no. 2953

Movement Generation with Circuits of Spiking Neurons*

Prashant Joshi, Wolfgang Maass

Institute for Theoretical Computer Science

Technische Universität Graz

A-8010 Graz, Austria

{joshi, maass}@igi.tugraz.at

December 10, 2004

*The work was partially supported by the Austrian Science Fund FWF, project # P15386, and *PASCAL*, project # IST2002-506778, of the European Union.

Abstract

How can complex movements that take hundreds of milliseconds be generated by stereotypical neural microcircuits consisting of spiking neurons with a much faster dynamics? We show that linear readouts from generic neural microcircuit models can be trained to generate basic arm movements. Such movement generation is independent of the arm-model used and the type of feedbacks that the circuit receives. We demonstrate this by considering two different models of a two-jointed arm, a standard model from robotics and a standard model from biology, that each generate different kinds of feedback. Feedbacks that arrive with biologically realistic delays of 50–280 ms turn out to give rise to the best performance. If a feedback with such desirable delay is not available, the neural microcircuit model also achieves good performance if it uses internally generated estimates of such feedback. Existing methods for movement generation in robotics that take the particular dynamics of sensors and actuators into account ("embodiment of motor systems") are taken one step further with this approach, which provides methods for also using the "embodiment of motion generation circuitry", i.e., the inherent dynamics and spatial structure of neural circuits, for the generation of movements.

1 Introduction

Using biologically realistic neural circuit models to generate movements is not so easy, since these models are made of spiking neurons and dynamic synapses which exhibit a rich inherent dynamics on several temporal scales. This tends to be in conflict with movement tasks that require sequences of precise motor commands on a relatively slow time scale. However we show that without the construction of any particular circuit, training a linear readout to take a suitable weighted sum (with *fixed* weights after training) of the output activity of a fairly large number of neurons in a generic neural microcircuit model provides a very general paradigm for movement generation. It is obviously reminiscent of a number of experimental results (see e.g. (Wessberg et al., 2000)) which show that a suitable weighted sum of the activity from a fairly large number of cortical neurons in monkeys predicts quite well the trajectory of hand positions for a variety of arm movements. Obviously the neural microcircuit model assumes here a similar role as a kernel for support vector machines in machine learning (see (Maass et al., 2004b) and (Maass et al., 2004a) for details).

This article demonstrates that controllers made from generic neural microcircuits are functionally "generic" in the sense that readouts from such circuits can learn to control the arm irrespective of the model that is used to describe the arm dynamics, the type of feedbacks used (visual or proprioceptive), and also the type of movements that are generated. This is shown here by teaching the same generic neural circuit to generate reaching movements for two different models, with different kinds of feedbacks. The first model

used (Model 1) is the standard model of a 2-joint robot arm described in (Slotine and Li, 1991). The other model (Todorov, 2003; Todorov, 2000) comes from biology and relates the activity of neurons in the cortical motor area M1 to the kinematics of the arm (Model 2).

It turns out that both the spatial organization of information streams, especially the population coding of slowly varying input variables, and the inherent dynamics of the generic neural microcircuit model have a significant impact on its capability to generate movements. In particular it is shown that the inherent dynamics of neural microcircuits allows these circuits to cope with rather large delays for proprioceptive and sensory feedback. In fact it turns out that the performance of this generic neurocontroller is optimal for feedback delays that lie in the biologically realistic range of 50 to 280 ms. Furthermore, it is shown that other readout neurons from the same neural microcircuit model can be trained simultaneously to estimate results of such feedbacks, and that in the absence of real feedbacks the precision of reaching movements can be improved significantly if the circuit gets access to these estimated feedbacks.

This work complements preceding work where generic neural microcircuit models were used in an open loop for a variety of sensory processing tasks ((Buonomano and Merzenich, 1995), (Maass et al., 2002), (Maass et al., 2004b)). It turns out that the demands on the precision of real-time computations carried out by such circuit models are substantially higher for closed-loop applications such as those considered in this article. The paradigm for movement generation discussed in this article is somewhat related to preceding work (Ijspeert et al., 2003), where a fixed parametrized system of differential equations was

used instead of neural circuits, and to the melody-generation and prediction of chaotic time series with artificial neural networks in discrete time of (Jäger, 2002; Jäger and Haas, 2004). In these other models no effort is made to choose a movement generator whose inherent dynamics has a similarity to that of biological neural circuits. It has not yet been sufficiently investigated whether feedback, especially feedback with a realistic delay, can have similarly beneficial consequences in these other models.

No effort was made in this article to make the process by which the neural circuit model (more specifically: the readouts from this circuit) learns to generate specific movement primitives in a biologically realistic fashion. Hence the results of this article only provide evidence that a generic neural microcircuit can hold the information needed to generate certain movement primitives, and that it can generate a suitable slow dynamics with high precision.

The structure of this article is as follows: Section 2 describes the neural microcircuit model. This is followed by the description of the robot arm model (Model 1) in section 3. Sections 4, 5, and 6 present results of computer simulations for Model 1. Section 7 repeats the experiment described in section 4 for the biologically motivated arm model (Model 2). Finally we discuss robustness issues related to our new paradigm for movement generation in section 8.

A preliminary version of some results from this article (for movements of just one fixed temporal duration, and without Model 2) have previously been presented at a conference (Joshi and Maass, 2004).

2 Generic neural microcircuit models

In contrast to common artificial neural network models, neural microcircuits in biological organisms consist of diverse components such as different types of spiking neurons and dynamic synapses, that are each endowed with an inherently complex dynamics of its own. This makes it difficult to construct neural circuits out of biologically realistic computational units that solve specific computational problems, such as generating arm movements to various given targets. In fact, the generation of a smooth arm movement appears to be particularly difficult for a circuit of spiking neurons, since the dynamics of arm movements takes place on a time scale of hundreds of milliseconds, whereas the inherent dynamics of spiking neurons takes place on a much faster time scale. We show that this problem can be solved, even with a generic neural microcircuit model whose internal dynamics has not been adjusted or specialized for the task of creating arm movements, by taking as activation command for a muscle at any time t a weighted sum $\mathbf{w} \times \mathbf{z}(t)$ of the vector $\mathbf{z}$ that describes the current firing activity of all neurons in the circuit.[1] The weight vector $\mathbf{w}$, which remains fixed after training, is the only part that needs to be specialized for the generation of a particular movement task. Each component of $\mathbf{z}(t)$ models the impact that a particular neuron v may have on the membrane potential of a generic readout neuron. Thus each spike of neuron v is replaced by a pulse of unit amplitude 1 that decays exponentially with a time constant of 30 ms. In other words: $\mathbf{z}(t)$ is obtained by

[1] As usual a constant component is formally included in $\mathbf{z}(t)$ so that the term $\mathbf{w} \times \mathbf{z}(t)$ may contain some fixed bias.

applying a low-pass filter to the spike trains emitted by the neurons in the generic neural microcircuit model. Note that it is already known that hand trajectories of monkeys can be recovered from the current firing activity $\mathbf{z}(t)$ of neurons in motor cortex through the same types of weighted sums as considered in this article (Wessberg et al., 2000).

In principle one can of course also view various parameters within the circuit as being subject to learning or adaptation, for example in order to optimize the dynamics of the circuit for a particular range of control tasks. However this has turned out to be not necessary for the applications described in this article, although it remains an interesting open research problem how unsupervised learning could optimize a circuit for motor control tasks. One advantage of just viewing the weight vector $\mathbf{w}$ as being plastic is that learning is quite simple and robust, since it just amounts to linear regression – in spite of the highly nonlinear nature of the control tasks to which this set-up is applied. Another advantage is that the same neural microcircuit could potentially be used for various other information processing tasks (e.g. prediction of sensory feedback, see section 6) that may be desirable for the same or other tasks.

