Computers Elect. Engng Vol. 19, No. 4, pp. 315–332, 1993
Printed in Great Britain. All rights reserved

0045-7906/93 $6.00 + 0.00
Copyright © 1993 Pergamon Press Ltd

A NEURAL NETWORK FOR ROBOT CONTROL: COOPERATION BETWEEN NEURAL UNITS AS A REQUIREMENT FOR LEARNING

THOMAS MARTINETZ and KLAUS SCHULTEN†
Beckman-Institute and Department of Physics, University of Illinois at Urbana-Champaign, 405 North Matthews Ave, Urbana, IL 61801, U.S.A.

(Received 5 December 1990; accepted in final revised form 15 October 1991)

Abstract—We present a self-organizing neural network for learning visuo-motor coordination and describe its application to the task of learning the end effector positioning of a simulated, three-jointed robot arm. In contrast to a neural network algorithm that we introduced earlier in Refs [1,2] (Martinetz *et al.*, *IEEE Trans Neural Networks* **1**, 131, 1990; Ritter *et al.*, *Neural Networks* **2**, 159, 1989) we now employ visual feedback which enables the robot arm to position its end effector with an accuracy of about 5% of the size of the workspace after only 100 learning steps and with a final accuracy of about 0.06% after 6000 learning steps. Learning proceeds without the need for an external teacher by a sequence of trial movements using input signals from a pair of cameras. The use of visual feedback enables the robot arm not only to adapt to slowly occurring miscalibrations, but also to compensate for sudden changes in its geometry, e.g. when picking up a tool. The results of a simulation and a mathematical analysis of the learning procedure show that cooperation between neural units during the course of learning, incorporated in the network, is essential for the robot arm system to succeed in learning.

1. INTRODUCTION

Motion control is one of the oldest tasks biological organisms have to solve in order to be able to survive. Most of the biological motor control systems which emerged during evolution still outperform today's robot control mechanisms in many respects, including flexibility, velocity and especially their learning capability. Therefore, we have good reason to assume that valuable knowledge can be gained by elucidating the control principles employed by biological organisms.

One of the few generally recognized organizational principles found in the nervous system of many organisms is the formation of so-called topology conserving mappings from input stimuli onto parts of the cortex. The most prominent examples are the topology conserving mapping from the retina to the visual cortex [3] and the mapping from the body surface to the somatosensory cortex [4]. Through these topology conserving mappings input stimuli from adjacent receptors of the retina or the body surface excite neighboring neural units on the corresponding cortex. Another kind of topology conserving mapping can be found in the motor cortex, where adjacent neural units tend to code similar limb configurations and similar movements. These mappings, also called topology conserving or topographic feature maps, are not fixed, but develop after birth by receiving input stimuli from the corresponding sensory system. In addition, these maps are able to adapt, if unforeseen changes occur in the sensory system [5].

Several approaches which mimic the formation and the adaptability of these topology conserving feature maps have been suggested [6–11]. One prominent model is Kohonen's feature map algorithm [9,10] which captures only the most essential properties of the formation of such maps for the benefit of remaining computationally tractable. This model can be extended for the purpose of learning input–output relations [12,13], which then can be applied to control, i.e. robot control tasks [1,2,12–16]. We will demonstrate how this extended version of Kohonen's feature map is able to learn the task of end effector positioning up to a very high precision. The learning rule we employ is an improved version of the adaptation schemc introduced earlier in Refs [1,2]. The new learning rule and the new control scheme for generating the arm movements incorporate an additional visual feedback mechanism which both accelerates the learning process and leads to a higher accuracy.

†To whom all correspondence should be sent.

The network now achieves an accuracy three times higher than in Ref. [1] with only a fifth of the training steps needed before.

A number of different robot control schemes based on neural network models have been developed [17–21]. One important approach which also employs topographic maps is the neural network model proposed by Kuperstein [17,18]. In Kupersteins model, however, these maps are fixed and do not develop with the course of learning, which requires more prior knowledge about the structure of the set of input data the network has to process. In addition, the use of fixed maps do not provide an automatic increase in the resolution of the representation of those movements which have been trained more frequently. In the model we will describe here the topology conserving maps develop with the course of learning and adapt to the structural features of the set of input data through Kohonen's feature map algorithm. Kohonen's feature map algorithm also provides a finer representation and, therefore, a higher accuracy of movements which have been trained more frequently.

2. KOHONEN'S FEATURE MAP ALGORITHM AND ITS EXTENSION

Figure 1 presents a computer graphic of the simulated robot system which has to learn to position its end effector to visually designated target locations. The robot system consists of a robot arm with three degrees of freedom and a pair of cameras providing the spatial information about the

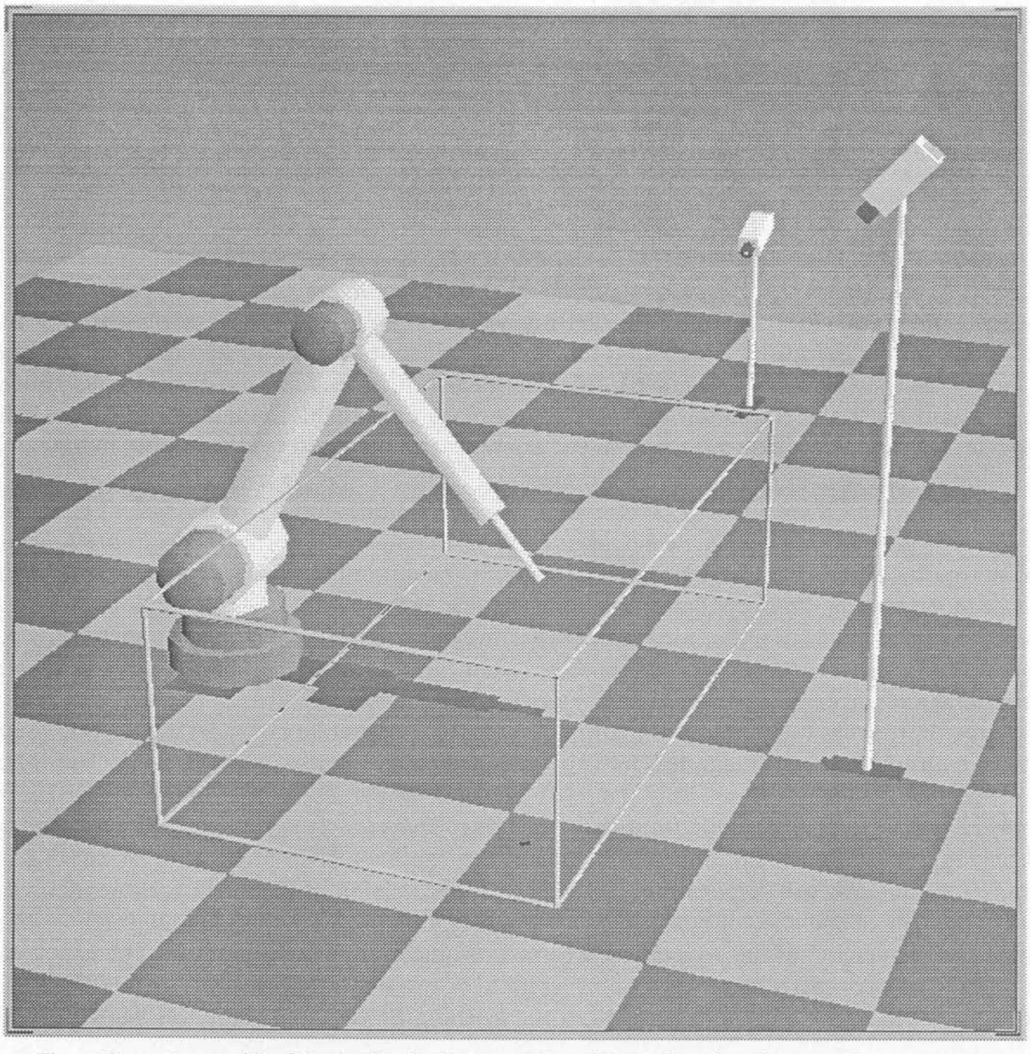


Fig. 1. Computer graphic of the simulated robot arm system. Depicted are the robot arm, the two cameras and the workspace.

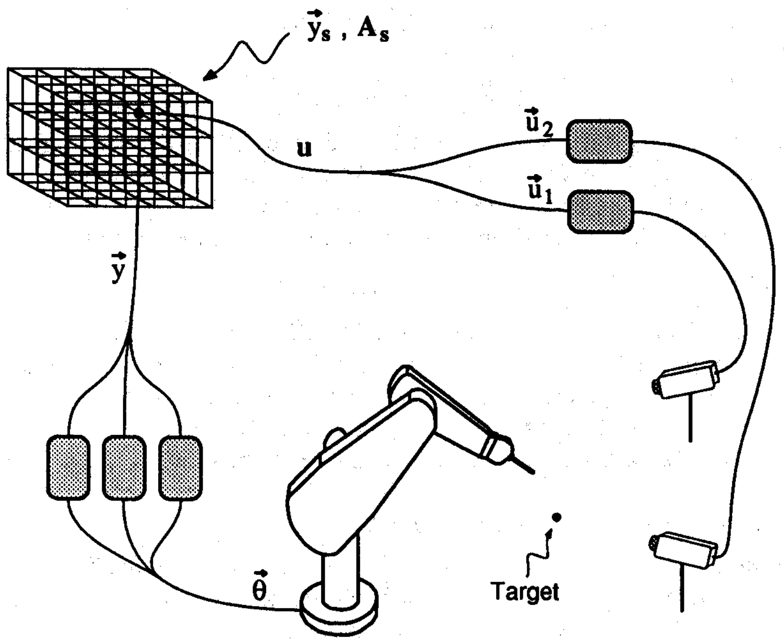


Fig. 2. Sketch of the robot system. The two 2-D camera coordinates $\vec{u}_1$ and $\vec{u}_2$ of the target object are combined to a 4-D vector **u** which is fed as an input signal to a 3-D lattice of neural units. The selected neural unit **s** which is responsible for the current input **u** provides a vector $\vec{y}_s$ and a Jacobian matrix $\mathbf{A}_s$ which determine the network's output $\vec{y}$. Through the joint angle controllers $\vec{y}$ generates joint angles $\vec{\theta}$ which lead the end effector to the target.

location of the object the robot arm reaches for. For each trial movement, the target location is chosen randomly within the depicted workspace. Figure 2 illustrates the configuration of the robot system more schematically. Each target within the 3-D workspace corresponds to a pair of 2-D coordinates, namely the locations of the two images of the object on the focal planes of the two cameras. We assume some form of visual preprocessing which is able to reduce the two images from the cameras to a pair of "retinal" coordinates $\vec{u}_1, \vec{u}_2$. In general this is not an easy task, but since we do not want to address issues of early vision in this paper, we assume the target object to be specific enough to be identifiable in front of the background.

