3D-Neural-Net for Learning Visuomotor-Coordination of a Robot Arm

Thomas M. Martinetz[†], Helge J. Ritter [†] and Klaus J. Schulten[‡]

Department of Physics

Technical University of Munich

D-8046 Garching, FRG

Abstract: An extension of Kohonen's self-organizing mapping algorithm together with an error-correction rule of Widrow-Hoff-type is applied to develop an unsupervised learning scheme for the visuo-motor-coordination of a simulated robot arm. Using input signals from a pair of cameras, the "closed" robot arm system is able to reduce its positioning error to about 0.3% 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.

Keywords: Visuo-Motor-Coordination, Topology-Conserving Maps, unsupervised Learning, Motor Control, Robotics.

1. The model

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 (Arbib 1981). Only recently, topology conserving maps have been recognized as important for the generation of output for motor control (Sparks and Nelson 1987) and theoretical approaches using topology conserving maps for robot control have been proposed (Ritter and Schulten 1986, 1987; Ritter, Martinetz, Schulten 1988; Grossberg and Kuperstein 1986; Kuperstein 1987, 1988).

Our approach is based on an extension of Kohonen's self-organizing mapping algorithm (Kohonen 1982a-c) suggested earlier by two of the authors (Ritter and Schulten 1986, 1987) and an error-correction rule of Widrow-Hoff-type (Widrow and Hoff 1960), which is capable of learning 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 learnt from a sequence of random movements of the arm, which are observed by cameras and used to gradually improve the map. The topology-conserving map allows neighboring units to cooperate during learning, which greatly contributes to the efficiency and robustness of the algorithm. To maximally exploit this feature, we choose a network topology which mimics the topology of the space of relevant input signals as closely as possible and therefore is three-dimensional (Ritter, Martinetz, Schulten 1988).

[†] On leave to Beckman Center, University of Illinois at Urbana-Champaign, IL 61801

[‡]University of Illinois at Urbana-Champaign, IL 61801

Figure 1 shows the simulated three joint 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 (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 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.

2. The mapping algorithm

Each target object to be grasped by the manipulator of the robot arm induces an intensity distribution on the retinas of camera 1 and camera 2. We describe this distribution by a vector $\nu_{\alpha 1}$ (and $\nu_{\alpha 2}$ for camera 2), in which the element $\nu_{\alpha 1}$ denotes the intensity at pixel element α of the CCD-module of camera 1. $\nu_{\alpha 1}$ and $\nu_{\alpha 2}$ yield the information for two filters, which have to extract the location of the object within the three-dimensional work space. For our purpose, because we want to investigate mainly the aspects of motorcontrol our robot system has to deal with, we use a very simple "filter", which extracts the center of the intensity distribution on each camera retina. The postions of these centers are given by

$$\vec{u}_1 = \frac{1}{S_1}\sum_{\alpha} \nu_{\alpha 1}\vec{u}_{\alpha 1} \qquad \vec{u}_2 = \frac{1}{S_2}\sum_{\alpha} \nu_{\alpha 2}\vec{u}_{\alpha 2} \qquad (1)$$

with $\vec{u}_{\alpha i}$ as location of pixel element α on camera retina i. S_i denotes the whole intensity camera i receives and normalizes the vector $\nu_{\alpha i}$. This simple filtering is sufficient, if, for example, the light intensity $I(\vec{r}\,)$ is extremely peaked at the object location within the work space.

We group both two-dimensional retina locations $\vec{u}_1$ and $\vec{u}_2$ to a four-dimensional vector **u** which then carries 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 $\vec{\theta}(\mathbf{u})$ 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 of the positions of both cameras relative to the work space and shall be learned automatically by an unsupervised learning procedure.

