Three-Dimensional Neural Net for Learning Visuomotor Coordination of a Robot Arm

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

Abstract—**An extension of Kohonen's self-organizing mapping algorithm together with an error-correction scheme based on the Widrow–Hoff learning rule is applied to develop a learning algorithm for the visuomotor coordination of a simulated robot arm. Learning occurs by a sequence of trial movements without the need of an external teacher. Using input signals from a pair of cameras, the "closed" robot arm system is able to reduce its positioning error to about 0.3 percent of the linear dimensions of its work space. This is achieved by choosing the connectivity of a 3D-lattice between the units of the neural net.**

I. 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 still outperform by far today's robot control algorithms [1].

Only recently, topology conserving maps have been recognized as important for the generation of output for motor control [2] and theoretical approaches using topology conserving maps for robot control have been proposed [3]–[12]. One important contribution using topology conserving maps for visuomotor coordination is Kuperstein's model [6]–[9]. The model, which already has been implemented and tested on a real robot arm system [8], learns during a training phase control of a robot arm with five degrees of freedom (four joints and a parallel jaw gripper) so that subsequently it can successfully reach for visually presented objects of cylindrical shape. This is achieved through the use of a set of topographic maps that represent the location of the target object and hold the adaptive weights determining the output to the arm actuators. However, in Kuperstein's model, each topographic map is only one dimensional and has a fixed topographic ordering, which is imposed initially. Only the output weights can adapt during the learning process. As a consequence, for the design of the system the range of the expected input values must be known beforehand and adaptive changes in the resolution of the neural population required for control are not possible. Furthermore, as the maps are one dimensional and their outputs for each actuator are summed linearly, they can approximate only a restricted class of control laws accurately (cf. Section VI).

In this paper we want to present an approach which can overcome both problems by using a network architecture which is an extension of Kohonen's model for the formation of topographically correct feature maps [13]–[15]. In this model, the ordering and the resolution of the topographic map evolve during learning by adjusting a layer of input weights determining the distribution of neuronal units over the task space, thereby overcoming the problem of fixed resolution. For the adaptation of the output weights we use an error-correction scheme based on the Widrow–Hoff learning rule for adaptive linear elements [16]. In addition, we use a network with a three-dimensional topology matched to the work space. This eliminates the restrictions arising from the additive coupling of several one-dimensional maps. In addition, the 3D-lattice-topology-conserving map allows many neighboring units to cooperate during learning, which greatly contributes to the efficiency and robustness of the algorithm.

In the following, we will present a simulation study of this approach for the control of a robot arm with three joints, shown in Fig. 1. The arm is controlled by an array of neural units, which receives its input from a pair of cameras observing the arm. The task consists of learning to position the end effector (manipulator) of the arm at a specified object location within the work space in front of the robot arm, i.e., to learn the kinematic visuomotor coordination between camera output and desired end effector location. To achieve this goal, the system learns a mapping between a (sensory) input space and a (motor) output space by establishing a topology-conserving map on an array of neuronal units. The map is learned from a sequence of trial movements of the robot arm, which are observed by cameras and used to gradually improve the map. These trial movements try to position the end effector of the arm at different target locations within the work space. An essential aspect is the "closedness" of the whole system, observing its own reactions and learning from them. As a consequence, much of the detailed "interfacing" to the outside world (e.g., input signals from cameras and output signals to joint motors) can be left to the adaptive capabilities of the internal map.

Manuscript received June 13, 1989; revised October 24, 1989. This work has been supported by the German Ministry of Science and Technology (ITR-8800-G9) and by the Volkswagen Foundation. An earlier version of this paper was presented at the 1989 International Joint Conference on Neural Networks, Washington, DC, June 18-22, 1989.

The authors are with the Department of Physics and with the Beckman Institute for Advanced Science and Technology, University of Illinois, Urbana, IL 61801.

IEEE Log Number 8933091.

