A Neural Network with Hebbian-like Adaptation Rules Learning Visuomotor Coordination of a PUMA Robot

Thomas Martinetz[†] and Klaus Schulten[‡]

[†]Siemens AG, Corporate Research and Development
Otto-Hahn-Ring 6, W-8000 Munich 83, Germany

[‡] Beckman Institute and Department of Physics
University of Illinois, Urbana-Champaign, Il 61801, USA

***Abstract*—A hybrid neural network algorithm which employs superpositions of linear mappings is presented, and its application to the task of learning the end effector positioning of a robot arm is described. The learning and the control of the positioning is accomplished by the network solely through visual input from a pair of cameras. In addition to the learning of the a priori unknown input-output relation from target locations seen by the cameras to corresponding joint angles, the network provides the robot with the ability to perform feedback-guided corrective movements. This allows one to divide the positioning movement into an initial, open-loop controlled positioning and subsequent feedback-guided corrections, a divison which resembles the strategy for fast goal-directed arm movements of humans. For the robot arm which we employed, a PUMA 560, the neural network algorithm achieves a final positioning error of about 1.3 mm, the lower bound given by the finite resolution of the cameras. Because of the feedback loops, the LMS error correction rules for the weights of the network have the form of Hebbian learning rules, except that instead of the product of the pre- and post-synaptic excitation, it is the product of their time derivatives that determines the adjustment.**

I. Introduction

Robot arm systems which are capable of learning their task autonomously by observing their own trial movements will play an important role in future applications. A number of adaptive movement control schemes based on artificial neural networks and dealing with a variety of different movement tasks have been proposed in recent years [1-3]. One of these neural network approaches which, in particular, has been developed for visuomotor coordination of a robot arm is based on Kohonen's topology conserving feature map [4-6].

In this paper we present a modified version of the neural network algorithm introduced in [5, 6] and report about the results we obtained with an implementation on a PUMA 560 robot arm. Instead of Kohonen's feature map algorithm [7] for representing the task space we now employ the so-called "neural-gas" algorithm which has been introduced recently [8]. Instead of an external lattice structure to which the neural units are assigned, the "neural-gas" network employs a "neighborhood-ranking" of the units within the task space for forming the representation. This allows a more flexible adaptation to topologically intricately structured task spaces since no *a priori* given lattice which has to match the task space structure is required [8]. At the same time, the "neighborhood-ranking" provides information about neural units which are tuned to similar

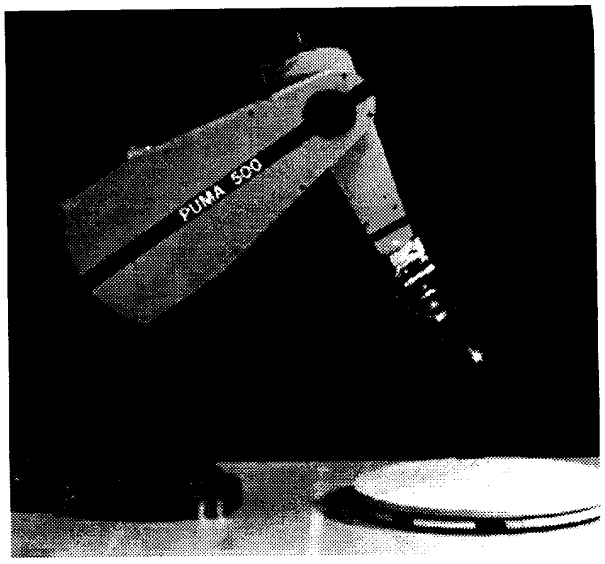

Fig. 1: The PUMA 560. In our experiments the end effector was marked by a small light bulb at its end to make it easily identifiable by the vision system.