The generic microcircuit models used for the closed loop control tasks described in this article were similar in structure to those that were earlier used for various sensory processing tasks in an open loop. More precisely, we considered circuits consisting of 600 leaky-integrate-and-fire neurons arranged on the grid points of a $20 \times 5 \times 6$ cube in 3D (see Fig. 1). 20 % of these neurons were randomly chosen to be inhibitory. Synaptic connections were chosen according to a biologically realistic probability distribution that favored local connections but also allowed some long range connections. Biologically

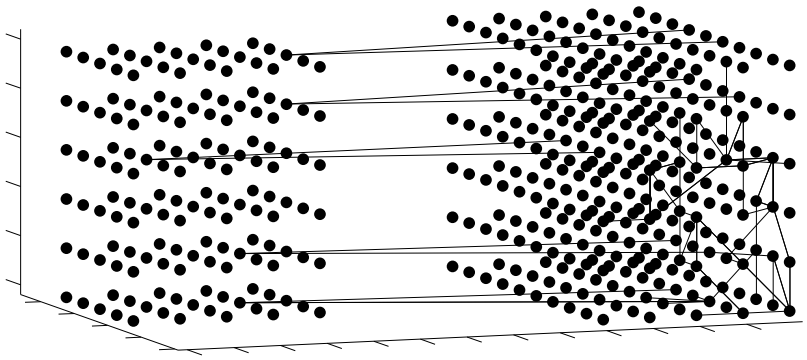

Figure 1: Spatial arrangement of neurons in the neural microcircuit models considered in this article. The neurons in the 6 layers on the left hand side encode the values of the 6 input/feedback variables $x_{dest}, y_{dest}, \theta_1(t-\Delta), \theta_2(t-\Delta), \tau_1(t), \tau_2(t)$ in a standard population code. Connections from these 6 input layers (shown for a few selected neurons), as well as connections between neurons in the subsequent 6 processing layers are chosen randomly according to a probability distribution discussed in the text (a typical example is shown).

realistic models for dynamic synapses were employed instead of the usual static synapses of artificial neural network models. Parameters of neurons and synapses were chosen to fit data from microcircuits in rat somatosensory cortex (based on (Gupta et al., 2000) and (Markram et al., 1998)), see the Appendix.

In order to test the noise robustness of movement generation by the neural microcircuit model the initial condition of the circuit was randomly drawn (initial membrane potential for each neuron drawn uniformly from the interval [13.5 mV, 14.9 mV], where 15 mV was the firing threshold). In addition a substantial amount of noise was added to the input current of each neuron throughout the simulation at each time-step, a new value for the noise input current with mean 0 and SD of 1 nA was drawn for each neuron and added (subtracted) to its input current.

The neural circuit receives in the case of the arm model that is considered in sections 3 - 6 (Model 1) analog input streams from 6 sources (from 8 sources in the experiment with internal predictions discussed in Fig. 8 and 9). A critical factor for the performance of these neurocontrollers is the way in which these time-varying analog input streams are fed into the circuit. The outcomes of the experiments discussed in this article would have been all negative if these analog input streams were fed into the circuit as time-varying input currents. Apparently the variance of the resulting spike trains were too large to make the information about the slowly varying values of these input streams readily accessible to the circuit. Therefore we employed instead a standard form of population coding ((Pouget and Latham, 2003). Each of the 6 time varying input variables was mapped onto an array of 50 symbolic input neurons with bell-shaped tuning curves (see Appendix). Thus the

value of each of the 6 input variables is encoded at any time by the output values of the associated 50 symbolic input neurons (of which at least 43 neurons output at any time the value 0). The neurons in each of these 6 input arrays are connected[2] with one of the 6 layers consisting of 100 neurons in the circuit of 100×6 $((20 \times 5) \times 6)$ integrate-and-fire neurons, providing a time-varying input current to a randomly selected subset of integrate-and-fire neurons on that layer; see Fig. 1.

3 A 2-joint robot arm as a benchmark nonlinear control task

We first trained a generic neural microcircuit model (see Fig. 1 and Fig. 2) to control a standard model for a 2-joint robot arm (Model 1), see Fig. 3. This model is used in (Slotine and Li, 1991) as a standard reference model for a complex nonlinear control task (see in particular ch. 6 and 9). It is assumed that the arm is moving in a horizontal plane, so that gravitational forces can be ignored.

Using the well-known Lagrangian equation in classical dynamics, the dynamic equations for this arm model are given by equation 1:

$$\begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + \begin{bmatrix} -h\dot{\theta}_2 & -h(\dot{\theta}_1 + \dot{\theta}_2) \\ h\dot{\theta}_1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{1}$$

[2]with a value of 3.3 for λ in the formula for the connection probability given in the Appendix.

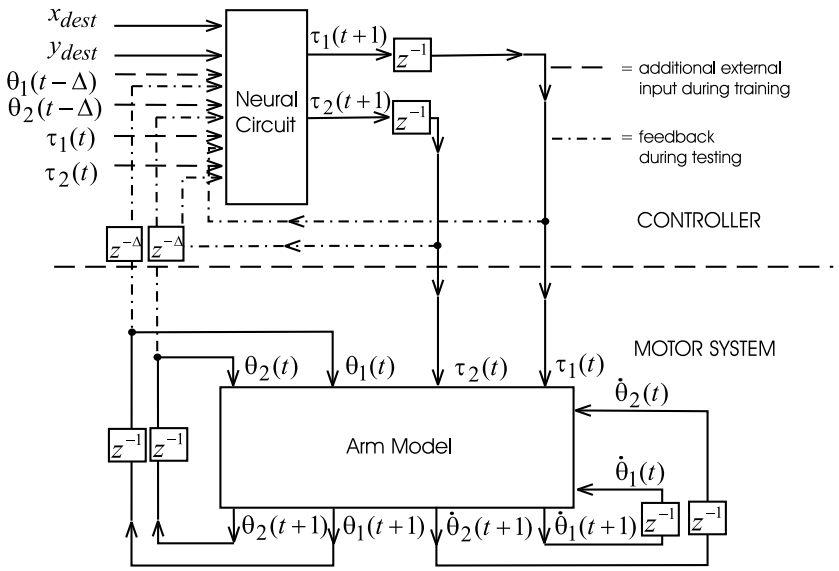

Figure 2: Closed loop application of a generic neural microcircuit. The weight vectors of the linear readouts from this circuit that produce the next motor commands $\tau_1(t+1), \tau_2(t+1)$ are the only parameters that are adjusted during training. After training the neural circuit receives in this closed loop as inputs a target position $x_{dest},\ y_{dest}$ for the tip of the robot arm (in cartesian coordinates; these input remain constant during the subsequent arm movement) as well as feedback $\theta_1(t-\Delta), \theta_2(t-\Delta)$ from the arm representing previous values of joint angles delayed by an amount Δ, as well as "efferent copies" $\tau_1(t), \tau_2(t)$ of its preceding motor commands. All the dynamics needed to generate the movement is then provided by the inherent dynamics of the neural circuit in response to the switching on of the constant inputs (and in response to the dynamics of the feedbacks). During training of the readouts from the generic neural circuit the proprioceptive feedbacks $\theta_1(t-\Delta), \theta_2(t-\Delta)$ and the efferent copies of previous motor commands $\tau_1(t), \tau_2(t)$ are replaced by corresponding values for a target movement which are given as external inputs to the circuit ("imitation learning").

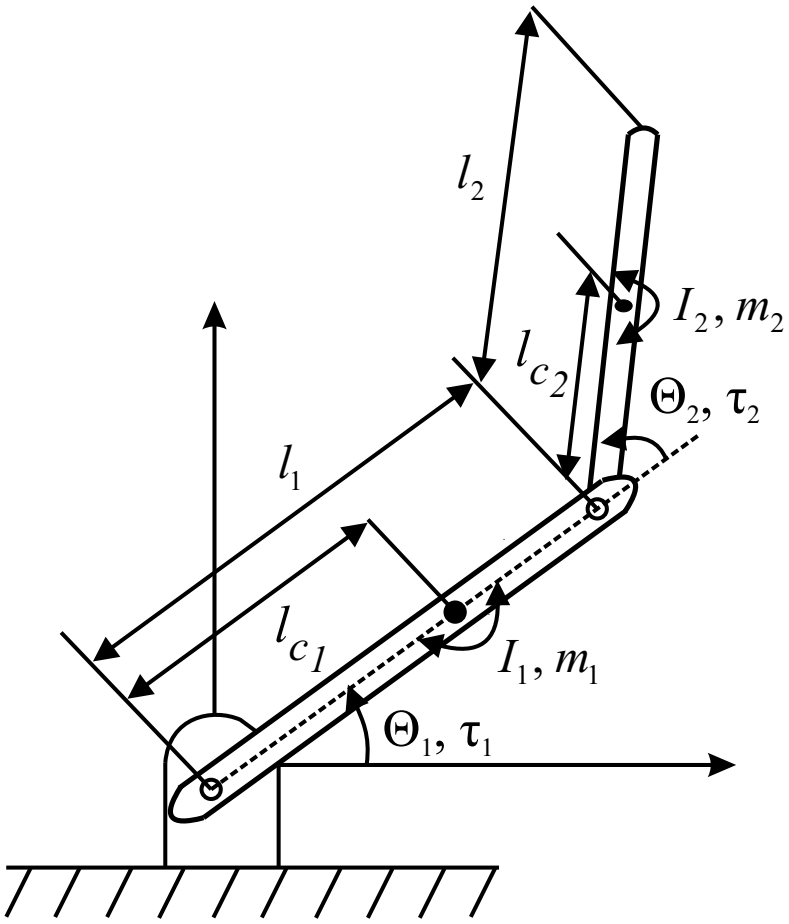

Figure 3: Standard model of a 2-joint robot arm.

