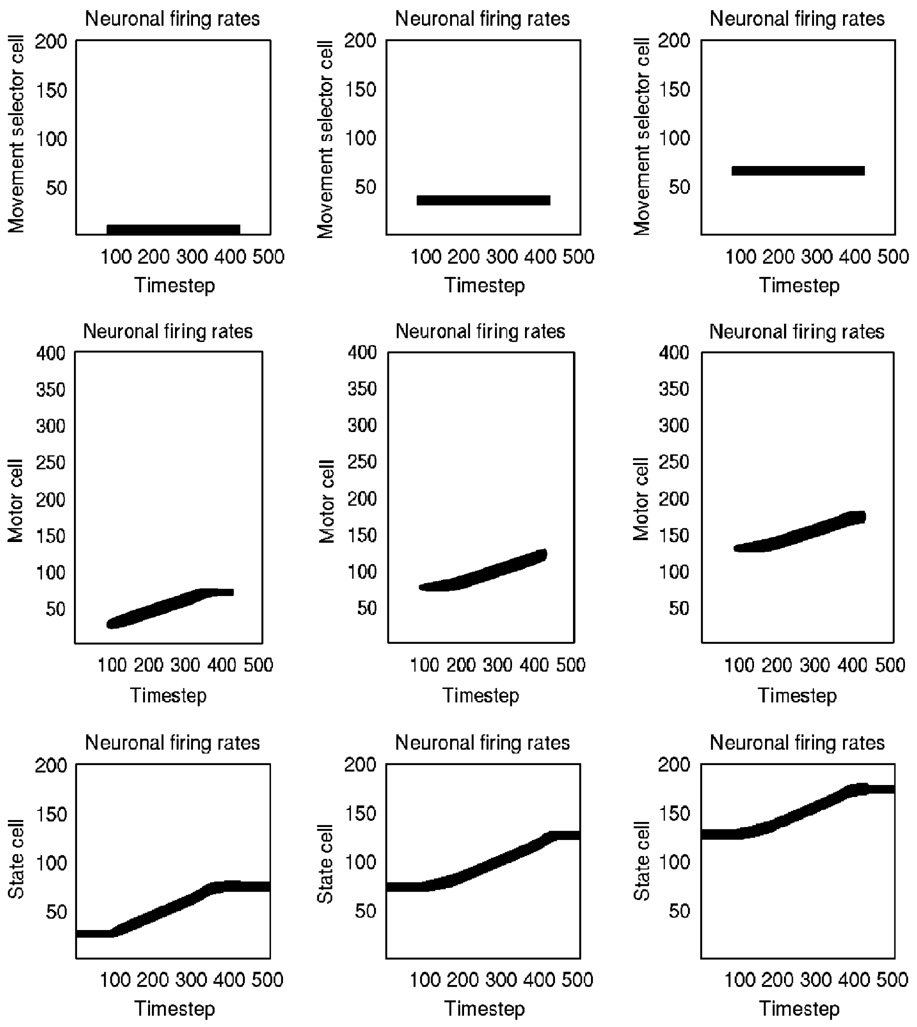

Fig. 2. Numerical results from the first stage of Experiment 1, in which the network is trained to perform the low-level motor primitives. This figure shows the motor primitives 1, 2 and 3, which encode small movements of the state of the agent in the positive x direction. The first column shows the 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.

At the start of the training with each primitive the trace values of the neuronal firing rates were initialised to zero. Motor primitives 1, 2 and 3 involved small movements through the state space in the positive x-direction. These primitives were represented by motor cells 1–200. As is evident from Figs. 2 and 3, for the first low-level motor primitive, the network was trained with the agent performing a motor sequence, with the postural state x of the agent and current motor activity running in lock-step from $x = 0.1$ to 0.37. During the training, the selection of this movement was represented by setting the firing rates r^{MS} of MS cells 1–10 to 1, with the firing rates of the remaining MS cells 11–200 set to 0. For the second low-level motor primitive, the network was trained with the state x of the agent and motor activity running in lock-step from $x = 0.37$ to 0.63. This movement was represented by setting the firing rates r^{MS} of MS cells from 31–40 to 1. For the third low-level motor primitive, the network was trained with the state x of the agent and motor activity running in lock-step from $x = 0.63$ to 0.9. This movement was represented by setting the firing rates r^{MS} of MS cells from 61–70 to 1. Motor primitives 4, 5 and 6 were similarly encoded, except that they involved small movements through the state space in the negative x-direction. These primitives were represented by motor cells 201–400.

In the testing phase the agent performed the motor primitives without the motor training signal t_i, and without the visual and proprioceptive inputs e_i. The aim was to show that the population of movement selector cells could produce the desired motor primitives once the relevant MS cells were activated. For the testing phase, the full 'leaky-integrator' dynamical equations (1), (2), (4) and (5) were implemented. Before the MS cells were activated, the network was set into a state of firing which represented the starting point of one of the motor primitives (by applying visual inputs for a short period).