Both "retinal" locations $\vec{u}_1, \vec{u}_2$ are combined to a 4-D vector **u** which then carries implicitly the visual information necessary for the network to extract the spatial position of the object. To be able to position its end effector correctly, the network has to know the transformation $\vec{y}(\mathbf{u})$ from "retinal" locations **u** to the required input $\vec{y} = (\vec{y}_1, \vec{y}_2, y_3)$ for the joint angle controllers which generate joint angles $\vec{\theta} = [\theta_1(y_1), \theta_2(y_2), \theta_3(y_3)]$ to lead the end effector to the desired target location. This transformation depends on the position of the cameras relative to the workspace, the cameras' imaging characteristics, the relation between the joint angle controllers' input and the resulting joint angles, and the geometry of the arm. None of these properties is known *a priori* and the neural network has to adapt to them in the course of learning.

The 4-D vector **u** of the "retinal" locations forms the input signal for a neural network which consists of neural units arranged within a 3-D grid (see Fig. 2). To produce the appropriate output values $\vec{y}$, the 3-D grid of neural units **r** works as a "winner-take-all" network. Each neural unit **r** is responsible for a small subset $F_\mathbf{r}$ (its "receptive field") of the 4-D input space U. The input space U consists of all possible "retinal" coordinates **u**. If $\mathbf{u} \in F_\mathbf{r}$, neural unit **r** wins and determines the output $\vec{y}$. To specify the subsets $F_\mathbf{r}$, a vector $\mathbf{w}_\mathbf{r} \in U$ which can be interpreted as synaptic weights are assigned to each neural unit **r**. The vectors $\mathbf{w}_\mathbf{r}$ are chosen as pairs $\mathbf{w}_\mathbf{r} = (\vec{w}_{\mathbf{r}1}, \vec{w}_{\mathbf{r}2})$ of two component vectors $\vec{w}_{\mathbf{r}1}, \vec{w}_{\mathbf{r}2}$. Vector $\vec{w}_{\mathbf{r}i}$ is a 2-D location on the "retina" of camera $i, i = 1, 2$. Therefore, each neural unit is "binocular" and "looks" essentially at two small spots centered at $\vec{w}_{\mathbf{r}1}$ and $\vec{w}_{\mathbf{r}2}$ on the two camera "retinas". The subset $F_\mathbf{s} \subseteq U$ for which neural unit **s** is responsible consists of all input signals $\mathbf{u}$ which are closer to $\mathbf{w}_s$ than to any other $\mathbf{w}_r$, $\mathbf{r} \neq \mathbf{s}$ and can be described mathematically by:

$$F_s = \{\mathbf{u} \in U \mid \|\mathbf{w}_s - \mathbf{u}\| \leqslant \|\mathbf{w}_r - \mathbf{u}\| \quad \forall \mathbf{r}\}. \tag{1}$$

To enable the neural network to provide the required output $\hat{y}$ for the joint angle controllers, in addition to $\mathbf{w}_r$ we assign to each neural unit $\mathbf{r}$ a vector $\hat{y}_r$ and a 3×4 matrix $\mathbf{A}_r$. The neural unit $\mathbf{s}$ which is responsible for the current input $\mathbf{u}$ produces the output by using its $\hat{y}_s$, $\mathbf{A}_s$ as the first two terms of a Taylor expansion of the transformation $\hat{y}(\mathbf{u})$, which yields:

$$\hat{y} = \hat{y}_s + \mathbf{A}_s(\mathbf{u} - \mathbf{w}_s). \tag{2}$$

The vector $\hat{y}_s$ and the matrix $\mathbf{A}_s$ have to be learned by the neural net and at the end of the course of learning have to approximate the exact transformation $\hat{y}(\mathbf{u})$ over the small subset F_s of input signals $\mathbf{u}$ for which neural unit $\mathbf{s}$ is responsible. Initially, all $\mathbf{w}_r$s, $\hat{y}_r$s and $\mathbf{A}_r$s are chosen randomly. The learning task of the neural network is to iteratively adjust the $\mathbf{w}_r$s, $\hat{y}_r$s and $\mathbf{A}_r$s in such a way that the control law $\hat{y}(\mathbf{u})$ is approximated as accurately as possible.

2.1. *The learning procedure for the receptive fields*

First, the learning algorithm has to efficiently distribute the receptive fields F_r over the input space U. Since the workspace from which the target locations are chosen is 3-D, input signals $\mathbf{u}$ only appear within a 3-D submanifold W of the 4-D input space U. Therefore, the learning procedure should assign the formal synaptic weights $\mathbf{w}_r$ which determine the receptive fields F_r only to this relevant subpart of U. The location of the submanifold W depends on the position of the cameras relative to the workspace and the cameras' imaging characteristics which are both not known *a priori*. To find W and to assign the receptive fields F_r only to this submanifold we employ Kohonen's algorithm for the formation of topology conserving feature maps for the adjustment of the $\mathbf{w}_r$s [9,10].

The adaptation step for the $\mathbf{w}_r$s, which is performed each time a target object is presented, is determined by:

$$\mathbf{w}_r^{\text{new}} = \mathbf{w}_r^{\text{old}} + \epsilon h_{rs}(\mathbf{u} - \mathbf{w}_r^{\text{old}}) \quad \text{for all } \mathbf{r}, \tag{3}$$

with $\mathbf{s}$ denoting the neural unit which had its $\mathbf{w}_s$ closest to the input signal $\mathbf{u}$. Essential here is a topological arrangement of the neural units. Each neural unit $\mathbf{r}$ occupies a position $\mathbf{r}$ in the 3-D lattice, and the coefficient h_{rs} is taken to be a unimodal function of Gaussian shape, depending on the lattice distance $\|\mathbf{r} - \mathbf{s}\|$ and with a maximum at $\mathbf{r} = \mathbf{s}$ (to remove the ambiguity in the scaling of ϵ, we require the normalization $h_{ss} = 1$). Hence, neighboring neural units in the lattice share adaptation steps and become tuned to similar inputs $\mathbf{u}$. Neural units which are adjacent in the lattice will assign their receptive fields, and centers of which are determined by the $\mathbf{w}_r$s, to locations which are close within the space of input signals U.

In addition to forming the topology conserving mapping, the adaptation step (3) will assign the $\mathbf{w}_r$s only to parts of the input space where input signals occurred. This ensures that the receptive fields F_r will be distributed only over the relevant 3-D submanifold W of the 4-D input space U. The process of finding the relevant submanifold W is illustrated schematically in Fig. 3. By presenting an input signal $\mathbf{u}$ within the submanifold W the neural units move their $\mathbf{w}_r$ towards $\mathbf{u}$, with the result that after several adaptation steps (3) all the $\mathbf{w}_r$s are located only within the relevant submanifold W.

Related to the property of assigning the $\mathbf{w}_r$s only to relevant parts of the input space, it can be shown that the size of the subsets F_r decreases with an increasing density of input stimuli [10,22]. Hence, in areas of the input space where input signals occur more frequently the density of receptive field centers $\mathbf{w}_r$ is higher, which results in a higher resolution of the representation of the input space in these regions. This property causes the network to learn most accurately those positioning movements which were practiced most frequently.

2.2. *The learning procedure for the output values*

After neural unit $\mathbf{s}$, which is responsible for the current target location, has been selected, the positioning of the end effector is carried out in two phases: (i) a gross-positioning movement; and

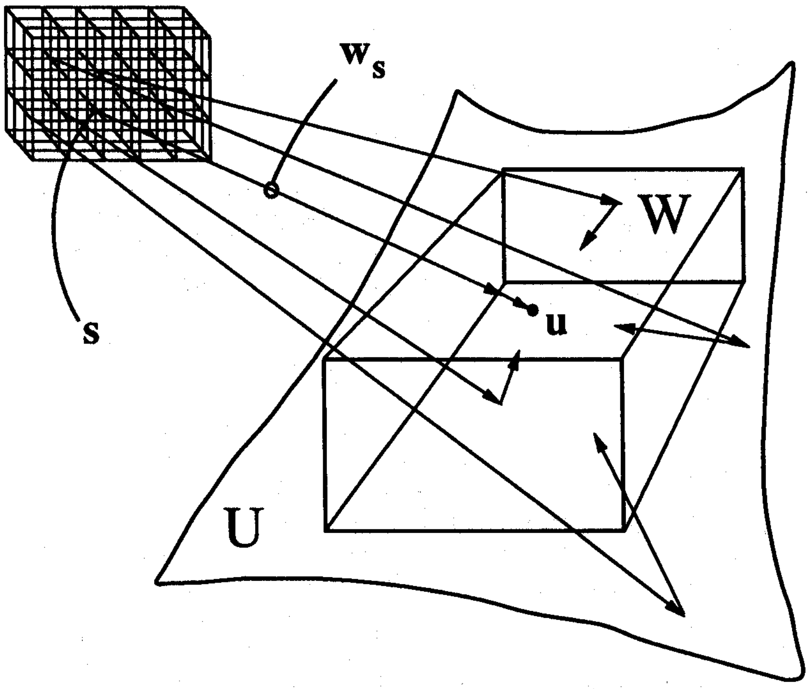


Fig. 3. The 3-D neural net assigning its neural units to the relevant 3-D submanifold W of the 4-D input space U. The neural unit $\mathbf{s}$ and its neighbors in the lattice move the centers $\mathbf{w_r}$ of their receptive fields towards the location $\mathbf{u}$ of the presented target object.