with $\boldsymbol{\theta} = [\theta_1 \ \theta_2]^T$ being the two joint angles, $\boldsymbol{\tau} = [\tau_1 \ \tau_2]^T$ being the joint input torques to the two joints, and

$$H_{11} = m_1 {l_{c_1}}^2 + I_1 + m_2[{l_1}^2 + {l_{c_2}}^2 + 2\ l_1 l_{c_2} \cos\theta_2] + I_2$$

$$H_{12} = H_{21} = m_2 l_1 l_{c_2} \cos\theta_2 + m_2 {l_{c_2}}^2 + I_2$$

$$H_{22} = m_2 {l_{c_2}}^2 + I_2$$

$$h = m_2 l_1 l_{c_2} \sin\theta_2 \ .$$

Equation 1 can be compactly written as:

$$H(\theta)\,\ddot{\boldsymbol{\theta}} + C(\theta,\dot{\theta})\,\dot{\boldsymbol{\theta}} = \boldsymbol{\tau}$$

where H represents the inertia matrix, and C represents the matrix of Coriolis and centripetal terms. I_1, I_2 are the moments of inertia of the two joints. The values of the parameters that were used in our simulations were: $m_1 = 1, m_2 = 1, l_{c_1} = 0.25, l_{c_2} = 0.25, I_1 = 0.03, I_2 = 0.03$.

The closed loop control system that we used is shown in Fig. 2. During training of the weights of the linear readouts from the generic neural microcircuit model the circuit was used in an open loop with target values for the output torques provided by equation 1 (for a given target trajectory $\{\theta_1(t), \theta_2(t)\}$), and feedbacks from the plant replaced by the target values of these feedbacks for the target trajectory. The delay Δ of the proprioceptive or sensory feedback is assumed to have a fixed value of 200 ms, except for section 6 where we study the impact of this value for the precision of the movement. For each such target trajectory 20 variations of the training samples were generated, for which at each time step[3] t a different noise value of $10^{-5} \times \rho$ was added to each of the input channels where ρ is a random number drawn from a gaussian distribution with mean 0 and SD 1, multiplied by the current value of that input channel. The purpose of this extended training procedure was to make the readout robust with regard to deviations from the target trajectory caused by faulty earlier torque outputs given by the readouts from the neural circuit (see section

[3]All time steps were chosen to have a length of 2 ms, except for the experiment reported in Fig. 6, where a step size of 1 ms was used to achieve a higher precision.

8). Each target trajectory had a time duration of 500 ms.

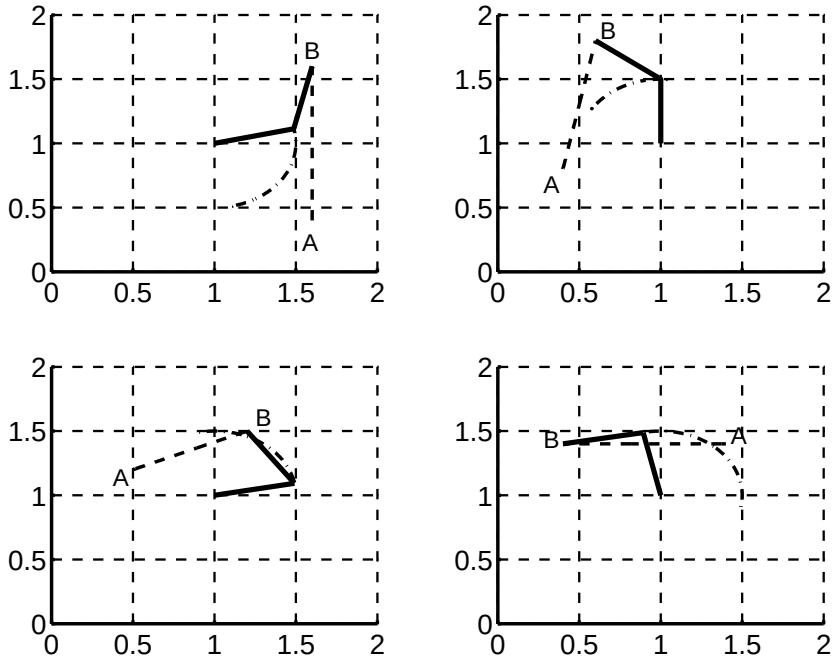

Figure 4: Initial position A and end position B of the robot arm (Model 1) for 4 target movements, scaled in meters. The target trajectory of the tip of the robot arm and of the elbow are indicated by dashed and dashed/dotted lines. One sees clearly that even simple linear movements of the tip to the arm require quite nonlinear movements of the elbow.

4 Teaching a generic neural microcircuit model to generate basic movements

As a first task, the generic neural microcircuit model described in section 2, was taught to generate with the 2-joint arm described in section 3, the 4 movements indicated in Fig. 4. In each case the task was to move the tip of the arm from point A to point B on a straight

line, with a biologically realistic bell-shaped velocity profile. The two readouts from the neural microcircuit model were trained by linear regression to output the joint torques required for each of these movements.[4]

20 noisy variations of each of the 4 target movements were used for the training of the two readouts by linear regression, as specified in section 3. Note that each readout is simply modeled as a linear gate with weight vector $\mathbf{w}$ applied to the liquid state $\mathbf{x}(t)$ of the neural circuit. This weight vector is fixed after training, and during validation all 4 movements are generated with this fixed weight vector at the readout.

The performance of the trained neural microcircuit model during validation in the closed loop (see Fig. 2) is demonstrated in Fig. 5. When the circuit receives as input the coordi-

[4] Training data were generated as follows: For a given start point $\langle x_{start}, y_{start} \rangle$ and target end point $\langle x_{dest}, y_{dest} \rangle$ of a movement (both given in cartesian coordinates) an interpolating trajectory of the tip of the arm was generated according to the following equation given in (Flash and Hogan, 1965):

$$x(t) = x_{start} + (x_{start} - x_{dest}) \cdot (15\tau^4 - 6\tau^5 - 10\tau^3)$$

$$y(t) = y_{start} + (y_{start} - y_{dest}) \cdot (15\tau^4 - 6\tau^5 - 10\tau^3)$$

where $\tau = t/MT$ and MT is the target movement time (in this case $MT = 500$ ms). From this target trajectory for the endpoint of the robot arm we had generated target trajectories of the angles Θ_1, Θ_2 of the robot arm by applying standard equations from geometry (see e.g. (Craig, 1955)). From these the target trajectories of the torques were generated according to equ. (1).

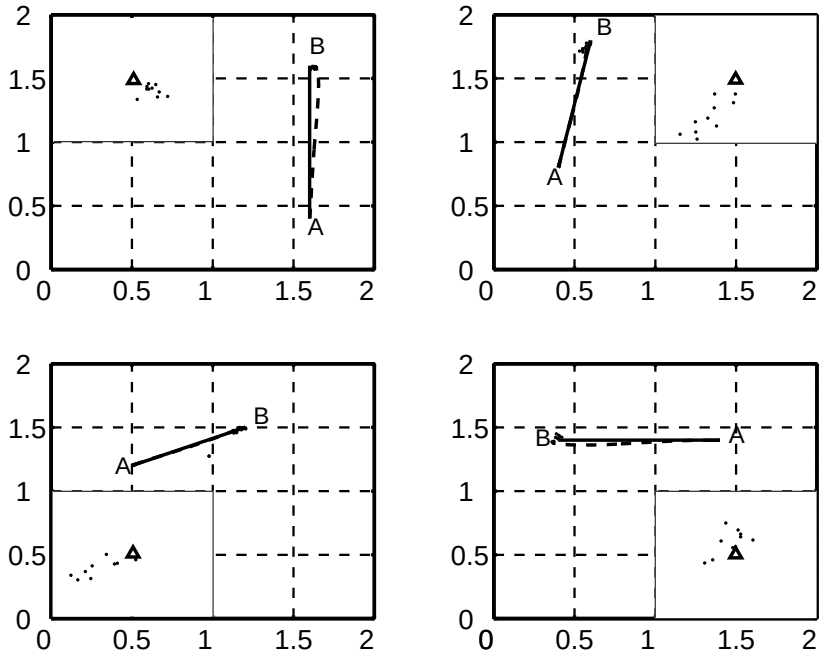

Figure 5: Target trajectories of the tip of the robot arm as in Fig. 4 (solid) and resulting trajectories of the tip of the robot arm in a closed loop for one of the test runs (dashed). The dots around the target end points show the end-points of the tip of the robot arm for 10 test runs for each of the movements (enlarged inserts show a 20 cm $\times$ 20 cm area with target end point B marked by a black open triangle. Differences are due to varying initial conditions and simulated inherent noise of the neural circuit. Nevertheless all movement trajectories converged to the target, with an average deviation from the target end point of 4.72 cm, and the SD of 0.85 cm (scale of figures in m).

nates $\langle x_{dest}, y_{dest} \rangle$ of the endpoint B of one of the target movements shown in Fig. 4, the circuit autonomously generates in a closed loop the torques needed to move the tip of the 2-joint arm from the corresponding initial point A to this endpoint B [5].