The control law is adaptively represented by a "winner-take-all"-network of formal neurons, receiving the sensory input **u** in parallel. Each neuron **r** is "responsible" for some small sub-

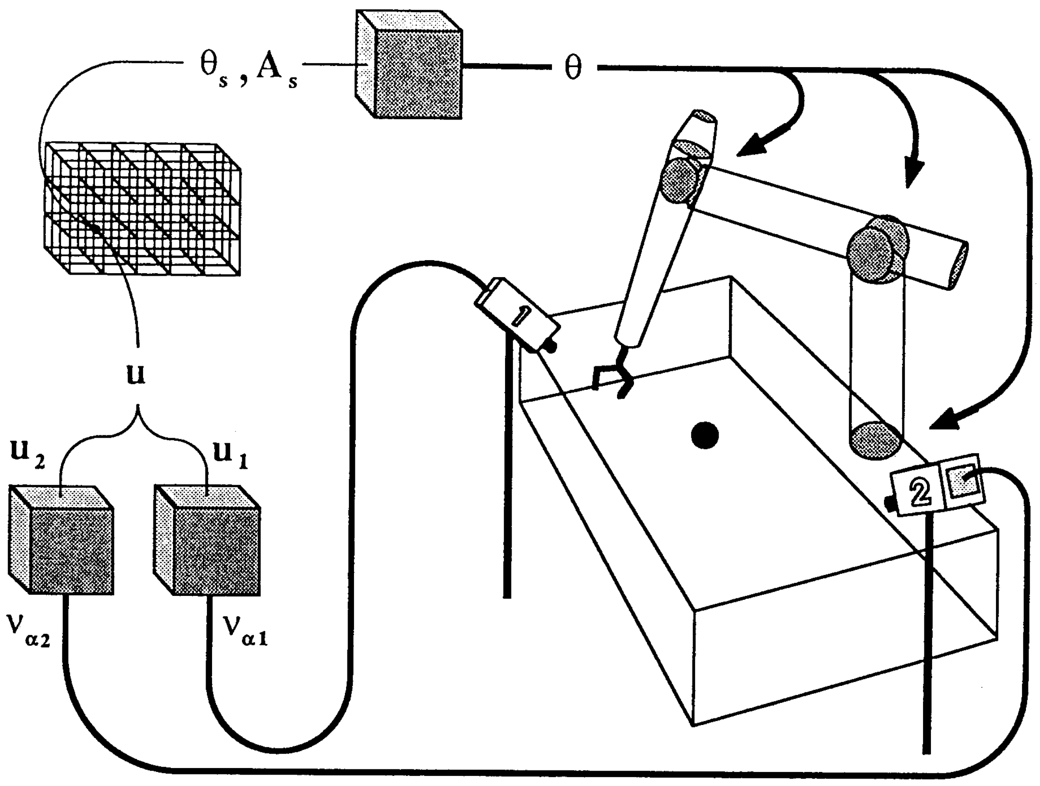

Fig.1 The simulated robot system. Two cameras observe the robot arm to the right of the work space whose borders are indicated schematically by lines. Each camera provides the intensity distribution ν_α on its "retina" to a filter, which extracts the retinal coordinates $\vec{u}$ of the object the manipulator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

set (its "receptive field") $F_\mathbf{r}$ of the four-dimensional input space U. Whenever $\mathbf{u} \in F_\mathbf{r}$, neuron $\mathbf{r}$ determines the output. In the nervous system, the output will be specified by the average behaviour of a localized subpopulation comprising many simultaneously active neurons with overlapping receptive fields (Georgopoulos et al. 1986). Their average behaviour 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 $F_\mathbf{r}$, a vector $\mathbf{w_r} \in U$ is associated with each neuron. The vectors $\mathbf{w_r}$ are chosen as pairs $\mathbf{w_r} = (\vec{w}_{\mathbf{r}1}, \vec{w}_{\mathbf{r}2})$ of two component vectors $\vec{w}_{\mathbf{r}1}$, $\vec{w}_{\mathbf{r}2}$. $\vec{w}_{\mathbf{r}i}$ 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 $\vec{w}_{\mathbf{r}1}$ and $\vec{w}_{\mathbf{r}2}$ on the two camera "retinas".