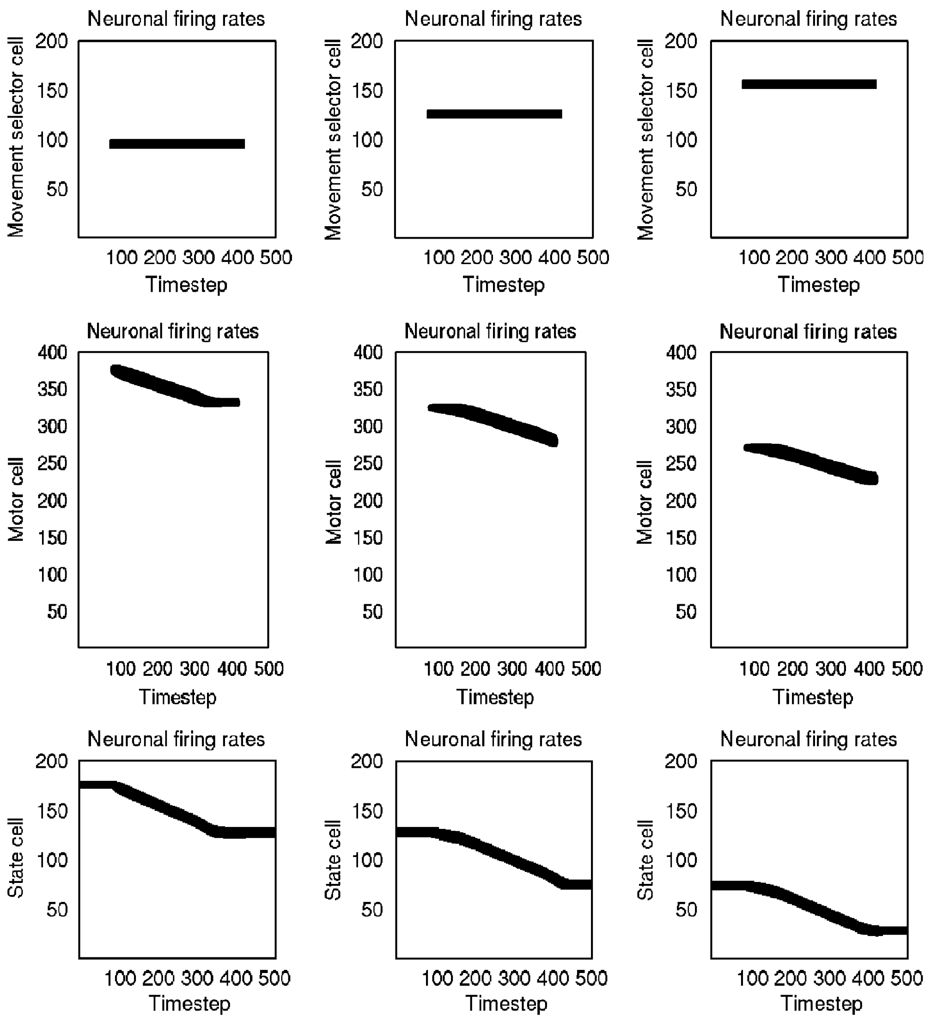

Fig. 3. Numerical results from the first stage of Experiment 1, in which the network is trained to perform the low-level motor primitives. This figure shows the motor primitives 4, 5 and 6, which encode small movements of the state of the agent in the negative x direction. Conventions as in Fig. 2.

In Figs. 2 and 3 we show numerical results from the first stage of Experiment 1. Fig. 2 shows the network performing motor primitives 1, 2 and 3. The first column in Fig. 2 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, motor network and state network through time as the agent performed the learned low-level motor primitive in the absence (in this testing phase) of the motor training signal t_i, and without the external (visual or proprioceptive) input e_i. Fig. 3 shows similar results for motor primitives 4, 5 and 6.

For each motor primitive, it is shown that applying steady activity to the relevant MS cells during timesteps 81–430 first activates (through the synapses w^3) the motor cells, the firing of which alters the state being represented by the network of state cells via connections w^2, which in turn shifts the motor state represented by the motor cells through synapses w^3. The results produced are continuously moving motor and postural states. In this way, the activity packets in the state and motor networks move together, with the network firing patterns running through the learned motor primitive. During timesteps 431–510 the MS cells stopped firing, and all the motor cells became quiescent (while the state cells continued firing representing the state reached).

Stage 2: *learning to perform the high-level motor programs*. In stage 2 of Experiment 1 the network learned to perform two high-level motor programs, each of which was composed of a temporal sequence of three particular primitives. Motor program 1 involved the agent moving in the positive x-direction, from $x = 0.1$ to 0.9. To do this the system needed to sequence appropriately primitives 1, 2 and 3 shown in Fig. 2. Motor program 2 involved the agent moving in the negative x-direction, from $x = 0.9$ to 0.1. To do this the system needed to sequence appropriately primitives 4, 5 and 6 shown in Fig. 3.

During this learning stage, the full 'leaky-integrator' dynamical equations (1), (2), (4) and (5) were implemented for the state cells and motor cells. To learn the first high-level motor program, the agent ran sequentially through the motor primitives 1, 2 and 3 in order. For this learning phase, the three motor primitives were initiated by firing the MS cells that had already learned to represent the motor primitives. In addition, a new set of HMS cells, 1–10, was activated. This set of HMS cells was used to represent the first high-level motor program being selected. As the agent performed the first high-level motor program, the synaptic weights w^4 of the connections from the set of active HMS cells, 1–10, onto the MS cells were updated according to Eq. (12). Training the network on the second high-level motor program was performed in a similar manner, except that the agent ran sequentially through the motor primitives 4, 5 and 6 in order. In addition, a different set of HMS cells, 31–40, was activated. This set of HMS cells was used to represent the second high-level motor program being selected.

After the learning phases for both of the high-level motor programs were completed, the network was tested to determine whether it could perform the two high-level motor programs when the relevant HMS cells were activated. For the testing phase, the full 'leaky-integrator' dynamical equations (1), (2), (4), (5), (10) and (11) were implemented. Fig. 4 shows numerical results from the second stage of Experiment 1, after the network has been trained to perform the two high-level motor programs. On the left are results for the first high-level motor program. At the start, an activity packet was initiated at position $x = 0.1$ (by application of a short visual input). It is shown that applying steady activity to the relevant HMS cells 1–10 during timesteps 81–900 drives the network through the high-level motor program composed of a temporal sequence of the three motor primitives 1, 2 and 3. From Fig. 4 it can be seen that the HMS cells drive the MS cells through the correct temporal sequence for the high-level motor program. Then as the MS cells fire, the signals from the MS cells drive the activity packets in the postural state and motor networks to run through the learned motor primitives. Further, because activity in the MS network is