(ii) a subsequent fine-movement which is usually only a small correction. For the gross-positioning movement the network employs the linear expansion (2) of $\hat{y}(\mathbf{u})$ around $\mathbf{w_s}$ to generate:

$$\hat{y}_i = \hat{y}_s + \mathbf{A_s}(\mathbf{u} - \mathbf{w_s}), \tag{4}$$

as input for the joint angle controllers. The position of the end effector on both camera "retinas" after this gross-movement is denoted by the 4-D vector $\mathbf{v}_i$ which is usually already close to the 4-D "retinal" target location $\mathbf{u}$. Then, a subsequent fine-movement is carried out. This fine-movement is based on the visual feedback given through $\mathbf{v}_i$ and employs the Jacobian matrix $\mathbf{A_s}$ to reduce the residual error $\mathbf{u} - \mathbf{v}_i$ by adjusting the network's output according to the linear correction:

$$\Delta\hat{y} = \mathbf{A_s}(\mathbf{u} - \mathbf{v}_i). \tag{5}$$

The correction $\Delta\hat{y} = \hat{y}_f - \hat{y}_i$ with $\hat{y}_f$ as the final output is usually sufficient to compensate very precisely for the error of the gross-movement. A further decrement of the residual positioning error can be achieved by repeating the corrective fine movement (5) several times.

The result of the corrective fine-movement is used to obtain an improved estimate $\mathbf{A}^*$ for the Jacobian matrix $\mathbf{A_s}$. The improved estimate $\mathbf{A}^*$ is determined via a Widrow–Hoff-type error correction rule [23] which minimizes the quadratic error:

$$E = \tfrac{1}{2}(\Delta\hat{y} - \mathbf{A_s}\,\Delta\mathbf{v})^2, \tag{6}$$

by using steepest descent, with $\Delta\mathbf{v} = \mathbf{v}_f - \mathbf{v}_i$ and $\mathbf{v}_f$ as the final position of the end effector seen by cameras 1 and 2. Together with (5) we obtain for $\mathbf{A}^*$:

$$\mathbf{A}^* = \mathbf{A_s} + \delta \cdot \mathbf{A_s}(\mathbf{u} - \mathbf{v}_f)\,\Delta\mathbf{v}^{\mathrm{T}}, \tag{7}$$

with δ as the step size of the adaptation step (7). For δ we choose its optimal value $\delta = 1/\|\Delta\mathbf{v}\|$.

If we denote by $\mathbf{A}(\mathbf{w_s})$ the correct Jacobian associated with $\mathbf{w_s}$ and by $\hat{y}(\mathbf{w_s})$ the correct zero-order term, then the relation $\hat{y}(\mathbf{w_s}) - \hat{y}_i = \mathbf{A}(\mathbf{w_s})(\mathbf{w_s} - \mathbf{v}_i)$ is valid (to linear order). This yields an expression for $\hat{y}(\mathbf{w_s})$ which at the same time determines an improved estimate $\hat{y}^*$ for the adaptation of the zero-order terms $\vec{y}_s$. Since after a number of learning steps (7) $\mathbf{A}(\mathbf{w}_s) \approx \mathbf{A}_s$, we obtain together with (4) and (5):

$$\begin{aligned}\vec{y}^* &= \vec{y}_s + \Delta\vec{y} \\ &= \vec{y}_s + \vec{y}_f - \vec{y}_i. \end{aligned} \tag{8}$$

The improved estimates $\vec{y}^*$ and $\mathbf{A}^*$ are not only used to improve the output values $\vec{y}_s$ and $\mathbf{A}_s$ of the selected neural unit **s**, but also to adjust the output values of the neural units neighboring **s** within the 3-D lattice. For this purpose we apply an adaptation step of the same form we employed for the adjustment of the weights $\mathbf{w}_r$, yielding:

$$\left.\begin{aligned}\vec{y}_r^{\text{new}} &= \vec{y}_r^{\text{old}} + \epsilon' h'_{rs}(\vec{y}^* - y_r^{\text{old}}) \\ \mathbf{A}_r^{\text{new}} &= \mathbf{A}_r^{\text{old}} + \epsilon' h'_{rs}(\mathbf{A}^* - \mathbf{A}_r^{\text{old}})\end{aligned}\right\} \text{ for all } \mathbf{r}, \tag{9}$$

with h'_{rs} as Gaussian determining the spatial decrease of the learning step within the 3-D lattice and ϵ' as a scaling factor for the overall size of the adaptation steps.

The application of an adaptation step of the form used in Kohonen's algorithm for the formation of topology conserving maps forces the output values of neural units which are adjacent within the 3-D grid [which, because of (3), also means adjacent within the input space] to adapt to values which are close within the output space. The output space consists of all 3-D vectors $\vec{y}$ and all 3×4 matrices **A**, respectively. This topologically correct mapping also with respect to the output space enforces a smoothness constraint on the output values, which is desirable, since we know that the transformation $\vec{y}(\mathbf{u})$ which has to be learned is continuous. The "spreading" of the adjustment of $\vec{y}_s$ and $\mathbf{A}_s$ to all neural units neighboring **s** provides an enormous increase in the speed of convergence of the learning algorithm and also results in a higher stability of the adaptation process. In a subsequent section we will show that the cooperation between neighboring neural units, introduced by (9), is even essential for the neural network to succeed in learning at all.

As we can see, the adaptation rules (7), (8) and (9) do not require any explicit information about joint angles at any time during the positioning movement. Information provided only by the cameras is sufficient for the presented adaptation process, which makes the described neural network algorithm applicable to robot arms which lack sensory systems for the measurement of joint angles.

3. RESULTS OF A SIMULATION

In the following, we describe a simulation of the learning process of the robot system. During this simulation, we chose target locations from a workspace the size of which was $1.0 \times 0.57 \times 0.33$ units. The robot arm segments, beginning at the base, had the lengths 0.14×0.44 and 0.47 units, respectively, and the lattice of neural units consisted of $7 \times 12 \times 4$ elements. The function h_{rs} was taken to be the Gaussian:

$$h_{rs} = \exp[-\|\mathbf{r} - \mathbf{s}\|^2/2\sigma^2(t)], \tag{10}$$

as was h'_{rs}. The step sizes ϵ, ϵ' and the widths σ, σ' all had the same time dependence $g(t) = g_i(g_f/g_i)^{t/t_{\max}}$, with t as the number of already performed learning steps and $t_{\max} = 10{,}000$. The parameter values were chosen as $\epsilon_i = 1$, $\epsilon_f = 0.005$, $\epsilon'_i = 0.9$, $\epsilon'_f = 0.5$, $\sigma_i = 3$, $\sigma_f = 0.1$, $\sigma'_i = 2$ and $\sigma'_f = 0.05$. For the relation $\vec{\theta}(\vec{u}) = [\theta_1(y_1), \theta_2(y_2), \theta_3(y_3)]$ between the input $\vec{y}$ for the joint angle controllers and the resulting joint angles $\vec{\theta}$, we chose $\vec{\theta} = \vec{y}$. The only necessary condition for $\vec{\theta}(\vec{y})$ is the existence of its inverse. As long as $\vec{\theta}(\vec{y})$ is smooth enough, the positioning accuracy obtained does not depend on the exact form of $\vec{\theta}(y)$.

Figures 4–6 show the result of the simulation. Figure 4 presents the state of the mapping $\mathbf{r} \mapsto \mathbf{w}_r$ initially, after 2000, and after 6000 learning steps, respectively. We show the state after 6000 learning steps as the final result, since the residual positioning error at this stage of the learning procedure has already decreased to its minimal value and the network does not change significantly anymore.

Each node **r** of the lattice is mapped to a location $\vec{w}_{r1}$ on the "retina" of camera 1 (left column) and a location $\vec{w}_{r2}$ on the "retina" of camera 2 (right column) with $\mathbf{w}_r = (\vec{w}_{r1}, \vec{w}_{r2})$. Locations associated with lattice neighbors are connected by lines to visualize the lattice topology. Initially,

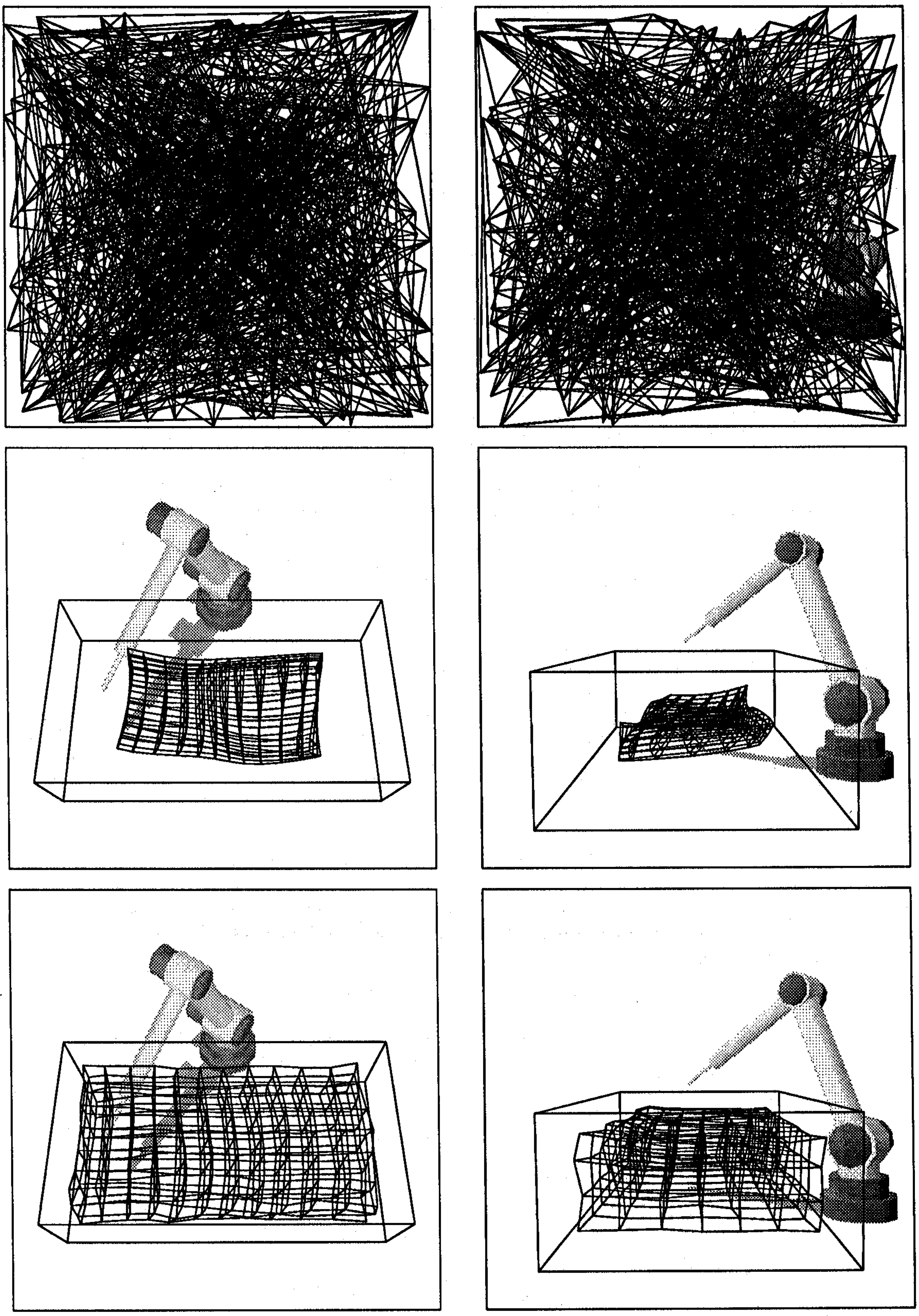


Fig. 4. The "retinal" locations $\mathbf{w}_r = (\vec{w}_{r1}, \vec{w}_{r2})$ projected on the focal plane of camera 1 (left column) and camera 2 (right column). Depicted are the initial state (top), the lattice after 2000 (center) and after 6000 adaptation steps (bottom).

