SELF-ORGANIZING MULTI-RESOLUTION GRID FOR MOTION PLANNING AND CONTROL

TIBOR FOMIN, TAMÁS ROZGONYI,* CSABA SZEPESVÁRI† and ANDRÁS LŐRINCZ‡,§
Department of Photophysics, Institute of Isotopes, The Hungarian Academy of Sciences, P.O. Box 77, Budapest, Konkoly–Thegeút 29–33, H–1525 Hungary

Received 27 June 1996
Revised 11 November 1996
Accepted 27 January 1997

A fully self-organizing neural network approach to low-dimensional control problems is described. We consider the problem of learning to control an object and solving the path planning problem at the same time. Control is based on the path planning model that follows the gradient of the stationary solution of a diffusion process working in the state space. Previous works are extended by introducing a self-organizing multigrid-like discretizing structure to represent the external world. Diffusion is simulated within a recurrent neural network built on this multigrid system. The novelty of the approach is that the diffusion on the multigrid is fast. Moreover, the diffusion process on the multigrid fits well the requirements of the path planning: it accelerates the diffusion in large free space regions while still keeps the resolution in small bottleneck-like labyrinths along the path. Control is achieved in the usual way: associative learning identifies the inverse dynamics of the system in a direct fashion. To this end there are introduced interneurons between neighboring discretizing units that detect the strength of the steady-state diffusion and forward control commands to the control neurons via modifiable connections. This architecture forms the Multigrid Position-and-Direction-to-Action (MPDA) map. The architecture integrates reactive path planning and continuous motion control. It is also shown that the scheme leads to population coding for the actual command vector.

1. Introduction

Controlling a manipulator can be enormously complex from the point of view of an analytical approach since it requires the sequential computation of the location of the target, the path to be followed to reach the target, the inverse joint kinematics that satisfies the constraints of the path and the obstacles, the inverse joint dynamics and eventually the command series while meeting the demand of changes of the plant's dynamics.

Biological evidence strongly suggests that such a task can be solved with the help of learning. Effort along this route include various inverse system identification methods, such as the direct identification method (Miller, 1987; Kawato *et al.*, 1987; Widrow *et al.*, 1978), the indirect method, that is based on the identification of the forward model (Jordan, 1990; Werbos, 1988; Widrow, 1986) and the feedback-error learning method, when the errors generated by a previously fixed stabilizing feedback controller are used to train the inverse system identification model (Lewis *et al.*, 1993; Miyamoto *et al.*, 1988). Excellent reviews have been published in the literature, see, e.g. (Dean and Wellman, 1991; Miller

*Also at Department of Physics Attila József University, Szeged, Hungary.
†Also at Bolyai Institute of Mathematics, Attila József University, Szeged, Hungary.
‡Also at Department of Adaptive Systems, Attila József University, Szeged, Hungary.
§To whom correspondence should be addressed.
E-mail: lorincz@iserv.iki.kfki.hu

et al., 1990; Narendra and Parthasarathy, 1990; Vemuri, 1993).

The main obstacles of manipulator controlling are:

- the problem of many degrees of freedom,
- the question of adaptivity, i.e. the adjusting of the system to changes of non-repetitive nature,
- the question of learning, i.e. the modifying of the system to respond for changes of repetitive nature.

The problem of many degrees of freedom has no general solution as of yet. Some of the approaches, just like that of the present paper, apply coarse coding discretization techniques that provide a natural interpolation scheme. Nevertheless, a discretization technique is of limited use when the configurations of a many-degrees-of-freedom manipulator are considered, since the full problem requires the discretization of the phase space that leads to a neuron number having the dimension of the phase space in its exponent.

The present approach features the following properties: it is a natural extension of previous path planning approaches that are based on the diffusion equation (Lei, 1990; Connolly and Grupen, 1993, Glasius *et al.*, 1995; Glasius *et al.*, 1996; Marshall and Tarassenko, 1994). These works were extended by Lőrincz *et al.* in a series of publications that provides a self-organizing formulation (Szepesvári *et al.*, 1994; Szepesvári and Lőrincz, 1996; Rozgonyi *et al.*, 1996) to the path planning architecture and provides a direct inverse system identification stage of associative nature for controlling (Fomin *et al.*, 1994; Szepesvári and Lőrincz, 1996a). The approach has the advantage that it provides a natural solution to the problem of adapting versus learning (Szepesvári and Lőrincz, 1996b) while relaxing the demand on the precision of the identification stage at the same time (Szepesvári and Lőrincz, 1996c).

The disadvantage of using the diffusion equation is its long relaxation time. This is a crucial point since the reactive path planning properties of the model rely on the continuous upgrading of the equilibrium state of the diffusion equation. If that upgrading is slow then the use of the full architecture is limited. This limitation will be resolved here by proposing the use of a self-organizing multigrid structure (Rozgonyi *et al.*, 1996) that can shorten the relaxation time considerably.

For a better inspection to the control problem let us formulate the mathematical framework: The neural *spreading activation* (SA) method we consider can be viewed as the discretization of the following diffusion like differential equation:

$$\dot{\sigma} = \Delta\sigma + I\,, \tag{1}$$

where $\sigma = \sigma(\mathbf{q}, t)$ is the activity, $I = I(\mathbf{q})$ is the external flow, $\mathbf{q} \in \mathbb{R}^n$ is the state vector of the plant and $\Delta = \partial^2/\partial x_1^2 + \cdots + \partial^2/\partial x_n^2$ is the Laplacean operator. Let us denote the stationary solution of Eq. (1) by $\sigma^* = \sigma^*(\mathbf{q})$. If the source (sink) activity corresponds to the start (target) state then the path planning problem is solved by following the gradient of the stationary solution. The desired equation of motion of the plant thus may be given as:

$$\dot{\mathbf{q}} = \kappa\nabla\sigma^*(\mathbf{q})\,, \tag{2}$$

where κ is a positive constant. In the present framework the Neumann boundary condition that requires the normal component of the flow with respect to the boundary of the free zone (F) to be zero:

$$\left.\frac{\partial\sigma}{\partial\mathbf{n}}\right|_{\partial F} = 0\,, \tag{3}$$

arises in a natural fashion.

Now, assume that the plant's dynamics is given by the following equation (Isidori, 1989):

$$\dot{\mathbf{q}} = \mathbf{b}(\mathbf{q}) + \mathbf{A}(\mathbf{q})\,\mathbf{u}\,, \tag{4}$$

where $\dot{\mathbf{q}}$ is the time derivative of $\mathbf{q}$, the state vector of the plant, $\mathbf{u} \in \mathbb{R}^m$ is the control signal, $\mathbf{b}(\mathbf{q}) \in \mathbb{R}^n$, and $\mathbf{A}(\mathbf{q}) \in \mathbb{R}^{n\times m}$. We assume that the domain (denoted by D) of the state variable $\mathbf{q}$ is compact and is simply connected; that $n \leq m$, and for each $\mathbf{q} \in D$ the rank of matrix $\mathbf{A}(\mathbf{q})$ is equal to n; that is, the matrix is non-singular. As a consequence the plant is strongly controllable. In this case the inequality $n < m$ means that there are more independent actuators than state vector components, i.e. the control problem is redundant. Another kind of redundancy, or ill-posedness occurs when $n > m$ in which case even $\mathbf{A}^{-1}$ is non-unique. Further, we assume that both of the matrix fields, $\mathbf{A}(\mathbf{q})$ and $\mathbf{A}^{-1}(\mathbf{q})$ are differentiable with respect to $\mathbf{q}$.

Let $\mathbf{v} = \mathbf{v}(\mathbf{q})$ be a fixed n dimensional vector field over D. The *speed field tracking task* is to find the

static state feedback control $\mathbf{u} = \mathbf{u}(\mathbf{q})$ that solves the equation

$$\mathbf{v}(\mathbf{q}) = \mathbf{b}(\mathbf{q}) + \mathbf{A}(\mathbf{q})\mathbf{u}(\mathbf{q})\,. \tag{5}$$

Note, that Eq. (2) defines a speed field tracking task. Speed fields must be carefully designed if $\mathbf{A}(\mathbf{q})$ is singular [as in the case of a robot arm (Hwang and Ahuja, 1992)].

Given the plant's dynamics by Eq. (4) the *inverse dynamics* of the plant may be written as follows:

$$\mathbf{P}(\mathbf{q},\,\dot{\mathbf{q}}) = \mathbf{A}^{-1}(\mathbf{q})(\dot{\mathbf{q}} - \mathbf{b}(\mathbf{q})) + (\mathbf{E} - \mathbf{A}^{-1}(\mathbf{q})\mathbf{A}(\mathbf{q}))\mathbf{y}(\mathbf{q},\,t)\,, \tag{6}$$

where $\mathbf{E}$ is the unit matrix and $\mathbf{y} = \mathbf{y}(\mathbf{q}, t)$ is an arbitrary function and $\mathbf{A}^{-1}(\mathbf{q})$ is the generalized inverse of $\mathbf{A}(\mathbf{q})$. Of course, the control signal $\mathbf{u}(\mathbf{q}) = \mathbf{P}(\mathbf{q}, \mathbf{v}(\mathbf{q}))$ solves the speed field tracking control task. Here we treat the learning of the inverse dynamics when the state space is discretized by means of a self-organizing multigrid.

The paper is organized as follows: first the proposed medium for the diffusion equation, the self-organizing multigrid is described. Then the diffusion process on the multigrid system is discussed and computer simulations are shown for illustration. Section 4 introduces the concept of interneurons, that serve as the bases of the inverse system identification by creating position-and-direction discretization and thus allowing to form a Multigrid Position-and-Direction-to-Action (MPDA) map. Control experiments with the MPDA map are also presented in this section. In the discussion it is shown that the model leads to population coding being the subject of recent interest in both biological experiments (Georgopoulos *et al.*, 1982; Georgopoulos *et al.*, 1988) and adaptive parallel algorithms (Pougets and Sejnowski, 1995). Conclusions are drawn in Sec. 6. Details of motion control computations and direct associative identification of the inverse dynamics are presented in the Appendix.

2. Multigrids

A multigrid is a hierarchical set of discretization-, or gridpoints with levels differing in their resolution. The multigrid may also include the neighboring relations, i.e. a graph structure that connects gridpoints that are nearest neighbors, or are close to each other. Both of the definitions acquire a meaning if the space to be discretized and the closeness relationship are defined. In the case of a pixel discretized 2D image the multigrid refers to the discretization of the 2D *space* and the *size*, (i.e. the resolution level) thus the multigrid discretizes a space of 3D. Neighbors then can be defined in this 3D space.