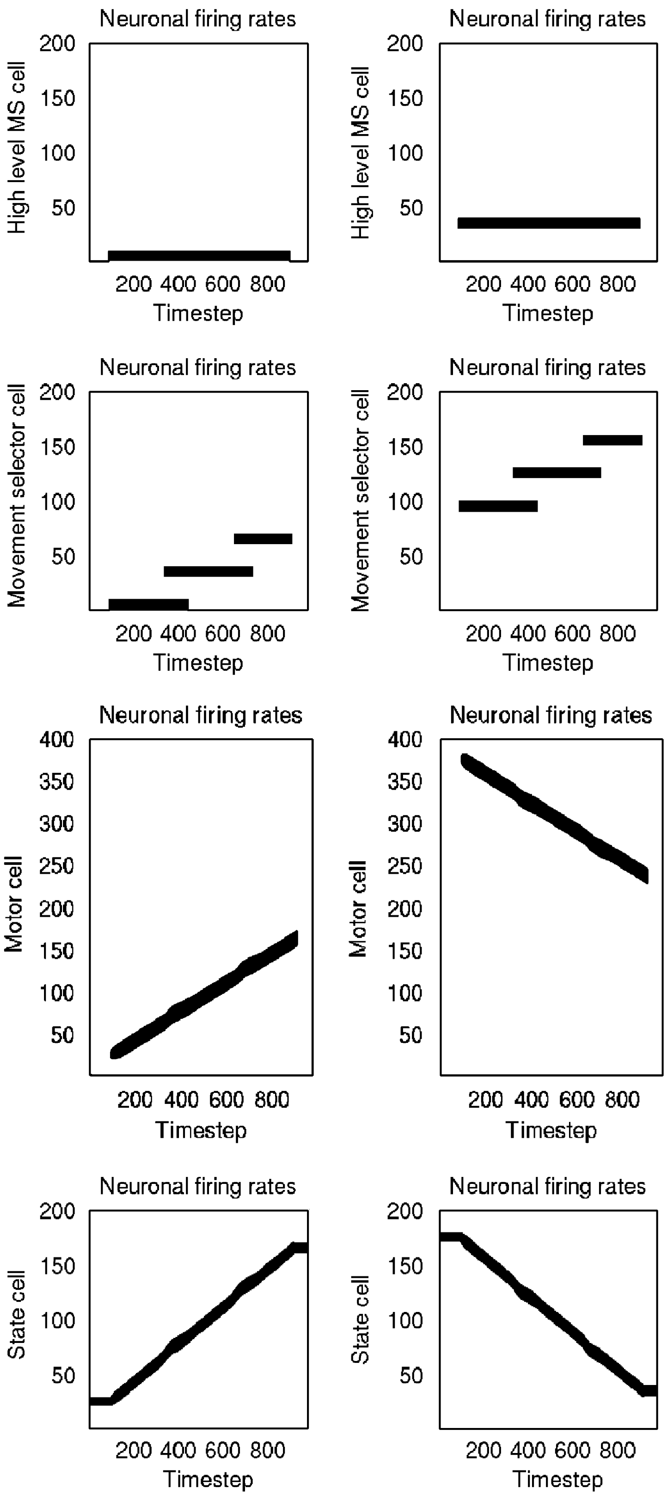

Fig. 4. This figure shows numerical results from the second stage of Experiment 1, after the network has been trained to perform two different high-level motor programs. On the left are results for the first high-level motor program. It is shown that applying steady activity to the relevant high-level movement selector cells 1–10 during timesteps 81–900 drives the network through the high-level motor program composed of a temporal sequence of the three motor primitives 1, 2 and 3. The high-level movement selector cells drive the movement selector cells through the correct temporal sequence for the high-level motor program. As the movement selector cells fire, 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. Because activity in the movement selector network is gated by the w^4 connections from the state cells (shown in Fig. 1), the movement selector cells for each motor primitive switch on automatically when the agent reaches the beginning state for that motor primitive, and switch off automatically when the agent reaches the end state for that primitive. Thus, there is no need for any further top-down input signal to guide the timing of the firing of the movement selector cells which represent the component motor primitives of the overall high-level motor program. The high-level motor program is performed without the training signal t_i^{MS} driving the movement selector cells which represent the selection of the motor primitives, without the motor training signal t_i, and without the external (visual or proprioceptive) input e_i. On the right are results for the second high-level motor program, in which the agent moves in the negative x-direction as a consequence of stimulating high-level movement selector cells 31–40.

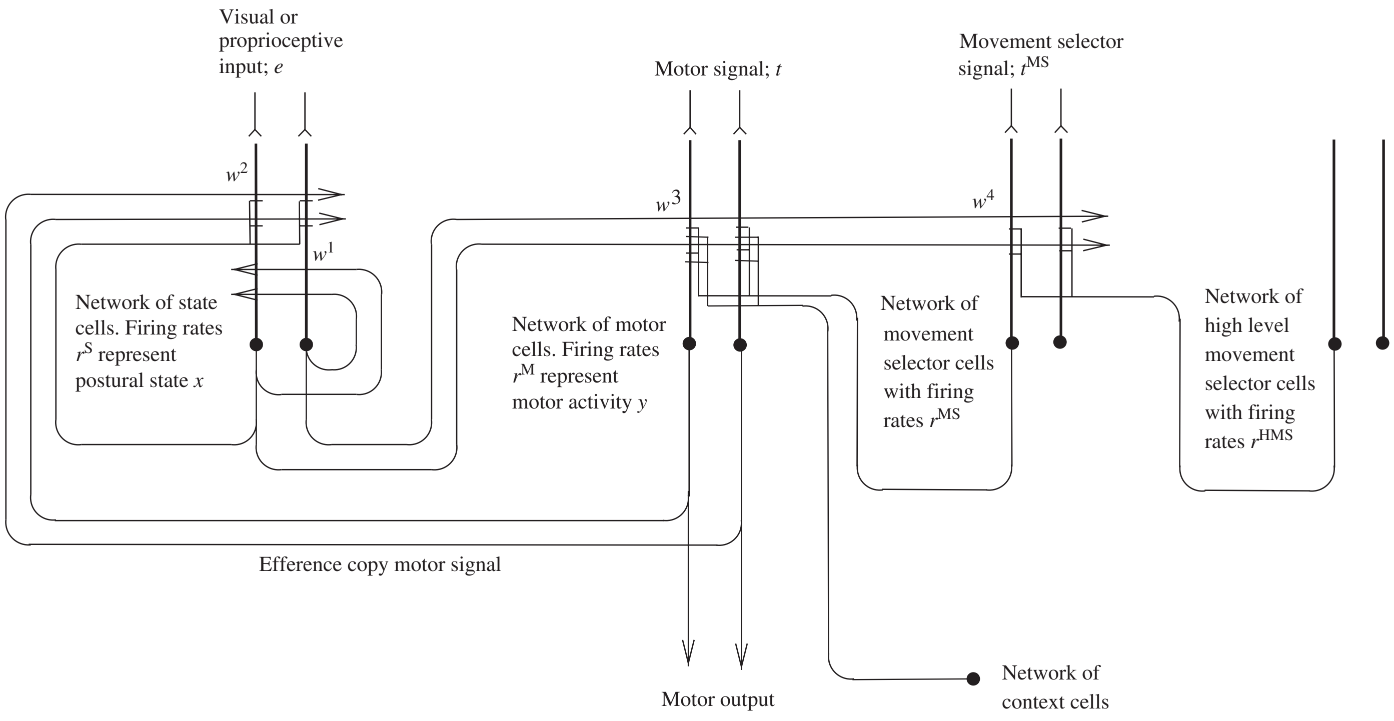
Visual or proprioceptive input; e
Motor signal; t
Movement selector signal; t^MS
w^2
w^1
w^3
w^4
Network of state cells. Firing rates r^S represent postural state x
Network of motor cells. Firing rates r^M represent motor activity y
Network of movement selector cells with firing rates r^MS
Network of high level movement selector cells with firing rates r^HMS
Efference copy motor signal
Motor output
Network of context cells

Fig. 5. Network architecture of the context augmented model used for Experiment 2. The network architecture for the model used for Experiment 2 is similar to that used in Experiment 1, except for the addition of a layer of context cells which are used to represent the context in which the motor actions are performed, and which exert their effect via the Sigma–Pi synapses w^3 driving the motor cells. The Sigma–Pi synapses w^3 driving the motor cells are from combinations of state cells, movement selector cells and context cells. This allows the context to modulate the motor activity given a particular motor primitive command represented by the movement selector cells and a particular state represented by the state cells.

gated by the w^4 connections from the state cells (shown in Fig. 1), the MS cells for each motor primitive switch on automatically when the agent reaches the beginning state for that motor primitive, and switch off automatically when the agent reaches the end state for that primitive. That is, there is no need for any further top-down input signal to guide the timing of the firing of the MS cells which represent the component motor primitives of the overall high-level motor program. The high-level motor program is performed without the training signal t_i^{MS} driving the MS cells which represent the selection of the motor primitives, without the motor training signal t_i, and without the external (visual or proprioceptive) input e_i. On the right of Fig. 4 are results for the second high-level motor program, in which the agent moves in the negative x-direction as a consequence of stimulating HMS cells 31–40.

In the above experiment we demonstrated how the HCL network model is able to learn and select different high-level motor programs which are composed of a temporal sequence of motor primitives. In further experiments (not illustrated) we demonstrated that the HCL network model is able to learn high-level motor programs which involve the *parallel* execution of a number of motor primitives. Thus, the HCL network model is able to learn high-level motor programs which are composed of complex sequences of many motor primitives blended together through time, such as might be needed to coordinate the two legs when walking.

