Neural Networks, Vol. 2, pp. 159–168, 1989
Printed in the USA. All rights reserved.

0893-6080/89 $3.00 + .00

ORIGINAL CONTRIBUTION

Topology-Conserving Maps for Learning Visuo-Motor-Coordination

HELGE J. RITTER, THOMAS M. MARTINETZ AND KLAUS J. SCHULTEN

Technical University of Munich

(*Received* 16 *September* 1988; *revised and accepted* 18 *October* 1988)

Abstract—*We investigate the application of an extension of Kohonen's self-organizing mapping algorithm to the learning of visuo-motor-coordination of a simulated robot arm. We show that both arm kinematics and arm dynamics can be learned, if a suitable representation for the map output is used. Due to the topology-conserving property of the map spatially neighboring neurons can learn cooperatively, which greatly improves the robustness and the convergence properties of the algorithm.*

Keywords—Visuo-motor-coordination, Topology-conserving maps, Learning, Motor control, Robotics.

1. INTRODUCTION

Control of their limbs is one of the oldest tasks biological organisms had to solve in order to survive successfully. Therefore we have good reason to assume that much will be gained by elucidating the principles of biological motor control systems, which outperform todays robot control algorithms still by far.

A major one of the few generally recognized organizational principles in the brain is its organization into a collection of two-dimensional "modules" in which neighboring neurons contribute to similar tasks. These modules often represent "topology-conserving maps" in which neurons are dedicated to input data in such a fashion that the most essential interrelationships of the data are captured in the two-dimensional spatial arrangement of the corresponding neurons (Knudsen, du Lac, & Esterly, 1987). Examples are the auditive maps of sound location in the hippocampus, the motor map of eye-movements in the superior colliculus (Lee, Rohrer, & Sparks, 1988; Robinson, 1972), or the crescent-shaped arrangement of motor-neuron-pools in the motor cortex innervating arm muscles (Murphy, Kwan, MacKay, & Wong, 1977).

It seems that the functional role of such maps consists in providing an organization of a two-dimensional storage and processing medium such that the most important communication and processing requirements can be satisfied by local interactions, spanning small distances only.

In the past, study of such maps, both experimentally and theoretically, has mostly centered on the side of sensory perception (Harris, 1986; Kaas, Nelson, Sur, Lin, & Merzenich, 1979). Only recently, these maps have also been recognized as important for the generation of output for motor control (Sparks & Nelson, 1987) and theoretical approaches using topology conserving maps for robot control have been proposed (Grossberg & Kuperstein 1986; Kuperstein 1987, 1988; Ritter & Schulten 1986, 1987, 1988b). In this paper, we want to pursue this approach further and demonstrate the control of movements of a simulated three-link robot arm.

Our approach is based on an extension of Kohonen's self-organizing mapping algorithm (Kohonen, 1982a, 1982b, 1982c; Ritter & Schulten, 1988a) suggested earlier by two of the authors (Ritter & Schulten, 1986, 1987), which is capable of learning a mapping between a (sensory) input space and a (motor) output space by establishing a topology-conserving map on a (usually) planar array of neuronal units. The map is learned from a sequence of random movements of the arm, which are observed by a camera and used to gradually improve the map. The topology-conserving map allows neighboring units to cooperate during learning, which greatly contributes the efficiency and robustness of the algorithm. We investigated two different situations with this approach.

This work has been supported by the German Ministry of Science (BMFT) under contract ITR-8800-G9.

Requests for reprints should be sent to Helge J. Ritter, Department of Physics, Technical University of Munich, D-8046 Garching, Federal Republic of Germany.

The first situation is a simulation of a robot arm controlled by a neural network, which receives its input from a pair of cameras observing the arm. The task consists of learning to position the end effector of the arm at a specified target location on a table, that is, to learn the kinematic visuo-motor-coordination between camera output and desired end effector location. An essential aspect is the "closedness" of the whole system, observing its own reactions and learning from them. As a consequence, the whole "interfacing" to the outside world (output signals to joint motors and input signals from cameras) can be left to the adaptive capabilities of the internal map.

The second situation also includes the dynamics of the robot arm. For heavy limbs and fast movements, independent feedback control of the individual joints becomes inaccurate due to inertial effects and feedforward control with precomputed joint torques is a much superior approach (Brady, Hollerbach, Johnson, Lozano-Perez, & Mason, 1984). However, these torques depend in a very complex way on the full dynamic and kinematic characteristics of the arm. Learning this relationship, together with the arm characteristics, is the focus of the second simulation. We consider the case of so-called "ballistic" movements, which are controlled by brief torque pulses and proceed freely in the intervals between the pulses. During a sequence of trial movements the system learns to accelerate the end effector from given postures to prespecified velocities without any prior knowledge about arm kinematics and dynamics.

Together, these simulations address two important issues of robot control: adaptive learning of arm kinematics and of arm dynamics. Although we recognize that there is still a significant way towards a level of practical applicability, the results so far obtained appear very promising to contribute to building a flexible, self-organizing neural controller for robot arm movements in the future.

2. THE MAPPING ALGORITHM

In this Section we will describe the learning algorithm used for the two simulations.

For both simulations, a control task must be solved. Sensory input (a visually designated end effector location in the first case, an arm configuration together with a desired velocity in the second case), specifying the desired state of the arm, must be mapped into suitable motor output signals. Our only assumption for this mapping will be (a) suitable and fixed dimensionalities of the input and output signals and (b) a continuous relation between both. Then input and output can be represented by vectors $\mathbf{x} \in X$ and $\mathbf{y} \in Y$ respectively. The system must learn an initially unknown control law $\mathbf{y} = \mathbf{y}(\mathbf{x})$ for the solution of the control task. The control law will be adaptively represented by a "winner-take-all"-network of formal neurons, receiving the sensory input $\mathbf{x}$ in parallel. Each neuron $\mathbf{r}$ is "responsible" for some small subset (its "receptive field") $F_\mathbf{r}$ of the input space X. Whenever $\mathbf{x} \in F_\mathbf{r}$, neuron $\mathbf{r}$ determines the output. In the nervous system, the output will be specified by the average behavior of a localized subpopulation comprising many simultaneously active neurons with overlapping receptive fields (Georgopoulos, Schwartz, & Kettner, 1986). Their average behavior is summarized by a single formal neuron in our model and the subsets $F_\mathbf{r}$ are non-overlapping. To specify for each neuron $\mathbf{r}$ the subset $F_\mathbf{r}$ and the required output, two vectors, $\mathbf{w}_\mathbf{r} \in X$ and $\mathbf{u}_\mathbf{r} \in Y$ are associated with each neuron. $\mathbf{u}_\mathbf{r}$ specifies the output if neuron $\mathbf{r}$ "wins", that is, for $\mathbf{x} \in F_\mathbf{r}$, and $F_\mathbf{r}$ consists of all points of X, which are closer to $\mathbf{w}_\mathbf{r}$ than to any other $\mathbf{w}_\mathbf{s}$, $\mathbf{s} \neq \mathbf{r}$, that is,

$$F_\mathbf{r} = \{\mathbf{x} \in X | \|\mathbf{w}_\mathbf{r} - \mathbf{x}\| \leq \|\mathbf{w}_\mathbf{s} - \mathbf{x}\| \forall \mathbf{s}\}. \quad (1)$$