To specify the required output, a vector $\vec{\theta}_\mathbf{r}$ together with a 3×4 matrix $\mathbf{A_r}$ are associated with each neuron in addition to $\mathbf{w_r}$. The system produces the joint angles $\vec{\theta}(\mathbf{u}) = (\theta_1, \theta_2, \theta_3)$ by using $\vec{\theta}_\mathbf{r}$ and $\mathbf{A_r}$ to specify the first two terms of a Taylor expansion, i.e.

$$\vec{\theta}(\mathbf{u}) = \vec{\theta}_\mathbf{s} + \mathbf{A_s}(\mathbf{u} - \mathbf{w_s}). \tag{2}$$

Here $\mathbf{s}$ is the neuron that was responsible for the received input. Neuron $\mathbf{s}$ "wins" whenever $\mathbf{u} \in F_\mathbf{s}$, where $F_\mathbf{s}$ consists of all points of U, which are closer to $\mathbf{w_s}$ than to any other $\mathbf{w_r}$, $\mathbf{r} \neq \mathbf{s}$. That means

$$F_\mathbf{s} = \{\mathbf{u} \in U \mid \|\mathbf{w_s} - \mathbf{u}\| \leq \|\mathbf{w_r} - \mathbf{u}\| \; \forall \mathbf{r}\}. \tag{3}$$

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

3. The learning procedure

During the learning phase, the objects the robot arm shall reach for are presented at different, randomly chosen locations within the work space. For each sensory input $\mathbf{u}$, induced by an object, the network output specified by $(\vec{\theta}, \mathbf{A})_\mathbf{s}$ 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 $(\vec{\theta}^*, \mathbf{A}^*)$ 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

$$\mathbf{w}_\mathbf{r}^{new} = \mathbf{w}_\mathbf{r}^{old} + \epsilon h_{\mathbf{rs}}(\mathbf{u} - \mathbf{w}_\mathbf{r}^{old}), \tag{4}$$
$$(\vec{\theta}, \mathbf{A})_\mathbf{r}^{new} = (\vec{\theta}, \mathbf{A})_\mathbf{r}^{old} + \epsilon' h'_{\mathbf{rs}}((\vec{\theta}, \mathbf{A})^* - (\vec{\theta}, \mathbf{A})_\mathbf{r}^{old}). \tag{5}$$

Here $\mathbf{s} = \mathbf{s}(\mathbf{u})$, the neuron selected by input $\mathbf{u}$, 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 $\mathbf{r}$ in a lattice, normally a two-dimensional sheet, and the coefficients $h_{\mathbf{rs}}$, $h'_{\mathbf{rs}}$ 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 normalisation $h_{\mathbf{ss}} = h'_{\mathbf{ss}} = 1$). Hence, neighboring neurons in the sheet share adaptation steps with the same input and get tuned to similar inputs $\mathbf{u}$. Kohonen 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 (Kohonen 1986a-c). Our algorithm extends his method by associating with each formal neuron a second piece of data, the output quantity $(\vec{\theta}, \mathbf{A})_\mathbf{r}$ (Ritter and Schulten 1986, 1987; Ritter, Martinetz, Schulten 1988). 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 $(\vec{\theta}, \mathbf{A})$, and the net. Both maps develop simultaneously and therefore get matched in such a way as to approximate the desired input-output-relationship $\vec{\theta}(\mathbf{u})$. The resulting representation is an adaptive discretization, which adjusts its resolution dynamically to the probability density of the required control actions $\vec{\theta}(\mathbf{u})$ by allocating neurons only to those regions of U actually required for representing the control law $\vec{\theta}(\mathbf{u})$.