3. Extension of the model to allow execution of the high-level motor program to be modulated by context

We now extend the model to allow the motor primitives represented by the MS cell firing to be modulated by context states. For example, if we consider the problem of an agent reaching to a target object in the environment, then a context state may be the location of an obstacle near the object. The augmented network architecture is shown in Fig. 5. The network architecture is similar to that used in Experiment 1, except for the addition of context cells which are used to represent the context in which the motor actions are performed, and which exert their effect via the Sigma–Pi synapses w^3. These Sigma–Pi synapses w^3 driving the motor cells reflect combinations of postural state cell firing, MS cell firing and context cell firing. This allows the context to modulate the motor primitive. In this case Eq. (4) governing the activation of the motor cells becomes

$$\tau \frac{\mathrm{d}h_i^{\mathrm{M}}(t)}{\mathrm{d}t} = -h_i^{\mathrm{M}}(t) + t_i + \frac{\phi_2}{C^{\mathrm{S}\times\mathrm{MS}\times\mathrm{C}}} \times \sum_{j,k,l} w^3_{ijkl} r^{\mathrm{S}}_j r^{\mathrm{MS}}_k r^{\mathrm{C}}_l, \qquad (13)$$

where r_l^{C} is the firing rate of a context cell l representing some additional state of the agent or its environment.[6] During learning the synaptic weights w_{ijkl}^3 are updated according to

$$\delta w_{ijkl}^3 = k^3 r_i^{\mathrm{M}} r_j^{\mathrm{S}} r_k^{\mathrm{MS}} r_l^{\mathrm{C}}. \tag{14}$$

3.1. Simulation results

We demonstrate the operation of this context augmented model in Experiment 2, which takes place in two stages.[7] During the first stage of the experiment the network was taught three motor primitives. The network was trained to perform each of the three motor primitives as represented in the MS cells under two contexts. In the simulation performed, each primitive produced the same sequence of postural states regardless of the context, but the motor cell firing required to do this in the different contexts was different. (A real-life example might be loading a limb in one of the contexts.) Therefore, for Experiment 2 the motor network contained 400 cells, where the three motor primitives were performed by motor cells 1–200 for the first context state, and performed by motor cells 201–400 for the second context state. (In this scenario, the motor primitives are different for the two contexts but the movements through the state space are the same. Context 1 might correspond to heavy loading, with one set of motor cells needed to achieve a particular movement. Context 2 might correspond to light loading, in which a different set of motor cells is required to achieve the same movement.) During the second stage of the experiment the network learns to perform a new high-level motor program which is composed of a temporal sequence of the three low-level motor primitives. However, during the second stage, the network needs to be trained on the high-level motor program for only one of the possible context states. The network model is then able to generalise the performance of the high-level motor program to the other context states with which the motor primitives were learned.

Stage 1: *learning to perform the three motor primitives*. The numerical simulation begins with the learning phase for the three motor primitives. For this experiment, the network of context cells contained only two neurons, where the first neuron was used to represent the first context state, and the second neuron was used to represent the second context state. For each context state, the firing rate of the cell that represented that context was set to 1, while the firing rate of the other context cell was set to 0. During context state 1, the three motor primitives were trained with motor neurons 1–200 producing the motor response. During context state 2, the three motor primitives were trained with motor neurons 201–400 producing the motor response. The postural states that evolved were the same in the two contexts. As the agent performed each motor primitive during learning, the firing rates of the state cells and the motor cells were set for each position using Gaussian functions as for Experiment 1, the firing rates of the movement selector cells used to represent each motor primitive were set to 1, and the firing rate of the relevant context cell was set to 1. During this learning phase, the synaptic weights, w_{ij}^1, w_{ijk}^2 and w_{ijkl}^3, were self-organised. However, while the motor primitives were learned, the HMS cells did not fire, and the synaptic weights w_{ijk}^4 were not altered.

Stage 2: *learning to perform the high-level motor program*. In stage 2 of Experiment 2 the network learned to perform the high-level motor program, which was composed of a temporal sequence of the three motor primitives. During the learning phase of stage 2, the full 'leaky-integrator' dynamical equations (1), (2), (13) and (5) were implemented. To learn the high-level motor program, the agent ran sequentially through the three motor primitives in order. However, this was done only for the first context state. The high-level motor program was not trained with the second context state. First, the firing rate of the first context cell was set to 1 to indicate the first context state. Then, the three low-level motor primitives were initiated by firing the MS cells that had already learned to represent the motor primitives. In addition, a set of HMS cells, 1–10, was activated. These HMS cells represented the high-level motor program. As the agent performed the high-level motor program, the synaptic weights w^4 of the connections from the set of HMS cells, 1–10, were updated according to Eq. (12).

After the learning phase for the high-level motor program was completed for the first context state, the network was tested to see if it could perform the high-level motor program when the HMS cells 1–10 were activated. In particular, the network was tested for both of the context states. For the testing phase, the full 'leaky-integrator' dynamical equations (1), (2), (13), (5), (10) and (11) were implemented. At the start of the testing phase, one of the context states was chosen, and the firing rate of the corresponding context cell was set to 1. Then an activity packet was initiated at position $x = 0.1$ (by application of a short visual input). The activity for the next 870 timesteps is shown in Fig. 6. The first column in Fig. 6 shows the network performing the high-level motor program for the first context state, and the second column shows the network performing the high-level motor program for the second context state. Within each column, we show the firing rate profiles within the networks, through time as the agent performs the learned high-level motor program. For each context state, it is shown that applying steady activity to HMS cells 1–10 during timesteps 81–790 drives the network through the high-level motor program composed of a temporal sequence of the three motor primitives. However, for the first context state,

[6]The term $C^{\mathrm{S}\times\mathrm{MS}\times\mathrm{C}}$ is the number of presynaptic connections received by each motor cell from couplings of state cells, MS cells, and context cells.

[7]For Experiment 2 we used the same parameter values as were used for Experiment 1, except for $\phi_1 = 1 \times 10^7$ and $w^{\mathrm{INH}} = 0.0055$.