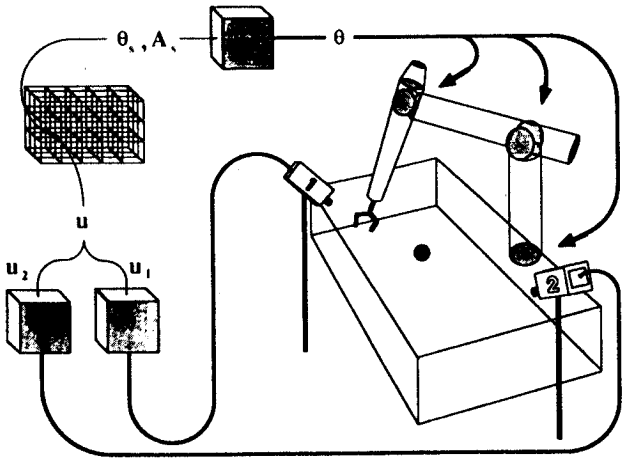


Fig. 1. The simulated robot system. Two cameras observe the robot arm to the right of the work space whose borders are indicated schematically by lines. Each camera provides a picture for a visual preprocessing unit, which extracts the "retinal" coordinates $\vec{u}_1$, $\vec{u}_2$ of the object the manipulator of the robot arm will reach for. The "retinal" coordinates of both cameras are grouped to a four-dimensional vector $u = (\vec{u}_1, \vec{u}_2)$ which is fed as input to a 3D-lattice of neurons. The output $\vec{\theta}_s, A_s$ of the neuron with the array vector w_s which matches the sensory input u best is used to specify the desired joint angles of the robot arm.

II. The Model

Each target object the robot arm reaches for is seen by cameras 1 and 2. As we do not want to address issues of early vision, we assume some form of visual preprocessing which is able to reduce the images of both cameras to a pair of "retinal" coordinates $\vec{u}_1$, $\vec{u}_2$ of the target object. Such preprocessing can, e.g., be realized by simple convolution and thresholding operations, provided that there is a single target object of high contrast. This is also the approach taken in Kuperstein's implementation.

We group both two-dimensional coordinates $\vec{u}_1$ and $\vec{u}_2$ to a four-dimensional vector u which then carries implicitly the whole information necessary to determine the position of the target. To be able to position its manipulator correctly the robot system has to know the transformation $\vec{\theta}(u)$ from "retina" locations u to angles $\vec{\theta}$ of the three-joint arm. This transformation depends on both the geometry of the robot arm and the positions of both cameras relative to the work space and will be learned automatically by the learning procedure described below.

The control law is adaptively represented by a "winner-take-all" network of formal neurons [17], receiving the sensory input u in parallel. Each neuron r is "responsible" for some small subset (its "receptive field") F_r of the four-dimensional input space U. U consists of the subset of all possible "retinal" coordinates $u = (\vec{u}_1, \vec{u}_2)$. Whenever $u \in F_r$, neuron r wins and 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 [18]. In our model the subsets F_r are nonoverlapping and a single formal neuron summarizes the average behavior of the localized population. To specify for each neuron r the subset F_r, a vector $w_r \in U$ is associated with each neuron. The vectors w_r are chosen as pairs $w_r = (\vec{w}_{r1}, \vec{w}_{r2})$ 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 required output, a vector $\vec{\theta}_r$ together with a 3×4 matrix A_r are associated with each neuron in addition to w_r. The system produces the joint angles $\vec{\theta}(u) = (\theta_1, \theta_2, \theta_3)$ by using $\vec{\theta}_r$ and A_r to specify the first two terms of a Taylor expansion of $\vec{\theta}(u)$, i.e.,

$$\vec{\theta}(u) = \vec{\theta}_s + A_s(u - w_s). \tag{1}$$

At the end of the learning procedure, this Taylor expansion shall approximate the exact transformation $\vec{\theta}(u)$ over the small compact subset of inputs u neuron s is responsible for. These subsets F_s are defined to be the subsets of U, which consists of all points which are closer to w_s than to any other w_r, $r \neq s$, i.e.,

$$F_s = \{u \in U \mid \|w_s - u\| \leq \|w_r - u\| \forall r\}. \tag{2}$$

Initially, w_r and $(\vec{\theta}, A)_r \equiv (\vec{\theta}_r, A_r)$ are assigned randomly, and the learning task is to gradually adjust them in such a way that the required control law $\vec{\theta}(u)$ is approximated as accurately as possible. This is achieved in the following way.