2.1. *Multiresolution elements*

The network suitable for forming multiresolution elements can be described as follows (Rozgonyi *et al.*, 1994; Rozgonyi *et al.*, 1996). Assume an n dimensional bounded Euclidean space $\mathbb{R}^n$ as the *external* space and its pixel discretized image as the *input* space. Let N_p denote the number of discretizing pixels. Thus the input space is the subset of $\mathbb{R}^{N_p}$. Each point of the input space corresponds to a set of configurations in the external space. We consider that an object in the external space is completely characterized by its position and size and is also represented by its N_p dimensional input-intensity distribution vector $\underline{x} = (x_1,\, x_2,\, \ldots,\, x_{N_p})$. (Vectors of the N_p dimensional space will be distinguished by underlining.) Assume a network consisting of N_n neurons each of which receives the input vector $\underline{x}$ via input weights. Let f_{il} $(i = 1, \ldots, N_n;\, l = 1, \ldots, N_p)$ denote the weight connecting the ith neuron with the lth component of the input vector. Neurons and their weights will be sometimes called weight vectors, or filters.

The development of the multigrid consists of two separate problems:

- that of finding the filters that are able to distinguish between input patterns of different sizes,
- that of finding a self-organizing learning rule that can lead to the development of such filters when the network is input by objects of different sizes.

Consider the training set of *local extended objects* modeled by Gaussian "excitation profiles" parametrized by $\omega_{\text{inp}} \in \mathbb{R}$ ("size" of the input pattern) and $\mathbf{r}_{\text{inp}} \in \mathbb{R}^n$ (position of the input pattern). A distribution of ω_{inp} and $\mathbf{r}_{\text{inp}}$ induces a distribution of input pattern vectors. Each input pattern vector $\underline{x}$ is generated by the expression:

$$x_l = N(\omega_{\text{inp}})e^{\left(-\frac{(\mathbf{r}_l - \mathbf{r}_{\text{inp}})^2}{\omega_{\text{inp}}{}^2}\right)}\,, \tag{7}$$

where $N(\omega_{\text{inp}})$ is a normalization factor ensuring $\sum_l {x_l}^2 = 1$, $\mathbf{r}_{\text{inp}}$ points to the center of excitation, $\mathbf{r}_l \in \mathbb{R}^n$ is the position vector of pixel l.

It was found (Rozgonyi *et al.*, 1994) that this training set has a suitable learning rule to form multigrids. This learning rule may be expressed as follows:

$$\Delta f_{il} = \varepsilon_f \delta_{is(\underline{x})}(-J_i(\underline{x}) f_{il} + x_l)\,, \qquad (8)$$

where Δf_{il} is the change of the lth component of the weight vector belonging to the ith neuron, ε_f is a positive learning rate, $J_i(\underline{x}) = \sum_l f_{il} x_l$ is the input activity of the ith neuron, and δ_{is} is the Kronecker-delta ($\delta_{is} = 1$ if $i = s$, otherwise $\delta_{is} = 0$), with indices determined by the winner-takes-all (WTA) mechanism, i.e. $s(\underline{x})$ is the index of the neuron of maximal input activity for input $\underline{x}$. Learning rule (8) — together with other learning schemes — was first investigated by Kohonen in his book (Kohonen, 1984).

This learning rule has been studied in detail both theoretically and numerically (Rozgonyi *et al.*, 1996). It corresponds to a stochastic gradient approximation and thus converges with a suitable choice of the learning rate time series. It minimizes the cost function $E(\mathbf{f}_1, \ldots, \mathbf{f}_{N_n})$ defined as $E = \sum_i (\sum_l {f_{il}}^2 - 1)^2$. That is, the weight vectors become normalized at the end of the training. As it has been shown (Rozgonyi *et al.*, 1996), the normalization of the weight vectors, the inner product expression for the input activities, and the WTA procedure, do favour the neural unit that best matches the input vector apart from a scalar factor. That property leads to the formation of filters of different sizes and positions in the natural ordering of pixel, i.e. to the formation of a multiresolution system, possibly a multigrid depending on the input set utilized. Computational simulations illustrate that under suitable conditions a multigrid is formed.

For the illustration of the advantages of multigrids in control tasks 400 filters in a two dimensional space of pixel size 18×18 were formed. The results are shown in Fig. 1. The filter positions, $(\mathbf{c}_i)$ and radii, (R_i) are respectively defined as

$$\mathbf{c}_i = \frac{\sum_l f_{il} \mathbf{r}_l}{\sum_l f_{il}}\,, \quad R_i = \sqrt{\frac{\sum_l f_{il} (\mathbf{r}_l - \mathbf{c}_i)^2}{\sum_l f_{il}}}\,. \qquad (9)$$

(Filter sizes of the figures are scaled down by a factor of 10 for better visualization.) The magnitude ordered filter radii are shown in the upper left subfigure of Fig. 1. Filters arrange themselves in size groups. The structure is fairly irregular. More regular systems with better separated subgrids can be developed with torus-like boundary conditions (Rozgonyi *et al.*, 1996). The control utilizing spreading activation methods (see later) does not allow, however, a straightforward development of the multigrid on a torus. Boundary effects then become strong and errors can easily freeze into the structure; the structure becomes more irregular. Nevertheless two smaller filter clusters have emerged together with a broad distribution of largers filters. It has been argued that the larger the size distribution the less stable the multigrid structure will be (Rozgonyi *et al.*, 1996). These figures indicate that in the case of a self-organizing discretization some — or even extensive — structural errors may freeze in and there is a need for self-organizing control methods that can compensate for those structural errors.

2.2. *Connections between neighboring multiresolution elements*

In order to allow a diffusion to spread, the set of discretization points should be extended by *neighboring connections*. Neighboring connections should connect filters of similar (close) positions and of similar sizes. Theoretical works on developing suitable neighboring connections have followed two routes, the one that used a WTA competition between the connections themselves (Martinetz, 1993; Martinetz and Schulten, 1994) and the other one that used leaky learning for the connections and assumed local extended objects (Szepesvári and Lőrincz, 1993; Szepesvári *et al.*, 1994; Szepesvári and Lőrincz, 1996). Here it suffices to use the method of Martinetz and Schulten (Martinetz, 1993; Martinetz and Schulten, 1994) which they termed competitive Hebbian learning (CHL). Below, the concepts developed by Martinetz and Schulten are shortly reviewed.

In the case of a WTA network the neurons divide the input space into winning domains. Let V_i denote the winning domain in $\mathbb{R}^{N_p}$ belonging to neuron i and let the V_{ij} sets be given by all the $\underline{x} \in \mathbb{R}^{N_p}$ for which the $J_i(\underline{x})$ and $J_j(\underline{x})$ are the two highest input activities. Let $M \subset \mathbb{R}^{N_p}$ denote the set of $\underline{x}$ vectors

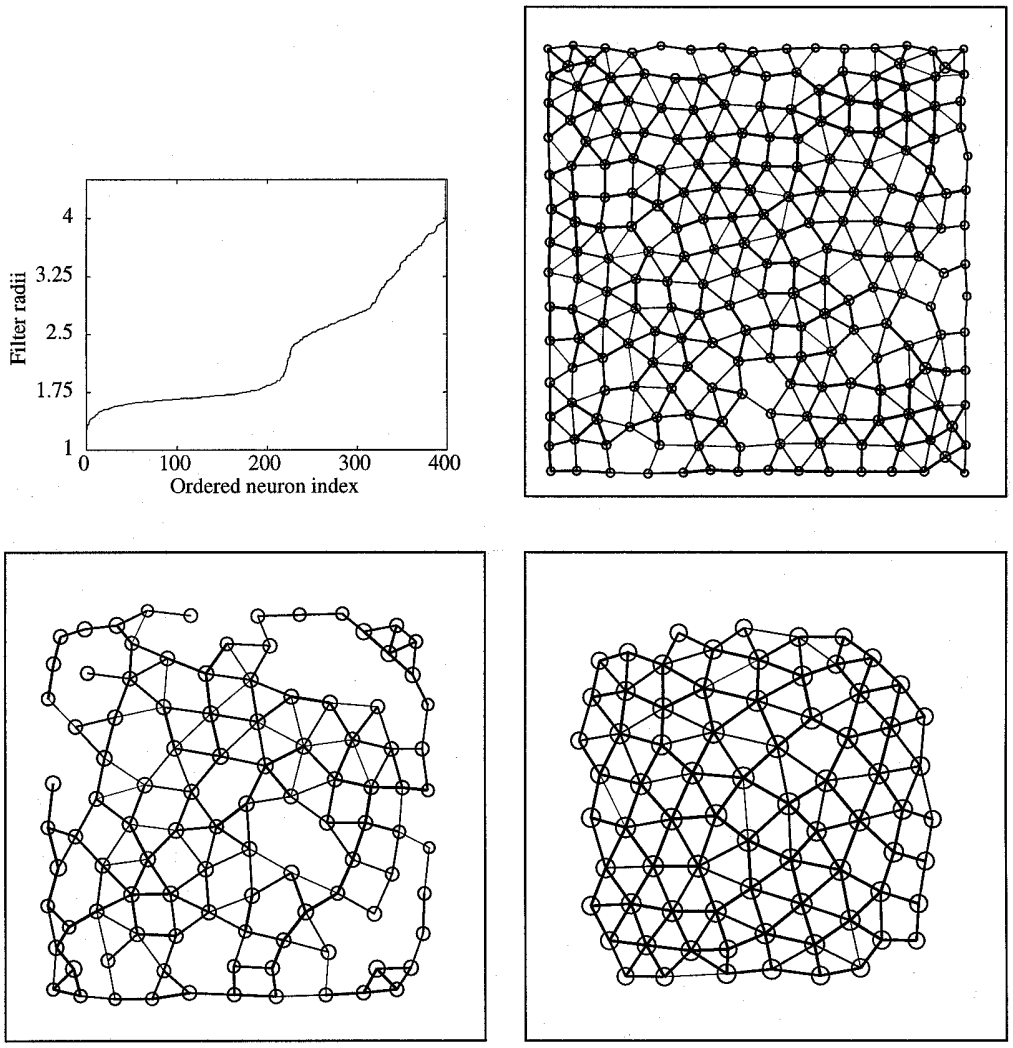

Fig. 1. *Upper left*: size ordered filters. *Upper right, lower left and lower right* subfigures show neural positions, the relative size of receptive fields and within group interneural connections for groups of neural radii between 0.0 and 2.1, 2.1 and 3.00, and above 3.00, respectively. The filter sizes are scaled down in the figure by a factor of 10 for better visualization.

that represent input objects, and let $V_i^{(M)}$ denote the common part of V_i and M. Furthermore let $V_{ij}{}^{(M)}$ denote the common part of V_{ij} and M; that is, $V_{ij}{}^{(M)} = V_{ij} \cap M$. $V_i^{(M)}$ and $V_{ij}{}^{(M)}$ are the so-called masked Voronoi polyhedra of first and second order, respectively. The neighborhood relations of neurons stands for the neighborhood relations of their winning domains:

Definition

Neuron i and neuron j are neighbors of each other if there exists $\underline{x} \in V_{ij}{}^{(M)}$.