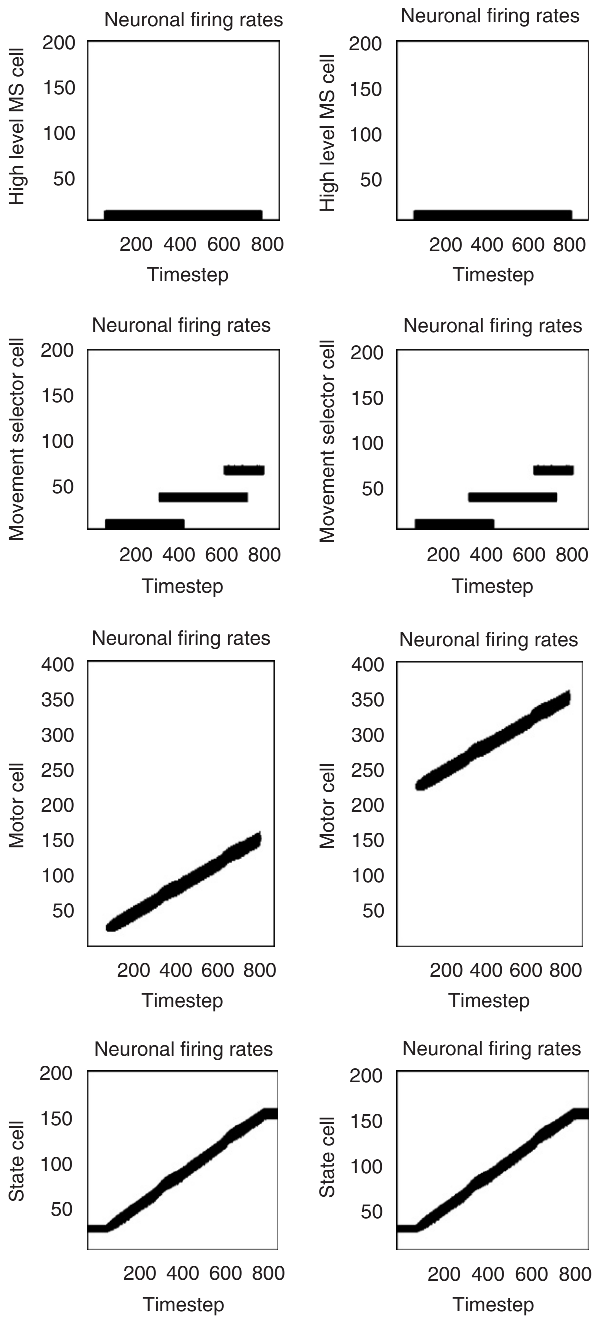

Fig. 6. Experiment 2 with the context augmented network model. This figure shows the high-level motor program learned by the network during the second phase of training. On the left is the high-level motor program for the first context state, and on the right is the high-level motor program for the second context state. Within each column, 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. For each context state, 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 three low-level motor primitives. In particular, during the execution of the motor program, the motor primitives are performed correctly according to the context. That is, for the first context state the motor activity occurs in cells 1–200 of the motor network, while for the second context state the motor activity occurs in cells 201–400 of the motor network.

the motor activity occurred in cells 1–200 of the motor network, while for the second context state, the motor activity occurred in cells 201–400 of the motor network. Thus, the execution of the high-level motor program depends correctly on the context state. Moreover, the high-level motor program was performed without the training signal t_i^{MS} for the MS cells which represent the selection of the motor primitives, without the motor training signal t_i, and without the external (visual or proprioceptive) input e_i.

The key result demonstrated in this experiment is that the network model is able to perform the high-level motor program correctly for both context states, even though the network was only trained with the high-level motor program for the first of the two context states. This is an extremely powerful property of the model. The model is able to generalise the performance of the high-level motor program to other context states with which the motor primitives were learned, but with which the high-level motor program is not trained. In particular, the sequence of motor firing performed during testing of the motor program with context state 2 was not the same sequence of motor firing performed during training. Thus, the network model is able to generate new motor sequences that have not been performed during training. This result thus demonstrates a potential solution to the problem of serial order in behaviour [8].

4. How could coupled state and motor representations self-organise in a single recurrent network?

In this section we show that the type of functionality described above (in e.g. Section 2) can be implemented in a single recurrent network, and further, how it could self-organise to have the appropriate synaptic connection strengths. This may be relevant for understanding the operation of some parts of the cerebral cortex, in which recurrent connectivity, and both sensory and motor cells are found. In such a network, all the state and motor cells would be fully connected to each other, with the same learning rules used for modifying the synaptic strengths between all types of cell. That is, the same learning rules would be used to modify synaptic connections from state cells to state cells, from state cells to motor cells, from motor cells to state cells, and from motor cells to motor cells. Moreover, there could be a variety of different kinds of synaptic connection, each with its own learning rule. In the brain, the different kinds of synaptic connection might correspond to synaptic contacts occurring on different parts of the cell dendrites and in different layers of the cortex. In such a network, the modality and firing properties of each cell would be determined during training by the external inputs to that cell, which might be for example a visual or proprioceptive input, or a motor training signal. These input signals during training would then be responsible for guiding the self-organisation of the synaptic connectivity such that the cells are able to function later as either state or motor cells, with the network as a

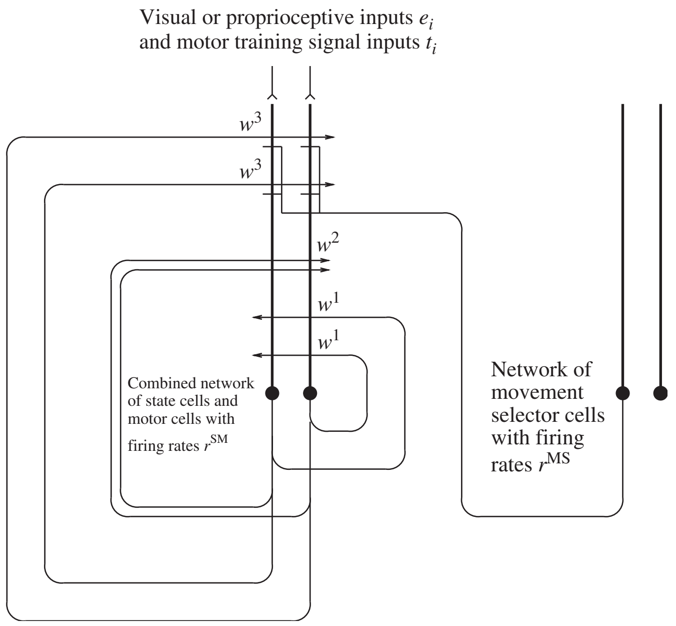
Visual or proprioceptive inputs e_i
and motor training signal inputs t_i
w^3
w^3
w^2
w^1
w^1
Combined network of state cells and motor cells with firing rates r^{SM}
Network of movement selector cells with firing rates r^{MS}

Fig. 7. Network architecture of the combined state and motor network model used for Experiment 3. There is a single combined network of state cells and motor cells, and a network of movement selector cells. Within the combined network of state cells and motor cells, the modality and firing properties of each cell are determined during training by the external inputs to that cell, which might be a visual or proprioceptive input, or a motor training signal. These input signals during training are responsible for guiding the self-organisation of the synaptic connectivity such that the cells are able to function after training as either state or motor cells. The combined state and motor network model has three types of synaptic connection: w^1, w^2, and w^3. These three types of synaptic connection function in a similar manner to the corresponding connections in the model used for Experiment 1. For each particular type of synaptic connection (w^1, w^2, or w^3), all of the state and motor cells are fully connected to each other, with the same learning rule used for modifying the synaptic strengths between all types of cell.

whole able to reproduce motor sequences learned during training. In this section we present a simple model which demonstrates one way in which this might be achieved. This is an important step towards explaining how the hierarchical network models described earlier in the paper might be self-organised in single, fully recurrently connected, brain areas. In [20] we have already shown how multiple spatial representations can be stably maintained in a single recurrent network, and updated as the agent moves through its environment.