III. The Learning Procedure

The objects the robot arm will reach for are presented at different, randomly chosen locations within the work space. For each sensory input u induced by an object, the network output specified by $(\vec{\theta}, A)_s$ is used to effect an actual position, which during learning will be subject to some error. Using an error-correction rule of Widrow–Hoff-type, this error is used to obtain an improved estimate $(\vec{\theta}^*, A^*)$ of what the correct output should have been (the details are given in the subsequent sections). Then for all neurons the following adaptation step is made

$$w_r^{\text{new}} = w_r^{\text{old}} + \epsilon h_{rs}(u - w_r^{\text{old}}) \tag{3}$$

$$(\vec{\theta}, A)_r^{\text{new}} = (\vec{\theta}, A)_r^{\text{old}} + \epsilon' h_{rs}((\vec{\theta}, A)^* - (\vec{\theta}, A)_r^{\text{old}}). \tag{4}$$

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

If $h_{rs} = h'_{rs} = \delta_{rs}$, the system is equivalent to a perceptron. However, an essential ingredient here is a topological arrangement of the neurons. Each neuron r is considered as occupying a position r in a lattice, normally a two-dimensional sheet, and the coefficients h_{rs}, h'_{rs} are taken to be unimodal functions of Gaussian shape, depending on the lattice distance $\|r - s\|$ and with a maximum at $r = s$ (to remove the ambiguity in the scaling and ϵ and ϵ', we require the normalization $h_{ss} = h'_{ss} = 1$). Hence, neighboring neurons in the sheet share adaptation steps for the same input and get tuned to similar inputs u. Ko-

honen was the first to recognize this property for the formation of abstract sensory maps onto normally two-dimensional sheets analogous to the sensory maps found in the brain [13]–[15]. Our algorithm extends his method by associating with each formal neuron a second piece of data, the output quantity $(\vec{\theta}, A)_r$. Hence in this case, there are two topology conserving maps, a map between the input space U and the neural net, and a map between the output space, defined by $(\vec{\theta}, A)$, and the net. Both maps develop simultaneously and therefore get matched in such a way as to approximate the desired input–output relationship $\vec{\theta}(u)$. The resulting representation is an adaptive discretization, which adjusts its resolution dynamically to the range and the probability density of the required control actions $\vec{\theta}(u)$ by allocating neurons only to those regions of U actually required for representing the control law $\vec{\theta}(u)$.

In the present case, these regions form a submanifold of the input space spanned by the camera signals u. As each point of this submanifold corresponds to a target location in the three-dimensional work space of the robot, the submanifold is also three dimensional. The effect of (3) and (4) is to adaptively distribute the values w_r associated with the neurons r over this submanifold, and choosing their density distribution and therefore the resolution of the representation according to the probability density of the movements occurring during training.

As lattice neighbors share their adaptation steps in (3) and (4), this process will be particularly efficient, if the topology of the lattice matches the topology of the submanifold. Therefore, we choose the topology of a 3 D-lattice of $7 \times 12 \times 4$ units for the arrangement of the neurons. The three dimensionality is not directly suggested from the situation in the cortex, where the neurons are geometrically arranged in a more sheet-like fashion; however, the geometric arrangement of the neurons need not necessarily reflect the topology inherent in the connectivity among the neurons. If one is willing to accept a fraction of long-distance connections among neurons, higher dimensional topologies can be implemented with two-dimensional geometric arrangements of neurons as well. The presence of axon bundles in the brain running over longer distances might well be indicative of such underlying higher dimensional wiring topologies.