The neighborhood relations of neurons are represented by the interneural connections. Let us denote the connection strength between neuron i and neuron k by w_{ik} (i, $k = 1, \ldots, N_n$). Taking $w_{ik} = 0\ \forall i, k$ as the initial conditions, competitive Hebbian learning can lead in an unsupervised way to w_{ik}-s, which differ from zero if and only if neuron i and neuron k are neighbors. This learning procedure was first suggested by (Martinetz and Schulten, 1994) for developing a correct representation of topology of the input space. An analog version of this learning rule, is the following:

$$\Delta w_{ik} = \varepsilon_w \delta_V^{(ik)}(-w_{ik} + J_i(\underline{x})J_k(\underline{x}))\,, \qquad (10)$$

where $\delta_V^{(ik)} = 1$ only if $\underline{x} \in V_{ik}{}^{(M)}$, otherwise $\delta_V^{(ik)} = 0$. The restriction of learning for the case when $\delta_V^{(ik)} = 1$ means a strong (WTA) competition between connections. ε_w is a positive learning rate. Interneural connection strength matrix will be symmetric.

The connection structure of the network trained on two dimensional patterns is shown only within each layer (see Fig. 1). The concepts of developing neighboring connections can be extended to the case of multigrids. It may be shown for multigrids that the CHL rule is suitable. More details may be found in (Rozgonyi *et al.*, 1996).

3. Diffusion on the Multigrid

In order to set up the path planning problem with start and target activities an external recognition system — not modelled here — is needed. It is assumed that this recognition system can identify and segment the object to be controlled and can input the segmented part of the external world to the self-organizing network. This input gives rise to the *start activities* and the respective neural activities shall be denoted by $\hat{s}_i$ ($\hat{s}_i = \sum_l f_{il}(\underline{x}_s)_l$, where $\underline{x}_s$ is the segmented start vector). Index i runs through the discretizing neurons. It is assumed that the target area can be identified and inputted to the network in similar fashion giving rise to the *target activities*, $\hat{t}_i$ ($\hat{t}_i = \sum_l f_{il}(\underline{x}_t)_l$, where $\underline{x}_t$ is the segmented target vector and $i = 1, \ldots, N_n$). Moreover, it is also the task of the recognition system to identify (segment) the obstacle ($\underline{x}_o$) and to input that to the network giving rise to the obstacle activities $\hat{o}_i$ ($\hat{o}_i = \sum_l f_{il}(\underline{x}_o)_l$). If the obstacle activity of a given neuron is above a predefined threshold then that node is thought to represent the obstacle region and is excluded from the free space region. Now the diffusion process [Eq. (1)] can be simulated on the self-organizing multigrid according to the following set of equations:

$$\dot{\sigma}_i = I_i + \sum_{k \in N_i \cap F} w_{ik}(\sigma_k - \sigma_i)\,, \quad i \in F\,, \qquad (11)$$

where F is the set of active neurons (corresponding to the free space), N_i denotes the set of neighboring grid points of grid point i and σ_i is the discretized version of the activity σ at position $\mathbf{c}_i$. The external signal is the summed contribution of start and target activities, $I_i = s_i - t_i$ where $s_i = \hat{s}_i/S$, $t_i = \hat{t}_i/T$ where $S = \sum_{i \in F} \hat{s}_i$ and $T = \sum_{i \in F} \hat{t}_i$ are the normalizing factors. It is easy to see that Eq. (11) with w_{ik} connections equal to the inverse of the nearest neighbor distance and discretization forming a homogeneous orthogonal lattice structure corresponds to the discrete approximation of a diffusion equation.

3.1. *Diffusion experiments*

We have compared relaxation rates on multigrid versus single grid structures. These experiments were conducted on the self-organizing networks. Initial activities are shown in the upper subfigure of Fig. 2, while the relaxed activities are shown in the lower subfigure of Fig. 2. The time dependences belonging to these figures are shown in Fig. 3. These figures were computed by summing up the absolute

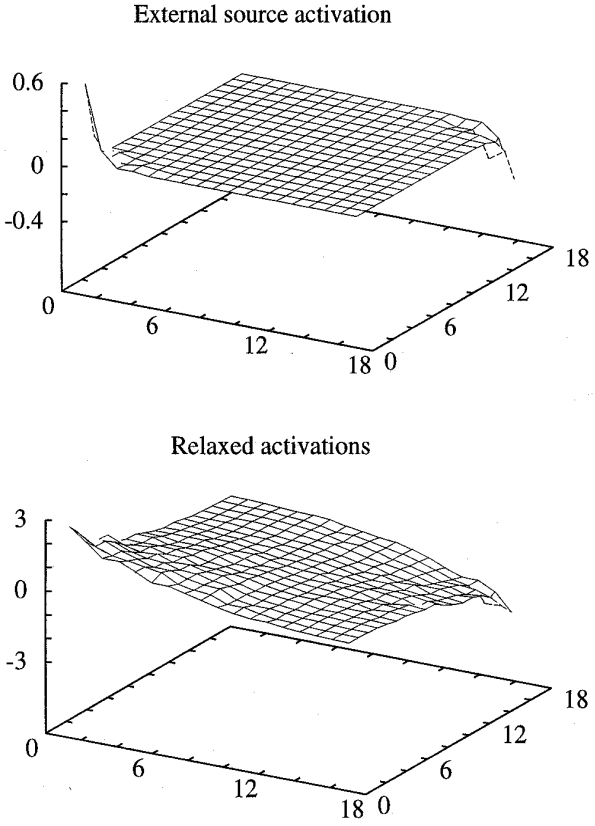

Fig. 2. *Top*: initial start and target activity distribution. *Bottom*: relaxed activity field.

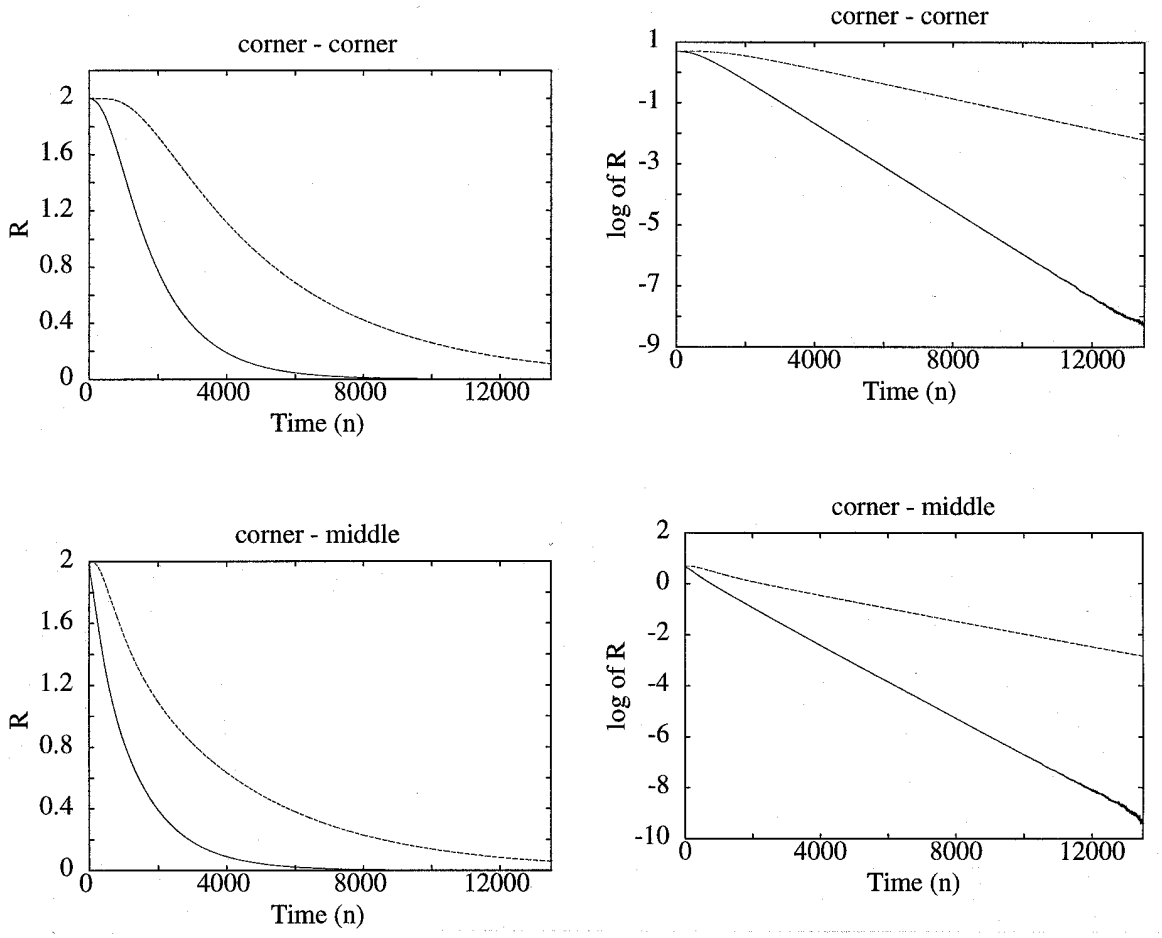

Fig. 3. Relaxation (R) of network activities on linear (*left*) and logarithmic scales (*right*). *Upper* and *lower* rows correspond to target placed to the upper right corner of the input space and start position placed to the lower left corner and to middle of the input space, respectively. Dashed line shows relaxation for the case when only the high resolution group was utilized for activation spreading. Solid line shows relaxation with the full network. Relaxation R is defined as the sum of the absolute values of the changes of spreading activities at each node at time t and computing this sum as a function of time $t = n\Delta t$.

values of the changes of activities at each node at time $t = n\Delta t$:

$$R(n) = \sum_i |\sigma_i(n\Delta t) - \sigma_i((n-1)\Delta t)| \qquad (12)$$

and computing this sum as a function of n [Δt was the discrete time step used in the numerical solution of Eq. (11)]. The left hand side and right hand side figures differ in the vertical scale: the left and right hand side figures are drawn on a linear and on a logarithmic scale, respectively. In all cases the target was positioned in the upper right corner of the two dimensional world. In the case of the top (bottom) figures the start position is in the lower left corner (center) of the free space. The dashed lines represent diffusion on the highest resolution layer, while the solid lines show the results for the diffusion process using all the neurons and all the interneural connections.