The network architecture for the model is shown in Fig. 7. There is a single combined network of state cells and motor cells, and a separate network of MS cells. Within the combined network of state cells and motor cells, the responses of each cell are determined during training by the external inputs to that cell, which might be a visual or proprioceptive input, or a motor training signal. These input signals during training guide the self-organisation of the synaptic connectivity such that the cells function after training as either state or motor cells. The combined state and motor network model has three types of synaptic connection: w^1, w^2, and w^3. These three types of synaptic connection function in a similar manner to the corresponding connections in the SCL network model used for Experiment 1. That is, the recurrent connections w^1 are set up through Hebbian associative learning, and help to support stable packets of activity among state cells. The synapses w^2 are Sigma–Pi connections which are set up using a traced learning rule, and are responsible for performing path integration. The synapses w^3 are Sigma–Pi connections which are set up using an associative learning rule, and are responsible for firing motor cells when the MS cells are active. For each particular type of synaptic connection (w^1, w^2 or w^3), all of the state and motor cells are fully connected to each other, with the same learning rule used for modifying the synaptic strengths between all types of cell.

The behaviour of cells within the combined network of state cells and motor cells during the testing phase is governed by the following 'leaky-integrator' dynamical equations. The activation h_i^{SM} of a cell i in the combined network of state cells and motor cells (denoted SM cells) is governed by

$$\tau \frac{\mathrm{d}h_i^{SM}(t)}{\mathrm{d}t} = -h_i^{SM}(t) + \frac{\phi_0}{C^{SM}} \sum_j (w_{ij}^1 - w^{INH}) r_j^{SM}(t) + e_i + t_i + \frac{\phi_1}{C^{SM\times SM}} \sum_{j,k} w_{ijk}^2 r_j^{SM} r_k^{SM} + \frac{\phi_2}{C^{SM\times MS}} \sum_{j,k} w_{ijk}^3 r_j^{SM} r_k^{MS}, \qquad (15)$$

where the activation h_i^{SM} is driven by the following terms. The term r_j^{SM} is the firing rate of a pre-synaptic SM cell j within the combined network of state cells and motor cells, w_{ij}^1 is the excitatory (positive) synaptic weight from SM cell j to SM cell i, and w^{INH} is a global constant describing the effect of inhibitory interneurons within the layer of SM cells.[8] Further terms in Eq. (15) are as follows. The term τ is the time constant of the system. The term e_i represents an external input to SM cell i, which carries information about the state of the agent and may be visual or proprioceptive. When there is no visual or proprioceptive input, the term e_i is set to zero. The term t_i represents an external input to SM cell i, which carries a motor training signal. In the absence of external inputs, there are two key terms driving the SM cell activations in Eq. (15). Firstly, the term $\sum_{j,k} w_{ijk}^2 r_j^{SM} r_k^{SM}$ represents a sum of coupled inputs from SM cells, where r_j^{SM} is the firing rate of SM cell j, r_k^{SM} is the firing rate of SM cell k, and w_{ijk}^2 is the corresponding strength of connection from these cells.[9] These terms are responsible for performing path integration within the

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

[9]The scaling factor $\phi_1/C^{SM\times SM}$ controls the overall strength of these SM×SM inputs, where ϕ_1 is a constant, and $C^{SM\times SM}$ is the number of w^2 connections received by each SM cell.

combined network of state cells and motor cells. Secondly, the term $\sum_{j,k} w^3_{ijk} r^{SM}_j r^{MS}_k$ represents a sum of coupled inputs from SM cells and MS cells, where r^{SM}_j is the firing rate of SM cell j, and r^{MS}_k is the firing rate of MS cell k.[10] These terms are responsible for firing the motor cells when the MS cells are active.

The firing rate r^{SM}_i of cell i is given by the sigmoid activation function

$$r^{SM}_i(t) = \frac{1}{1 + e^{-2\beta(h^{SM}_i(t)-\alpha)}}, \tag{16}$$

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

During training, the three different types of synaptic connection, w^1, w^2 and w^3, are self-organised according to biologically plausible local learning rules, which are dependent on local cell quantities such as the (traced) firing rates of the pre- and post-synaptic neurons, and the external inputs to the postsynaptic neurons.

The learning rule used to update the recurrent synapses w^1_{ij} is the associative Hebb rule

$$\delta w^1_{ij} = k^1 r^{SM}_i r^{SM}_j e_i. \tag{17}$$

Learning rule (17) operates somewhat similarly to Eq. (6), except that the pre- and post-synaptic cells are SM cells rather than state cells, and except for the additional factor e_i. The factor e_i represents an external input to SM cell i carrying information about the state of the agent, which may be visual or proprioceptive. The input e_i will only be present (i.e. non-zero) for a portion of SM cells, which will then learn during training to function as state cells. We note that post-synaptic regulation of Hebbian associativity has been described by [12].

The learning rule used to update the synapses w^2_{ijk} is

$$\delta w^2_{ijk} = k^2 r^{SM}_i \bar{r}^{SM}_j \bar{r}^{SM}_k e_i. \tag{18}$$

Learning rule (18) is a traced learning rule which operates similarly to Eq. (7), except that the pre- and post-synaptic cells are SM cells rather than state cells and motor cells, and except for the additional factor e_i.

The synaptic weights w^3_{ijk} are updated during learning according to

$$\delta w^3_{ijk} = k^3 r^{SM}_i r^{SM}_j r^{MS}_k t_i. \tag{19}$$

Learning rule (19) is an associative learning rule which operates similarly to Eq. (9), except that the state and motor cells have been combined as SM cells, and except for the additional factor t_i. The factor t_i represents an external input to SM cell i carrying a motor training signal. The input t_i will only be present (i.e. non-zero) for a portion of SM cells, which will then learn during training to function as motor cells.

[10]The scaling factor $\phi_2/C^{SM\times MS}$ controls the overall strength of these SM×MS inputs, where ϕ_2 is a constant, and $C^{SM\times MS}$ is the number of w^3 connections received by each SM cell.

The combined state and motor network model described in this section and shown in Fig. 7 combines the state cells and motor cells into a single recurrent network, and so has a simpler network architecture than the SCL model presented in Section 2 which has separate state and motor networks with specific types of synaptic connection between them. Moreover, through the use of a combination of carefully designed learning rules, the combined state and motor network model is still able to self-organise during training such that it is able to perform a desired motor sequence, as shown below. However, after training, the operation of the combined state and motor network model is in fact more complex than the SCL model because for each particular type of synaptic connection (w^1, w^2 or w^3), all of the state and motor cells are fully connected to each other, with the same learning rule used for modifying the synaptic strengths between all types of cell. This gives rise to additional interactions between the cell types (state and motor) that are not present in the SCL model. For example, the learning rule shown in Eq. (17), not only strengthens connections from state cells to state cells, but also strengthens connections from motor cells to state cells. Similarly, learning rule (18), as well as strengthening connections from state cells and motor cells to state cells, also strengthens connections from pairs of state cells to other state cells, and from pairs of motor cells to state cells. Finally, learning rule (19), as well as strengthening connections from pairs of state cells and MS cells to motor cells, also strengthens connections from pairs of motor cells and MS cells to motor cells.

4.1. Simulation results

Experiment 3 demonstrates the ability of the combined state and motor network model to learn to perform a motor sequence.[11] The network architecture was as shown in Fig. 7, where the combined network of state cells and motor cells was composed of 200 state cells and 200 motor cells, and the movement selector network contained 200 cells. The network was trained to perform a motor sequence with the postural state x of the agent and the motor activity y running in lock-step from $x = 0.1$ to 0.9 with $y = x$. During the training, the selection of this movement was represented by setting the firing rates r^{MS} of MS cells 1–10 to 1, with the firing rates of the remaining MS cells 11–200 set to 0. This training was performed in a similar manner to that described for the SCL model in Experiment 1.