[5] In these experiments no effort was made to stabilize the endpoint of the arm at or near the target position. Rather the movement was externally halted at the end of the allotted

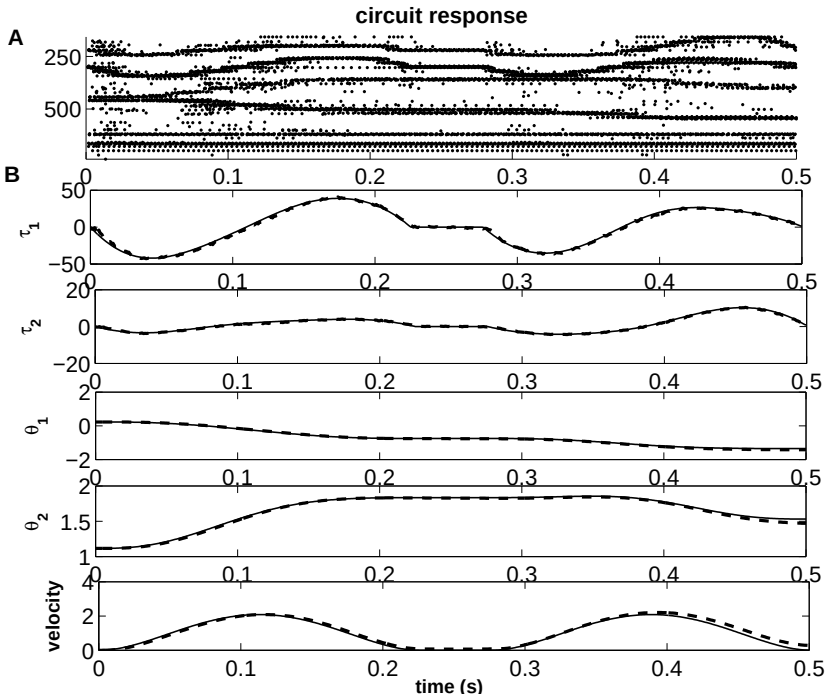

Figure 6: Demonstration of the temporal integration capability of the neural controller. The data shown are for a validation run for a circuit that has been trained to generate a movement that requires an intermediate stop and then autonomous continuation of the movement after 50 ms. **a)** Spike raster of the 600 neurons on the right hand side of Fig. 1. Note that the readout neurons receive at time t only information about the last few spikes before time t (more precisely: they receive at time t the liquid state $x(t)$ of the circuit as their only input). **b)** Target time courses of the joint angles θ_1, θ_2, joint torques τ_1, τ_2 and of the velocity of the tip of the robot arm are shown as solid line, actual time courses of these variables during a validation run in closed loop as dashed lines.

Obviously temporal integration capabilities of the controller are needed for the control of many types of movements. The next experiment was designed to test explicitly this capability of neurocontrollers constructed from generic circuits of spiking neurons. Fig. 6 shows results for the case where the readouts from the neural microcircuit have been trained to generate an arm movement with an intermediate stop of all movement from 225 to 275 ms (see the velocity profile at the bottom of Fig. 6). The initiation of the continuation of the movement at time t = 275 ms has to take place without any external cue, just on the basis of the inherent temporal integration capability of the neural circuit. For the sake of demonstration purposes we chose for the experiment reported in Fig. 6 a feedback delay of just 1 ms, so that all circuit inputs are constant during 49 ms of the 50 ms while the controller has to wait, forcing the readouts to decide just on the basis of the inherent circuit dynamics when to move on. Nevertheless the average deviation of the tip of the robot arm for 20 test runs (with noisy initial conditions and noise on feedbacks as before) was just 6.86 cm, and the bottom part of Fig. 6 shows (for a sample test run) that the tip of the robot arm came to a halt during the period from 225 to 275 ms, and then autonomously continued to move.

---

time period of 500 ms. Hence the neural circuit model acts as a movement generator, rather than as a controller. However we are not aware of a fundamental obstacle which would make it impossible to teach a circuit to stabilize the arm once it has reached the target position (which the circuit receives as an extra input).

5 Generalization capabilities

The trained neurocontroller (with the weights of the linear readouts being the only parameters that were adjusted during training) had some limited capabilities to generate arm-reaching movements to new targets. For the experiment reported in Fig. 7, the circuit was trained to generate from a common initial position reaching movements to 8 different target positions, given in terms of their cartesian coordinates as constant inputs $\langle x_{dest}, y_{dest} \rangle$ to the circuit. After training the circuit was able to generate with fairly high precision reaching movements to other target points never used during training, provided that they were located between target points used for training. The autonomously generated reaching movements moved the tip of the robot arm on a rather straight line with bell-shaped velocity profile, just as for those reaching movements to targets that were used for training.

6 On the role of feedback delays and autonomously generated feedback estimates

Our model assumes that the neural circuit receives as inputs in addition to the constant target end points and efferent copies $\tau_1(t), \tau_2(t)$ of its movement commands with very little delay, also proprioceptive or visual feedback that provides at time t information about the values of the angles of the joints at time $t - \Delta$. Whereas it is quite difficult to construct circuits or other artificial controllers for imitating movements that can benefit

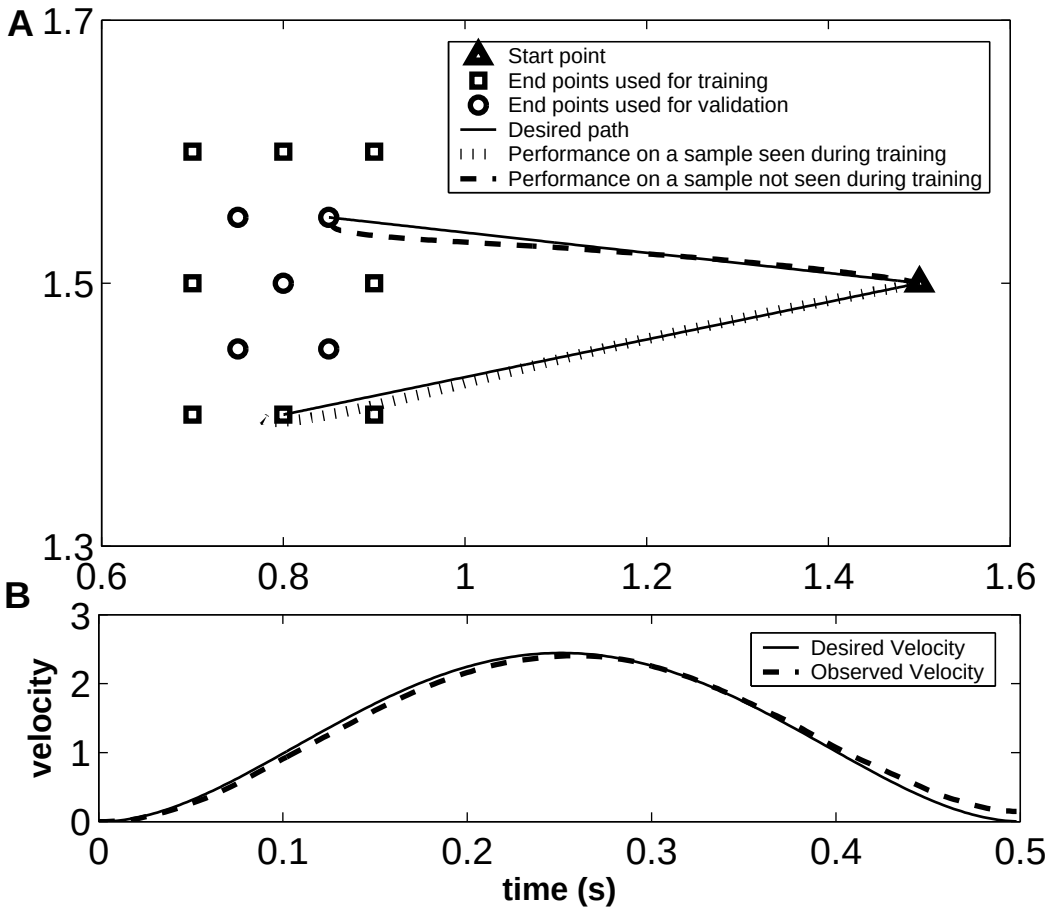

Figure 7: Generation of reaching movements to new target end points that lie between end points used for training. **a)** Generalization of movement generation to 5 target end points (small circles) which were not among the 8 target end points (small squares) that occurred during training. Movements to a new target end point was initiated by giving its cartesian coordinates as constant inputs to the circuit. Average deviation for 15 runs with new target end points: 10.3 cm (4.8 cm for target end points that occurred during training). **b)** The velocity profile for one of the movements to a new target end point (solid line is ideal bell-shaped velocity profile, actual - dashed).