For the neurons, we choose the topological arrangement of a 3D-lattice of $7 \times 12 \times 4$ units. The three-dimensionality is not directly suggested from the situation in the cortex, where the neurons are arranged in a more sheet-like fashion, but for our technical application, however, the three-dimensional topology offers an essential advantage: Both retinal coordinates are combined to the input vector $\mathbf{u} = (\vec{u}_1, \vec{u}_2) \in U$ and define a four-dimensional space. However, because the given target locations are all chosen from a three-dimensional work space, every $\mathbf{u}$ produced by an object within this work space is an element of a subspace of U, which is only three-dimensional as well. Therefore, it is convenient to take as connectivity between the neurons of the net the topology of a 3D-lattice, which reflects the dimensionalty of the space occupied by the input signals. The location of the relevant three-dimensional subspace in U is determined by the positions of the cameras relativ to the work space and therefore initially unknown to the learning algorithm of the robot system. To be able to learn the required task, the robot system has to find this subspace by its own. This is done by the Kohonen-algorithm. $\mathbf{w}_\mathbf{r}$ determines the receptive field, the part of the four-dimensional space the neuron $\mathbf{r}$ has to look at. As we know, the Kohonen-algorithm organizes the receptive fields of the neural net by using the sensory inputs actually received by the network. Hence, the Kohonen-algorithm has the feature only to map these sensory inputs, in our case the target locations $\mathbf{u}$, which really occur during the learning phase. That means, at the end the neurons of our 3D-lattice are allocated only to these inputs belonging to the relevant subspace of U.

4. The error-correction rule

In the absence of any further information, starting values for $\mathbf{w}_\mathbf{r}$ and $\vec{\theta}_\mathbf{r}, \mathbf{A}_\mathbf{r}$ 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 $\mathbf{u}$ for this target location selects a neuron $\mathbf{s}$ "looking" at $\mathbf{u}$, i.e. $\mathbf{u} \in F_\mathbf{s}$, for determining this movement. From the actual outcome of the movement we derive an improved estimate $(\vec{\theta}^*, \mathbf{A}^*)$ and perform an adjustment according to (4) and (5). To obtain $\vec{\theta}^*$ and $\mathbf{A}^*$, the following strategy is used (Ritter, Martinetz, Schulten 1988). First the array generates motor output for a "gross movement", which results from setting the joint angles to the values $\vec{\theta}_\mathbf{s}$ 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 $\mathbf{v}_i$. The gross movement is followed by a "fine movement" by switching on the linear correction term in (2). Denoting the resulting retinal coordinates of the end effector by $\mathbf{v}_f$, we take as improved estimates $\vec{\theta}^*$, $\mathbf{A}^*$

$$\vec{\theta}^* = \vec{\theta}_\mathbf{s} + \mathbf{A}_\mathbf{s}(\mathbf{u} - \mathbf{v}_f), \tag{6}$$
$$\mathbf{A}^* = \mathbf{A}_\mathbf{s} + \mathbf{A}_\mathbf{s}(\mathbf{u} - \mathbf{w}_\mathbf{s} - \mathbf{v}_f + \mathbf{v}_i)(\mathbf{v}_f - \mathbf{v}_i)^T||\mathbf{v}_f - \mathbf{v}_i||^{-2} \tag{7}$$

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

$$\mathbf{A}^* = \mathbf{A}_\mathbf{s} + (\Delta\vec{\theta} - \mathbf{A}_\mathbf{s}\Delta\mathbf{u})\Delta\mathbf{v}^T||\Delta\mathbf{v}||^{-2}, \tag{8}$$

where $\Delta\mathbf{v} = \mathbf{v}_f - \mathbf{v}_i$ and $\Delta\vec{\theta} = \mathbf{A}_\mathbf{s}^{true}\Delta\mathbf{v}$ 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 $\mathbf{A}_\mathbf{s}^{true}$ to which $\mathbf{A}_\mathbf{s}$ shall converge, (8) is seen to be equivalent to

$$\mathbf{A}^* = \mathbf{A}_\mathbf{s} + (\mathbf{A}_\mathbf{s}^{true} - \mathbf{A}_\mathbf{s})\Delta\mathbf{v}\Delta\mathbf{v}^T||\Delta\mathbf{v}||^{-2}, \tag{9}$$

i.e. a linear error correction rule for $\mathbf{A}_\mathbf{s}$.