In the testing phase the agent performed the motor sequence without the motor training signal t_i, and without

[11]For Experiment 3 we used the following parameter values. The parameters governing the learning were: $\eta = 0.9$, $k^1 = 0.001$, $k^2 = 0.001$, and $k^3 = 0.001$. The parameters governing the 'leaky-integrator' dynamical equation (15) were: $\tau = 1$, $\phi_0 = 3 \times 10^5$, $\phi_1 = 5 \times 10^6$, $\phi_2 = 1.25 \times 10^7$, and $w^{INH} = 0.429$. The parameters governing the sigmoid activation functions for the SM cells were as follows: $\alpha^{HIGH} = 0.0$, $\alpha^{LOW} = -20.0$, $\gamma = 0.5$, and $\beta = 0.1$.

the visual and proprioceptive inputs e_i. The aim was to show that the population of MS cells could produce the desired motor sequence once the relevant MS cells were activated. For the testing phase, the full 'leaky-integrator' dynamical equations (15) and (16) were implemented. Before the MS cells were activated, the network was set into a state of firing which represented the starting point of the motor sequence (by applying visual inputs for a short period).

Numerical results from Experiment 3 are presented in Fig. 8, which shows the neuronal firing rates within the network through time as the agent performs the learned motor sequence. Top: firing rates within the MS network. Middle: firing rates of the motor cells contained within the combined state and motor network. Bottom: firing rates of the state cells contained within the combined state and motor network. It is shown that applying steady activity to the relevant MS cells during timesteps 201–1600 resulted in the continuously moving motor and postural states. In this way, the activity packet supported by the state cells and the activity packet supported by the motor cells moved together, with the network firing patterns running through the learned motor sequence maintaining the relation $x = y$. During timesteps 1601–1800 the MS cells stopped firing, and all the motor cells became quiescent while the state cells continued firing representing the state reached. The motor sequence was performed without the motor training signal t_i, and without the external (visual or proprioceptive) input e_i.

5. Discussion

The self-organising networks we described are novel in their application to learning hierarchical motor function. Most approaches to motor function have been in the arena of a servo system, whether feedforward or feedback, that attempts to minimise an error with respect to a defined target. The approaches we describe utilise a dynamical framework in which the controlling network itself generates the dynamics of the movement. This is achieved by using a continuous attractor network to represent the current state, e.g. postural state, and then utilising a mechanism for driving the packet of activity in the continuous attractor network, in order to generate the movement. The driving is achieved in the network we describe by utilising the Sigma–Pi w^2 synapses in Fig. 1 (or Fig. 5) (which were set up during training using a trace delay) to enable the current state of the network, in conjunction with the current motor signal (from the motor cell network M), to produce the next state of the postural state network. That next state of the postural state network then produces via Sigma–Pi synapses w^3 in Fig. 1 (or Fig. 5), in conjunction with the movement selector (MS) cell firing, the next pattern of motor (M) neuron firing. Because the network is a dynamical system with time constants in the updates of each stage, the whole movement evolves continuously in time.

The Single Command Layer (SCL) network model, which consists of the architecture shown in Fig. 1 but

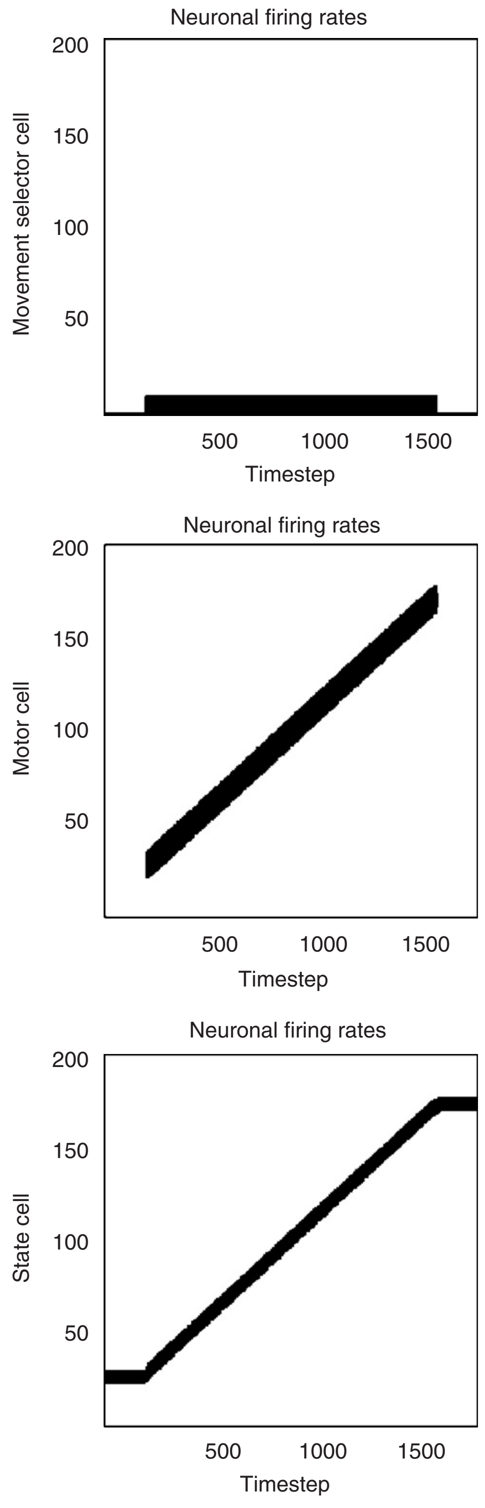

Fig. 8. Numerical results from Experiment 3, in which the combined state and motor network model is trained to perform a motor sequence. We show the neuronal firing rates within the network through time as the agent performs the learned motor sequence. Top: firing rates within the movement selector network. Middle: firing rates of the motor cells contained within the combined state and motor network. Bottom: firing rates of the state cells contained within the combined state and motor network. The motor sequence is performed without the motor training signal t_i, and without the external (visual or proprioceptive) input e_i.

without the high-level movement selector cells and the w^4 synapses, can learn hierarchical motor programs as illustrated by the following. After training on a set of three motor primitives represented by three patterns of MS cell firing, the three MS cell patterns could be presented in sequence while a fourth set of MS cells were active for the whole sequence, in order to train this fourth set of MS cells as a command set for the overall motor program. In simulations not illustrated here, it was shown that after training, when this fourth set of MS cells was made active, the whole motor program, consisting of the three primitives performed in succession, unfolded in time.