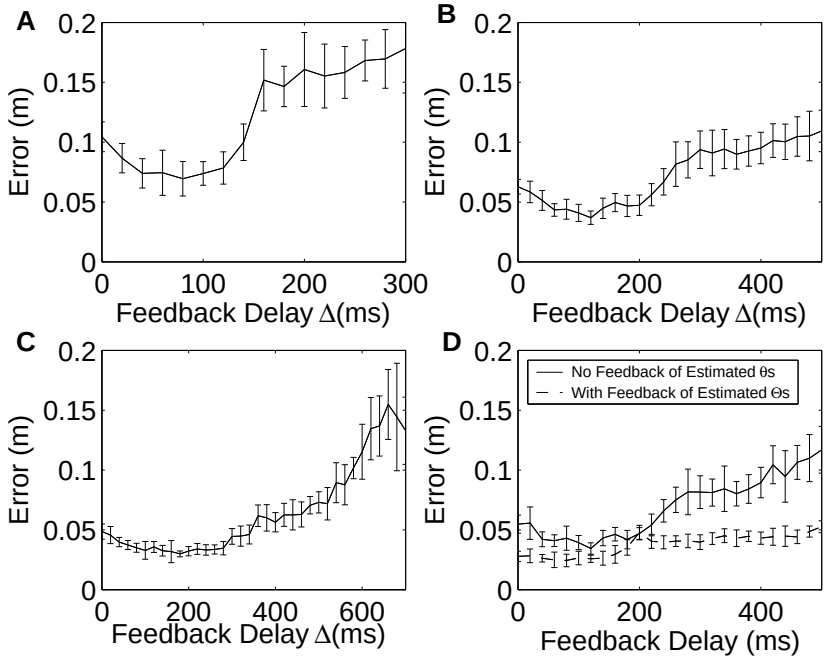

Figure 8: Influence of feedback delay Δ on movement error. Error is defined as the difference of desired and observed end-point of movement. The delay Δ is for proprioceptive feedbacks $\theta_1(t-\Delta), \theta_2(t-\Delta)$. The curves show the averages and the vertical bars show the SD of the data achieved for 400 movements for each value of Δ (4 different movements as shown in Fig. 4 repeated 10 different times with different random initial conditions of the circuit and different online noise for each of 10 randomly drawn generic neural microcircuit models). Panels a), b), c) show these data for 3 different movement durations: 300, 500, and 700 ms. Panel d) shows in the upper curve results for a slightly larger neural circuit (consisting of 800 instead of 600 early integrate-and-fire neurons). The lower (dashed) curve in d) shows the performance of the same circuits when internally generated estimates of proprioceptive feedbacks (for a delay of 200 ms) were fed back as additional inputs to the neural circuit. Note that the use of such internally estimated feedbacks not only improves the movement precision for all values of the actual feedback delay Δ expect for $\Delta = 200\,\text{ms}$, but also reduces the SD of the precision achieved for different circuits considerably.

significantly from feedback (for example with the approach of (Ijspeert et al., 2003)), especially if this feedback is significantly delayed, we show in Fig. 8 that neurocontrollers built from generic neural microcircuit models are able to generate and control movements for feedbacks with a wide range of delays. In fact, Fig. 8 shows that the smallest deviation between the target end point $\langle x_{dest}, y_{dest} \rangle$ and the actual end point of the tip of the robot arm is not achieved when this delay Δ has a value of 0, but for a range of delays between 50 and 280 ms. In order to make sure that this surprising result is not an artifact of some particular randomly drawn neural microcircuit model or a particular arm movement, it has been tested on each of 10 randomly drawn neural microcircuits with 12 different movements (4 trajectories as shown in figure 4, each created at 3 different speeds, resulting in movement times of 300, 500 and 700 ms). The result of these statistical experiments are reported in Fig. 8 a), b), c). The rightmost point on each of the 3 curves shows the performance achieved without any feedback (since for this point the delay of the feedback is as large as the duration of the whole movement). Compared with that, feedback with a suitable delay reduces the imprecision of the movement by at least 50 %. Altogether these data show that the best values for the feedback delay lie in the range of 50 to 280 ms. The upper bound for this interval depends somewhat on the duration of the movement. A possible explanation for the fact that feedbacks with a delay of less than 50 ms are less helpful is that in this case the current target circuit output is very similar to the currently arriving feedback, and hence it is more difficult for the circuit to learn the map from current feedback to current target output in a noise-robust fashion. In addition a delayed feedback complements the inherent temporal integration property of

the neural microcircuit model (see (Maass et al., 2004b)), and therefore tends to enlarge the time constant for the fading of memory in the closed loop system. Hence these neurocontrollers perform best for a range of feedback delays that contain typical values of actual delays for proprioceptive and visual feedback measured in a variety of species (e.g. 120 ms for proprioceptive feedback and 200 ms for visual feedback is reported in (van Beers et al., 2002)).

In another computer experiment we have examined the potential benefit of using estimated feedback for the neurocontroller under consideration. Estimation of feedback is very easy for such neural architecture, since the generic neural microcircuit model that generates (via suitable readouts) the movement commands has not been specialized in any way for this movement generation task, and can simultaneously be used as information reservoir for estimating feedbacks. More precisely, 2 additional readouts were added and trained to estimate at any time t the values of the joint angles θ_1 and θ_2 at time $t - 200$ ms, i.e., 200 ms earlier. These delayed values were chosen as targets for these 2 additional readouts during training, since the previously reported results (see in particular Fig. 8b)) show that a feedback of the actual values of θ_1 and θ_2 with a delay of 200 ms is quite beneficial for the precision of the movement that is generated. After training, the weights of these 2 additional readouts were frozen (like for the first 2 readouts which produce the movement commands).[6] The outputs of these 2 additional readouts were also fed back into the circuit

[6]Since neither the training of the readouts for movement commands nor the training of the readouts for retrodiction of sensory feedback changes the neural circuit itself, it does not matter whether these readouts are trained sequentially or in parallel. In our

(without delay), see Fig. 9. Compared with the architecture shown in Fig. 2 the neural circuit receives now 2 additional time-varying inputs. These were fed into the circuit in the same way as the other 6 inputs (described in section 2). Thus 2 additional arrays consisting of 50 neurons each were used for a population coding of these time-varying input variables, and 2 "columns" consisting of 100 neurons each were added of the neural circuit that received the outputs of these 2 additional input-arrays.

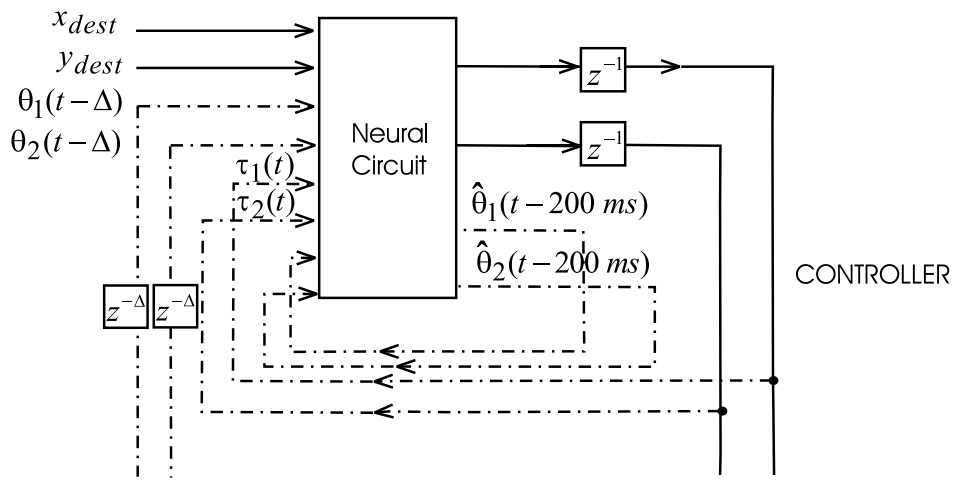

Figure 9: Information flow for the case of autonomously generated estimates $\hat{\theta}(t - 200 \text{ ms})$ of delayed feedback $\theta(t - 200 \text{ ms})$. Rest of the circuit as in Fig. 2.