The mapping Φ represented by the network is

$$\Phi: \mathbf{x} \in X \longrightarrow \mathbf{y} = \mathbf{u}_{\mathbf{s}(\mathbf{x})} \in Y, \quad (2)$$

where $\mathbf{s}(\mathbf{x})$ is defined by the condition $\mathbf{x} \in F_{\mathbf{s}(\mathbf{x})}$. Initially, the vectors $\mathbf{w}_\mathbf{r}$ and $\mathbf{u}_\mathbf{r}$ are assigned randomly, and the task of the learning phase is to gradually adjust them in such a way, that Φ approximates the required control law $\mathbf{y}(\mathbf{x})$ as accurately as possible. This is achieved in the following way. For each sensory input $\mathbf{x}$, specifying a desired movement, the output $\mathbf{u}_{\mathbf{s}(\mathbf{x})}$ of the network is used to effect an actual movement which during learning will be subject to some error. Using an error-correction rule of Widrow-Hoff-type (Widrow & Hoff, 1960), this error is used to obtain an improved estimate $\mathbf{u}^*$ of what the correct output should have been (the details differ for the two simulations and are given in the subsequent sections). Then for all neurons the following adaptation step is made

$$\mathbf{w}_\mathbf{r}^{\text{new}} = \mathbf{w}_\mathbf{r}^{\text{old}} + \epsilon h_{\mathbf{rs}}(\mathbf{x} - \mathbf{w}_\mathbf{r}^{\text{old}}), \quad (3)$$

$$\mathbf{u}_\mathbf{r}^{\text{new}} = \mathbf{u}_\mathbf{r}^{\text{old}} + \epsilon' h'_{\mathbf{rs}}(\mathbf{u}^* - \mathbf{u}_\mathbf{r}^{\text{old}}). \quad (4)$$

Here $\mathbf{s} = \mathbf{s}(\mathbf{x})$, the neuron selected by input $\mathbf{x}$, ϵ and ϵ' scale the overall size and $h_{\mathbf{rs}}$ and $h'_{\mathbf{rs}}$ determine the spatial variation of the adaptation steps.

If $h_{\mathbf{rs}} = h'_{\mathbf{rs}} = \delta_{\mathbf{rs}}$, the system is equivalent to a perceptron. However, an essential ingredient here is a topological arrangement of the neurons. Each neuron $\mathbf{r}$ is considered as occupying a position $\mathbf{r}$ (chosen from a discrete square lattice for computational convenience) in a two-dimensional sheet[1], and the coef-

[1] The two-dimensionality is suggested from the situation in the cortex, where the neurons are essentially arranged in a sheet-like fashion. For technical applications, however, more elaborate topologies may offer advantages.

ficients h_{rs}, h'_{rs} are taken to be unimodal functions of Gaussian shape, depending on the distance $\|\mathbf{r} - \mathbf{s}\|$ and with a maximum at $\mathbf{r} = \mathbf{s}$ (to remove the ambiguity in the scaling of ϵ and ϵ', we require the normalization $h_{ss} = h'_{ss} = 1$). Hence, neighboring neurons in the sheet share adaptation steps with the same input and get tuned to similar inputs $\mathbf{x}$. Kohonen was the first to recognize this property for the formation of abstract sensory maps onto two-dimensional sheets analogous to the sensory maps found in the brain (Kohonen, 1982a, 1982b, 1982c). Our algorithm extends his method by associating with each formal neuron a second piece of data, the output quantity $\mathbf{u}_r$ (Ritter & Schulten, 1986, 1987). Hence in this case, there are two topology conserving maps, a map between the input space X and the neuron sheet, and a map between the output space Y and the sheet. Both maps develop simultaneously and therefore get matched in such a way as to approximate the desired input–output-relationship $\mathbf{y}(\mathbf{x})$. The resulting representation is an adaptive discretization, which adjusts its resolution dynamically to the probability density of the required control actions $(\mathbf{x}, \mathbf{y}(\mathbf{x})) \in X \otimes Y$ by allocating a higher proportion of neurons to regions of $X \otimes Y$ for which this probability density is high, that is, for which control actions are needed especially frequently. In particular, neurons are only allocated to those regions of $X \otimes Y$ actually required for representing the control law $\mathbf{y}(\mathbf{x})$. This results in a very economical use of the available storage elements. As a second important benefit, we have found that the cooperation (4) in the learning steps greatly improves the convergency for the output map (Ritter & Schulten, 1987). Often, the learning rule yielding the improved estimate $\mathbf{u}^*$ has only a limited radius of convergency. If the initial starting guess $\mathbf{u}_r$ is outside of this convergency zone, having the neurons learn their outputs in isolation (i.e., $h'_{rs} = \delta_{rs}$) will be unsuccessful. However, if the range of h'_{rs} is nonzero, each neuron participates in learning steps resulting from the actions of a whole subset of neighboring neurons, each trying to learn a similar output due to the topology-conserving map. Hence, false adjustments from neurons with bad starting values will tend to cancel, and the contributions from correct adjustments, which are similar for neighboring neurons, will be favored. This contributes significantly to the robustness of the method against poor starting values.

3. LEARNING ROBOT ARM KINEMATICS

In this Section we will present a demonstration of the above algorithm for the learning of the kinematics part of the visuo-motor-coordination of a simulated robot arm.

The robot arm is shown in Figure 1. It consists of three links and is mounted behind a table. Two cameras provide two pairs $\vec{x}_i = (x_{i1}, x_{i2})$, $i = 1, 2$ of "retinal coordinates" for any chosen target point in the scene. These are grouped into the four-component vector $\mathbf{x} = (\vec{x}_1, \vec{x}_2)$, which is fed as sensory input to a rectangular array of 12×21 neuronal units. The desired output of the array is the triple of joint angles positioning the end effector of the arm at the target location specified by $\mathbf{x}$.