The new concept introduced in this paper is about extending the basic SCL approach [22] to hierarchical control. The advantages are that context can be introduced (which allows novel sequences to be performed during different contexts as shown in Experiment 2), and that the learning signal can use a delayed reward as shown elsewhere [19]. The extended HCL model that can deal with context was demonstrated with the new architecture shown in Fig. 5. The context signal might reflect for example different loading on a limb, or guidance round an obstacle. After training the network on the three motor primitives with each of the contexts present but without HMS cell firing, then the HMS firing is present during subsequent training of the w^4 synapses with one pattern of HMS inputs active, and in the presence of only one of the contexts. During later testing with the other context signal present, the same HMS cell firing leads to the motor response sequence appropriate for the second context, without further training. In this way, this network demonstrates generalisation from one context to another, where quite a different motor cell firing may be needed to produce the same movement (i.e. change of postural states S) in the two contexts. In particular, the sequence of motor firing performed during testing of the motor program with context state 2 was not the same sequence of motor firing performed during training. Thus, the HCL network model is able to generate new motor sequences that have not been performed during training. This demonstrates a potential solution to the problem of serial order in behaviour [8].

A very novel aspect of the approach to understanding motor function described here is that we have been able to combine the necessary functionality into a single self-organising recurrent network (Experiment 3). This is an interesting advance, for networks with recurrent connectivity and intermingled neurons with different types of response are a feature of the cerebral cortex. In the single network we describe, all the state and motor cells are fully connected to each other, with the same learning rules used for modifying the synaptic strengths between both types of cell. After training, the activity of the state and motor neuronal populations evolves dynamically to produce a full dynamical sequence corresponding to a movement with just a single MS command as the input to the single network. We know of no similar self-organising model in which the sensory and motor neurons are combined in a single network that can implement many functions fundamental in the organisation of movements.

One interesting property of the networks described is that their dynamical states do not unfold blindly with respect to the state of the agent, but instead the evolution in time is guided by the dynamics of the postural state network and its interactions with the other networks, which are set during training to reflect the time course of the changing visual or proprioceptive inputs e to the postural state (S) network. In this way, the network learns and becomes tuned to the actual dynamics of the system being controlled.

The simulations described are for a one-dimensional space, although the network can operate in higher dimensional spaces. In higher dimensions the network can learn arbitrary paths. Indeed, part of the utility of the context inputs is that they could allow the agent to learn trajectories in which the paths crossed, as for example in drawing the number '8'. 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.

As noted in the Introduction, there has been little previous work in applying self-organising 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. [22] 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 the correct movement. 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 we describe does provide a possible computational basis for what could be implemented (with different implementation details) in the brain. The networks described above do 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 incorporate context effects such as altered loading forces on the system. In addition, and in the direction of biological plausibility, we have described elsewhere how Sigma–Pi neurons can be replaced with neurons that self-organise using competitive learning to represent combinations of the input signals that can then be correctly mapped using pattern association learning [15].

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

[1] S. Amari, Dynamics of pattern formation in lateral-inhibition type neural fields, Biol. Cybern. 27 (1977) 77–87.

[2] E. Bizzi, A. Polit, Processes controlling visually evoked movements, Neuropsychologia 17 (1979) 203–213.

[3] G. Deco, E.T. Rolls, Attention and working memory: a dynamical model of neuronal activity in the prefrontal cortex, Eur. J. Neurosci. 18 (2003) 2374–2390.

[4] C. Ghez, J. Krakauer, The organisation of movement, in: E.R. Kandel, J.H. Schwartz, T.M. Jessell (Eds.), Principles of Neural Science, McGraw-Hill, New York, fourth ed., 2000, pp. 653–673 (Chapter 33).

[5] J. Hughlings Jackson, Selected writings of John Hughlings Jackson 2, in: J. Taylor (Ed.), Hodder and Staughton, London, 1878 (edited in 1932).

[6] M. Jueptner, C.D. Frith, D.J. Brooks, R.S.J. Frackowiak, R.E. Passingham, Anatomy of motor learning. II. subcortical structures and learning by trial and error, J. Neurophysiol. 77 (1997) 1325–1337.

[7] F. Lacquaniti, C. Terzuolo, P. Viviani, The law relating the kinematic and figural aspects of drawing movements, Acta Psychol. 54 (1983) 115–130.

[8] K.S. Lashley, The problem of serial order in behaviour, in: L.A. Jeffress (Ed.), Cerebral Mechanisms in Behaviour, Wiley, New York, 1951, pp. 112–136.

[9] J.L. Laszlo, The performance of a single motor task with kinesthetic sense loss, Q. J. Exp. Psychol. 18 (1966) 1–8.

[10] J.L. Laszlo, Training of fast tapping with reduction of kinesthetic, tactile, visual, and auditory sensation, Q. J. Exp. Psychol. 19 (1967) 344–349.

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

[12] M. Nishiyama, K. Hong, K. Mikoshiba, M. Poo, K. Kato, Calcium stores regulate the polarity and input specificity of synaptic modification, Nature 408 (2000) 584–588.

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

[14] E.T. Rolls, G. Deco, Computational Neuroscience of Vision, Oxford University Press, Oxford, 2002.

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

[16] E.T. Rolls, A. Treves, Neural Networks and Brain Function, Oxford University Press, Oxford, 1998.

[17] R.A. Schmidt, Motor Control and Learning: A Behavioural Emphasis, Second ed. Human Kinetics, Champaign, IL, 1987.

[18] R.A. Schmidt, Motor and action perspectives on motor behaviour, in: O.G. Meijer, K. Roth (Eds.), Complex Motor Behaviour: The Motor-Action Controversy, Elsevier, Amsterdam, 1988, pp. 3–44.

[19] S.M. Stringer, E.T. Rolls, P. Taylor, Learning movement sequences with a delayed reward signal in a hierarchical model of motor function, Neural Networks, 2006, in press, doi:10.1016/j.neunet.2006.01.016.

[20] S.M. Stringer, E.T. Rolls, T.P. Trappenberg, Self-organising continuous attractor networks with multiple activity packets, and the representation of space, Neural Networks 17 (2004) 5–27.

[21] S.M. Stringer, E.T. Rolls, T.P. Trappenberg, I.E.T. De Araujo, Self-organising continuous attractor networks and path integration: two-dimensional models of place cells, Network Comput. Neural Syst. 13 (2002) 429–446.

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

[23] S.M. Stringer, T.P. Trappenberg, E.T. Rolls, I.E.T. De Araujo, Self-organising continuous attractor networks and path integration: one-dimensional models of head direction cells, Network Comput. Neural Syst. 13 (2002) 217–242.

[24] J.G. Taylor, Neural 'bubble' dynamics in two dimensions: foundations, Biol. Cybern. 80 (1999) 393–409.

[25] D.M. Wolpert, Z. Ghahramani, Computational principles of movement neuroscience, Nature Neurosci. 3 (Suppl.) (2000) 1212–1217.

[26] D.M. Wolpert, R.C. Miall, M. Kawato, Internal models in the cerebellum, Trends Cognitive Sci. 2 (1998) 338–347.

Dr. Simon M. Stringer has been a research mathematician at Oxford University for twelve years. During this time he has conducted research in computational aerodynamics, mathematical epidemology and theoretical neuroscience. Dr. Stringer is currently based in the Department of Experimental Psychology, where he is developing computer models of vision, spatial processing and motor behaviour.

Professor Edmund T. Rolls is Professor of Experimental Psychology at the University of Oxford. He has written the book Computational Neuroscience of Vision (with G. Deco, 2002, Oxford University Press), and papers shown at www.cns.ox.ac.uk.