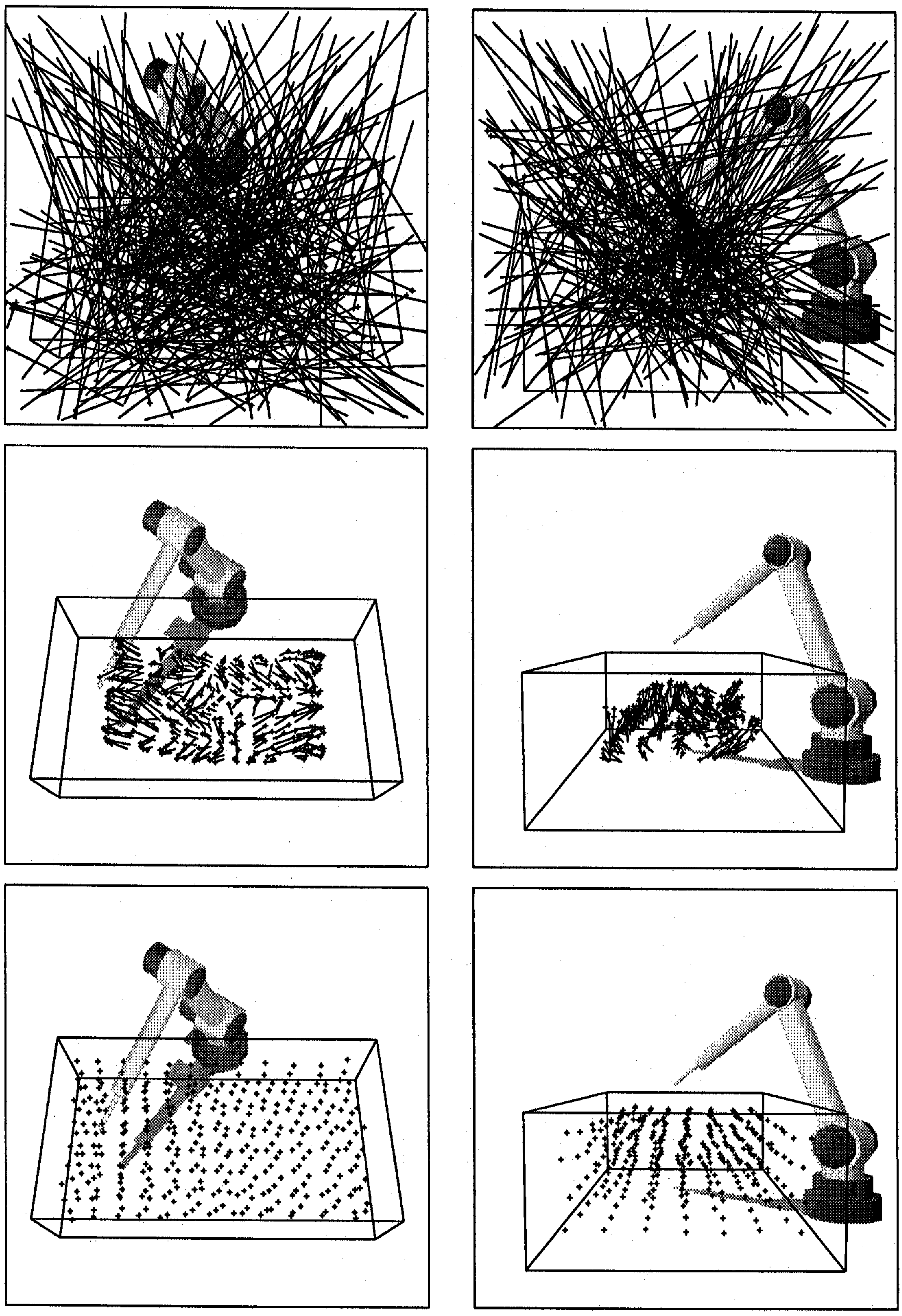


Fig. 5. The end effector locations (cross marks) generated by $\vec{y}_r$, as seen from camera 1 (left column) and camera 2 (right column). Their deviations from the required values $\vec{w}_{r1}$ and $\vec{w}_{r2}$, respectively, are indicated by appended lines. Depicted are the initial state (top), the lattice after 2000 (center) and after 6000 adaptation steps (bottom).

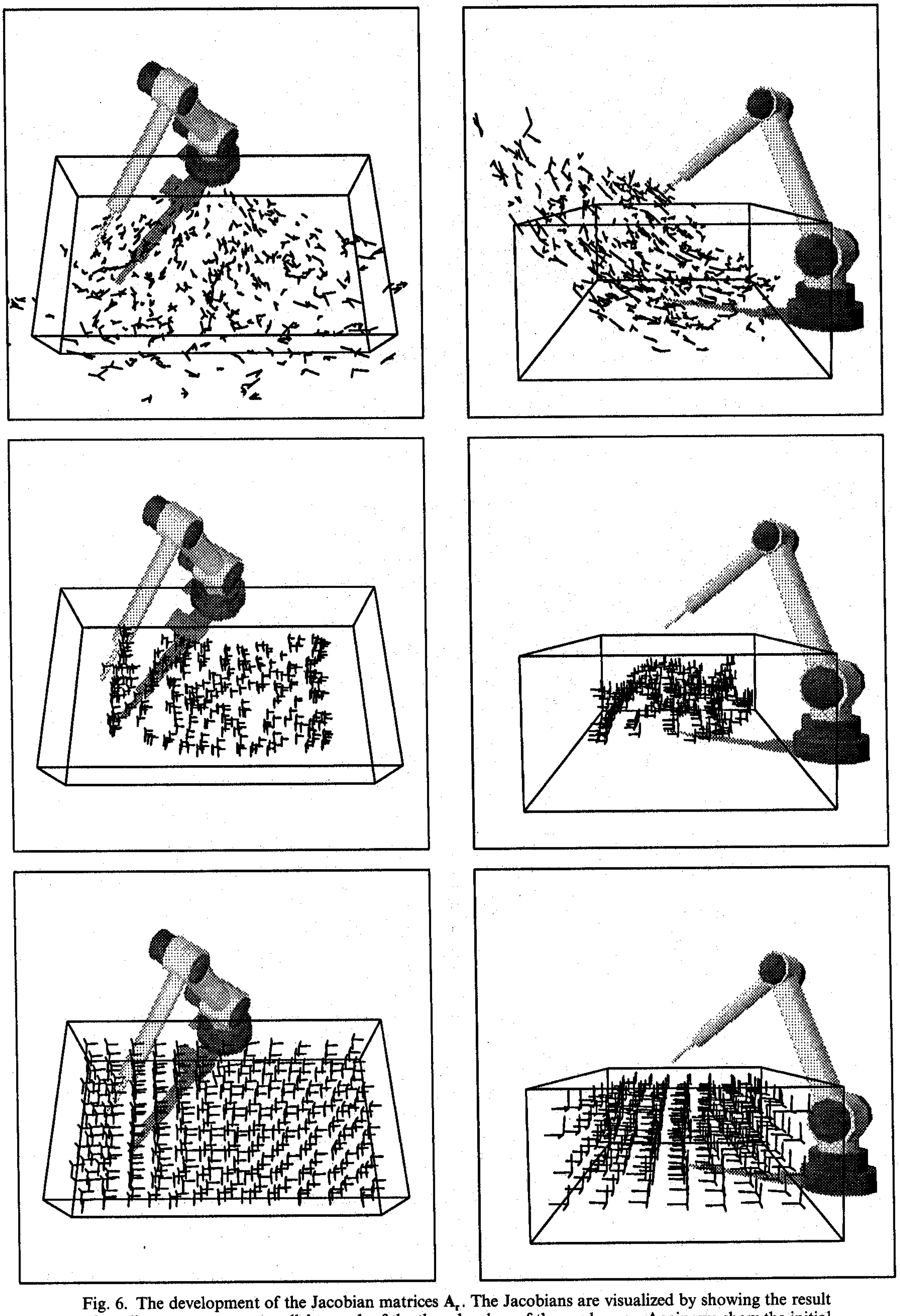


Fig. 6. The development of the Jacobian matrices $\mathbf{A}_r$. The Jacobians are visualized by showing the result of small test movements parallel to each of the three borders of the workspace. Again, we show the initial state (top), the lattice after 2000 (center) and after 6000 adaptation steps (bottom).

the vectors $\tilde{w}_{r1}$ and $\tilde{w}_{r2}$ are distributed randomly in the image plane of their camera. This provides a homogeneous random distribution of the $\mathbf{w}_r$s over the 4-D input space, and the corresponding image of the lattice is highly irregular (top). After only 2000 learning steps the initial distribution has retracted to the relevant 3-D subspace which corresponds to the workspace (center). Finally (bottom), a very regular distribution of the nodes has emerged, indicating a good representation of the workspace through the $\mathbf{w}_r$s.