The top solid line in Fig. 8 d) shows the result (computed in the same way as in the experiments both types of readouts were trained simultaneously, while the target values of both $\theta_1(t - \Delta), \theta_2(t - \Delta)$ and $\theta_1(t - 200), \theta_2(t - 200)$ were given to the circuits as additional inputs during training (where Δ is the assumed actual feedback delay plotted on the x-axis in Fig. 8 d)).

other panels of Fig. 8 for the case when the values of the estimates of $\theta_1(t - 200$ ms) and $\theta_2(t - 200$ ms) produced by the 2 additional readouts were not fed back into the circuit. The bottom dashed line shows the result when these estimates were available to the circuit via feedback. Although these additional feedbacks do not provide any new information to the circuit, but only collect and redistribute information within the neural circuit; this additional feedback significantly improved the performance of the neurocontroller for all values of the actual delay Δ of feedback about the values of θ_1 and θ_2 (except for $\Delta = 200$ ms). The value on the rightmost point of the lower curve for $\Delta = 500$ ms shows the improvement achieved by using estimated sensory feedback in case when no feedback arrives at all, since the total movements lasted for 500 ms. Altogether the use of internally estimated feedback improved the precision of the movement by almost 50 % for most values of the delay of the actual feedback.

7 Application to a biological model for arm control

Whereas in the preceding section we have focused on a model for a robot arm as a standard example for a highly nonlinear control task, we will demonstrate in this section that the same paradigm for movement generation can also be applied to a well known model for cortical control of arm movements in primates (Todorov, 2000; Todorov, 2003). This model proposes a direct relationship between the firing rate c_j of individual neuron j in primary motor cortex M1 (relative to some baseline firing rate C) and the kinematics (in cartesian coordinates) and endpoint force f_{ext} of the hand, which is viewed here simply

as the tip of a 2-joint arm:

$$c_j(t-d) = \frac{\mathbf{u}_j^T}{2}\left(F^{-1}\mathbf{f}_{ext}(t) + m\ddot{\mathbf{x}}(t) + k\mathbf{x}(t)\right) + b\lfloor \mathbf{u}_j^T\dot{\mathbf{x}}(t)\rfloor \quad . \tag{2}$$

The vector $\mathbf{u}_j$ denotes the direction in which the end point force is generated due to activation of muscles by neuron j (assuming cosine tuning of neurons). In our simulations we simply took 4 unit vectors $\mathbf{u}_j$ pointing up, down, left, right. $\mathbf{f}_{ext}(t)$ is the endpoint force that the hand applies against external objects. $\mathbf{x}, \dot{\mathbf{x}}$, and $\ddot{\mathbf{x}}$ are the position, velocity and acceleration of the hand respectively (we usually write $\langle x, y\rangle$ for the hand position $\mathbf{x}$ in a 2-dimensional space).

Although the precise relationship between the activity of neurons in motor cortex and the activation of individual muscles is extremely complicated and highly nonlinear, a derivation given in in (Todorov, 2000) suggested that equation 2 provides a quite good (almost linear) local approximation to multijoint kinematics over a small workspace. As a consequence we have applied this model only for arm movements when the hand moves on the boundaries of a 28.28 cm $\times$ 28,28 cm square.

Since we are only concerned with the movement of the hand in its workspace, and do not require the hand to exert an endpoint force on external world objects, $\mathbf{f}_{ext}(t)$ can be set to 0.

For the sake of simplicity we have also set the transmission delay d from cortex to muscles to 0 (but our model would work just as well for other values of d). This simplifies the model to:

$$c_j(t) = \frac{\mathbf{u}_j^T}{2}\left(m\ddot{\mathbf{x}}(t) + k\mathbf{x}(t)\right) + b\lfloor \mathbf{u}_j^T \dot{\mathbf{x}}(t) \rfloor \tag{3}$$

In our computer experiments we applied this model with the parameter values $b = 10$ Ns/m, $k = 50$ N/m, and $m = 1$kg suggested in (Todorov, 2000).

In order to produce a paradigm for arm control by cortical circuits we took a generic cortical microcircuit model consisting of 800 neurons as described in section 2. 4 readouts that received inputs from all neurons in this microcircuit model were trained by linear regression to assume the role of these 4 neurons in motor cortex that directly control arm muscles resulting in hand movements according to equation 3.[7] We trained these readout neurons to produce 4 different hand movements along the edges of a 28.28 cm $\times$ 28.28 cm square whose diagonals were parallel to the x- and y-axis respectively.

The inputs to the neural microcircuit were the coordinates $\langle x_{dest}, y_{dest} \rangle$ of the desired target end point of the hand, efferent copies of the outputs $c_1(t), \ldots, c_4(t)$ of motor neurons $1, \ldots, 4$, and feedback $x(t-200)$ ms, $y(t-200)$ ms about preceding hand positions with a delay of 200 ms that is biologically realistic for visual feedback into motor cortex. The

[7] Target trajectories of the endpoint of the arm were generated for training as described in footnote 4. Target outputs $c_j(t)$ for the readouts were generated from these trajectories by equation 3.

values of these 8 inputs were fed into the 800 neuron microcircuit model in the same way as for the 8 input circuit discussed at the end of the preceding section. The results for this experiment are shown in 10. The average deviation over 40 runs of the tip of the arm from the desired end-point was 0.13 cm with a SD of 7.2295×10^{-2} cm.

It is interesting to note that the generic neural microcircuit can also learn to generate movements for this quite different arm model. Another point of interest is that the control performance of the generic neural microcircuit is independent of the kind of feedbacks that it is receiving (*c.f.* angles in the earlier model and position coordinates in this model).

8 How to make the movement generation noise-robust

Mathematical results from approximation theory (see the appendix of (Maass et al., 2002) and (Maass and Markram, 2004) for details) imply that a sufficiently large neural microcircuit model (which contains sufficiently diverse dynamic components to satisfy the separation property) can in principle (if combined with suitable static readouts) uniformly approximate any given time-invariant fading memory filter F.

Additional conditions have to be met for successful applications of neural microcircuit models in closed-loop movement generation tasks, such as those considered in this article. First of all, one has to assume that the approximation target for the neural microcircuit, some movement generator F for a plant P, is a time-invariant fading memory filter (if considered in an open loop). But without additional constraints on the plant and/or target movement generator F one cannot guarantee that neural microcircuits L that

uniformly approximate F in an open loop can successfully generate similar movements of the plant P. Assume that F can be uniformly approximated by neural microcircuit models L, i.e., there exists for every $\varepsilon > 0$ some neural microcircuit model L so that $\|(Fu)(t) - (Lu)(t)\| \leq \varepsilon$ for all times t and all input functions $u(\cdot)$ that may enter the movement generator. Note that the feedback f from the plant has to be subsumed by these functions $u(\cdot)$, so that $u(t)$ is in general of the form $u(t) = \langle u_0(t), f(t)\rangle$, where $u_0(t)$ are external movement commands[8] and $f(t)$ is the feedback (both $u_0(t)$ and $f(t)$ are in general multi-dimensional). Assume that such microcircuit model L has been chosen for some extremely small $\varepsilon > 0$. Even if the plant P has the common bounded input bounded output (BIBO) property, it may magnify the differences $\leq \varepsilon$ between outputs from F and outputs from L (which may occur even if F and L receive initially the same input u) and produce for these two cases feedback functions $f_F(s), f_L(s)$ whose difference is fairly large. The difference between the outputs of F and L for these different feedbacks $f_F(s), f_L(s)$ as inputs may become much larger than ε, and hence the outputs of F and L with plant P may eventually diverge in this closed loop. This situation does in fact occur in the case of a 2-joint arm as plant P. Hence the assumption that L approximates F uniformly within ε cannot guarantee that $\|(Fu_F)(t) - (Lu_L)(t)\| \leq \varepsilon$ for all t (where $u_F(t) := \langle u_0(t), f_F(t)\rangle$ and $u_L(t) := \langle u_0(t), f_L(t)\rangle$), since even $\|(Fu_F)(t) - (Fu_L)(t)\|$ may already become much larger than ε for sufficiently large t.