It may be seen that the start of relaxation on the high resolution layer alone is very slow, being more pronounced for the case when start and target are in the opposite corners. This point will be discussed later.

4. Controlling the Plant

4.1. *Gradient estimation*

The path planning task involves the design of a route that connects the start region given by the start activities to the target region given by the target activities. The best a control system can do in our case that it follows the gradient of the steady state activity map provided by the diffusion system [Eq. (2)]. Below it is argued that a local approximation utilizing the steady state diffusion flow can estimate the gradient and the conditions of the estimation are given. It is detailed why these conditions are only partially satisfied for our self-organizing structure. The errors thus introduced are compensated by the direct, associative identification of the inverse dynamics (Fomin *et al.*, 1994; Szepesvári and Lőrincz, 1996a) being shortly reviewed in the Appendix. The efficiency of this identification method is illustrated by the computer simulations of the forthcoming sections.

The gradient can be approximated by directional derivatives of the equilibrium activity map. Let us first introduce the concept of geometry vectors. The geometry vector between neuron i and j is the vector that points from the position $\mathbf{c}_i$ of neuron i towards the position $\mathbf{c}_j$ of neuron j: $\mathbf{g}_{ij} = \mathbf{c}_j - \mathbf{c}_i$. Let σ_i^* denote the stationary activity at node i. The gradient of the stationary activity at node i, i.e. the gradient of the steady-state result of the discretized diffusion equation (11) is approximated by

$$\mathbf{d}_i = \sum_{j \in N_i \cap F} I_{ij} \mathbf{e}_{ij} \,, \tag{13}$$

where $\mathbf{e}_{ij} = w_{ij}\mathbf{g}_{ij}$ is a unit vector and I_{ij} is defined as

$$I_{ij} = (\sigma_j^* - \sigma_i^*) w_{ij} \,. \tag{14}$$

Equation (13) is just the first order approximation of the gradient of the steady state activity (limited only by the discretization error) provided that (i) the lattice is a regular lattice and (ii) w_{ij} connection set is equal to $\hat{w}_{ij}$:

$$\hat{w}_{ij} = \frac{1}{||\mathbf{c}_j - \mathbf{c}_i||} \,, \quad j \in N_i \,. \tag{15}$$

where N_i denotes the set of neurons neighboring neuron i. Neither of these conditions are, however, fully satisfied for the self-organizing maps, since there are structural errors in the system and the self-organized connection strengths are not necessarily tuned to the values given by Eq. (15). The approximation of using the w_{ij} values instead of the expression given by Eq. (15) is, however, justified since the receptive fields (roughly) approximate closed-packing within each subgrids and thus are uniformly placed resulting in similar neigboring connection strengths. Nevertheless it is clear that the learning rule of the controlling scheme should compensate for strong structural errors.

The coarse coding technique that we utilize allows further averaging and the gradient of the activity map at the state of the plant is approximated by the sum of the gradients of activity at discretizing points (neurons) weighted by the coarse coded activities of the state of the plant:

$$\mathbf{d} = \sum_{i \in F} s_i \mathbf{d}_i \,. \tag{16}$$

Note that in Eq. (13) it is necessary to restrict the summation for active nodes (elements of F) since activities, and thus I_{ij} values are only defined for active nodes. The I_{ij} values may be interpreted as the activity flow along the neighboring connection between neurons i and j. The architecture utilizes simple linear elements, the *interneurons* that sense these flows, and through their connections to the control units they activate the control units in proportion with the sensed activity flows and the connection strengths — the output of the network that is the control signal (or control vector) $\mathbf{u}$ is given as

$$\mathbf{u} = \sum_i \sum_{j \in N_i \cap F} s_i I_{ij} \mathbf{u}_{ij} \,, \tag{17}$$

where the $\mathbf{u}_{ij}$-s are the individual interneural command vectors (or command connections) associated to the interneural connection from neuron i to neuron j. After learning, $\mathbf{u}_{ij}$ should move the plant from position $\mathbf{c}_i$ to $\mathbf{c}_j$ (see Sec. 7). It is the tuning of the $\mathbf{u}_{ij}$ values and the association made possible by the introduction of the new units, the interneurons, that can compensate for the structural errors. The tuning procedure is detailed in the Appendix. The result of the tuning procedure, i.e. the relation between geometry connections and actions is called the Position-and-Direction-to-Action (PDA) map for a single discretization layer and can be termed as Multigrid PDA (MPDA) map for the case of the

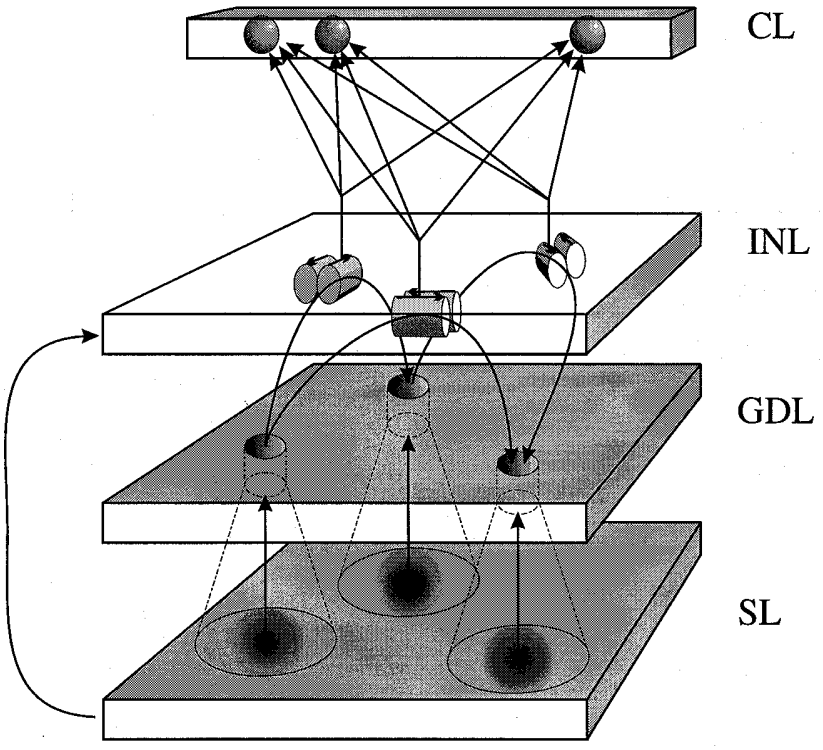

Fig. 4. Scheme of the self-organizing architecture. *CL* is control layer, *INL* is interneuronal layer, *GDL* is geometry discretizing layer and *SL* is sensory layer.

multigrid. The schematic structure of the (M)PDA map is shown in Fig. 4. The figure depicts the sensory layer (SL), the feedforward spatially tuned connection structure that excites the discretizing units of the geometry discretizing layer (GDL). Units of the GDL are connected by the neighboring connections that input the interneurons of the interneural layer (INL). The connections between the interneural layer and the control layer (CL) develop the directional actions (interneural command vectors) that can compensate for structural errors by means of learning. The working of the network is detailed in Fig. 5.

4.2. *Self-organized command vectors*

The controlling experiments were performed on the discretization shown by Fig. 1, thus $\mathbf{q}$, $\dot{\mathbf{q}} \in \mathbb{R}^2$. The controller consisted of four control neurons and the plant's equation had the simple form of $\dot{\mathbf{q}} = \mathbf{A}\mathbf{u}$ where $\mathbf{A} \in \mathbb{R}^{2\times 4}$ was a fixed matrix, the four control units correspond to left, right, up and down directions.

Results of learning the command connections with the help of direct associative identification of the MPDA map are shown in Fig. 6 (left). The learning equations are given in Sec. 7. They may be compared with the prewired case of Fig. 6 (right), i.e. the case when the $\mathbf{u}_{ij}$ command connections were prewired according to Eq. (21), that any individual interneural command vector moved the plant exactly in the interneural direction.

Figure 6 was developed as follows: First the relaxation process was initialized in a way that the upper-left pixel was set to -1 and the corresponding neural initial target activities were computed. Then one single pixel at a specified position was set to $+1$ and again the corresponding neural initial source activities were computed. The algebraic sum of the two sets gave the initial activities for the spreading relaxation procedure. After relaxation the different parts of interneuron activities were used as the population code for the control signal. These respective parts are summed up providing the control signal $\mathbf{u}$. The