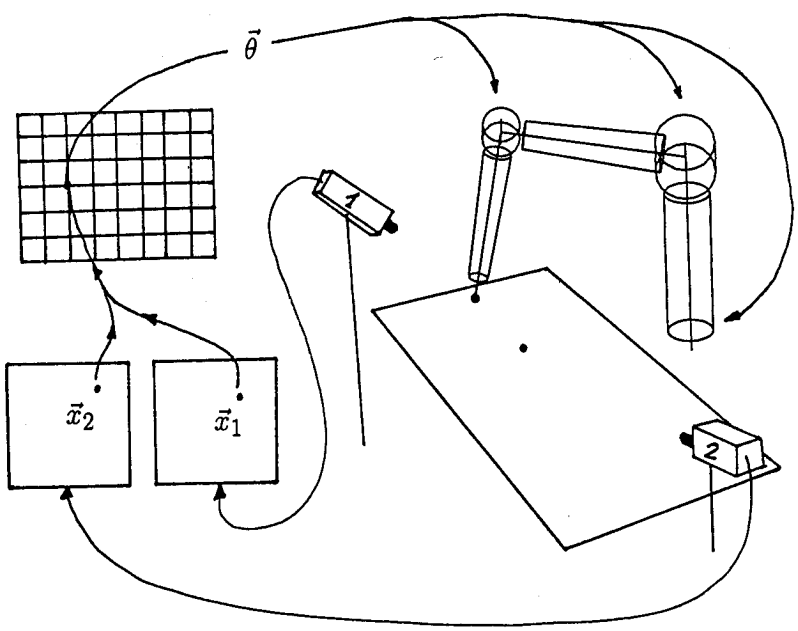

FIGURE 1. The simulated system. Two cameras observe the robot arm to the right of the table. Each camera has a quadratic "retina" and maps a specified location from the scene to a pair of retinal coordinates $\vec{x}_i (i = 1, 2)$. The pair $\mathbf{x} = (\vec{x}_1, \vec{x}_2)$ is fed as input to the array of neurons. The output of the array is determined by the neuron $\mathbf{s}$ whose vector $\mathbf{w}_s$ matches the sensory input $\mathbf{x}$ best.

The vectors $\mathbf{w}_r$ associated with the neurons are chosen as pairs $\mathbf{w}_r = (\vec{w}_{r1}, \vec{w}_{r2})$ of two component vectors $\vec{w}_{r1}$, $\vec{w}_{r2}$. $\vec{w}_{ri}$ is a two-dimensional location on the "retina" of camera i, $i = 1, 2$. Therefore each neuron is "binocular" and "looks" essentially at two small spots centered at $\vec{w}_{r1}$ and $\vec{w}_{r2}$ on the two camera "retinas."

To specify the output associated with neuron $\mathbf{r}$, we use a 15-component vector $\mathbf{u}_r$. Three components are joint angles, grouped into a triple $\vec{\theta}_r = (\theta_1, \theta_2, \theta_3)$. After learning, $\vec{\theta}_r$ shall represent the joint angles for which the end effector is at the target location specified by retinal coordinates $\mathbf{w}_r$.

To reduce the discretization error due to the limited number of neurons, we include the first term of a Taylor expansion for each value pair $(\mathbf{w}_r, \vec{\theta}_r)$. The remaining 12 components of $\mathbf{u}_r$ specify the necessary 3×4-Jacobians $\mathbf{A}_r$. If $\mathbf{x} \in F_s$, that is, neuron $\mathbf{s}$ is selected, its output then is given by

$$\vec{\theta} = \vec{\theta}_s + \gamma(t) \cdot \mathbf{A}_s(\mathbf{x} - \mathbf{w}_s). \tag{5}$$

Here $\gamma(t) \in [0, 1]$ is a time-dependent prefactor explained below.

In the absence of any further information, starting values for $\mathbf{w}_r$ and $\mathbf{u}_r = (\vec{\theta}_r, \mathbf{A}_r)$ may be chosen randomly. It is the task of the learning algorithm to adjust these to their correct final values. Each learning step involves execution of a trial movement of the end effector to some randomly designated target location. The camera output $\mathbf{x}$ for this target location selects a neuron $\mathbf{s}$ "looking" at $\mathbf{x}$, that is, $\mathbf{x} \in F_s$, for determining this movement. From the actual outcome of the movement we derive an improved estimate $\mathbf{u}^* = (\vec{\theta}^*, \mathbf{A}^*)$ for $\mathbf{u}_s$ and perform an adjustment according to (3) and (4). To obtain $\vec{\theta}^*$ and $\mathbf{A}^*$, the following strategy is used. First the array generates motor output for a "gross movement," which results from setting the joint angles to the values $\vec{\theta}_s$ associated with the selected neuron. This corresponds to an initial choice of $\gamma(0) = 0$ and brings the end effector to a location in the vicinity of the desired target point. The retinal coordinates of the end effector after this gross movement are denoted by $\mathbf{x}_I$. The gross movement is followed by a "fine movement" by increasing $\gamma(t)$ to its final value of unity, that is, switching on the linear correction term in (5). Denoting the resulting retinal coordinates of the end effector by $\mathbf{x}_F$, we take as improved estimate $\vec{\theta}^*$, $\mathbf{A}^*$

$$\vec{\theta}^* = \vec{\theta}_s + \mathbf{A}_s(\mathbf{x} - \mathbf{x}_F), \tag{6}$$

$$\mathbf{A}^* = \mathbf{A}_s + \mathbf{A}_s(\mathbf{x} - \mathbf{w}_s - \mathbf{x}_F + \mathbf{x}_I) \times (\mathbf{x}_F - \mathbf{x}_I)^T \|\mathbf{x}_F - \mathbf{x}_I\|^{-2}. \tag{7}$$

The first equation can be recognized as a linear error correction rule for the discretization values $\vec{\theta}_s$. The motivation for the second equation is more obvious, if it is written as

$$\mathbf{A}^* = \mathbf{A}_s + (\Delta\vec{\theta} - \mathbf{A}_s\Delta\mathbf{x})\Delta\mathbf{x}^T\|\Delta\mathbf{x}\|^{-2}, \tag{8}$$

where $\Delta\mathbf{x} = \mathbf{x}_F - \mathbf{x}_I$ and $\Delta\vec{\theta} = \mathbf{A}_s^{\text{true}}\,\Delta\mathbf{x}$ are the changes in the retinal coordinates of the end effector and the joint angles during the fine movement phase. As these are related by the matrix $\mathbf{A}_s^{\text{true}}$ to which $\mathbf{A}_s$ shall converge, (8) is seen to be equivalent to

$$\mathbf{A}^* = \mathbf{A}_s + (\mathbf{A}_s^{\text{true}} - \mathbf{A}_s)\Delta\mathbf{x}\Delta\mathbf{x}^T\|\Delta\mathbf{x}\|^{-2}, \tag{9}$$

that is, a linear error correction rule for $\mathbf{A}_s$.