IV. The Error-Correction Rule

In the absence of any further information, starting values for w_r and $\vec{\theta}_r$, 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 u for this target location selects a neuron s "looking" at u, i.e. $u \in F_s$, for determining the movement. From the actual outcome of the movement we derive an improved estimate $(\vec{\theta}^*, A^*)$ for $(\vec{\theta}_s, A_s)$ and perform an adjustment according to (3) and (4). To obtain $\vec{\theta}^*$ and 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 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 v_i. The gross movement is followed by a "fine movement" by switching on the linear correction term in (1). This moves the "retinal" coordinates of the end effector to their final position v_f, leaving a final error of $u - v_f$ between desired and achieved end effector position in the visual field of the cameras. From u, v_i, and v_f, provided by the cameras by observing the robot arm during the performance of its task, the neural network determines improved estimates of $\vec{\theta}^*$, A^* by using the error correction rules

$$\vec{\theta}^* = \vec{\theta}_s + A_s(u - v_f), \tag{5}$$

$$A^* = A_s + \|v_f - v_i\|^{-2} \cdot A_s(u - w_s - v_f + v_i)(v_f - v_i)^T. \tag{6}$$

The first equation can be recognized as a linear error correction rule for the discretization values $\vec{\theta}_s$. At the end of the learning procedure, when $A_s \approx A_s^{\text{true}}$ with A_s^{true} as the exact Jacobi-matrix at $x = v_f$ of the transformation $\vec{\theta} = \vec{\theta}(x)$ between the four-dimensional "retinal" coordinates of the target object and the related joint angles, $A_s(u - v_f)$ gives the accurate direction of steepest descent for the correction of $\vec{\theta}_s$ as long as $u - v_f$ is small. The motivation for the second equation is more obvious, if it is written as

$$A^* = A_s + \|\Delta v\|^{-2} \cdot (\Delta\vec{\theta} - A_s\Delta v)\Delta v^T \tag{7}$$

where $\Delta v = v_f - v_i$ and $\Delta\vec{\theta} = A_s(u - w_s)$ are the changes in the "retinal" coordinates of the end effector and the related joint angles during the fine movement phase. In this form (6) can be recognized as an error-correction rule of Widrow–Hoff-type for the Jacobians A_s. The prefactor $\|\Delta v\|^{-2}$ determines the size of the adaptation step.

As we see in (5) and (6), our robot system only needs information provided by the cameras during the robot's positioning movement. Thus it is able to attain its learning goal without the need for any kind of external "teacher."

V. Simulation Results

In the following simulation we chose target locations from the work space, which is indicated by lines in Fig. 1. The size of the work space is $0.7 \times 0.4 \times 0.2$ 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

$$h_{rs} = \exp\left(-\|r - s\|^2/2\sigma^2(t)\right) \tag{8}$$

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 $t_{\max} = 30\,000$. The parameter values were chosen as

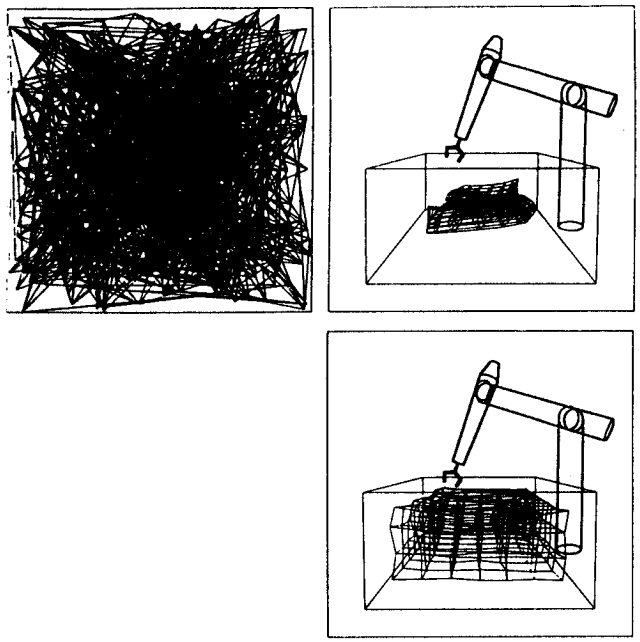


Fig. 2. The "retinal" locations $\vec{w}_{r2}$ the neurons get tuned to from the view of camera 2. The robot arm and the work space are indicated schematically by lines. The leftmost picture shows the initial state after chosing w_r randomly. In the rightmost picture at the top we see the state after 6000 learning steps, and the third picture shows the locations the elements of the 3D-lattice are assigned to after 30 000 iterations.

follows: $\epsilon_i = 1$, $\epsilon_f = 0.005$, $\epsilon_i' = 1$, $\epsilon_f' = 0.7$, $\sigma_i = 2.5$, $\sigma_f = 0.1$, $\sigma_i' = 1.5$, and $\sigma_f' = 0.05$.