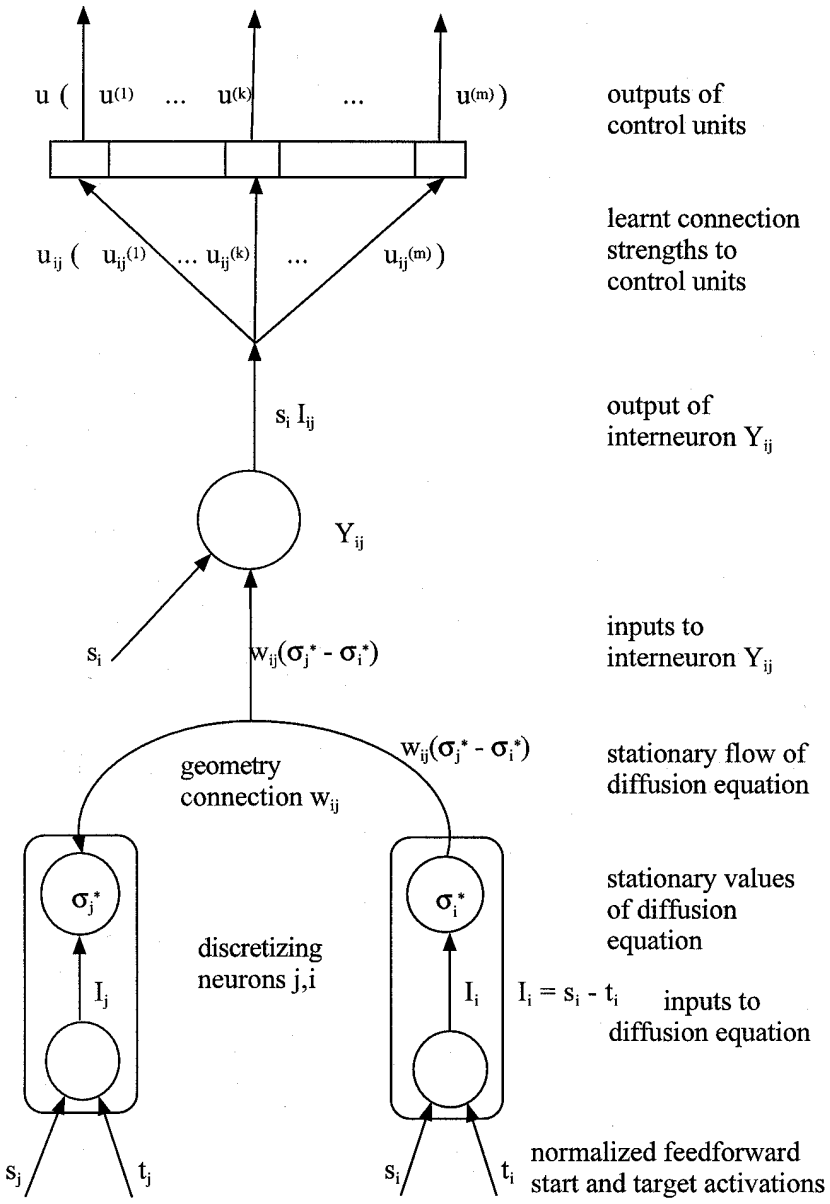

Fig. 5. Working of the network.

control signal was computed according to Eq. (24). In the figure the normalized speed vectors $\mathbf{Au}/||\mathbf{Au}||$ are shown at the position of the start pixel. In Fig. 6 the upper and lower rows show cases when (i) all the neurons and when (ii) only the neurons of the high resolution layer may contribute to the control signal, respectively. The left and right subfigures correspond to learned (left) and prewired (right) control command connections. The prewired and the learnt cases are rather similar. The boundary condition of the diffusion process resulted in the "off-from-the-boundary" direction of the speed vectors situated at the boundaries in both cases. This behavior would fully disappear if torus were used in the experiments, or if the Neumann boundary condition was modified.

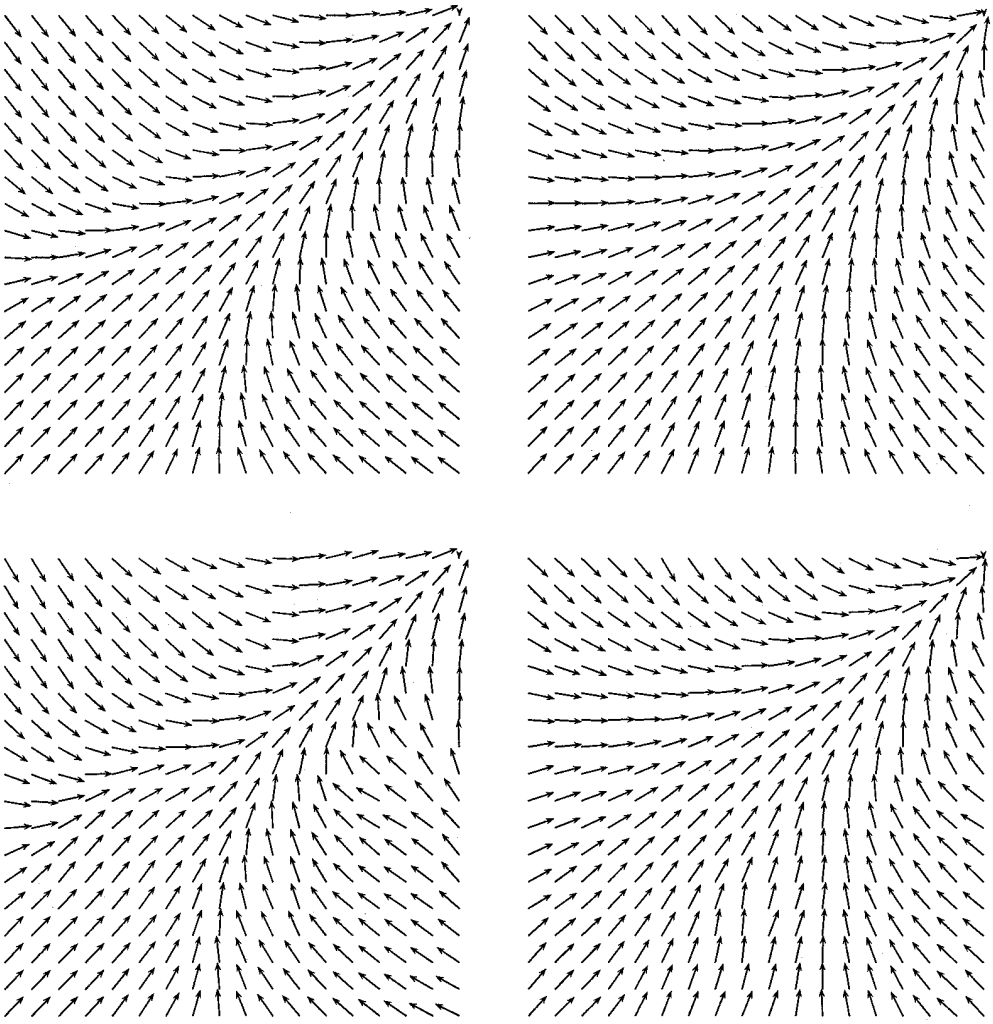

Fig. 6. Normalized speed vectors with learned command connections (*left*) and prewired command connections (*right*). *Top*: All neurons play a role in the relaxation. *Bottom*: Only the neurons of the high resolution group play a role in the relaxation. (For more details see text.)

4.3. *Motion generation*

The control protocol is a closed-loop static state feedback procedure that continuously upgrades the sensory information: To every instant the relaxation procedure was executed and the plant was moved in accord with the vectorial sum of the start neuron activity governed interneural command vectors. At the new position the relaxation procedure was again executed, and so on. The system was used for closed-loop motion generation in the presence of an obstacle. The plant started at each case from the lower left corner. The diameter of the plant was 5.0 pixel. The three curves of Fig. 7 correspond to three different threshold values that determine the obstacle nodes and thus the extent of the free space; neurons with above threshold obstacle activations are excluded from the free space. The curve that hits the rectangular obstacle corresponds to the move in the space without obstacle. It curves away from the edge of the 2D region as expected from the learned speed vectors. The two other curves have different obstacle thresholds. A threshold with 10% of the maximal obstacle input activity ($\max_i \hat{o}_i$) results in a curve that would hit the obstacle due to the finite

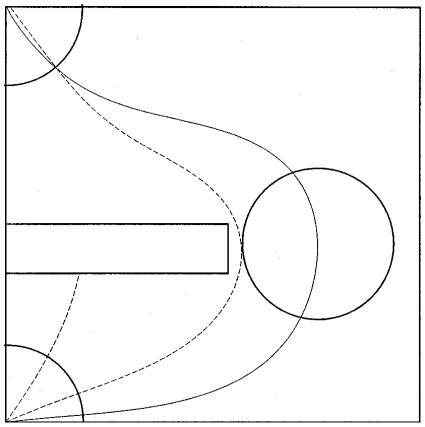

Fig. 7. Motion in the presence of rectangular obstacle.

size of the plant under control (dashed line). More stringent threshold value (1%) moves the plant in a collision-free path (solid line).

As may be seen, the controlling experiment has a better precision then the precision of the discretization being the result of coarse coding: the center of the plant approaches the target within the size of both the plant and the pixel in all cases.

A somewhat more complex labyrinth is depicted in the upper left subfigure of Fig. 8. Two rectangular obstacles are now placed at the two sides of the figure and the plant should move on a doubly curved path. The size of the plant was 0.75 pixel. The fully connected network is shown in the upper right part of Fig. 8. The lower left subfigure depicts the free

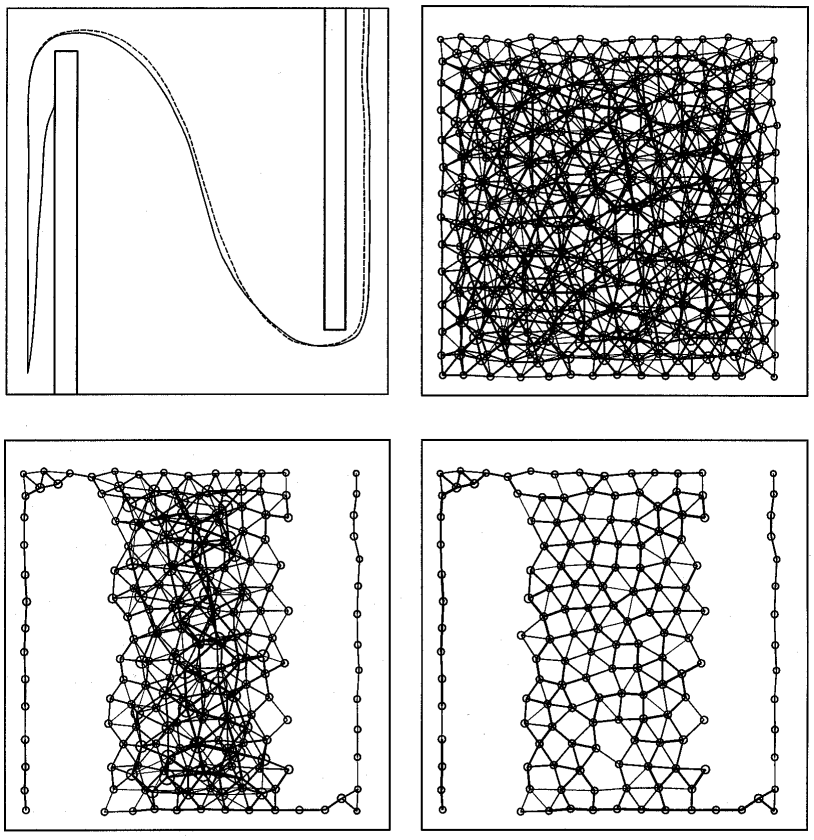

Fig. 8. The 'labyrinth' problem. Two rectangular objects are placed into the 2D space and are shown in the *upper left* subfigure. The *upper right*, the *lower left* and the *lower right* subfigures depict the full connectivity structure, the free space units with their connectivity structure for the multigrid (obstacle threshold value is 0.002), and the free space units with their connectivity structure for the small receptive field subgrid (obstacle threshold value is 0.00475), respectively. The path for the multigrid are depicted by solid line in the *upper left* subfigure. Obstacle threshold values of 0.00475 and 0.002 result in a collision and in a collision free path, respectively. The path for the small receptive field subgrid (obstacle threshold value is 0.00475) is depicted by the dashed line.

space units and their connectivity structure for the same network. The lower right subfigure shows the free space and the connectivity structure for the case when only the small receptive field neurons are considered. The two cases require different thresholding: in the case of the full multigrid an obstacle threshold value of 0.00475 gives rise to collision, while a threshold value of 0.00200 allowed collision free motion. In the case of the small receptive field subgrid, an obstacle threshold value of 0.00200 deletes too many neurons, breaking the connection structure into disjunct pieces that cannot provide a continuous path for diffusion. A threshold value of 0.00475 proved to be satisfactory for this case. The three paths are shown in the upper left subfigure of Fig. 8; the solid lines depict the path formed by means of the multigrid for the two different cases, while the dashed line depicts the path formed by means of neurons of the small receptive field subgrid.