[8]In our experiments $u_0(t)$ was a very simple 2-dimensional function with value $\langle 0, 0\rangle$ for $t < 0$ and value $\langle x_{dest}, y_{dest}\rangle$ for $t \geq 0$. All other external inputs to the circuit were only given during training.

This instability problem can be solved by training the readout from the neural circuit L to create an "attractor" around the trajectory generated by F in the noisefree case. This is possible because the current liquid state of the circuit depends not just on the most recent feedback to the circuit, but also on the preceding stream of feedbacks (therefore the liquid state also contains information about which particular part of the movement has to be currently carried out) as well as on the target end-position $\langle x_{dest}, y_{dest} \rangle$. If one trains the readout from circuit L to ensure that $\|(Lu_F)(t) - (Lu_L)(t)\|$ stays small when $u_F(\cdot)$ and $u_L(\cdot)$ did not differ too much at preceding time steps, one can bound $\|(Fu_F)(t) - (Lu_L)(t)\|$ by $\|(Fu_F)(t) - (Lu_F)(t)\| + \|(Lu_F)(t) - (Lu_L)(t)\| \leq \varepsilon + \|(Lu_F)(t) - (Lu_L)(t)\|$ and thereby avoid divergence of the trajectories caused by F and L in the closed-loop system.

This makes clear why it was necessary to train the readouts of the neural microcircuit models L to produce the desired trajectory not just for the ideal feedback $u_F(t)$ but also for noisy variations of $u_F(t) = \langle u_0(t), f_F(t) \rangle$ that represent possible functions $u_L(t)$ that arise if the approximating circuit L is used in the closed loop.

9 Discussion

Whereas traditional models for neural computation had focused on constructions of neural implementations of Turing machines or other offline computational models, more recent results have demonstrated that biologically more realistic neural microcircuit models consisting of spiking neurons and dynamic synapses are well-suited for real-time com-

putational tasks ((Buonomano and Merzenich, 1995), (Maass et al., 2002), (Maass et al., 2004b), (Natschläger and Maass, 2004)). Previously only sensory processing tasks such as speech recognition or visual movement analysis ((Buonomano and Merzenich, 1995), (Maass et al., 2002), (Legenstein et al., 2003)) were considered in this context as benchmark tests for real-time computing. In this article we have applied such generic neural microcircuit models for the first time in a biologically more realistic closed loop setting, where the output of the neural microcircuit model directly influences its future inputs.

Obviously closed loop applications of neural microcircuit models provide a harder computational challenge than open loop sensory processing, since small imprecisions in their output are likely to be amplified by the plant to yield even larger deviations in the feedback, which is likely to increase even further the imprecision of subsequent movement commands. This problem can be solved by teaching the readout from the neural microcircuit during training to ignore smaller recent deviations reported by feedback, thereby making the target trajectory of output torques an attractor in the resulting closed-loop dynamical system. After training, the learned reaching movements are generated completely autonomously by the neural circuit once it is given the target end position of the tip of the robot arm as (static) input.

We have demonstrated that the capability of the neural circuit to generate reaching movements generalizes to novel target end positions of the tip of the arm that lie between those which occured during training (see Fig. 7). The velocity profile for these autonomously generated new reaching movements exhibits a bell-shaped velocity profile, like for the previously taught movements. We propose to view the basic arm movements that are

generated in this way as possible implementations of muscle synergies, i.e. of rather stereotypical movement templates (d'Avella et al., 2003). In this interpretation, the learning of a larger variety of arm movements requires superposition of time-shifted versions of several different basic movement templates of the type as are considered in this article. Such learning on a higher level is a topic of currrent research.

Surprisingly the performance of the neural microcircuit model for generating movements not only deteriorates if the (simulated) proprioceptive feedback is delayed by more than 280 ms, or if no feedback is given at all, but also if this feedback arrives *without* any delay. Our computer simulations suggest that the best performance of such neurocontrollers is achieved if the feedback arrives with a biologically realistic delay in the range of 50 to 280 ms. If the delay assumes other values, or is missing altogether, a significant improvement in the precision of the generated reaching movements can be achieved if additional readouts from the same neural microcircuit models that generate the movements are taught to estimate the values of the feedback with an optimal delay of 200 ms, and if the results of these internally generated feedback estimates are provided as additional inputs to the circuit (see Fig. 8 d).

Apart from these effects resulting from the interaction of the inherent circuit dynamics with the dynamics of externally or internally generated feedbacks, also the spatial organization of information streams in the simulated neural microcircuit plays a significant role. The capability of such a circuit to generate movements is quite bad if information about slowly varying input variables (such as externally or internally generated feedback) is provided to the circuit in the form of a firing rate of a single neuron (not shown), rather

than through population coding (see description in section 2) as implemented for the experiments reported in this article.

Another interesting point to be noted is that our model for motor control can successfully learn to control the arm movement irrespective of the model that is used to describe the dynamics of the arm-movement and the types of feedbacks that the circuit is receiving. One of the two arm models that was tested (see section 7) is a model for cortical control of muscle activation. Hence our model also provides a new hypothesis for the computational function of neural circuits in the motor cortex.

Altogether the results presented in this article may be viewed as a first step towards an exploration of the role of the "embodiment of motion generation circuitry", i.e., of concrete spatial neural circuits and their inherent temporal dynamics, in motor control. This complements the already existing work on the relevance of the embodiment of actuators to motor control (Pfeifer, 2002).

References

Buonomano, D. V. and Merzenich, M. M. (1995). Temporal information transformed into a spatial code by a neural network with realistic properties. *Science*, 267:1028–1030.

Craig, J. J. (1955). *Introduction to Robotics: Mechanics and Control*. Addison-Wesley, Reading, (MA), 2nd edition.

d'Avella, A., Saltiel, P., and Bizzi, E. (2003). Combination of muscle synergies in the

construction of a natural motor behavior. *Nature Neuroscience*, 6(3):300–308.

Flash, T. and Hogan, N. (1965). The coordination of arm movements: An experimentally confirmed mathematical model. *The Journal of Neuroscience*, 5(7):1699–1703.

Gupta, A., Wang, Y., and Markram, H. (2000). Organizing principles for a diversity of GABAergic interneurons and synapses in the neocortex. *Science*, 287:273–278.

Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2003). Learning attractor landscapes for learning motor primitives. In Becker, S., Thrun, S., and Obermayer, K., editors, *Proc. of NIPS 2002, Advances in Neural Information Processing Systems*, volume 15, pages 1547–1554. MIT Press.

Jäger, H. (2002). Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the ”echo state network” approach. GMD Report 159, German National Research Center for Information Technology.

Jäger, H. and Haas, H. (2004). Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science*, 304:78–80.

Joshi, P. and Maass, W. (2004). Movement generation and control with generic neural microcircuits. *Proc. of the First International Workshop on Biologically Inspired Approaches to Advanced Information Technology (Bio-Adit 2004)*, pages 16–31. Online available as #151 from http://www.igi.tugraz.at/maass/publications.html.

Legenstein, R. A., Markram, H., and Maass, W. (2003). Input prediction and autonomous movement analysis in recurrent circuits of spiking neurons. *Re-*

views in the Neurosciences (Special Issue on Neuroinformatics of Neural and Artificial Computation), 14(1–2):5–19. Online available as #140 from http://www.igi.tugraz.at/maass/publications.html.

Maass, W., Legenstein, R. A., and Bertschinger, N. (2004a). Methods for estimating the computational power and generalization capability of neural microcircuits. *submitted for publication.* Online available as #160 from http://www.igi.tugraz.at/maass/publications.html.

Maass, W. and Markram, H. (2004). On the computational power of recurrent circuits of spiking neurons. *Journal of Computer and System Sciences*. in press. Online available as #135 from http://www.igi.tugraz.at/maass/publications.html.

Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560. Online available as #130 from http://www.igi.tugraz.at/maass/publications.html.

Maass, W., Natschläger, T., and Markram, H. (2004b). Computational models for generic cortical microcircuits. In Feng, J., editor, *Computational Neuroscience: A Comprehensive Approach*, chapter 18, pages 575–605. Chapman & Hall/CRC, Boca Raton. Online available as #149 from http://www.igi.tugraz.at/maass/publications.html.

Markram, H., Wang, Y., and Tsodyks, M. (1998). Differential signaling via the same axon of neocortical pyramidal neurons. *Proc. Natl. Acad. Sci.*, 95:5323–5328.

Natschläger, T. and Maass, W. (2004). Information dynamics and emergent computation in recurrent circuits of spiking neurons. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Proc. of NIPS 2003, Advances in Neural Information Processing Systems*, volume 16, pages 1255–1262, Cambridge. MIT Press. Online available as #150 from http://www.igi.tugraz.at/maass/publications.html.