The splitting of the movement into a first "gross movement" phase and a second "fine movement" phase has a counterpart in human arm movements (Keele, 1981). Although we do not claim that learning of such movements strictly adheres to the rules suggested here, it is interesting to note that the formulation of these rules was significantly simplified by such division of the movement and separate extraction of the "differential information" obtained in the fine movement phase (i.e., $\Delta\vec{\theta}$, $\Delta\mathbf{x}$).

In the following simulation we chose target locations from the table surface only. However, we also have carried out simulations without this restriction and with similar results, using an array with a three-dimensional topology. The table size is 0.7×0.4 units and the robot arm segments, beginning at the base, have lengths of 0.5, 0.4, and 0.4 units, respectively. Function h_{rs} was taken to be the Gaussian

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

and h'_{rs} likewise. Parameters ϵ, ϵ' and the widths σ, σ' all had the same time dependence $p(t) = p_i(p_f/p_i)^{t/t_{\max}}$ with t as the number of the already performed learning steps and $t_{\max} = 20{,}000$. The values were chosen as follows: $\epsilon_i = 1$, $\epsilon_f = 0.005$, $\epsilon'_i = 1$, $\epsilon'_f = 0.5$, $\sigma_i = 3$, $\sigma_f = 0.2$, $\sigma'_i = 2$, and $\sigma'_f = 0.05$.

Figure 2 shows the results of the simulation from the view of camera 1. Upper, center, and bottom rows show the state of the network initially, after 4000 and after 20,000 learning steps, respectively. Robot arm and table borders are indicated schematically by lines.

The leftmost picture of each row represents the part of the mapping $\mathbf{r} \to \mathbf{w}_r$ relevant to camera 1. Each node $\mathbf{r}$ of the lattice is mapped to a location $\vec{w}_{r1}$ in the image plane of camera 1. Initially the vectors $\vec{w}_{r1}$ and $\vec{w}_{r2}$ were distributed randomly in the image plane of their camera. This provided a homogenous distribution of the values $\mathbf{w}_r$ over the four-dimensional input space and the corresponding image of the lattice is highly irregular (top diagram). After only 4000 learning steps the initial distribution has retracted to the relevant two-dimensional subset corresponding to the table surface (center left). Fi-