To visualize the adaptation process of the zero-order terms $\tilde{y}_r$, we show in Fig. 5 the mismatches between intended target positions and actually achieved end effector locations which occur with the gross-movement (4) for the special subset of visual inputs $\mathbf{u} = \mathbf{w}_r$. Each end effector position, after setting the joint angles through $\tilde{y}_r$ [the linear term $\mathbf{A}_s(\mathbf{u} - \mathbf{w}_s)$ in (4) vanishes for $\mathbf{u} = \mathbf{w}_s$], is indicated by a cross mark, and the associated positioning error of the end effector is indicated by an appended line segment. The initial values of $\tilde{y}_r$ are chosen randomly (with the only restriction being that the resulting end effector positions should lie in the space in front of the robot) and consequently the errors are very large for the initial state (top). However, after 2000 learning steps all errors have markedly decreased (center), until finally (6000 steps, bottom) mismatches are no longer visible.

The special subset of target locations $\mathbf{u} = \mathbf{w}_r$ was chosen to visualize the accuracy of the zero-order terms $\tilde{y}_r$ of the Taylor expansion (2). In general, during operation, the target objects may be located anywhere in the work space and need not coincide with one of the discretization points $\mathbf{w}_r$. The deviation from the closest discretization point $\mathbf{w}_s$ is taken into account by the first-order term $\mathbf{A}_s(\mathbf{u} - \mathbf{w}_s)$ of the Taylor expansion (2). As the Jacobian matrices $\mathbf{A}_r$ cannot easily be visualized directly, we instead show for each location $\tilde{y}_r$ the reaction of the end effector to three pairwise orthogonal test movements. These test movements are of equal length and directed parallel to the borders of the workspace. If the $\mathbf{A}_r$s are correct, the end effector will trace out patterns in the shape of little "tripods", thereby testing each $\mathbf{A}_r$ along the three orthogonal space directions. The gradual convergence of these three test movements, as seen from both cameras, is shown in Fig. 6. The Jacobian matrices are initialized by assigning a random value from the interval $[-10,10]$ independently to each element of $\mathbf{A}_r$. Therefore, the initial test movements are very poor (top).

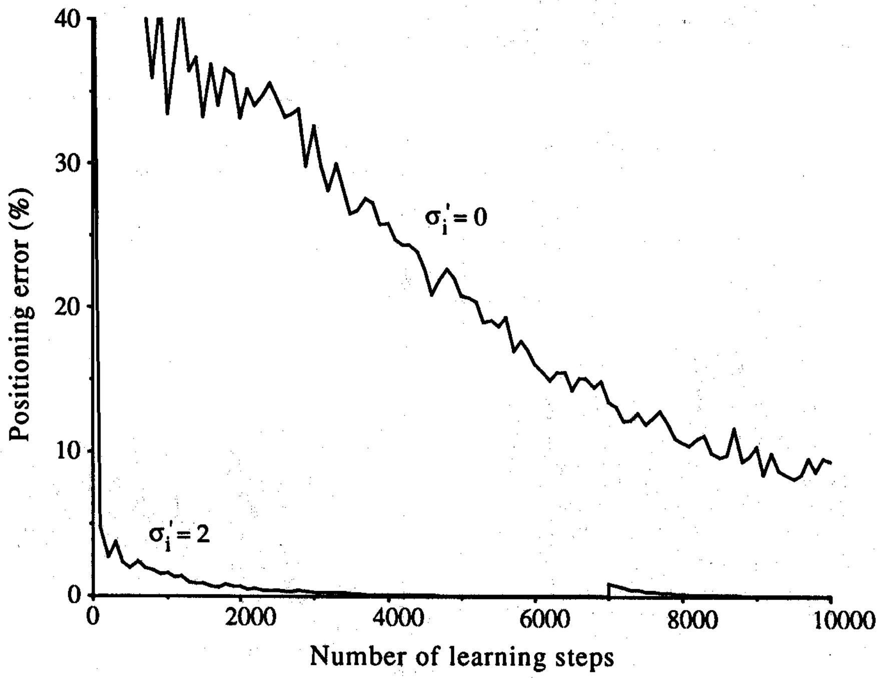


Fig. 7. The average positioning error vs the number of performed learning steps. σ_i' denotes the initial range of cooperation between neural units. With cooperation ($\sigma_i' = 2$), the positioning error decreases very rapidly against zero, whereas without cooperation ($\sigma_i' = 0$) a significant residual error remains at the end of the learning procedure. After 7000 trials the geometry of the robot arm was changed. For $\sigma_i' = 2$, the robot has regained its previous accuracy of 0.06% after a few hundred additional adaptation steps.

However, after 2000 iterations the accuracy of the test movements has improved significantly (center) and after 6000 learning steps, they are traced out very accurately (bottom).

In Fig. 7 we plotted the average positioning error vs the number of learning steps (the graph corresponding to $\sigma_i' = 2$). The error decreases rapidly in the beginning and reaches 5% of the length of the workspace already after 100 trial movements. After 6000 iterations, the positioning error has decreased to its final residual value of 0.06%. If the lengths of the arm segments of the robot were given in meters, the robot arm system would have accomplished a positioning accuracy of several tenths of a millimeter.

After 6000 learning steps the robot has learned the required task and is able to perform accurate positioning movements. However, during performance, the measure of the joint angles, the limbs and the positions of the cameras may become miscalibrated. Therefore, the neural network must be able to permanently readjust its weights $\mathbf{w}_r$, $\hat{y}_r$ and $\mathbf{A}_r$. To demonstrate this capability of the neural network, after 7000 trials we increased the length of the last arm segment by about 10% of the robot's dimensions, a situation which could result from connecting different tools to a robot's end effector. As we see from Fig. 7, the increase in the positioning error caused by this perturbation decreases with further trial movements until the robot has regained its previous accuracy. As we can also see in Fig. 7, the 10% change in the length of the last arm segment caused an increase in the positioning error of only 1.2%. The error increase of only 1.2 instead of 10% is due to the feedback guided fine-movement which tries to reduce the actual distance $\mathbf{u} - \mathbf{v}_i$ remaining after the preceding gross-movement. This corrective fine-movement enables the robot system to compensate immediately for sudden and unexpected changes in the geometry of the arm.

The described simulation was performed on a Silicon Graphics 4D/320 VGX. The CPU time needed for the whole training, with 10,000 trial movements, was about 300 s.

4. COOPERATION BETWEEN NEURAL UNITS AS REQUIREMENT FOR LEARNING

In Fig. 7 we also show the development of the positioning error in the case that the range σ' of the neighborhood function h'_{rs} is equal to zero, which corresponds to $h'_{rs} = \delta_{rs}$ with δ_{rs} as the *Kronecker-delta*. For $\sigma' = 0$ the cooperation between adjacent neural units introduced by (9) is "switched off", since now only the weights of neural unit $\mathbf{s}$ which was selected for providing the output are adjusted. As we see in Fig. 7, in the noncooperating case the robot arm system is not able to reduce the positioning error to a reasonable degree. The residual error for $\sigma' = 0$ was still about 10% of the length of the workspace at the end of the course of learning.

As has been demonstrated in Refs [2,13], for the noncooperating case, only a subset of the neural units is able to adapt their output $\hat{y}_r$, $\mathbf{A}_r$ to the required values, whereas all the other units which happened to be initialized with less appropriate starting values do not converge properly. We will show mathematically for the 1-D version of the presented algorithm, that by "switching on" the cooperation, i.e. by setting $\sigma' > 0$, the neural units which did not converge without cooperation are enabled to adjust their output values properly as well. We will show that, with cooperation, the probability of the neural net to learn as desired increases exponentially to one with the size of the network.

For our mathematical analysis of the behavior of the learning process we look at a small subset of adjacent lattice nodes (neural units). Since the transformation $\hat{y}(\mathbf{u})$ which the network has to learn is continuous, we can assume that within a sufficiently small subset of adjacent nodes $\mathbf{r}$, the Jacobians $\mathbf{A}_r$ all have to adapt to the same matrix $\mathbf{A}_0$. The required vectors $\hat{y}_r^0$ for the zero order terms $\hat{y}_r$, however, may still differ from unit to unit in the following analysis. We also assume that the receptive fields determined by $\mathbf{w}_r$ are already learned properly and do not change anymore. Then, we only have to examine the adaptation of the output elements $\hat{y}_r$ and $\mathbf{A}_r$.

By $\hat{y}_r^0 = \hat{y}(\mathbf{w}_r)$ and $\mathbf{A}_0 = \mathbf{A}(\mathbf{w}_r)$ we denote the correct zero-order term and the correct Jacobian matrix associated with each $\mathbf{w}_r$. Since the relation $\hat{y}_s^0 - \hat{y}_i = \mathbf{A}_0(\mathbf{w}_s - \mathbf{v}_i)$ is valid on the small subset of units, the adaptation step (8), together with (4) and (5), leads to $\hat{y}^* = \hat{y}_s^0$ if $\mathbf{A}_s = \mathbf{A}_0$. Therefore, if the selected neural unit $\mathbf{s}$ was able to adapt its Jacobian $\mathbf{A}_s$ to the correct matrix $\mathbf{A}_0$, then its zero order term $\hat{y}_s$ will also converge to the required value. Hence, in the following we only have to consider the learning process of the Jacobians $\mathbf{A}_\mathbf{r}$. As long as they are learned properly, the zero order terms $\tilde{y}_\mathbf{r}$ will also adapt as required.