5. The 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 $h_{\mathbf{rs}}$ was taken to be the Gaussian

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

and $h'_{\mathbf{rs}}$ 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 values were chosen as follows: $\epsilon_i = 1$, $\epsilon_f = 0.005$, $\epsilon'_i = 1$, $\epsilon'_f = 0.7$, $\sigma_i = 2.5$, $\sigma_f = 0.1$, $\sigma'_i = 1.5$ und $\sigma'_f = 0.05$. Figure 2 and figure 3 show the results of the simulation from the view of camera 2. Figure 2 shows the state of the mapping $\mathbf{r} \mapsto \mathbf{w}_\mathbf{r}$ relevant to camera 2 initially, after 6 000 and after 30 000 learning steps respectively. Each node $\mathbf{r}$ of the lattice is mapped to a location $\vec{w}_{\mathbf{r}2}$ in the image plane of camera 2. Values associated with lattice neighbors are connected by lines to visualize the lattice topology. Initially the vectors $\vec{w}_{\mathbf{r}1}$ and $\vec{w}_{\mathbf{r}2}$ were distributed randomly in the image plane of their camera. This provided a homogenous distribution of the values $\mathbf{w}_\mathbf{r}$ over the four-dimensional input space and the corresponding image of the lattice is highly irregular (left diagram). After only

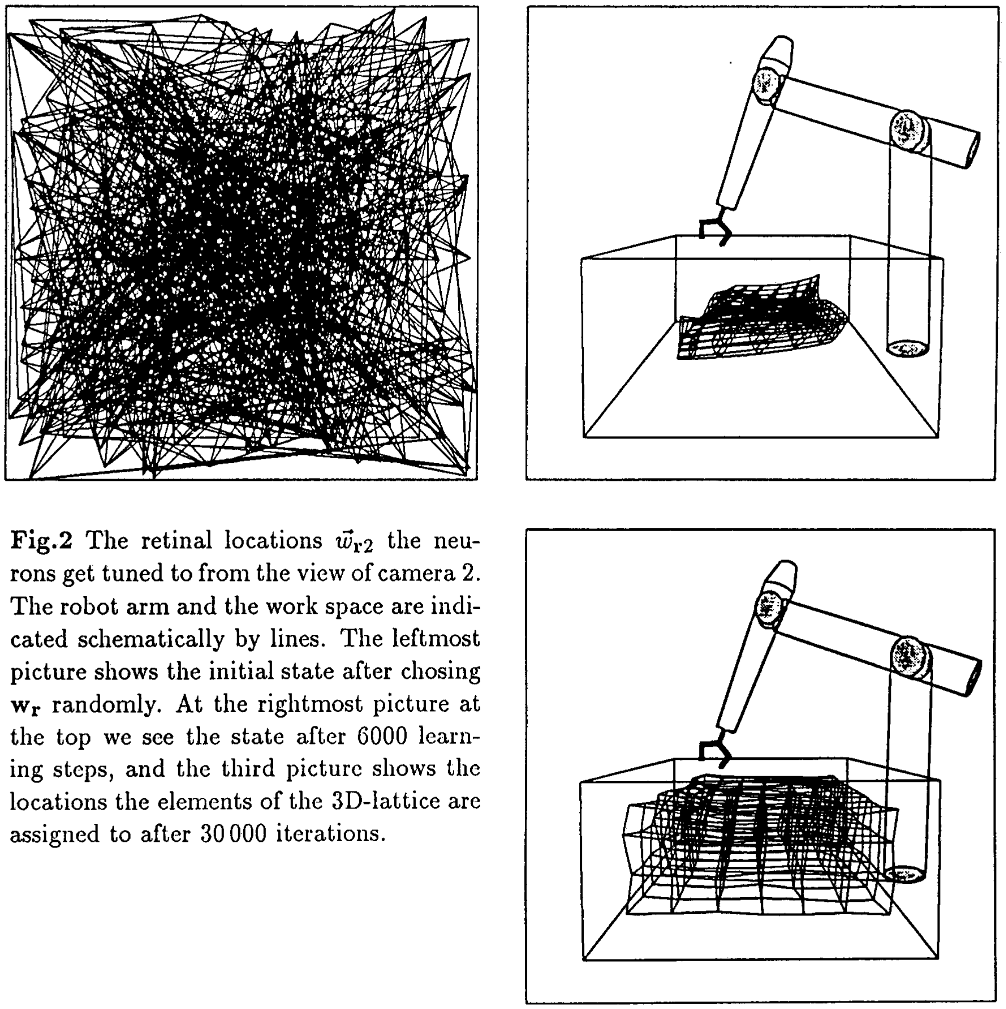

Fig.2 The retinal locations $\vec{w}_{r2}$ the neurons get tuned to from the view of camera 2. The robot arm and the work space are indicated schematically by lines. The leftmost picture shows the initial state after chosing $\mathbf{w_r}$ randomly. At the rightmost picture at the top we see the state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

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 $\mathbf{w_r}$.