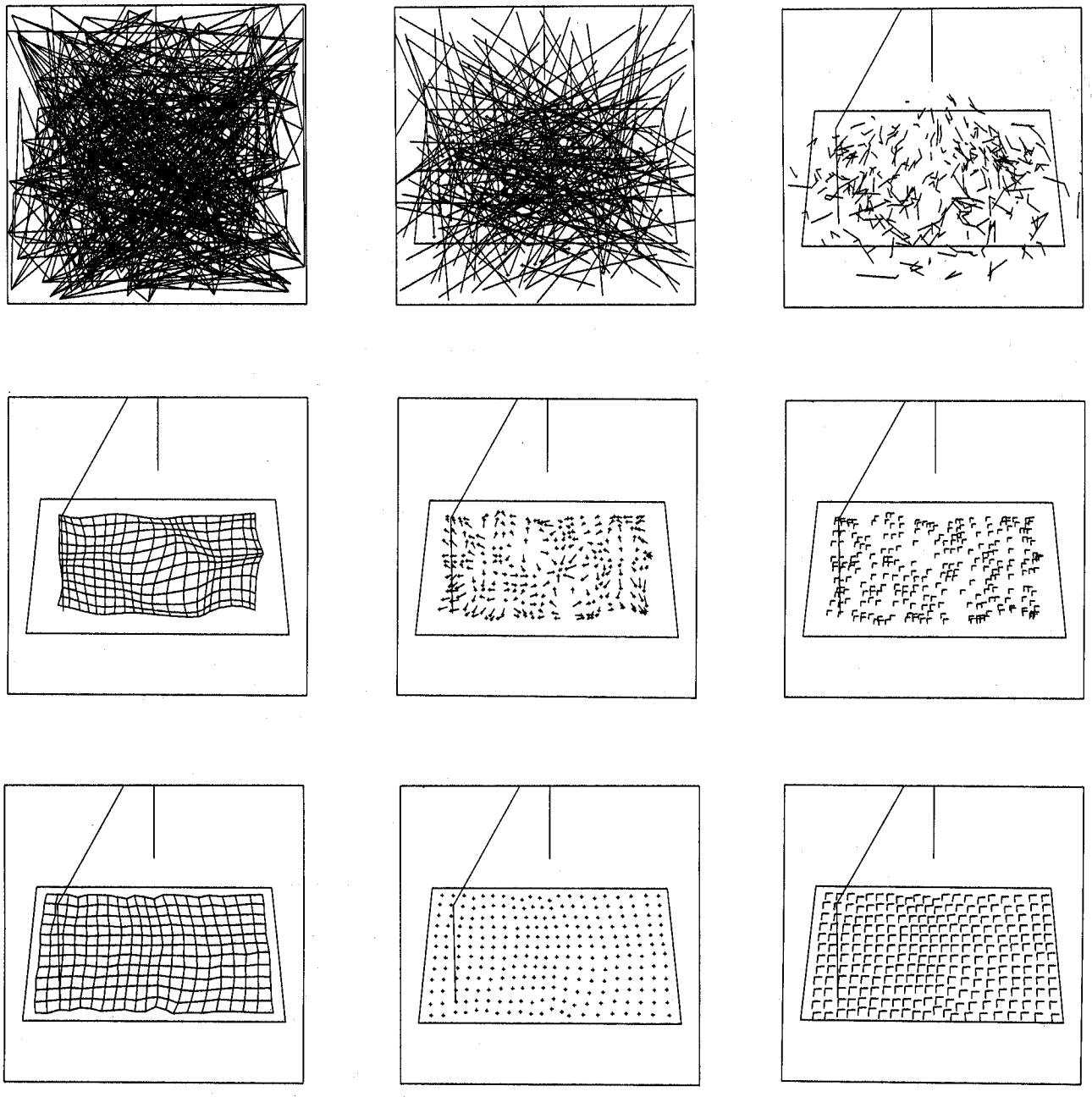

FIGURE 2. Simulation results from the view of camera 1. Robot arm and table surface are indicated schematically by lines. Each leftmost picture shows the retinal locations $\tilde{w}_{r1}$ the neurons get tuned to. The pictures in the center show the end effector locations x_r (cross marks) resulting from the joint angles $\vec{\theta}_r$, together with their deviation (appended line) from the retinal locations $\tilde{w}_{r1}$ associated with $\vec{\theta}_r$. Each rightmost picture visualizes the convergence of the matrices A_r by showing the reaction of the end effector to a "L"-shaped test movement. Upper row: initial state, values of w_r, $\vec{\theta}$ and A_r chosen randomly. Middle row: after 4000 learning steps only small deviations are left and test movements are performed to good approximation. Bottom row: final state after 20,000 learning steps. Deviations are no longer visible and test movements are traced out very accurately.

nally (bottom left) a very regular distribution of the nodes has emerged, indicating a good representation of the workspace by the discretization points $\mathbf{w}_r$.

The picture in the center of each row shows the mismatches between target positions and end effector locations which occur for the subset of visual inputs $\mathbf{x} = \mathbf{w}_r$. Each target position is indicated by a cross mark and the associated positioning error of the end effector by an appended line segment. The initial values of $\vec{\theta}_r$ were chosen randomly (with the only restriction 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 (topmost, center). However, after 4000 learning steps all errors have markedly decreased (center), until finally (20,000 steps, bottom) mismatches are no longer visible.

As the 12-dimensional Jacobians $\mathbf{A}_r$ cannot easily

be visualized directly, we instead show for each location x_l the reaction of the end effector to two test movements. Both movements are of equal length and directed parallel to the borders of the table. If $\mathbf{A}_r$ is correct, the end effector will trace out a little "L"-shaped right angle in the table plane, testing $\mathbf{A}_r$ along two orthogonal space directions. The gradual convergence of these two movements, as seen from camera 1, are shown in the rightmost picture of each row. The initial Jacobians were chosen by assigning a random value from the interval $[-100, 100]$ to each element of $\mathbf{A}_r$. Therefore, the initial test movements are very poor. However, after 4000 iterations the movements are seen to be already much better, and after 20,000 learning steps, the desired "L"-shape is traced out very accurately.

In Figure 3 we have plotted the average positioning error versus the number of learning steps. With cooperation ($\sigma' = 2$) between the neurons, the error decreases very rapidly to a final value of 0.4×10^{-3} after 20,000 iterations. Without cooperation (i.e., $\sigma' = 0$ and hence $h'_{rs} = \delta_{rs}$), the convergence is very unsatisfactory and a significant residual error remains.

4. LEARNING ROBOT ARM DYNAMICS

The previous simulation was restricted to learning the unknown kinematics of the robot arm. In this section, we want to extend the task to also include learning of unknown arm dynamics. This will be demonstrated for the case of ballistic movements driven by short torque pulses at the joints.

Instead of finding the joint angles for each target location, we now ask for the joint torques necessary

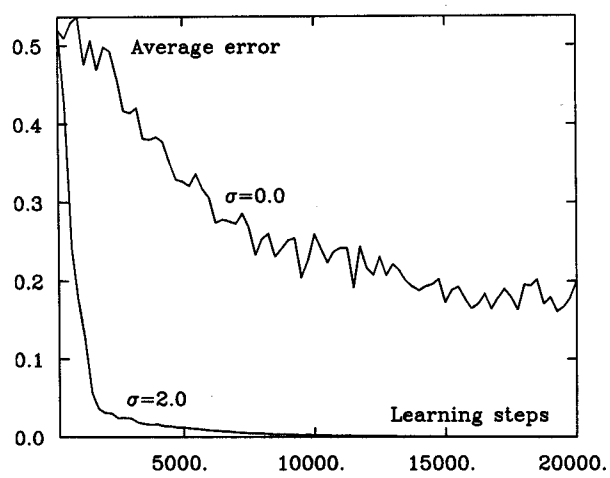

FIGURE 3. Average positioning error versus the number of learning steps. With cooperation between the neurons ($\sigma = 2.0$) the error decreases rapidly to a final value of 0.4×10^{-3} after 20,000 learning steps. If we switch off the cooperation ($\sigma = 0.0$), the decrease is much slower and a significant residual error remains even after 20,000 iterations.

to accelerate the end effector from a given initial position to a specified velocity $\mathbf{v}$. As we have demonstrated in the previous section that the joint angles can be learned from the camera output, we now assume that the task is specified by providing the initial joint angles $\vec{\theta}$ and the desired velocity $\mathbf{v}$ directly. The equations of motion of the arm can be shown to be (see, e.g., Brady et al., 1984)

$$d_i(t) = \sum_{j=1}^{3} A(\vec{\theta})_{ij}\ddot{q}_j + \sum_{j,k=1}^{3} B(\vec{\theta})_{ijk}\dot{q}_j\dot{q}_k + g_i(\vec{\theta}). \tag{11}$$

Here $\mathbf{d} = (d_1, d_2, d_3)$ are the applied joint torques, $A_{ij}(\vec{\theta})$ and $B_{ijk}(\vec{\theta})$ are configuration dependent matrix elements, depending on the dynamic properties of the arm and $\mathbf{q} = (q_1, q_2, q_3)$ are the Cartesian coordinates of the end effector location. The terms $g_i(\vec{\theta})$ account for the effect of gravity. A brief torque pulse, idealized by

$$\mathbf{d}(t) = \vec{\tau} \cdot \delta(t), \tag{12}$$

changes the velocity of the end effector to a value $\mathbf{v} = \dot{\mathbf{q}}$ obeying