The difference in the relaxation time for the multigrid and the single subgrid is illustrated by Fig 9. The figure depicts the integrated relaxation times needed for the closed-loop control procedure. In both cases the network was allowed to relax first. The speed vector was computed and a fixed step length (0.25 pixel) was executed into the direction of the speed vector. The new position was used to upgrade the input activities of the diffusion equation and the network was allowed to relax. The approximate relaxation rate $R(n)$ was computed. When that rate decreased to the one tenth of the value just after the step was executed, i.e. just at the time when the new relaxation procedure was started, the relaxation was considered as 'finished' and the relaxation time of that step was recorded, and so on. The integrated relaxation times are depicted in Fig. 9. The curve for the multigrid and the small receptive field subgrid are depicted by the solid line and the dashed line, respectively. There is a large difference in the open space region where the multigrid integrates very little time — the curve is almost horizontal. The advantage of the multigrid, however manifests itself in the region when the plant is close to the obstacle too, since the relaxation procedure concerns the whole network always. The difference between the network relaxation times is not large when the full network is between the source and sink nodes and the diffusion is limited by the narrow stripes, since the free space region gives just a small contribution. In the other extreme, however, when the source and the sink are at the same part of the free space, the 'hanging' network may considerably lengthen the relaxation time and this lengthening is more pronounced for the single subgrid structure.

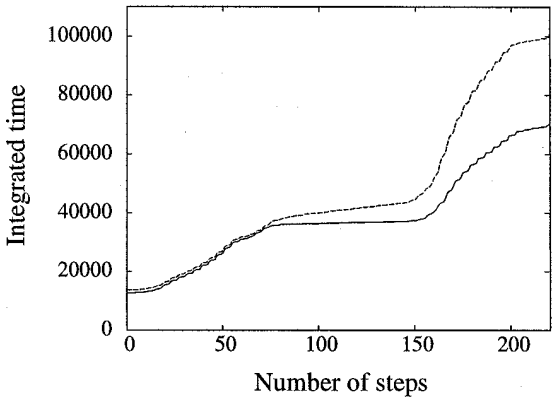

Fig. 9. Integrated relaxation time. The solid and the dashed lines depict the integrated relaxation time for the multigrid and the single subgrid, respectively. After each relaxation procedure a fixed length step (0.25 pixel) was executed, the new sensory information was inputted, the source activities of the diffusion equation were recomputed and the diffusion equation was allowed to relax, etc. The integrated relaxation times are shown in arbitrary units.

5. Discussion

5.1. *Role of multigrid: the maze problem*

The multigrid approach is useful when the problem of path planning and motion control is composed of large free spaces as well as thin maze-like structures. It is then advantageous that diffusion in the large free spaces will relax quickly with the help of the large receptive field neurons, while diffusion can still proceed in the bottlenecks via neurons with small sized filters. The hierarchy of large and small filters optimizes itself to a given diffusion problem if the Neumann boundary condition is applied by identifying the obstacle regions and forbidding diffusion along the connections of neurons positioned mostly (i.e. above some predefined threshold) within this region. Large receptive field neurons will take part in the diffusion only if they are situated within large free spaces, while narrow corridors will be discovered for path planning by the diffusion field if resolution

— set by the size of the objects themselves — allows. It may be important to mention that our goal is not to approximate the original diffusion equation as closely as possible but to create a speed field for control.

The fast relaxation of the diffusion is important since the presented control architecture represents a feedforward controller working with static state feedback closed-loop strategy and thus it requires fast updating. Considering the massively parallel architecture and a similarly structured hardware implementation, it is the diffusion that would be the slowest part of the controller, since the controller has three connection sets, two of which are simple feedforward connections while the third one corresponds to the recurrent interneural connections that govern the diffusion. It is then clear that this stage will determine the delay time of the feedforward controller.

It is also important that connections between the filters of different sizes have been developed: this allows the diffusion to spread in the fastest possible way allowed by the obstacles.

The uniform coverage of the external space by the filters in both position and size is also important since that can decrease the structural error of the controller. This requirement has minor restrictions for competitive systems where the structural error is kept low since competing units tend to arrange themselves in dense (close) packing. In the case of simple connected external spaces, the least structural error may belong to a multigrid structure. However, it is also possible that extremely small learning rates are needed to form the regular structure or that such structures may not be formed. In 1D and 2D the discretization grids are rare exceptions since in 1D and 2D local and global close packing structures are the same. Already in 3D, however, the local close packing has an icosahedral structure, while global close packing has hexagonal layers. That is one can freeze errors into the global structure. It is also possible that the external space of the control problem is not known in advance. Then the competitive net that optimizes the discretization structure, demonstrated by the multigrid formation, is an attractive choice. It is, however, not important for the discretization to be completely regular. This compromise is offered by the self-organizing associative direct identification procedure of the inverse dynamics.

The improvement in the speed of the relaxation of the diffusion is shown by the simulations when the multigrid structure is used is apparent. The improvement should be even larger for systems having a broader range of resolution units and/or in larger externals spaces. Here a particular reference should be made to the initial slowly changing parts of the diffusion figures (Fig. 3). This portion of the diffusion is not too pronounced in our computer runs due to our limitations in network sizes. This portion, however, has approximately a $t^{-d/2}$ time dependence, where d is the dimension of the external space (Carslaw and Jaeger, 1954), and may severely constrain the relaxation time of a high resolution single layer discretizing grid.

5.2. *Learning versus adaptation*

It has been shown that controllers that approximate the inverse dynamics of the controlled plant — just like the present architecture — can be used in the so called Static and Dynamic State (SDS) Control Mode to compensate inhomogeneous, non-linear, non-additive perturbations (Szepesvári and Lőrincz, 1996c). The SDS control mode offers a solution to the learning versus adapting problem (Szepesvári and Lőrincz, 1996b): The architecture, as is demonstrated here, can be tuned. At the same time this tuning is (i) slow and (ii) requires a thorough exploration of the possible configurations. Time constraints may prevent one from utilizing this learning capability in case of small and transient perturbations, such as varying loads and temperature effects on friction coefficients. The SDS scheme suggests to use the architecture with the learnt and approximate inverse dynamics in *two* identical copies. One copy acts as the original closed loop controller while the other identical copy can be used to develop the compensatory signal necessary to counteract the actual unforeseen external or internal structural perturbations — as it can be shown in a rigorous manner (Szepesvári and Lőrincz, 1996c; Szepesvári and Lőrincz, 1996b). The compensatory signal that can be developed without the need of influencing the learnt weights represents the adaptation capability of the system. As a consequence, by using the two identical copies of architecture one can control the plant more precisely than just by the closed loop feedforward controller alone.

Since the SDS Controller utilizes two identical copies of the very same architecture thus the credit assignment problem is considerably eased: Learning simply concerns the association of actual directions (i.e. interneural activities) and control vectors of the single architecture at all times.

5.3. *The generality of the approach*

The fully self-organizing nature of the architecture allows one to think of problems beyond robotics that could be subject of self-organizing controlling. As a miscellaneous example consider the problem of a plug flow reactor with distributed sensory and control systems designed to control complex reactions (Rojnuckarin *et al.*, 1993). The plug flow reactor model permits control by the chemical and heat fluxes added through the side wall of the reactor. The sensory systems may detect temperature, pressure, spectral information and/or concentration along the reactor in a distributed fashion. The multigrid may be built up by transients (local extended objects) as they proceed along the reactor. Such a representation is not suitable for traditional competitive schemes since inputs are not presented in a random fashion. Nevertheless, learning can be stabilized by means of a suppression mechanism (Tavitian *et al.*, 1996). The dimensionality of the representation is not well defined: it is at least 1, since the flow is one-dimensional. Additional dimensions may arise, however, from the competing chemical reactions. The general formulation of the MPDA map as well as the possibly low dimensionality of the control problem may allow the development of a self-organized control strategy. The complexity of the chemical processes could be overwhelmingly challenging for the straightforward application of the simple scheme described in the paper. The control of the plug-flow reactor, however, clearly calls for robust, self-organizing and fast (massively parallel) computational schemes.

5.4. *Population coding and biological relevance*

It may be worth mentioning that the present self-organizing network has some resemblance to the motor-like map in the primary motor cortex. According to the experiments of Georgopoulos and coworkers who studied how neuronal activity varies when monkeys move a handle to one of several targets arranged around a central starting position, the activity of individual neurons vary with the direction of the movement: they fired most briskly for movements in a preferred direction and fell silent during movements in the opposite direction. The directional tuning of all recorded neurons was broad. Individual neurons contribute predominantly to movements in a preferred direction but also to lesser degrees to movements in other directions (Georgopoulos *et al.*, 1982). Georgopoulos proposed that movement in a particular direction is determined not by the action of single neurons but by the net action of a broad population of neurons. Furthermore, he suggested that the contributions of each neuron to movement in a particular direction could be represented as a vector whose length depended on the degree of activity during movements in that particular direction. He then added the contributions of individual cells vectorially to produce a population vector that was directed in the correct direction.