The adaptation step for the improved estimate $\mathbf{A}^*$ is given by the gradient of (6) and can be written as:

$$\mathbf{A}^* = \mathbf{A}_\mathbf{s} + \delta \cdot (\Delta\tilde{y} - \mathbf{A}_\mathbf{s}\Delta\mathbf{v})\Delta\mathbf{v}^\mathrm{T}, \tag{11}$$

with $\Delta\tilde{y} = \tilde{y}_f - \tilde{y}_i$ as the change in the network's output caused by the fine-movement. Locally, the expression $\Delta\tilde{y} = \mathbf{A}_0\Delta\mathbf{v}$ with $\Delta\mathbf{v} = \mathbf{v}_f - \mathbf{v}_i$ is valid, which yields together with the fine-movement (5):

$$\Delta\mathbf{v} = \mathbf{A}_0^{-1}\mathbf{A}_\mathbf{s}(\mathbf{u} - \mathbf{v}_i), \tag{12}$$

assuming that the inverse of $\mathbf{A}_0$ exists. If in (11) we substitute (12) for $\Delta\mathbf{v}$ and (5) for $\Delta\tilde{y}$, we obtain:

$$\mathbf{A}^* = \mathbf{A}_\mathbf{s} + \delta \cdot \mathbf{A}_\mathbf{s}(\mathbf{1} - \mathbf{A}_0^{-1}\mathbf{A}_\mathbf{s})(\mathbf{u} - \mathbf{v}_i)(\mathbf{u} - \mathbf{v}_i)^\mathrm{T}\mathbf{A}_\mathbf{s}^\mathrm{T}(\mathbf{A}_0^{-1})^\mathrm{T}. \tag{13}$$

Substituting (13) for $\mathbf{A}^*$ in the adaptation step (9), and introducing $\mathbf{B}_\mathbf{r} = \mathbf{A}_0^{-1}\mathbf{A}_\mathbf{r} - \mathbf{1}$ with $\mathbf{1}$ as the identity matrix, yields:

$$\Delta\mathbf{B}_\mathbf{r} = -\epsilon'\delta h'_{\mathbf{rs}}(\mathbf{B}_\mathbf{s} + \mathbf{1})\mathbf{B}_\mathbf{s}(\mathbf{u} - \mathbf{v}_i)(\mathbf{u} - \mathbf{v}_i)^\mathrm{T}(\mathbf{B}_\mathbf{s} + \mathbf{1})^\mathrm{T} + \epsilon' h'_{\mathbf{rs}}(\mathbf{B}_\mathbf{s} - \mathbf{B}_\mathbf{r}), \tag{14}$$

as the change of $\mathbf{B}_\mathbf{r}$ with an adaptation step.

In the following we assume the step size ϵ' to be small which allows us to average the adaptation step (14) over many $\mathbf{u} \in F_\mathbf{s}$ while the matrices $\mathbf{B}_\mathbf{s}, \mathbf{B}_\mathbf{r}$ are quasi-constant. In addition, by keeping ϵ' small $\mathbf{B}_\mathbf{r}$ is changing smoothly in time which may then be described by:

$$\dot{\mathbf{B}}_\mathbf{r} = -\delta h'_{\mathbf{rs}}(\mathbf{B}_\mathbf{s} + \mathbf{1})\mathbf{B}_\mathbf{s}\langle(\mathbf{u} - \mathbf{v}_i)(\mathbf{u} - \mathbf{v}_i)^\mathrm{T}\rangle(\mathbf{B}_\mathbf{s} + \mathbf{1})^\mathrm{T} + h'_{\mathbf{rs}}(\mathbf{B}_\mathbf{s} - \mathbf{B}_\mathbf{r}), \tag{15}$$

with $\langle(\mathbf{u} - \mathbf{v}_i)(\mathbf{u} - \mathbf{v}_i)^\mathrm{T}\rangle$ as the matrix $(\mathbf{u} - \mathbf{v}_i)(\mathbf{u} - \mathbf{v}_i)^\mathrm{T}$ averaged over many target locations $\mathbf{u} \in F_\mathbf{s}$. We presume the directions of the vectors $\mathbf{u} - \mathbf{v}_i$ to be distributed isotropically, which yields $\langle(\mathbf{u} - \mathbf{v}_i)(\mathbf{u} - \mathbf{v}_i)^\mathrm{T}\rangle \propto \mathbf{1}$. If we consider the modification of $\mathbf{B}_\mathbf{r}$ not only for input signals $\mathbf{u} \in F_\mathbf{s}$ but averaged over all possible $\mathbf{u}$, we finally obtain for the time development of $\mathbf{B}_\mathbf{r}$:

$$\dot{\mathbf{B}}_\mathbf{r} = -\delta \cdot \sum_\mathbf{s} h'_{\mathbf{rs}}(\mathbf{B}_\mathbf{s} + \mathbf{1})\mathbf{B}_\mathbf{s}(\mathbf{B}_\mathbf{s} + \mathbf{1})^\mathrm{T} + \sum_\mathbf{s} h'_{\mathbf{rs}}(\mathbf{B}_\mathbf{s} - \mathbf{B}_\mathbf{r}), \tag{16}$$

with $\Sigma_\mathbf{s}$ denoting the sum over all neural units.

We will discuss the time development of $\mathbf{B}_\mathbf{r}$ for two different cases: (i) $\sigma' = 0$ which corresponds to the non-cooperating case with $h'_{\mathbf{rs}} = \delta_{\mathbf{rs}}$ and (ii) $\sigma' > 0$ for which the cooperative learning between the neural units is "switched on". We will limit the discussion to the case where the vectors $\tilde{y}_\mathbf{r}$ are 1-D and the Jacobians $\mathbf{A}_\mathbf{r}$, in the following denoted by $a_\mathbf{r}$, are scalars. We will see that the properties of the 1-D version of the learning algorithm are in qualitative agreement with the behavior observed in Section 3 when simulating the higher-dimensional version of the algorithm. This allows us to presume that, by reducing the dimensionality, the principal behavior of the learning algorithm is left unchanged and, therefore, our mathematical analysis will reflect the main features of the adaptation process.

4.1. Learning without cooperation

For the 1-D case and with $h'_{\mathbf{rs}} = \delta_{\mathbf{rs}}$, equation (16) reduces to:

$$\dot{b}_\mathbf{r} = -\delta \cdot b_\mathbf{r}(b_\mathbf{r} + 1)^2. \tag{17}$$

The time development of each $b_\mathbf{r}$ now corresponds to a gradient descent in a potential of the form:

$$V(b_\mathbf{r}) = \frac{b_\mathbf{r}^4}{4} + \frac{2b_\mathbf{r}^3}{3} + \frac{b_\mathbf{r}^2}{2}, \tag{18}$$

and is equivalent to the instructive picture of a pointlike mass moving overdamped in the potential given by (18). Figure 8 illustrates the form of this potential. We can discern that at $b_\mathbf{r} = 0$ the potential $V(b_\mathbf{r})$ exhibits a global minimum which corresponds to the required final value a_0 for the Jacobian $a_\mathbf{r}$. However, at $b_\mathbf{r} = -1$ the potential $V(b_\mathbf{r})$ has also a local minimum which corresponds to $a_\mathbf{r} = 0$. Since the change of $b_\mathbf{r}$ in this potential is overdamped, neural units with initial values

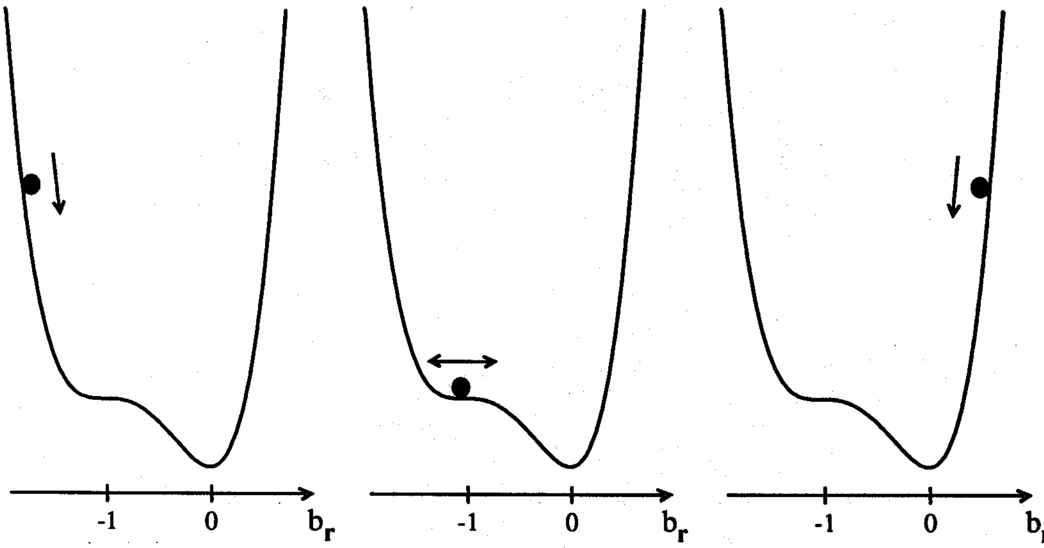


Fig. 8. Illustration of the potential $V(b_r)$ and the convergence for two different initial conditions. The time development corresponds to an overdamped motion of a pointlike mass. For starting values $b_r < -1$ the mass remains at the local minimum which corresponds to the wrong value $a_r = 0$. For initial values $b_r > -1$, however, the output b_r converges to the required value at the global minimum at $b_r = 0$ which corresponds to $a_r = a_0$.

$b_r(t_0) < -1[a_r(t_0) < 0]$ are not able to adapt their output properly since their b_rs remain at the local minimum at $b_r = -1$. Neural units which had their Jacobians initialized within the range $b_r(t_0) > -1[a_r(t_0) > 0]$ will learn properly. Their b_rs are attracted by the global minimum and will finally converge to the required value $b_r = 0$. If $P(a_r)$ describes the probability distribution of the initial values $a_r(t_0)$ and:

$$P_0 = \int_0^\infty P(a)\,\mathrm{d}a, \tag{19}$$

denotes the probability of a neural unit to have its output assigned to positive starting values, the fraction $N \cdot P_0$ of the neural units will converge properly, whereas the rest $N \cdot (1 - P_0)$ of the units are not able to learn the required Jacobians. This result corresponds to the observation that for simulations with $\sigma' = 0$ the output of only a fraction of the neural units adapt as required [2,13].