Figs. 2 and 3 show the results of the simulation from the view of camera 2. Fig. 2 shows the state of the mapping $r \rightarrow w_r$ relevant to camera 2 initially, after 6000 and after 30 000 learning steps, respectively. Each node r of the lattice is mapped to a location $\vec{w}_{r2}$ ($w_r = (\vec{w}_{r1}, \vec{w}_{r2})$) in the image plane of camera 2. Values associated with lattice neighbors are connected by lines to visualize the lattice topology. Initially, the vectors $\vec{w}_{r1}$ and $\vec{w}_{r2}$ were distributed randomly in the image plane of their camera. This provided a homogeneous random distribution of the values w_r over the four-dimensional input space and the corresponding image of the lattice is highly irregular (left diagram). After only 6000 learning steps the initial distribution has retracted to the relevant three-dimensional subspace corresponding to the work space (top right). Finally (bottom right) a very regular distribution of the nodes has emerged, indicating a good representation of the work space by the discretization points w_r.

To visualize the performance of the gross movements, i.e., the values of the zero-order terms of the Taylor expansion (1), we show in the leftmost pictures of Fig. 3 the mismatches between intended target positions and actually achieved end effector locations which occur for the special subset of visual inputs $u = w_r$. Each end effector position after adjusting the joint angles by using $\vec{\theta}_r$ 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 $\vec{\theta}_r$ were chosen randomly (with the only restriction that the resulting end effector

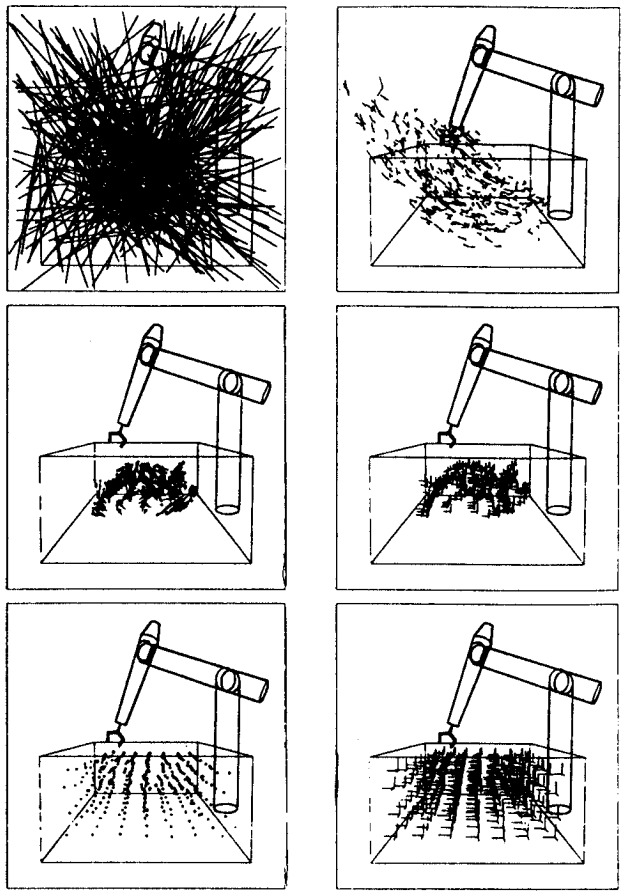


Fig. 3. The performance of the output at the start, after 6000 and after 30 000 trial movements. The leftmost pictures show the end effector locations v_i (cross marks) resulting from visual input $u = w_r$ (i.e., joint angles are $\vec{\theta}_r$), together with their deviation (appended line) from the target locations associated with $\vec{\theta}_r$. The rightmost pictures visualize A_r by showing the reaction of the end effector to small test movements parallel to the borders of the work space.

positions should lie in the space in front of the robot) and consequently the errors are very large for the initial state (topmost left). However, after 6000 learning steps all errors have markedly decreased (center), until finally (30 000 steps, bottom) mismatches are no longer visible.