The working of the model presented here is very similar. Figure 10 shows the individual speed vectors $\mathbf{A}s_i I_{ij}\mathbf{u}_{ij}$ corresponding to the individual interneural command vectors $\mathbf{u}_{ij}$ and the speed vector corresponding to the control signal, $\mathbf{Au}$ [where $\mathbf{u}$ was computed according to Eq. (24)], with the source being at the center of the figure and the goal being at eight different positions. The command vectors are depicted at the goal position for the ease of inspection. The largest vector in each group represents the net response of the network (direction of motion) in the appropriate position. The speed vectors of the individual interneurons are scaled in the figure in the following way: the largest individual speed vector is scaled up to 75% of the full control signal in each subfigure. The upper left, upper right and lower left figures show the results for the small, medium and large neural groups, respectively. The lower right figure shows the result for the full system. Command directions cover angles larger than 180 degrees, nevertheless the vectorial sum points to the appropriate direction.

From the point of view of Georgopoulos' findings the present model offers an explanation for why both broad tuning and well directed vectorial sum may emerge simultaneously. It should be stressed,

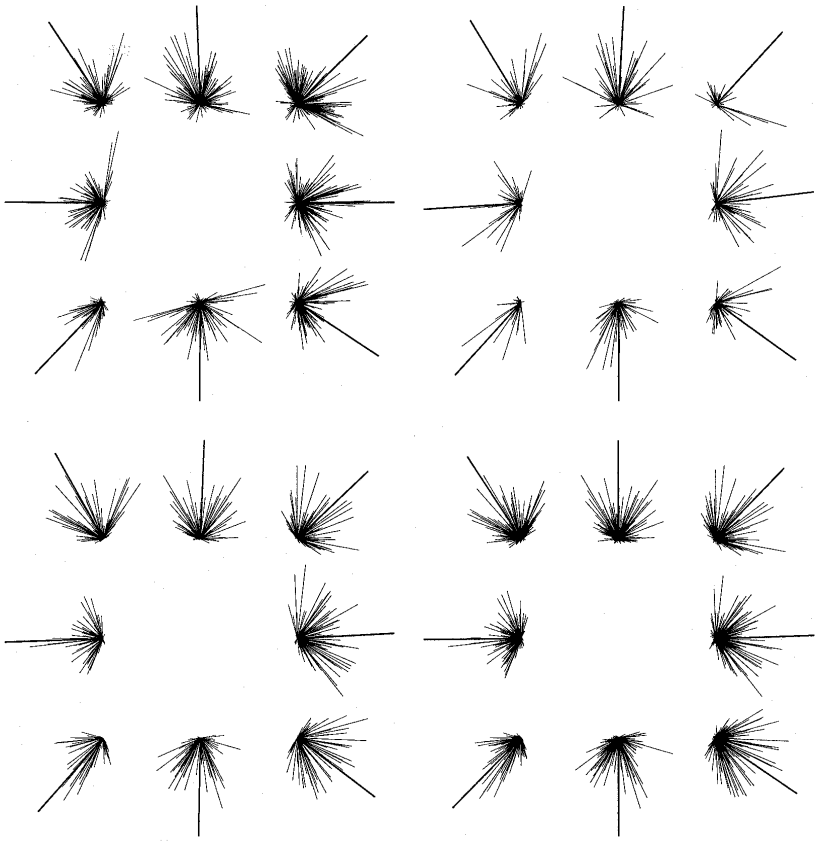

Fig. 10. Georgopoulos representation with small input objects and learned interneurons. *Left top*: small sized neurons contribute, *right top*: medium sized neurons contribute, *left bottom*: large sized neurons contribute, *right bottom*: all neuron contribute to the population coding. Details of the figure are given in the text.

however, that the network presented here is not a biological model by any means.

6. Conclusions

The MPDA architecture and its working have the following features:

- it is a fully self-organizing architecture with a discretizing neural layer and geometry connections between discretizing neurons,
- the diffusion-type relaxation procedure on the discretizing neural layer is accelerated by means of the self-organizing multigrid structure,
- the interneurons introduced to each geometry connections allow the direct associative identification of the inverse dynamics of the plant,
- the learnt command vectors lead to population coding with a broad range of angles for the many command vectors, resulting in a robust scheme,
- the architecture offers an attractive solution for the *credit assignment* problem for combining feedforward and feedback controllers,
- the architecture offers a natural solution for *the learning versus adapting* dilemma.

Acknowledgments

This work was partially supported by OTKA Grants T017110, T014566 and the grant of the US-Hungarian joint fund JF519.

7. Appendix

7.1. *Following the gradient*

In order to realize Eq. (5) in practice, we have to represent the inverse dynamics of the plant in a

suitable way. First let us fix an arbitrary point $\mathbf{q}$ in the space. This point may be chosen as a discretization point. Let $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{R}^n$ denote k "direction" vectors ($k \geq n$). One might think of the geometrical vectors belonging to a discretization point. Assume, that the control vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k \in \mathbb{R}^m$ satisfy the equalities

$$\mathbf{v}_i = \mathbf{b}(\mathbf{q}) + \mathbf{A}(\mathbf{q})\mathbf{u}_i\,, \quad i = 1, \ldots, k\,. \tag{18}$$

Assume, that the k direction vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ span the n dimensional space. We propose that the k control vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ are sufficient for controlling the plant at the state $\mathbf{q}$. To show this, assume that the plant is to be moved into the direction $\mathbf{d}$ from the point $\mathbf{q}$ and $\mathbf{d}$ is expressed as

$$\mathbf{d} = \sum_{i=1}^{k} \alpha_i \mathbf{v}_i\,, \quad \text{with} \sum_{i=1}^{k} \alpha_i = 1\,. \tag{19}$$

Note that if there are at least $n+1$ vectors among the vectors $\mathbf{v}_i$ that are affine independent (i.e. any n vector span $\mathbb{R}^n$), then coefficients that satisfy $\sum_i \alpha_i = 1$ may be found. Let us consider the control vector

$$\mathbf{u} = \sum_{i=1}^{k} \alpha_i \mathbf{u}_i\,. \tag{20}$$

Substituting Eq. (20) into Eq. (4) we have that the control vector $\mathbf{u}$ yields the speed vector $\mathbf{d}$.

Now assume that we are given a path planning problem and the recurrent network has already relaxed in a stationary state. One may then determine the control signal $\mathbf{u}$ that is able to move the plant in the desired direction in the following way: The speed vector of the plant must correspond to the gradient of the equilibrium activity map at the start state. In Sec. 4.1 it was shown that the gradient of the flow at the start state can be estimated by $\mathbf{d} = \sum_{i \in F} s_i \mathbf{d}_i$, where $\mathbf{d}_i$ is the approximated gradient at neuron i: $\mathbf{d}_i = \sum_{j \in N_i \cap F} I_{ij}\mathbf{e}_{ij}$, where $I_{ij} = w_{ij}(\sigma_j^* - \sigma_i^*)$. According to the previous section, if one is given the control command vectors $\mathbf{u}_{ij}$, $j \in N_i$, satisfying

$$\mathbf{e}_{ij} = \mathbf{b}(\mathbf{c}_i) + \mathbf{A}(\mathbf{c}_i)\mathbf{u}_{ij}\,, \tag{21}$$

then control vector

$$\mathbf{u}_i = \sum_{j \in N_i \cap F} s_i I_{ij} \mathbf{u}_{ij} \tag{22}$$

moves the plant into the direction $\mathbf{d}_i$ provided that the plant is in state $\mathbf{c}_i$.[a] Taking into account the coarse coding of the state of the plant, i.e. that the state of the plant is given by the full blob, we get that the control vector is approximated by

$$\mathbf{u} = \sum_i \mathbf{u}_i\,. \tag{23}$$

Equations (22) and (23) fit well to the recurrent architecture that compute the stationary flow.

Let us now assume that the path planning architecture is extended by interneurons and control neurons in a way described in Sec. 4.1. Interneurons correspond to neighboring connections and monitor the activities that flow along the connections and provide proportional outputs with the monitored signals, i.e. the output of interneuron Y_{ij} is given by $s_i I_{ij}$. Interneurons are connected to control neurons via command connections. Let the command connection that starts from interneuron Y_{ij} and ends on control neuron k be the kth component of $\mathbf{u}_{ij}$: u_{ij}^k. The individual control command of interneuron Y_{ij} is thus equal to $s_i I_{ij} \mathbf{u}_{ij}$. Control neurons provide control signals $\mathbf{u}$ by summing up these individual control commands:

$$\mathbf{u} = \sum_i \sum_{j \in N_i \cap F} s_i I_{ij} \mathbf{u}_{ij}\,. \tag{24}$$

Considering Eqs. (21)–(24), it can be seen that the learning problem is to tune the $\mathbf{u}_{ij}$ values that they each move the plant in their respective directions $\mathbf{e}_{ij}$ provided that the plant is in state $\mathbf{c}_i$. The motion planning and execution procedure can be summarized as follows:

Step 1. Develop the coarse coding of the path planning task on the recurrent network.

Step 2. Compute the equilibrium activity map by activation spreading.

Step 3. Interneurons compute the directional derivatives of the flow weighted by the coarse coding activities of the actual state of the plant.

Step 4. Interneurons send their outputs through the command connections to the control neurons. The sum of received activities is the control signal.

[a] In Eq. (22) the I_{ij} coefficients should have been normalized but according to our numerical experiments Eq. (22) can work equally well.

The architecture and its working are shown in Figs. 4 and 5.

7.2. *Direct, associative identification of the inverse dynamics*

This learning scheme is discussed in details in (Fomin *et al.*, 1994, Szepesvári and Lőrincz, 1996a). The algorithm learns the control vectors as follows:

Step 1. Develop the coarse coding that corresponds to the state of the plant. Store it as start activities.

Step 2. Choose a random control signal and feed it into the plant.

Step 3. Compute the coarse coding of the resulting state of the plant and use it as the target activities.

Step 4. Compute the equilibrium activity map according to these start and target activities.

Step 5. Associate the control signal to interneurons weighted by the output of the interneurons.

In Step 5, the signal Hebbian learning rule may be used, i.e.

$$\dot{\mathbf{u}}_{ij} = \alpha_{ij}(s_i I_{ij}\mathbf{u} - \mathbf{u}_{ij})\,, \tag{25}$$

where $\mathbf{u}$ is computed by the network according to Eq. (24). $0 < \alpha_{ij} < 1$ is the learning rate of interneuron Y_{ij}, that can be time-dependent or stationary. It is reasonable to choose a Robbins-Monro type time dependence (Robbins and Monro, 1951; Wasan, 1969) in order to ensure the convergence of $\mathbf{u}_{ij}$ to the average of learning samples with respect to the input distribution and also to maintain adaptivity forever. However, in this case, adaptivity may become extremely slow by time. If the learning rate is kept constant then adaptivity may be kept above a predefined level. In this case $\mathbf{u}_{ij}$ may be considered as a stochastic variable with mean given by the sample average and deviation magnitude proportional to $\sqrt{\alpha_{ij}}$ (Amari, 1967) after enough training epochs have passed. The additive term $\mathbf{b}(\mathbf{q})$ of the inverse dynamics can be easily learnt by using the plant's zero dynamics; i.e. by setting $\mathbf{u} = 0$ and learning the resulting movements.