The leftmost pictures of Fig.3 show the mismatches between intended target positions and actually achieved end effector locations which occur for the subset of visual inputs $\mathbf{u} = \mathbf{w_r}$. 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 left). However, after 6 000 learning steps all errors have markedly decreased (center), until finally (30 000 steps, bottom) mismatches are no longer visible.

As the 3×4 Jacobians $\mathbf{A_r}$ cannot easily be visualized directly, we instead show for each location $\mathbf{v}_i$ the reaction of the end effector to three required test movements. These test movements are of equal length and directed parallel to the borders of the work space. If $\mathbf{A_r}$ is correct, the end effector will trace out little three-legged patterns, testing $\mathbf{A_r}$ 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 Jacobians were chosen by assigning a random value from the interval $[-20, 20]$ to each element of $\mathbf{A_r}$. Therefore, the initial test movements are very poor. However, after 6000 iterations the test movements are seen to be already much better, and after 30 000 learning steps, they are traced out very accurately.

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.

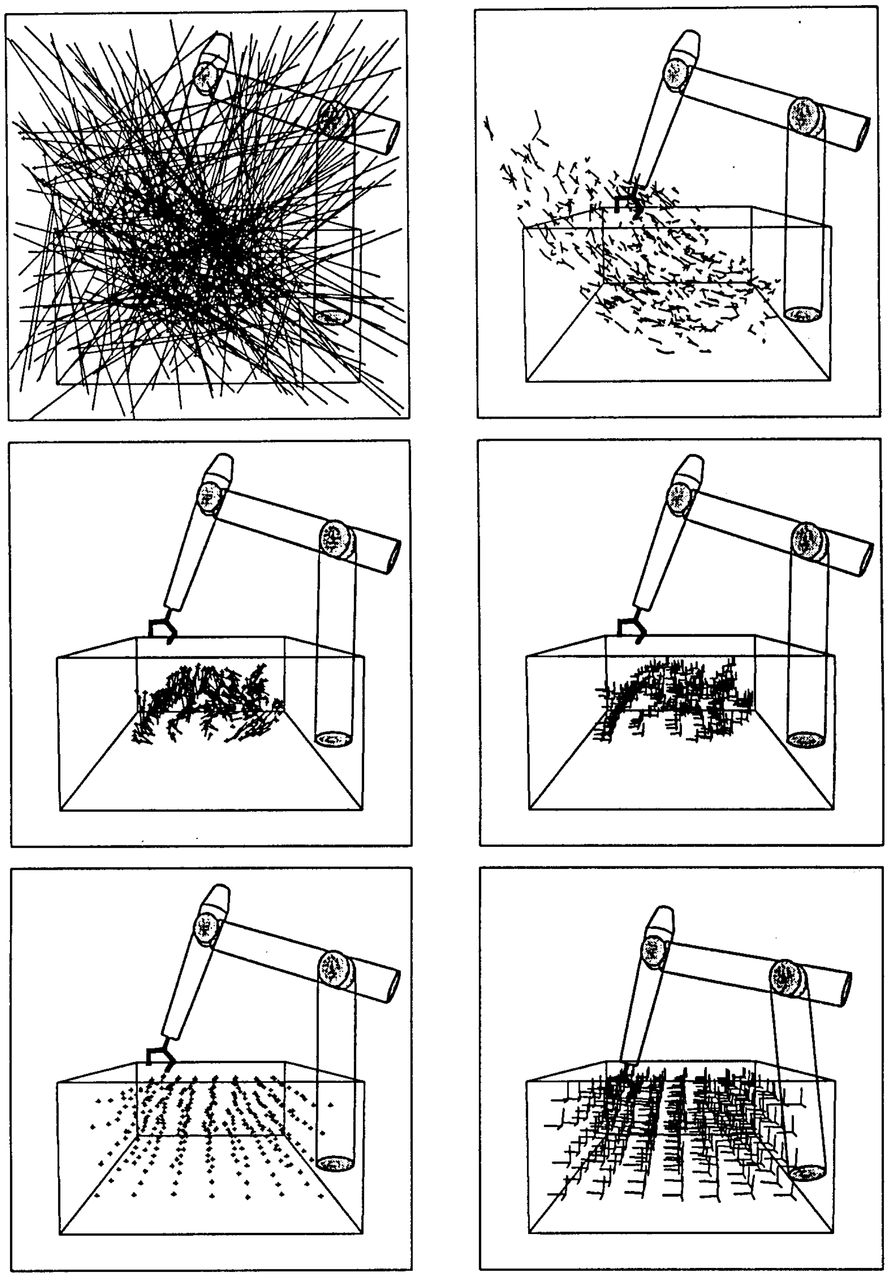

Fig.3 The performence of the output at the start, after 6000 and after 30 000 trial movements. The leftmost pictures show the end effector locations $\mathbf{v}_i$ (cross marks) resulting from visual input $\mathbf{u} = \mathbf{w}_\mathbf{r}$ (i.e joint angles are $\vec{\theta}_\mathbf{r}$), together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

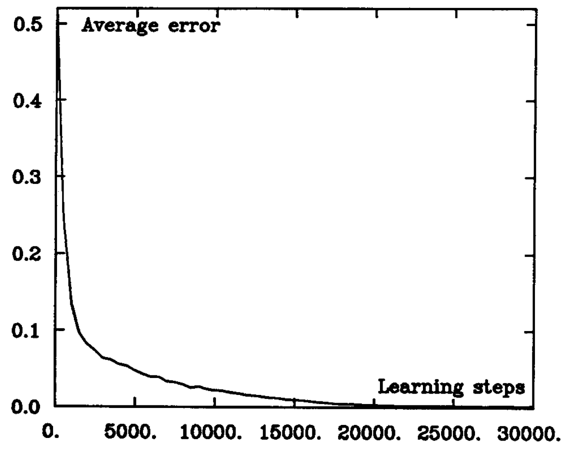

Abb.4 Average positioning error versus the number of learning steps. The error decreases rapidly to a final value of $1.7 \cdot 10^{-3}$ units after 30 000 learning steps.