The special subset of target locations $u = w_r$ was chosen to visualize the accuracy of the zero-order terms $\vec{\theta}_r$ of the Taylor expansion (1). 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 w_r. The deviations from the discretization points are taken into account by the first-order terms $A_r(u - w_r)$ of the Taylor expansion (1). As the 3×4 Jacobians A_r cannot easily be visualized directly, we instead show for each location $\vec{\theta}_r$ 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 work space. If A_r is correct, the end effector will trace out little "three-legged" patterns, testing A_r along the three orthogonal space directions. The gradual convergence of these three test movements, as seen from camera 2, are shown in the rightmost pictures. The initial

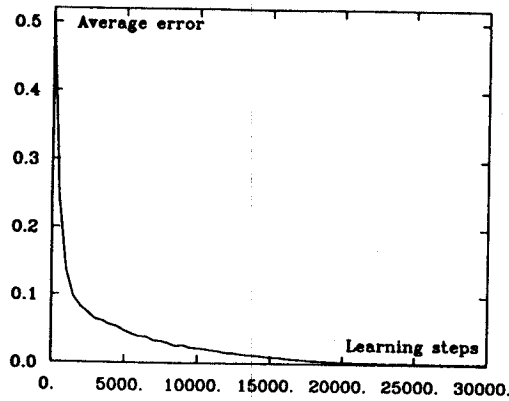


Fig. 4. Average positioning error versus the number of learning steps. The error decreases rapidly to a final value of 1.7×10^{-3} units after 30 000 learning steps.

Jacobians were chosen by assigning a random value from the interval $[-20, 20]$ independently to each element of A_r. Therefore, the initial test movements are very poor (Fig. 3, top). However, after 6000 iterations the accuracy of the test movements has significantly improved (Fig. 3, center), and after 30 000 learning steps, they are traced out very accurately (Fig. 3, bottom).

In Fig. 4 we have plotted the average positioning error versus the number of learning steps. The error decreases very rapidly to a final value of $1.7 \cdot 10^{-3}$ units after 30 000 iterations. At this stage 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 formal synaptic strengths w_r, $\vec{\theta}_r$ and A_r. This is possible as long as the plasticity of the network, i.e., the sizes of the adaptation steps after each positioning movement, described by the parameters ϵ and ϵ', are kept at small but nonvanishing values. In our simulation the final values of ϵ and ϵ' were chosen to 0.005 and 0.7, respectively, which guarantees a permanent adaptability to unforeseen changes in the behavior of the robot arm system during performance.

VI. Discussion

We have shown that an extension of Kohonen's algorithm for the formation of topologically correct feature maps together with an error-correction rule of Widrow-Hoff-type is able to learn the control of robot arm movements by using only the input signals of two cameras. 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 visuomotor coordination of a three-link robot arm.

This approach differs from previous approaches to neural robot control, such as, e.g., [11], [12] in that it makes explicit use of a topographically organized map, a biologically important form of network organization. It differs in several important aspects from a related, earlier approach by Kuperstein, also based on topographic maps for the control of a robot arm [6]–[9]. The first difference is the use of a single, three-dimensional map instead of a set of additively coupled, one-dimensional maps in [9]. This has an important bearing on the class of input-output transformations which can be implemented accurately. In Kuperstein's work each map processes only a single variable. As a consequence, each of the outputs a_m to the actuators is necessarily of the form (cf. [9]):

$$a_m = F_1(x_L) + F_2(x_R) + F_3(y_L) + F_4(y_R) + F_5(x_L - x_R) + F_6(y_L - y_R) \quad (9)$$