$$\vec{\tau} = \mathbf{A}(\vec{\theta})\mathbf{v}. \tag{13}$$

$\vec{\tau} = (\tau_1, \tau_2, \tau_3)$ is the vectorial amplitude of the torque pulse. In particular, coefficients B_{ijk} and gravity g_i do not affect the velocity immediately attained after the pulse (in the further course of the free movement, $\mathbf{v}$ will change due to the B_{ijk} and g_i. This change will not be considered here). To learn the relationship between the inputs $(\vec{\theta}, \mathbf{v})$ and the torque amplitude $\vec{\tau}$, we could in principle proceed straightforwardly and associate with each neuron a 6-dimensional vector $\mathbf{w}_r$ from the space spanned by $\vec{\theta}$ and $\mathbf{v}$, and a 3-dimensional vector $\mathbf{u}_r$ for specifying $\vec{\tau}$. However, the linear relationship (13) suggests a much more economical representation. For each neuron we take a three-dimensional vector $\mathbf{w}_r$ from the space of joint angles only, and a 9-component output vector $\mathbf{u}_r$ specifying an approximation $\mathbf{A}_r$ to the matrix $\mathbf{A}(\vec{\theta})$ in (13). For joint angles $\vec{\theta} \in F_s$ and desired end effector velocity $\mathbf{v}$, the array returns the output

$$\vec{\tau} = \mathbf{A}_s\mathbf{v}. \tag{14}$$

The trial movements for the learning steps (3), (4) were generated by choosing some random arm configuration $\mathbf{x} = \vec{\theta}$ and requiring some random velocity $\mathbf{v}$ (isotropically distributed and with uniform distribution of magnitude in the unit interval) for each step. Using the actual velocity $\bar{\mathbf{v}}$ resulting from the response (14) of the array

$$\mathbf{A}^* = \mathbf{A}_s + \frac{\epsilon''}{\|\bar{\mathbf{v}}\|^2}(\vec{\tau} - \mathbf{A}_s\bar{\mathbf{v}})\bar{\mathbf{v}}^T \tag{15}$$

was taken as an improved estimate of the output matrix $\mathbf{A}_s$.

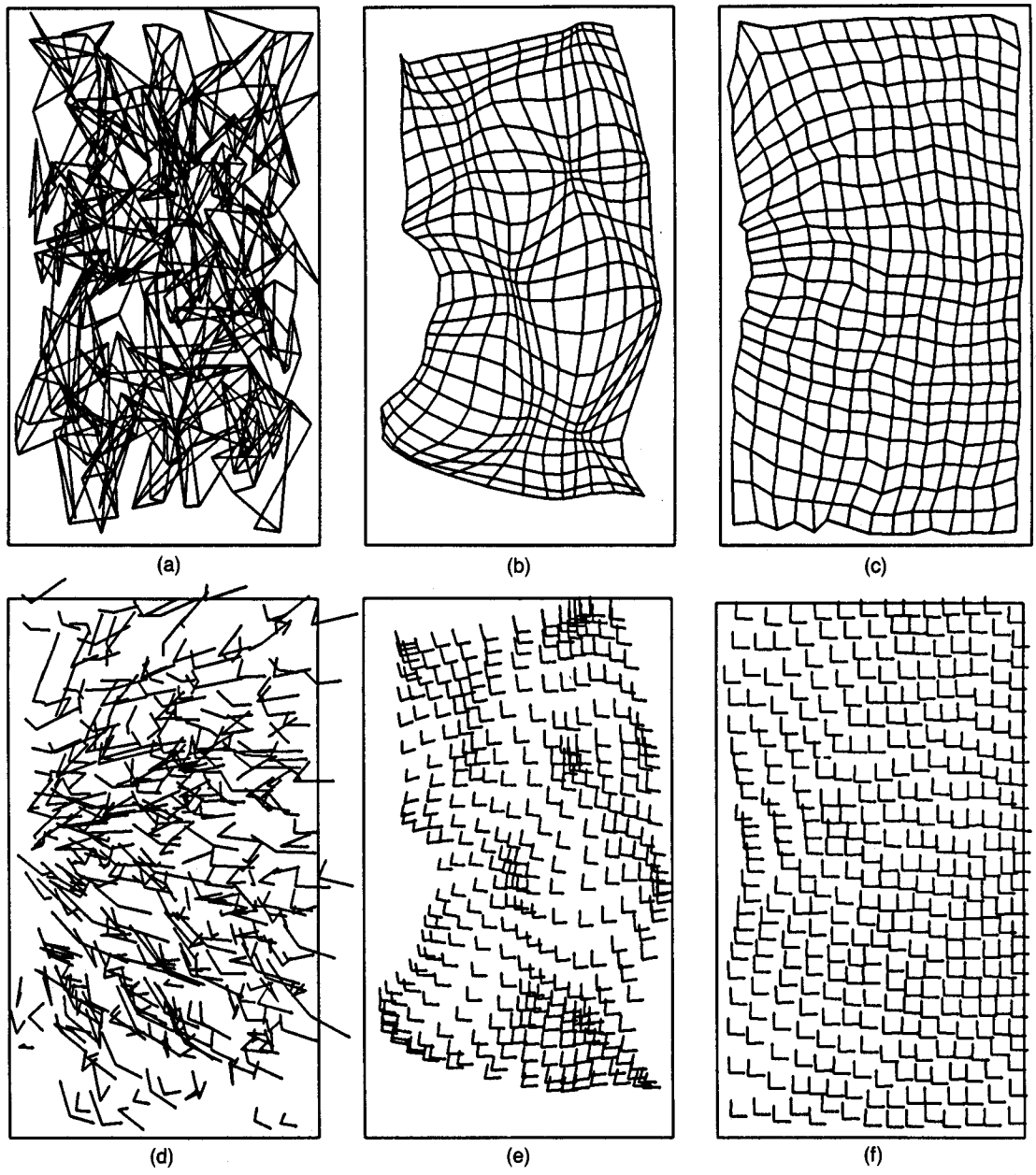

FIGURE 4. Development of the motor map represented by the array. The top row shows for each neuron r the corresponding end effector position on the table (viewed from above) associated with the joint angles w_r. End effector positions associated with neighboring neurons in the lattice are connected by lines. In the course of learning the initially random correspondence between lattice locations r and end effector locations (Figure 4a) gradually changes into an well ordered mapping preserving the topology of the array (Figure 4b, 500 trials and Figure 4c, 10,000 trials). The bottom row shows learning of the matrices A_r. To this end, for each of the end effector locations of the upper figures the reaction of the robot arm to two test movements is shown. Each test movement requires accelerating the arm to unit velocity parallel to one of the table edges. The figures show the response of the arm, that is, the actual movement velocities obtained. Initially, these conform very poorly to the required task, (Figure 4d), however, after 500 movement trials (Figure 4e), the movements are approximately correct and are performed very accurately after 10,000 trials (Figure 4f).