Pfeifer, R. (2002). On the role of embodiment in the emergence of cognition: Grey walter's turtles and beyond. In *Proc. of the Workshop "The Legacy of Grey Walter"*, Bristol. See http://www.ifi.unizh.ch/groups/ailab/.

Pouget, A. and Latham, P. E. (2003). Population codes. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, pages 893–897. MIT Press.

Slotine, J. J. E. and Li, W. (1991). *Applied Nonlinear Control*. Prentice Hall, Englewood Cliffs, New Jersey.

Todorov, E. (2000). Direct control of muscle activation in voluntary arm movements: a model. *Nature Neuroscience*, 3(4):391–398.

Todorov, E. (2003). On the role of primary motor cortex in arm movement control. In Latash, M. L. and Levin, M., editors, *Progress in Motor Control III*, pages 125–166. Human Kinetics.

van Beers, R. J., Baraduc, P., and Wolpert, D. M. (2002). Role of uncertainty in motor control. *Phil. Trans. R. Soc. Lond. B*, 357:1137–1145.

Wessberg, J., Stambaugh, C. R., Kralik, J. D., Beck, P. D., Laubach, M., Chapin, J. K., Kim, J., Biggs, S. J., Srinivasan, M. A., and Nicolelis, M. A. (2000). Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. *Nature*, 408(6810):361–365.

Appendix: Specification of Generic Neural Microcircuit Models

Neuron parameters: membrane time constant 30 ms, absolute refractory period 3 ms (excitatory neurons), 2 ms (inhibitory neurons), threshold 15 mV (for a resting membrane potential assumed to be 0), reset voltage drawn uniformly from the interval [13.8, 14.5 mV] for each neuron, constant non-specific background current I_b uniformly drawn from the interval [13.5 nA, 14.5 nA] for each neuron, noise at each time-step I_{noise} drawn from a gaussian distribution with mean 0 and SD of 1nA, input resistance 1 MΩ. For each simulation, the initial conditions of each I&F neuron, i.e., the membrane voltage at time $t = 0$, were drawn randomly (uniform distribution) from the interval [13.5 mV, 14.9 mV]. The probability of a synaptic connection from neuron a to neuron b (as well as that of a synaptic connection from neuron b to neuron a) was defined as $C \cdot \exp(-D^2(a,b)/\lambda^2)$, where $D(a,b)$ is the Euclidean distance between neurons a and b and λ is a parameter which controls both the average number of connections and the average distance between neurons that are synaptically connected (we set $\lambda = 1.2$). Depending on whether the

pre- or postsynaptic neuron were excitatory (E) or inhibitory (I), the value of C was set according to (Gupta et al., 2000) to 0.3 (EE), 0.2 (EI), 0.4 (IE), 0.1 (II).

We modeled the (short term) dynamics of synapses according to the model proposed in (Markram et al., 1998), with the synaptic parameters U (use), D (time constant for depression), F (time constant for facilitation) randomly chosen from Gaussian distributions that model empirically found data for such connections.

Depending on whether a and b were excitatory (E) or inhibitory (I), the mean values of these three parameters (with D,F expressed in seconds, s) were chosen according to (Gupta et al., 2000) to be .5, 1.1, .05 (EE), .05, .125, 1.2 (EI), .25, .7, .02 (IE), .32, .144, .06 (II). The SD of each parameter was chosen to be 50% of its mean. The mean of the scaling parameter A (in nA) was chosen to be 70 (EE), 150 (EI), -47 (IE), -47 (II). In the case of input synapses the parameter A had a value of 70 nA if projecting onto a excitatory neuron and -47 nA if projecting onto an inhibitory neuron. The SD of the A parameter was chosen to be 70% of its mean and was drawn from a gamma distribution. The postsynaptic current was modeled as an exponential decay $\exp(-t/\tau_s)$ with $\tau_s = 3$ ms ($\tau_s = 6$ ms) for excitatory (inhibitory) synapses. The transmission delays between neurons were chosen uniformly to be 1.5 ms (EE), and 0.8 ms for the other connections.

We applied the following input convention. Each input variable is first scaled into the range [0,1]. This range is linearly mapped onto an array of 50 symbolic input neurons. At each time step, one of these 50 neurons, whose number $n(t) \in \{1, \ldots, 50\}$ reflects the current value $i_n(t) \in [0, 1]$ which is the normalized value of input variable $i(t)$ (e.g.

$n(t) = 1$ if $i_n(t) = 0$, $n(t) = 50$ if $i_n(t) = 1$). The neuron $n(t)$ then outputs at time t the value of $i(t)$. In addition the 3 closest neighbors on both sides of neuron $n(t)$ in this linear array get activated at time t by a scaled down amount according to a gaussian function (the neuron number n outputs at time step t the value $i(t) \cdot \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(n-n(t))^2}{2\pi^2}}$, where $\sigma = 0.8$).

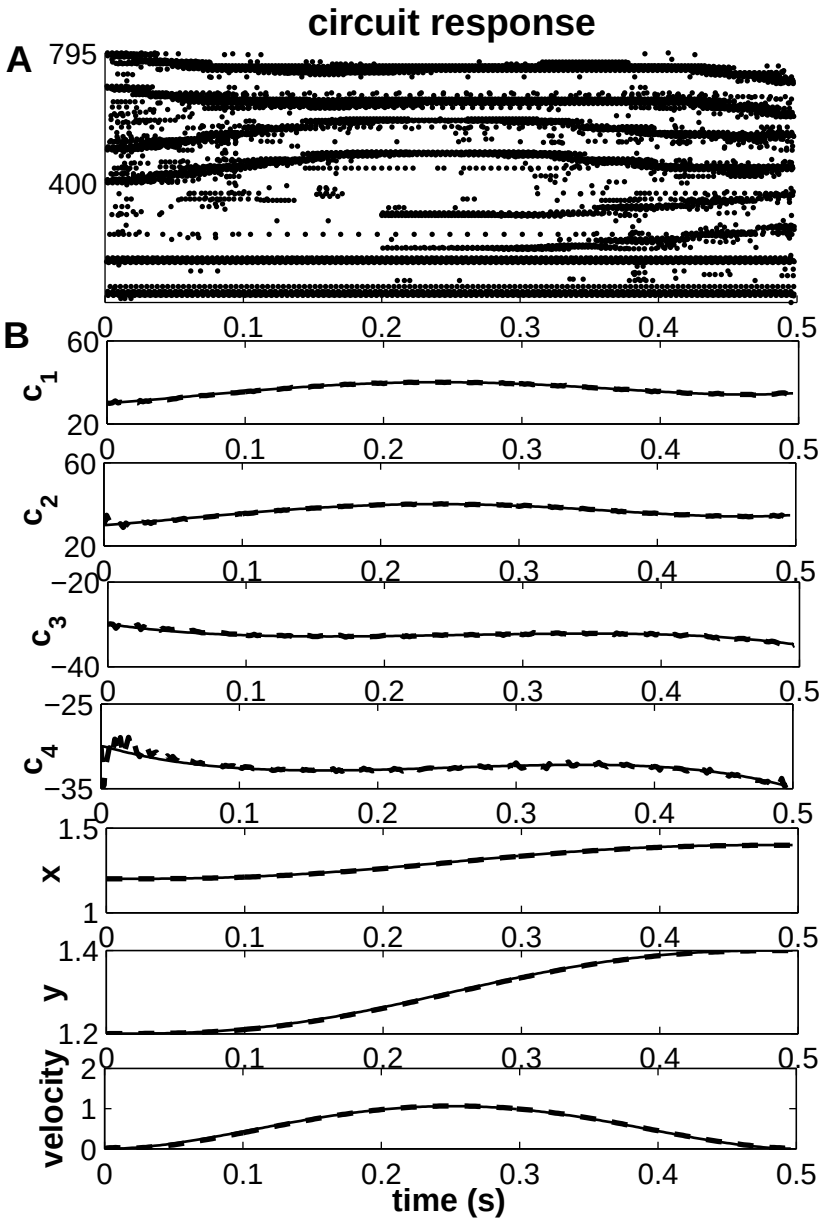

Figure 10: Generation of an arm movement for a biological model for cortical control of muscle activations. **a)** Spike raster analogous to Fig. 6. **b)** Solid lines denote target values and dashed lines show performance of simulated readouts $c_1, \ldots, c_4$ from a simulated microcircuit in motor cortex that receives significantly delayed information about earlier hand positions as feedback (simulating visual feedback to motor cortex). Scales for $c_1, \ldots, c_4$ in N, for x, y in m, for the velocity of the hand in m/s.