where x_L, y_L, x_R, y_R are the (Cartesian) coordinates of the target in the image plane of the left and right stereo camera respectively, and $F_1 \ldots F_6$ are functions for which the system can learn essentially arbitrary (apart from smoothness constraints) realizations. In our approach, each unit becomes responsible for a small subset of input values $\vec{u}_1 \equiv (x_L, y_L)$ and $\vec{u}_2 \equiv (x_R, y_R)$ and can associate an output value $F(x_L, y_L, x_R, y_R)$ with this subset which can (again apart from smoothness constraints) be independent from the values associated with any of the other subsets. Hence, arbitrary correlations between the four arguments x_L, y_L, x_R, y_R can be implemented, whereas the more restricted form (9) is biased towards functions that can be decomposed into a sum of independent contributions, each depending on a single coordinate value or one of the "disparities" $x_L - x_R$, $y_L - y_R$ only, and may encounter problems if the control function requires correlations between two or more variables, such as, e.g., the product $x_L \cdot y_L$. If one wishes to make no prior assumptions about the system, the relevance of such correlations is difficult to anticipate and may vary with details of the chosen implementation, such as, e.g., geometry of the robot arm or positioning of the cameras. In this case our less biased scheme seems more desirable. This increased flexibility, however, comes at a price. Having to assign units to a higher (here three) dimensional space would require a significantly higher number of units for a density of spacing that is comparable to that in six one-dimensional maps. The use of the interpolation matrices A_r bypasses this problem and allows the achievement of a high degree of accuracy with a moderate total number of adaptable weights (1680 output weights per degree of freedom + 1344 input weights in the present simulation).

The second major difference is the use of a map with an adaptable topographic ordering, in contrast to the fixed ordering in Kuperstein's model. This allows the map not only to adapt its outputs, but also to adapt to the range of its input signals, thereby eliminating any need for matching the system by hand to the range of the sensory signals to be processed. For instance, if the environment requires positioning the stereo cameras closer together, reduced

values of the observed stereo disparities will result. The corresponding reduction in stereo sensitivity of the system can then partly be counterbalanced by the adaptive capabilities of the input map. The plasticity of the input map also allows the system to make adaptive changes in the resolution of the represented control law. Frequently required movements become automatically represented at a higher resolution than less frequent movements. This behavior is reminiscent of biological motor systems, where frequently practiced movements also become more accurate.

For low dimensional spaces, the method may also offer an interesting alternative to backpropagation [19]. In contrast to backpropagation, a localized representation of the mapping is learned. This avoids the development of "hidden units," which is usually a slow process. The cost of training many independent units is kept small by imposing a topology among the units so that this topology matches the topology of the work space and each learning step is spread over a subset of neighboring units. Initially, the subsets are chosen large, resulting in rapid learning of the coarse mapping. As learning progresses, the size of the subsets is gradually reduced to refine the mapping more and more locally. This strategy allows efficient and accurate training of many units and should facilitate scaling up the system to a higher number of nodes for further improved accuracy.

REFERENCES

[1] M. A. Arbib, "Perceptual structures and distributed motor control," in *Handbook of Physiology: The Nervous System II. Motor Control*, V. B. Brooks, Ed. Bethesda, MD, 1981, pp. 1449-1480.

[2] D. L. Sparks and J. S. Nelson, "Sensory and motor maps in the mammalian superior colliculus," *TINS*, vol. 10, pp. 312-317, 1987.

[3] H. Ritter and K. Schulten, "Topology conserving mappings for learning motor tasks," in *Neural Networks for Computing, AIP Conf. Proc. 151*, J. S. Denker, Ed. (Snowbird, UT), 1986, pp. 376-380.

[4] H. Ritter and K. Schulten, "Extending Kohonen's self-organizing mapping algorithm to learn ballistic movements," in *Neural Computers*, R. Eckmiller and C. von der Malsburg, Eds. Heidelberg, W. Germany: Springer, 1987, pp. 393-406.

[5] H. Ritter, T. Martinetz, and K. Schulten, "Topology-conserving maps for learning visuomotor coordination," *Neural Networks*, vol. 2, pp. 159-168, 1988.

[6] S. Grossberg and M. Kuperstein, *Neural Dynamics of Adaptive Sensory-Motor Control*. Amsterdam, Holland: North-Holland, 1986.

[7] M. Kuperstein, "Adaptive visual-motor coordination in multijoint robots using parallel architecture," in *Proc. IEEE Int. Automat. Robotics* (Raleigh, NC), 1987, pp. 1595-1602.