The velocities $\mathbf{v}$ were allowed to point along any 3-dimensional direction and the workspace was restricted to the table surface as before. Functions h_{rs} and h'_{rs} were Gaussians with $\epsilon' = 1$ and the same functional dependence for $\sigma(t)$, $\sigma'(t)$, $\epsilon(t)$ and $\epsilon''(t)$ as before. The movement of the robot arm was simulated using an algorithm of Walker and Orin (1982). For the mass distribution we chose three point masses of equal magnitude, located at the end effector and the two outer joints. The array consisted of a rectangular lattice of 15 × 24 neurons. Initially each vector $\mathbf{w}_r$ was assigned the joint angles belonging to a randomly chosen end effector position and $\mathbf{A}_r$ was set to

$$(\mathbf{A}_r)_{ij} = \mathbf{A}_{ij} + \alpha\|\mathbf{A}\| \cdot \eta. \qquad (16)$$

Here, $\mathbf{A}$ is the correct matrix for joint angles $\mathbf{w}_r$ and $\eta \in [-1, 1]$ is a uniformly distributed random variable. This procedure was used to generate statistical initial errors whose magnitude can be controlled by a parameter α. The simulation parameters were $\alpha = 0.25, \epsilon_i = 1, \epsilon_f = 0.5, \epsilon_i'' = 0.8, \epsilon_f'' = 0.02, \sigma_i = \sigma_i' = 3, \sigma_f = \sigma_f' = 0.2$, and $t_{\max} = 10{,}000$ trials. Figures 4a–c show the development of the mapping between arm configurations and the lattice. To this end, each figure shows a perpendicular view of the table plane, together with the end effector positions associated with the neurons. Positions associated with nearest lattice neighbors are connected by lines to display the degree of topological ordering of the map. Figures 4a–c show the progression from the initial random mapping (Figure 4a) to a more ordered intermediate state (Figure 4b, after 500 steps) and to the final well ordered mapping obtained after 10,000 learning steps (Figure 4c).

To visualize the convergence of the matrices $\mathbf{A}_r$, we again use test movements. Each test movement required to accelerate the end effector from one of the postures $\mathbf{w}_r$ to unit velocity parallel to one of the table axes. The resulting velocities are depicted as arrows in Figures 4d–f. Figure 4d shows the initial poor performance of the array. However, after 500 trial movements, the performance has improved significantly (Figure 4e), until finally (10,000 movement trials, Figure 4f) all test movements are performed to good accuracy.

To demonstrate the automatic adaptation of the resolution of the mapping for movements required with different frequency, Figure 5 shows the resulting map between end effector locations and neurons for a simulation, in which movements from a central circular region of the table were performed three times as frequently as those from the region outside. It can be seen that the map has assumed a particularly high resolution for this part of the table. This is a very useful property which seems to be characteristic of biological motor systems too.

The good convergence of the output map is again largely a consequence of the participation of a whole subpopulation of neighboring units in each adaptation step. To demonstrate this, Figure 6 shows the result of the same simulation, if $h_{rs}' = \delta_{rs}$, that is, each neuron performs its adaptation step in isolation. In this case, only part of the neurons manage to learn the correct matrix $\mathbf{A}_r$. This is shown in a more quantitative way in Figure 7, which displays the average error $\|\mathbf{v} - \mathbf{v}^{\text{true}}\|$ between desired and actual movement over the number of movement trials. In this case, averaging was done over all neurons and an isotropic distribution of desired velocities of unit magnitude. The three graphs belong to the same starting state (with $\alpha = 2$, i.e., even stronger initial errors than for Figure 4), but different initial values

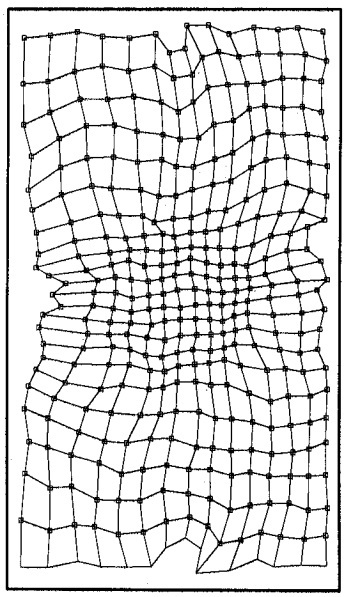

FIGURE 5. Non-uniform mapping between neurons and end effector locations resulting from requiring movements from a circular subregion of the table especially frequently. As a result, more neurons are used to represent these movements allowing for a finer resolution.

$\sigma_i' = 0.5, \sigma_i' = 1.0$ and $\sigma_i' = 2.0$ for the radius of the adjusted neuron population at each step. All other simulation parameters were as before. In the case of large initial diameter $\sigma_i' = 2$ the error declines the fastest and to a very small final value. However, for the smaller diameters $\sigma_i' = 1$ and $\sigma_i' = 0.5$ only part of the neurons manage to converge, giving rise

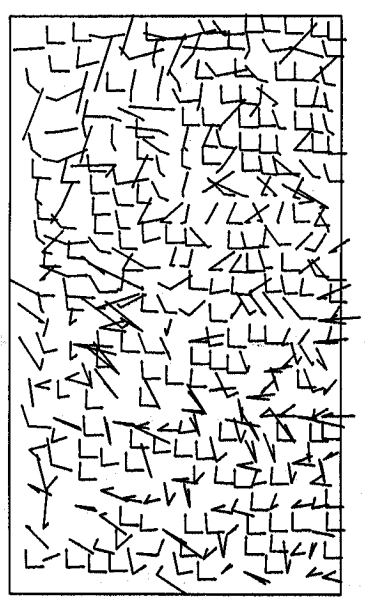

FIGURE 6. Resulting poor convergency in the absence of cooperation between the neurons. Displayed are reactions to test movements after 10,000 iterations, as in Figure 4f. However, each adjustment step now was confined to the selected neuron only (i.e., $h_{rs}' = \delta_{rs}$). As a consequence, only very few neurons manage to learn the correct matrix A_r.