The above result is valid for $N \to \infty$. For finite Ns there is still a small chance that all the a_rs are initialized with positive starting values, however, this chance decreases exponentially to zero with N. Since the probability of a neural unit to be initialized with $a_r(t_0) > 0$ is P_0, the probability $W(N)$ of a system with N neural units to converge completely to the required state, i.e. each unit adapts its output to $a_r = a_0$, is determined by:

$$W(N) = P_0^N. \tag{20}$$

4.2. Learning with cooperation

If we increase σ' to $\sigma' > 0$ the cooperation between the neural units is "switched on." We will study the case of a small range of neighborhood cooperation and a small step size, i.e. $\sigma' \ll 1$ and $\delta \ll 1$. Then we may write (16) as:

$$\dot{b}_r = -\delta \cdot b_r(b_r + 1)^2 - \delta h' \cdot \sum_{\langle s\rangle} b_s(b_s + 1)^2 - \delta h'^2 \cdot \sum_{\langle\langle s\rangle\rangle} b_s(b_s + 1)^2 - \cdots$$
$$+ h' \cdot \sum_{\langle s\rangle} (b_s - b_r) + h'^2 \cdot \sum_{\langle\langle s\rangle\rangle} (b_s - b_r) + \cdots \tag{21}$$

with $h' = \exp(-1/2\sigma'^2)$ [see (10)] and $\langle s\rangle$ denoting a summation over all nearest neighbors and $\langle\langle s\rangle\rangle$ denoting a summation over all next nearest neighbors of $\mathbf{r}$. Since $\sigma' \ll 1$ and $\delta \ll 1$, we may

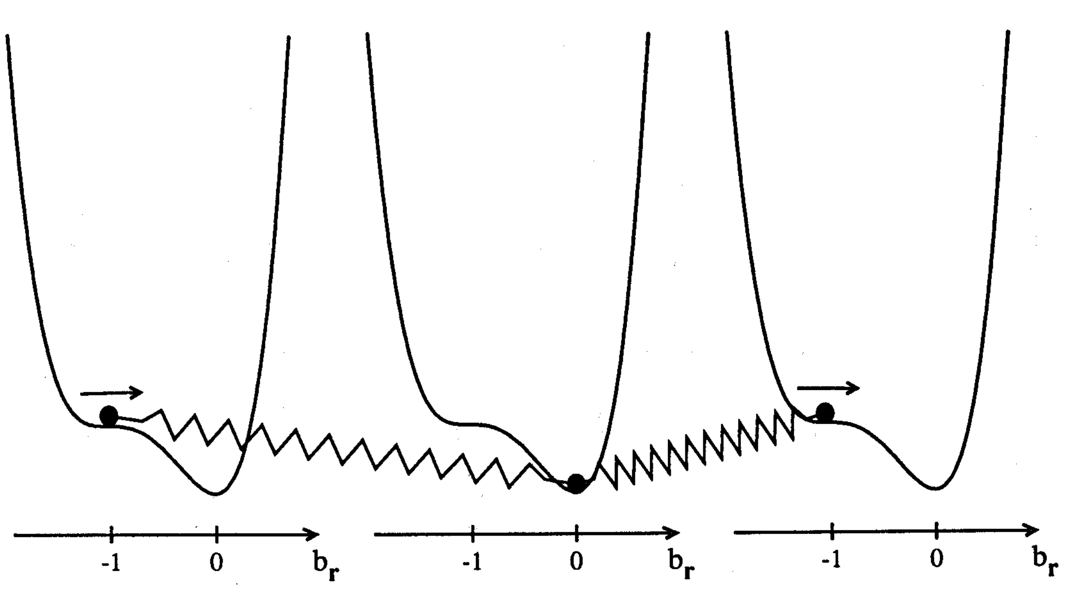


Fig. 9. Illustration of cooperative learning. The term describing the cooperation between neural units is equivalent to a harmonic coupling between the pointlike masses moving in the potential $V(b_r)$. Due to this coupling represented by the springs, point-masses which remain at the local minimum at $b_r = -1$ are pulled or pushed towards the global minimum and, therefore, may reach the required value $b_r = 0$ as well.

neglect the terms in (21) of higher order in h', δ or products of them. This yields for the time development of b_r:

$$\dot{b}_r = -\delta \cdot b_r(b_r + 1)^2 + h' \cdot \sum_{\langle s \rangle} (b_s - b_r), \tag{22}$$

which can be interpreted mathematically as adding the coupling term $h' \cdot \Sigma(b_s - b_r)$ to the time development of b_r without cooperation. The time development of b_r described by (22) corresponds again to an overdamped motion of a pointlike mass in the potential $V(b_r)$, however, the additional harmonic coupling $h' \cdot (b_s - b_r)$ among nearest neighbors in the lattice adds spring-like connections between the point-masses. For three neural units the new situation is illustrated in Fig. 9. Without cooperation each point-mass moved by itself within the potential $V(b_r)$ without any interaction with its neighbors. As depicted in Fig. 9, with the additional spring-like connections, point-masses which "got stuck" at the local minimum at $b_r = -1$ are pulled or pushed towards the desired global minimum by their neighboring point-masses. Hence, one expects the percentage of neural units, which are able to adapt their output to the required values, to increase by the additional coupling between the units.

If we multiply both sides of (22) by b_r and sum the result over all neural units $\mathbf{r}$, we obtain:

$$\sum_r b_r \dot{b}_r = -\delta \cdot \sum_r b_r^2 (b_r + 1)^2 + h' \cdot \sum_r \sum_{\langle s \rangle} (b_s b_r - b_r^2). \tag{23}$$

Denoting $1/N\, \Sigma_r b_r^2$ by $\langle b_r^2 \rangle$, equation (23) is equivalent to:

$$\langle \dot{b_r^2} \rangle = -\frac{2\delta}{N} \sum_r b_r^2 (b_r + 1)^2 - \frac{h'}{N} \sum_r \sum_{\langle s \rangle} (b_s - b_r)^2. \tag{24}$$

Equation (24) shows that the mean square value $\langle b_r^2 \rangle$ never increases since the right-hand side of (24) is never positive. If we look at the phase space which is defined by the N-dimensional space of all possible values of the b_rs, $\langle \dot{b_r^2} \rangle = 0$ is valid only for:

$$b_r = 0 \quad \forall \mathbf{r} \quad \text{and} \quad b_r = -1 \quad \forall \mathbf{r}. \tag{25}$$

At these two points in the phase space, holds $\dot{b}_r = 0$ for each $\mathbf{r}$ and, therefore, (25) describes the only two fixpoints of the dynamics of the adaptation process.

Condition (25) together with (24) allows us to determine the probability $W(N)$ of the network to succeed in learning for the cooperating case, too. For this purpose we assume $\delta \gg h'$. Then we may approximate the time development (22) of the output by neglecting the term $h' \cdot \Sigma_{\langle s \rangle}(b_s - b_r)$ as long as $b_r(b_r + 1)^2$ is not close to zero. Hence, each neural unit first modifies its b_r as in the non-cooperating case determined by (17), until it has adapted its Jacobian either to the local minimum at $b_r = -1$ or to the global minimum at the required value $b_r = 0$. At this point in time, the terms $b_r(b_r + 1)^2$ are zero and the coupling $h' \cdot \Sigma_{\langle s \rangle}(b_s - b_r)$ starts to contribute to the time development of b_r, which forces the whole network to converge to either of the two fixpoints determined by (25).

If we denote by M the number of neural units which have already adapted their Jacobians to the required value $b_r = 0$ at the time when the coupling starts to affect the time development, we obtain:

$$\langle b_r^2 \rangle = \frac{N - M}{N}, \tag{26}$$

for the mean square value $\langle b_r^2 \rangle$ at this point in time, which is smaller than one for $M \neq 0$. Since the mean square value $\langle b_r^2 \rangle$ always decreases until, because of (25), it reaches either $\langle b_r^2 \rangle = 0$ or $\langle b_r^2 \rangle = 1$ as a final value, the neural network converges to the required state, i.e. all the neural units adapt their output to $b_r = 0$, if $M \neq 0$.

The probability of M being zero, i.e. the probability for all the neural units to have adapted their output to the local minimum at $b_r = -1$ when the coupling term starts to contribute to the time development, is equal to the probability of initializing all the a_rs with values $a_r < 0$. The probability of initializing the output a_r of one neural unit with a value $a_r < 0$ is given by $1 - P_0$. Hence, for a network of N cooperating neural units the probability $W(N)$ of adapting completely to the required state, i.e. each unit adapts its output to $a_r = a_0$, is given by:

$$W(N) = 1 - (1 - P_0)^N, \tag{27}$$

which, in contrast to the noncooperating case, converges exponentially to one with an increasing number of neural units N.

In Fig. 10 we compare the analytical results for $W(N)$ with the results obtained through Monte-Carlo simulations of the learning process. For the comparison we simulated the adaptation

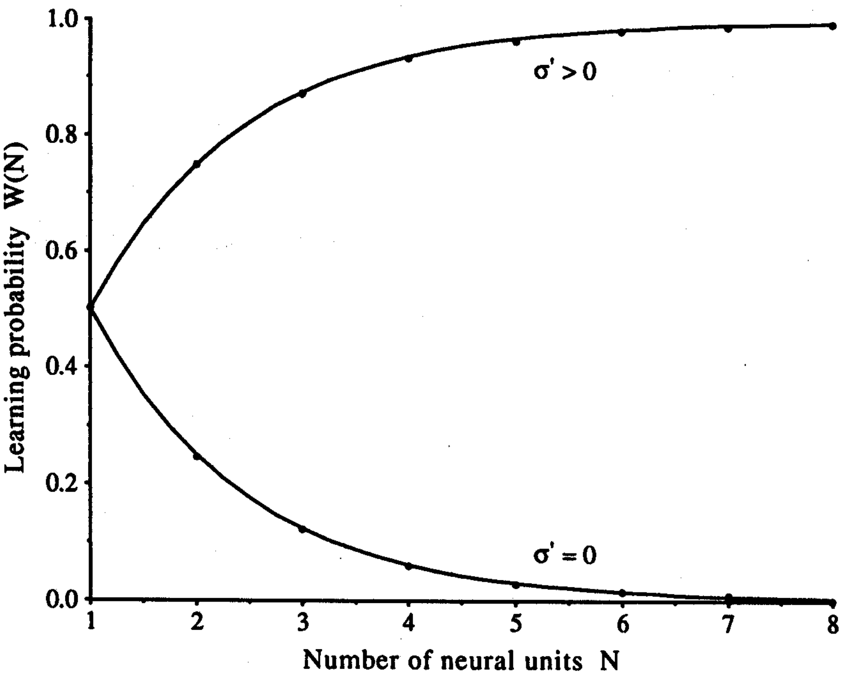


Fig. 10. The probability $W(N)$ of a neural net of N units to completely succeed in learning, i.e. the output a_r of each neural unit converges to $a_r = a_0$. The dot marks show the result obtained through a Monte-Carlo simulation of the learning process, and the two graphs show the result of our mathematical analysis. The predicted exponential decay of $W(N)$ for $\sigma' = 0$ and exponential increase for $\sigma' > 0$ is in very good agreement with the actual behavior of the learning process.

steps (7), (8) and (9) for $a\tilde{y}_r$ one-dimensional and with $\mathbf{A}_r$ being a scalar. As in our mathematical analysis, the $\mathbf{w}_r$s were fixed and each a_r had to adapt to the same a_0. The N neural units were arranged in a linear chain and the range σ' of the neighborhood cooperation as well as the step sizes δ and ϵ' were kept constant during the simulation. For a_0, σ', δ and ϵ' we chose the values $a_0 = 1$, $\sigma' = 0.2$, $\delta = 0.1$ and $\epsilon' = 0.5$. For the initialization we assigned to each a_r a random value from the interval $[-2,2]$, which yields $P_0 = 0.5$.