Another possible learning method would be to replace Step 5 by storing the "best action vector so far". However, this method may be sensitive to state errors, while the Hebbian learning does average and can handle noisy inputs too.

Yet another possible method is to restrict learning to the immediate neighborhood of the neuron with the highest s_i activity, and further to the interneuron that has the highest flow I_{ij}, that is

$$\dot{\mathbf{u}}_{ij} = \begin{cases} \alpha_{ij}(\mathbf{u}-\mathbf{u}_{ij}) & \text{if } s_i = \max\limits_j s_j \text{ and} \\ & \quad I_{ij} = \max\limits_l I_{il}\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{26}$$

Learning rule (26) can be considered as the "soft" version of Eq. (25). For this latter rule one can prove that the limit of the $\mathbf{u}_{ij}$ command connection vector satisfies Eq. (21) and thus the algorithm can learn the inverse dynamics of any plant up to the precision of discretization.

Again, if the time dependence of the α_{ij} values satisfy $\sum_t \alpha_{ij}(t) = \infty$ and $\sum_t \alpha_{ij}^2(t) < \infty$, where t is incremented only if interneuron (i, j) learns, then the convergence of $\mathbf{u}_{ij}$ is guaranteed.

In order to prove this note that the limit of command connection vector $\mathbf{u}_{ij}$ corresponding to interneuron Y_{ij} can be written in the form

$$\overline{\mathbf{u}}_{ij} = \int_W dP(\mathbf{x}) \int_{Y(\mathbf{x})} u(\mathbf{x}, \mathbf{y}) dP(\mathbf{y}|\mathbf{x})\,,$$

where $u(\mathbf{x}, \mathbf{y}) = \mathbf{A}^{-1}(\mathbf{x} - \mathbf{b}) + (\mathbf{E} - \mathbf{A}^{-1}\mathbf{A})\mathbf{y}$ with $\mathbf{A} = \mathbf{A}(\mathbf{c}_i)$, $\mathbf{b} = \mathbf{b}(\mathbf{c}_i)$ and W is the set of $\mathbf{x}$ direction vectors for which the geometry connection (i, j) is the "winner", and $Y(\mathbf{x})$ represents an arbitrary measurable set. Substituting the expression for $u(\mathbf{x}, \mathbf{y})$ in the above equation yields

$$\overline{\mathbf{u}}_{ij} = \mathbf{A}^{-1}(\overline{\mathbf{x}} - \mathbf{b}) + (\mathbf{E} - \mathbf{A}^{-1}\mathbf{A}) \times \int_W dP(\mathbf{x}) \int_{Y(\mathbf{x})} \mathbf{y} dP(\mathbf{y}|\mathbf{x})\,,$$

where $\overline{\mathbf{x}} = \int_W \mathbf{x} dP(\mathbf{x})$. If W is symmetrical with respect to $\mathbf{g}_{ij}$ (e.g. the discretization is regular), then $\overline{\mathbf{x}} = \mathbf{g}_{ij}$ and thus in this case $\overline{\mathbf{u}}_{ij}$ satisfies Eq. (21) (i.e. $\overline{\mathbf{u}}_{ij} = \mathbf{u}_{ij}$). Moreover, if the sampling of $\mathbf{y}$-s is also symmetrical for each given $\mathbf{x}$, that is if $Y(\mathbf{yx})$ is centrally symmetric, then the second term in the above equation disappears, too.

References

S. Amari 1967, "Theory of adaptive pattern classifiers," *IEEE Trans. Elect. Comput.* **16**, 299–307.

H. S. Carslaw and J. C. Jaeger 1954, *Conduction of Heat in Solids* (Oxford University Press, London).

C. I. Connolly and R. A. Grupen 1993, "On the applications of harmonic functions to robotics," *J. Robotic Sys.* **10**(7), 931–946.

T. L. Dean and M. P. Wellman 1991, *Planning and Control* (Morgan Kaufmann, San Mateo, CA, USA).

T. Fomin, C. Szepesvári and A. Lőrincz 1994, "Self-organizing neurocontrol," in *Proc. IEEE Int. Conf. Neural Networks. IEEE World Cong. Comput. Intell.* (Orlando, Florida), pp. 2777–2780.

A. P. Georgopoulos, J. F. Kalaska, R. Caminiti and J. T. Massey 1982, "On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex," *J. Neurosci.* **2**, 1527–1537.

A. P. Georgopoulos, R. E. Kettner and A. B. Schwartz 1988, "Primate motor cortex and free arm movements to visual targets in three-dimensional space. II: Coding of the direction of movement by a neuronal population," *J. Neurosci.* **8**, 2928–2937.

R. Glasius, A. Komoda and S. Gielen 1995, "A biologically inspired neural net for trajectory formation and obstacle avoidance," *Biol. Cybern.* **84**, 511–520.

R. Glasius, A. Komoda and S. Gielen 1995, "Neural network dynamics for trajectory formation and obstacle avoidance," *Neural Networks* **8**, 125–133.

Y. K. Hwang and N. Ahuja 1992, "Gross motion planning — a survey," *ACM Comput. Surveys* **24**(3), 219–291.

A. Isidori 1989, *Nonlinear Control Systems* (Springer-Verlag, Berlin).

M. I. Jordan 1990, "Learning and the degrees of freedom problem," in *Attention and Performance, XIII* (Hillsdale, NJ: Erlbaum).

M. Kawato, K. Furukawa and R. Suzuki 1987, "A hierarchical neural-network model for control and learning of voluntary movements," *Biol. Cybern.* **57**, 169–185.

T. Kohonen 1984, *Self-Organization and Associative Memory* (Springer-Verlag, Berlin).

G. Lei 1990, "A neural model with fluid properties for solving labyrinthian puzzle," *Biol. Cybern.* **64**, 61–67.

F. L. Lewis, C. T. Abdallah and D. M. Dawson 1993, *Control of Robot Manipulators* (MacMillan, New York).

T. Martinetz 1993, "Competitive Hebbian learning rule forms perfectly topology preserving maps," in *Proc. Int. Conf. Art. Neur. Networks* (Springer Verlag, Berlin), pp. 427–434

T. Martinetz and K. Schulten 1994, "Topology representing networks," *Neural Networks* **7**(3), 507–522.

W. T. Miller 1987, "Sensor based control of robotic manipulators using a general learning algorithm," *IEEE J. Robotics and Automation* **3**, 157–165.

W. T. Miller, R. S. Sutton and P. J. Werbos 1990, *Neural Networks for Control* (MIT Press, Cambridge, MA).

H. Miyamoto, M. Kawato, T. Setoyama and R. Suzuki 1988, "Feedback-error-learning neural network for trajectory control of a robotic manipulator," *Neural Networks* **1**, 251–265.

K. Narendra and K. Parthasarathy 1990, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks* **1**(1), 4–27.

A. Pouget and T. J. Sejnowski 1995, "Dynamic remapping," in *The Handbook of Neural Theory and Neural Networks* (Bradford Books/MIT Press, Cambridge), pp. 335–338.

H. Robbins and S. Monro 1951, "A stochastic approximation method," *Ann. Mat. Stat.* **22**, 400–407.

A. Rojnuckarin, C. A. Floudas, H. Rabitz and R. A. Yetter 1993, "Optimal control of a plug flow reactor with complex reaction mechanism," *J. Phys. Chem.* **97** 11689–11695.

T. Rozgonyi, L. Balázs, T. Fomin and A. Lőrincz 1996, "Self-organized formation of a set of scaling filters and their neighboring connections," *Biol. Cybern.* **75**, 37–47.

T. Rozgonyi, T. Fomin and A. Lőrincz 1994, "Self-organizing scaling filters for image segmentation," in *Proc. Int. Conf. Art. Neur. Networks*, Sorrento, Italy (Springer-Verlag), pp. 1129–1132.

C. Szepesvári, L. Balázs and A. Lőrincz 1994, "Topology learning solved by extended objects: A neural network model," *Neural Comput.* **6**, 439–456.

C. Szepesvári and A. Lőrincz 1996, "Approximate geometry representations and sensory fusion," *Neurocomput.* **12**, 267–287.

C. Szepesvári and A. Lőrincz 1996a, "Neurocontrol I: Self-organizing speed–field tracking," *Neural Network World*, **6**(6), 875–896.

C. Szepesvári and A. Lőrincz 1996b, "Neurocontrol II: High precision control achieved using approximate inverse dynamics models," *Neural Network World*, **6**(6), 897–920.

C. Szepesvári and A. Lőrincz 1996c, "Inverse dynamics controllers for robust control: Consequences for neurocontrollers," in *Proc. Int. Conf. Art. Neur. Networks* Bochum, Germany (Springer-Verlag), pp. 697–702.

C. Szepesvári and A. Lőrincz 1993, "Topology learning solved by extended objects: A neural network model," in *Proc. Int. Conf. Art. Neur. Networks* Amsterdam (Springer-Verlag, London), p. 678.

S. Tavitian, T. Fomin and A. Lőrincz 1996, "Stabilizing competitive learning during on-line training with an anti-Hebbian weight modulation," in *Proc. Int. Conf. Art. Neur. Networks* Bochum, Germany (Springer-Verlag), pp. 791–796.

V. Vemuri 1993, "Artificial neural networks in control applications," *Advances in Computers* **36**, 203–254.

T. Wasan 1969, *Stochastic Approximation* (Cambridge University Press, London).

P. J. Werbos 1988, "Generalization of back propagation with applications to a recurrent gas market model," *Neural Networks* **1**, 339–356.

B. Widrow 1986, "Adaptive inverse control," in *Proc. of the Second IFAC Workshop on Adaptive Systems in Control and Signal Processing*, Lund, Sweden (Lund Institute of Technology), pp. 1–5.

B. Widrow, J. McCool and B. Medoff 1978, "Adaptive control by inverse modeling," in *20th Asilomar Conference on Circuits, Systems and Computers.*