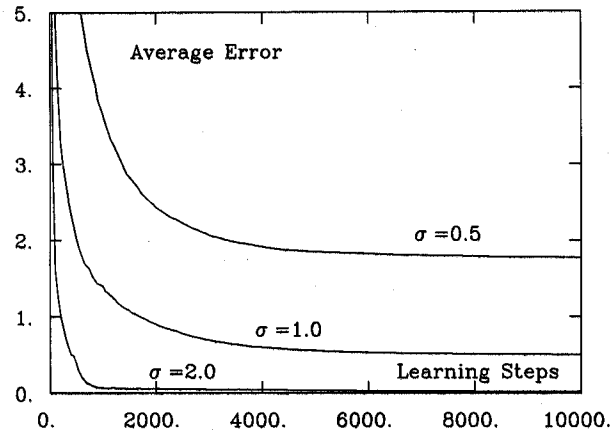

FIGURE 7. Improvement of convergence by cooperation between neurons. The diagram shows the average deviation between desired and achieved end effector velocity versus number of movement trials for three different initial values of σ_i. For weak cooperation (i.e., small values of σ_i) a significant residual error remains.

to slower decrease of the average error and its saturation at a significant non-zero final value, which is larger for the smaller diameter. The saturation shows that these errors cannot be avoided by employing more movement trials. The wrong responses are due to additional, undesired fixed points of the learning algorithm. A mathematical analysis of this behavior has been given in Ritter and Schulten (1987). Therefore, the cooperation of a whole population of neurons at each learning step greatly facilitates learning and significantly improves the robustness of our learning algorithm. This is a very interesting feature which may be worthwhile of further study in the context of other learning algorithms, too.

5. CONCLUSION

Topology conserving maps seem to play an important role in the organization of higher brains. We have investigated the potential of such maps for the control of robot arm movements. Our algorithm is an extension of Kohonen's algorithm for the formation of topologically correct feature maps. The basic idea is to use an input and an output map evolving simultaneously on the same sheet of neurons, thereby automatically matching corresponding input–output pairs in a topology-preserving fashion. This approach allows robust and flexible learning of continuous input–output-relations from a sequence of examples. We applied our method to learn the required transformations for visuo-motor-coordination of a robot arm. Using suitable representations, both, kinematic and dynamic properties of the arm can be learned. The use of an adaptive topology conserving map results in at least two benefits: (a) the map can assume a variable resolution, representing frequent movements more accurately, and (b) spreading adjustments over local subpopulations of neurons greatly increases the convergency speed and robustness of the output map. For low dimensional spaces, the method may offer an interesting alternative to back-propagation (Rumelhart, Hinton, & Williams, 1986). An interesting future task will be the coupling of several maps to represent more complicated and higher-dimensional input–output relationships.

REFERENCES

Brady, M., Hollerbach, J. M., Johnson, T. L., Lozano-Perez, T., Mason, M. T. (1984). *Robot motion: Planning and Control.* Cambridge, Ma: MIT Press.

Georgopoulos, A. P., Schwartz, A. B., & Kettner, R. E. (1986). Neuronal population coding of movement direction. *Science,* **233**, 1416–1419.

Grossberg, S., & Kuperstein, M. (1986). *Neural dynamics of adaptive sensory-motor control.* Amsterdam: North Holland.

Harris, W. A. (1986). Learned topography: The eye instructs the ear. *Trends in Neuroscience*, March, 97–99.

Kaas, J. H., Nelson, R. J., Sur, M., Lin, C. S., & Merzenich, M. M. (1979). Multiple representations of the body within the primary somatosensory cortex of primates. *Science*, **204**, 521–523.

Keele, S. W. (1981). Behavioral analysis of movement. In V. B. Brooks (Ed.), *Handbook of physiology: The nervous system II. Motor control* (pp. 1391–1414). Bethesda, MD: American Physiological Society.

Knudsen, E. I., du Lac, S., & Esterly, S. D. (1987). Computational maps in the brain. *Annual Reviews of Neuroscience*, **10**, 41–65.

Kohonen, T. (1982a). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, **43**, 59–69.

Kohonen, T. (1982b). Analysis of a simple self-organizing process. *Biological Cybernetics*, **44**, 135–140.

Kohonen, T. (1982c). Clustering, taxonomy and topological maps of patterns. *Proceedings of the 6th International Conference on Pattern Recognition* (pp. 114–128). Munich: IEEE Computer Society.

Kuperstein, M. (1987). Adaptive visual-motor coordination in multijoint robots using parallel architecture. *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 1595–1602). Raleigh, NC: IEEE Computer Society.

Kuperstein, M. (1988). Neural model of adaptive hand-eye coordination for single postures. *Science*, **239**, 1308–1311.

Lee, C., Rohrer, W. H., & Sparks, D. L. (1988). Population coding of saccadic eye movements by neurons in the superior colliculus. *Nature*, **332**, 357–360.

Murphy, J. T., Kwan, H. C., MacKay, W. A., & Wong, Y. C. (1977). Spatial organization of precentral cortex in awake primates. III. Input-output coupling. *Journal of Neurophysiology*, **41**, 1132–1139.

Ritter, H., & Schulten, K. (1986). Topology conserving mappings for learning motor tasks. In J. S. Denker (Ed.), *Neural networks for computing, AIP Conference Proceedings 151*, (pp. 376–380). Snowbird, Utah.

Ritter, H., & Schulten, K. (1987). Extending Kohonen's self-organizing mapping algorithm to learn ballistic movements. In R. Eckmiller & C. von der Malsburg (Eds.), *Neural computers*, (pp. 393–406). Heidelberg: Springer.

Ritter, H., & Schulten, K. (1988a). Convergency properties of Kohonen's topology conserving maps: Fluctuations, stability and dimension selection. *Biological Cybernetics*, **60**, 59–71.

Ritter, H., & Schulten, K. (1988b). Kohonen's self-organizing maps: Exploring their computational capabilities. *IEEEE ICNN 88 Conference*, (pp. 109–116). San Diego: IEEE Computer Society.

Robinson, D. A. (1972). Eye movements evoked by collicular stimulation in the alert monkey. *Vision Research*, **12**, 1795–1808.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, **323**, 533–536.

Sparks, D. L., & Nelson, J. S. (1987), Sensory and motor maps in the mammalian superior colliculus. *Trends in Neurosciences*, **10**, 312–317.

Walker, M. W., & Orin, D. E. (1982). Efficient dynamic computer simulation of robotic mechanisms. *Journal of Dynamic Systems, Measurement and Control*, **104**, 205–211.

Widrow, B., & Hoff, M. E. (1960). Adaptive switching circuits. *WESCON Convention Record*, P. IV, pp. 96–104.