As we can see in Fig. 10, the analytical results and the results of the Monte-Carlo simulations are in very good agreement. For $\sigma' = 0$ the probability $W(N)$ decreases exponentially to zero with the size N of the network, whereas for $\sigma' > 0$, i.e., for $\sigma' = 0.2$ in the simulations, the probability of the network succeeding in learning increases exponentially to one with its size. This shows very clearly the significant influence of the cooperation between neural units on the learning process. For a neural network consisting of a large number of units the cooperation is essential for the system to succeed in learning.

5. SUMMARY

We described a self-organizing neural network model based on Kohonen's feature map which is capable of learning input–output relations, i.e. the input–output transformation from camera coordinates of a visually designated target object to appropriate joint angles for the positioning of a robot arm's end effector. The described robot system, together with the neural network which consists of neural units arranged in a 3-D lattice, learns to position its end effector to target objects by performing a number of trial movements. Each trial movement consists of an open-loop gross positioning and a subsequent corrective fine-movement which is generated by employing visual feedback. The learning of the end effector positioning is achieved without the need for an external teacher. For the adaptation of the network only information provided by a pair of cameras is necessary, which makes the network algorithm applicable to robot arms without joint angle sensors. The positioning error of the robot arm decreases very rapidly within the first few hundred trials and in the described simulation reaches a residual final value of 0.06% of the length of the workspace after 6000 learning steps.

By increasing the length of one of the robot arm's segments at a final stage of the learning procedure we tested the capability of the neural network to adapt to changes in the robot system's geometry, which may result from worn-out joints, picking up, e.g. a tool, or changes in the camera positions. After a slight immediate increase in the robot system's positioning error the accuracy returned to its previous value through an additional number of trial movements. The slight increase in the positioning error of 1.2% was much less than the additional error which one would expect from a 10% change in the length of the robot arm's last segment. This discrepancy is due to a visual feedback mechanism incorporated in the control of the positioning movement which enables the robot system to react to changes in its configuration immediately. After several further adaptation steps the network had readjusted the quantities $\mathbf{w}_r$, $\tilde{y}_r$, $\mathbf{A}_r$ and the previous accuracy was completely regained.

We investigated the cooperative learning of the output values which is incorporated in the network through an adaptation step which adjusts the output of adjacent neural units in a concerted way. The concerted modification forces neighboring neural units to adapt to similar output values, which is desirable, since the *a priori* unknown input–output transformation is continuous. The concerted adjustment of the output values not only yields a significant increase in the speed of convergence, but, as the result of a simulation and a mathematical analysis showed, is even necessary for the network to succeed in learning at all.

A mathematical analysis and Monte-Carlo simulations of the 1-D version of the learning procedure showed that cooperation in the adjustment of the output values causes the probability of the network to learn properly to increase to one with the number of employed neural units. Without cooperation only a subset of the neural units, namely those which happened to be initialized properly, are able to adapt their output to the required values. The behavior of the learning algorithm in the non-cooperating as well as in the cooperating version could be illustrated by pointing out the mechanical analogy to an overdamped motion of a pointlike mass in a potential with a local and a global minimum. In this analogy cooperation between neural units corresponds to a harmonic coupling, i.e. to connecting the pointlike masses by springs, which causes masses which "got stuck" at the local minimum to be "pushed" or "pulled" towards the global minimum, until they reach their required final values as well.

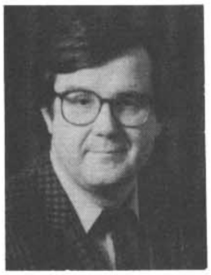
The significant effect of the cooperative learning in the described neural network suggests the assumption that the utilization of clusters of cortical neurons dedicated to the same task not only serves the purpose of robustness against failures of single units, but eventually is even necessary for neural systems to exhibit learning capabilities at all. This interesting aspect of supporting learning through lateral interaction between computing elements might have a valuable impact on distributed processes in general.

Acknowledgements—The authors would like to thank Helge Ritter for many valuable discussions. This work has been supported by the National Science Foundation under Grant DIR-90-15561, by the German Ministry of Science and Technology (ITR-8800-G9) and by a fellowship from the Volkswagen Foundation to TM.

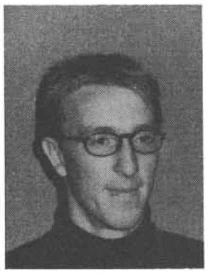
REFERENCES

1. T. Martinetz, H. Ritter and K. Schulten, Three-dimensional neural net for learning visuomotor-coordination of a robot arm. *IEEE-Trans. Neural Networks* **1,** 131–136 (1990).
2. H. Ritter, T. Martinetz and K. Schulten, Topology conserving maps for learning visuomotor-coordination. *Neural Networks* **2,** 159–168 (1989).
3. D. Whitteridge, Projection of optic pathways to the visual cortex. In *Visual Centers in the Brain* (R. Jung, Ed.). Springer, Berlin. *Handbook of Sensory Physiology*, Vol. VII/3B, pp. 247–268 (1973).
4. J. H. Kaas, R. J. Nelson, M. Sur, C. S. Lin and M. M. Merzenich, Multiple representations of the body within the primary somatosensory cortex of primates. *Science* **204,** 521–523.
5. J. H. Kaas, M. M. Merzenich and H. P. Killackey (1983) The reorganization of somatosensory cortex following peripheral nerve damage in adult and developing mammals. *Annl Rev. Neurosci.* **6,** 325–356 (1983).
6. D. J. Willshaw and C. von der Malsburg, How patterned neural connections can be set up by self-organization. *Proc. R. Soc. Lond.* **B-194,** 431–445 (1976).
7. C. von der Malsburg and D. J. Willshaw, How to label nerve cells so that they can interconnect in an ordered fashion. *Proc. Natl Acad. Sci. U.S.A.* **74,** 5176–5178 (1977).
8. A. Takeuchi and S. Amari, Formation of topographic maps and columnar microstructures. *Biol. Cybernet.* **35,** 63–72 (1979).
9. T. Kohonen, Self-organized formation of topologically correct feature maps. *Biol. Cybernet.* **43,** 59–69 (1982).
10. T. Kohonen, Self-organization and associative memory. *Springer Series in Information Sciences 8.* Heidelberg (1984).
11. K. J. Overton and M. A. Arbib, The branch arrow model of the formation of retino-tectal connections. *Biol. Cybernet.* **45,** 157–175 (1982).
12. H. Ritter and K. Schulten, Topology conserving mappings for learning motor tasks. In *Neural Networks for Computing* (J. S. Denker, Ed.), *AIP Conf. Proc. 151*, Snowbird, Utah, pp. 376–380 (1986).
13. H. Ritter and K. Schulten, Extending Kohonen's self-organizing mapping algorithm to learn ballistic movements. In *Neural Computers*, pp. 393–406. Springer, Heidelberg (1987).
14. D. H. Graf and W. R. LaLonde, A neural controlled for collision-free movement of general robot manipulators. In *IEEE Int. Conf. on Neural Networks*, San Diego, pp. 77–84 (1988).
15. T. Martinetz, H. Ritter and K. Schulten, Learning of visuomotor coordination of a robot arm with redundant degrees of freedom. In *Proc. Int. Conf. on Parallel Processing in Neural Systems and Computers*, Düsseldorf 1990 (ICNC-90) (R. Eckmiller, G. Hartmann, G. Hauske, Eds), North-Holland, Amsterdam (1990), pp. 431–434; and in *Proc. Third Int Symp. on Robotics and Manufacturing*, Vancouver, (*ISRAM-90*), pp. 521–526 (1990).
16. T. Martinetz and K. Schulten, Hierarchical neural net for learning control of a robot's arm and gripper. *IJCNN-90, Conf. Proc.*, San Diego, Vol.III, pp. 747–752 (1990).
17. M. Kuperstein, Neural model of adaptive hand–eye coordination for single postures. *Science* **239,** 1308–1311 (1988).
18. M. Kuperstein and J. Rubinstein, Implementation of an adaptive neural controller for sensory-motor coordination. *IEEE Control Syst. Mag.* **9,** 25–30 (1989).
19. S. H. Lane, D. A. Handelman and J. J. Gelfand, A neural network computational map approach to reflexive motor control. *Proc. 1988 IEEE Conf. on Intelligent Control* (1988).
20. B. W. Mel, A robot that learns by doing. *AIP Proc. 1987, Neural Information Processing System Conf.*, Denver, CO (1987).
21. W. T. Miller, Real-time application of neural networks for sensor-based control of robots with vision. *IEEE Trans. Syst. Man. Cybernet.* **19,** 825–831 (1989).
22. H. Ritter and K. Schulten, On the stationary state of Kohonen's self-organizing sensory mapping. *Biol. Cybernet.* **54,** 99–106 (1986).
23. B. Widrow and M. E. Hoff, Adaptive switching circuits. *WESCON Conv. Record*, Part IV, pp. 96–104 (1960).

AUTHORS' BIOGRAPHIES



Thomas Martinetz—Thomas Martinetz studied Physics and Mathematics and received his Diploma degree in physics at the Technical University of Munich in 1988. He is a fellow of the Volkswagen Foundation and is currently with the Department of Physics and the Beckman Institute for Advanced Science and Technology at the University of Illinois at Urbana-Champaign. He has coauthored one book and published several articles on neural networks.



Klaus Schulten—Klaus Schulten received his Diploma degree in physics at the University of Münster, Germany in 1969 and his Ph.D. in chemical physics at Harvard University in 1974. In 1980 he became Professor for theoretical physics at the Technical University of Munich. In 1988 he moved to the University of Illinois at Urbana-Champaign where he is Professor of physics, chemistry and biophysics, and a member of the new Beckman Institute.