6. Conclusion

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 continous 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. For low dimensional spaces, the method may offer an interesting alternative to backpropagation (Rumelhart et al. 1986).

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

References:

Arbib MA (1981) Perceptual Structures and distributed Motor Control. In: Handbook of Physiology: The Nervous System II. Motor Control (Brooks VB Ed.) 1449-1480. Bethesda, Maryland.

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. North Holland, Amsterdam.

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 Int. Conf. on Pattern Recognition, Munich pp 114-128

Kuperstein M. (1987) Adaptive Visual-Motor Coordination in Mulitjoint Robots using Parallel Architecture. Proc. IEEE Int. Conf. Automat. Robotics 1595-1602, Raleigh NC

Kuperstein M. (1988) Neural Model of Adaptive Hand-Eye Coordination for Single Postures. Science 239:1308-1311

Ritter H., Schulten K. (1986) Topology Conserving Mappings for Learning Motor Tasks. In Denker J.S. (Ed), Neural Networks for Computing, AIP Conference Proceedings 151, Snowbird, Utah, pp. 376-380

Ritter H., Schulten K. (1987) Extending Kohonen's Self-Organizing Mapping Algorithm to Learn Ballistic Movements. In Eckmiller R. and von der Malsburg C. (Eds.), Neural Computers, Springer, Heidelberg, pp. 393-406

Ritter H., Martinetz T., Schulten K. (1988) Topology-Conserving Maps for Learning Visuomotor-Coordination. Neural Networks 2 (in press).

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. TINS 10:312-317

Widrow B., Hoff M.E. (1960) Adaptive switching circuits, WESCON Convention Record, part IV pp. 96-104