[8] M. Kuperstein, "Neural model of adaptive hand-eye coordination for single postures," *Science*, vol. 239, pp. 1308-1311, 1988.

[9] M. Kuperstein and J. Rubinstein, "Implementation of an adaptive neural controller for sensory-motor coordination," *IEEE Control Syst. Mag.*, vol. 9, no. 3, pp. 25-30, 1989.

[10] Lane, Handelman, Gelfand, "A neural network computational map approach to reflexive motor control," in *Proc. 1988 IEEE Conf. Intelligent Control*, 1988.

[11] B. W. Mel, "A robot that learns by doing," in *AIP Proc. 1987, Neural Inform. Processing Syst. Conf.* (Denver, CO), 1987.

[12] W. T. Miller, "Real-time application of neural networks for sensor-based control of robots with vision," in *IEEE Trans. Syst., Man, Cybern.*, vol. 19, no. 4, pp. 825-831, 1989.

[13] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biolog. Cybern.*, vol. 43, pp. 59-69, 1982a.

[14] T. Kohonen, "Analysis of a simple self-organizing process," *Biolog. Cybern.*, vol. 44, pp. 135-140, 1982b.

[15] T. Kohonen, "Clustering, taxonomy, and topological maps of patterns," in *Proc. 6th Int. Conf. Pattern Recognition* (Munich, W. Germany), 1982c, pp. 114-128.

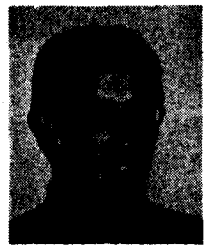
[16] B. Widrow and M. E. Hoff, "Adaptive switching circuits," in *WESCON Conv. Record*, pt. IV, 1960, pp. 96-104.

[17] S. Grossberg, "Contour enhancement, short-term memory, and constancies in reverberating neural networks," *Studies Appl. Math.*, vol. 52, pp. 213-257, 1973.

[18] A. P. Georgopoulos, A. B. Schwartz, and R. E. Kettner, "Neuronal population coding of movement direction," *Science*, vol. 233, pp. 1416-1419, 1986.

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533-536, 1986.

*



Thomas M. Martinetz was born in 1962. He studied physics and mathematics at the Universities of Cologne and Munich, Federal Republic of Germany. He received the Diplom degree in physics from the Technical University of Munich in 1988.

He is a fellow of the Volkswagen Foundation and is currently with the Department of Physics and with the Beckman Institute for Advanced Science and Technology at the University of Illinois at Urbana-Champaign. His main research is focused on the design of neural network architectures for robot and movement control, signal processing, and pattern recognition. As a Theoretical Physicist he is also interested in the theoretical and mathematical aspects of biological information processing and in analogies between physical systems and neural networks. He has coauthored one book and written several published articles on neural networks.

*

Helge J. Ritter was born in 1958. He studied physics and mathematics at the Universities of Bayreuth, Heidelberg, and Munich, Federal Republic of Germany. He received the Diplom degree in physics from the University of Heidelberg in 1983 and the Ph.D. degree in physics from the Technical University of Munich in 1988.

Since 1985 he has been engaged in research in the field of neural networks. His main interests are the study of biologically plausible strategies of massively parallel computation, in particular the analysis of neural network architectures for adaptive signal processing and for the coordination of robot movements. He is the author of several papers on neural algorithms, coauthor of one book, and was session co-chair of the IJCNN-89 Conference on Neural Networks. In 1989 he spent two months at the Laboratory of Computer and Information Science at Helsinki University of Technology. Recently he has moved to the University of Illinois at Urbana-Champaign, where he is carrying out research in the Department of Physics and at the newly established Beckman Institute.

*

Klaus J. Schulten was born in 1947. He received the Diplom degree in physics from the University of Münster, Federal Republic of Germany, in 1969, and the Ph.D. degree in chemical physics from Harvard University in 1974.

In 1974 he joined the Max-Planck-Institute for Biophysical Chemistry in Göttingen and in 1980 became Professor of 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. His research areas are many-particle physics, statistical mechanics, molecular biophysics, and computational neural science. In the latter area he studies currently the problems of motion synthesis in robots and data compression through adaptive filters.