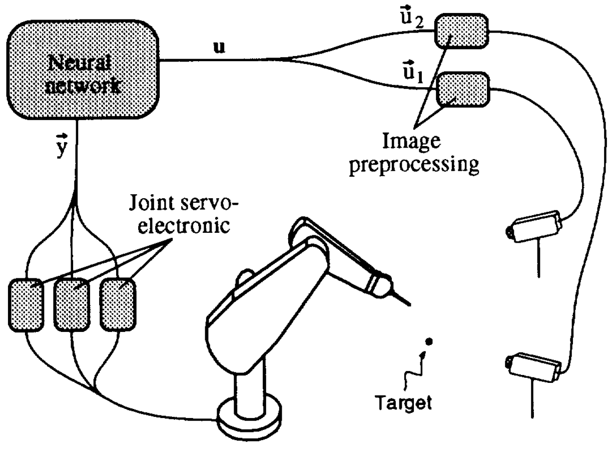

Fig. 2: The set-up of the robot system. The image preprocessing board of each camera provides a two-dimensional vector $\vec{u}$, the coordinates of the image of the target object on the camera's image plane. Both vectors $\vec{u}$ are combined into a four-dimensional vector $\mathbf{u}$ which is the input of the neural network. The output of the neural net is a three-dimensional vector $\vec{y}$, each component of which specifies the angle of one of the three joints of the robot.

tasks and, therefore, the output values of which may be adapted similarly, corresponding to the concerted adaptation of neighboring units within a Kohonen lattice. Another modification compared to the algorithms in [5, 6] concerns the use of the local linear mappings learned by the neural network to generate its output values. These local linear mappings enable the robot arm to perform feedback-guided corrective movements. We will show that by performing a whole sequence of these corrective movements the adaptation rules for the weights of the network take on a form similar to Hebb's learning rule [9].

II. The Task and the Configuration of the Robot System

In Fig. 1 we see the robot arm, a commercially available PUMA 560. The task of the robot arm is to position its end effector to visually designated target objects which during the learning phase are randomly chosen within the work space in front of the robot. Since we are interested in the problems involved in learning the robot control task and not in the problems of image segmentation and object recognition, we choose as the target object a small light bulb which can easily be identified by the vision system. Neither the geometry of the robot arm nor the position of the two cameras which provide the visual information about the target object's location are known *a priori*. The robot can rotate around the vertical axes and move its upper and lower arm. Since the wrist is fixed, the robot arm has three degrees of freedom. Figure 2 shows a sketch of the configuration. Two CCD cameras with a resolution of 560 × 480 pixels, respectively, observe the work space. Each camera has attached to it an image preprocessing board (ICS-400, Androx Inc.) which extracts the pixel coordinates $\vec{u}$ of the image of the target object on the camera's image plane. The resulting pair of two-dimensional coordinates are combined to a four-dimensional vector $\mathbf{u} = (\vec{u}_1, \vec{u}_2)$, the input signal for the neural network which is simulated on a SUN 4/370. For a more detailed description of the set-up see [10].

The robot arm's movement for approaching the presented target object consists of two phases, comparable to fast goal-directed arm movements of humans [11]. The first phase is an open-loop controlled gross positioning of the end effector, whereas in the second phase visual feedback is used to perform the fine positioning. The visual feedback which is provided by the cameras consists of information about the deviation of the current end effector position from its target location.

By using the pixel coordinates $\mathbf{u} = (\vec{u}_1, \vec{u}_2)$ of the two images of the target object the neural network controller generates its first output $\vec{y}(t_0)$, a set of three joint angle values which are fed into the servocontrollers of the robot arm and lead to the gross positioning of the end effector (the neural network controller and how it generates the output is explained in the next section). The resulting end effector location is observed by the cameras which yield its pixel coordinates, a four-dimensional vector $\mathbf{v}_0$. The residual deviation $\mathbf{u} - \mathbf{v}_0$ from the desired target location is used to perform the first feedback-guided corrective movement. For this purpose the sum $\mathbf{u} + (\mathbf{u} - \mathbf{v}_0)$ instead of $\mathbf{u}$ is taken as input for the neural network which leads to a second output $\vec{y}(t_1)$ and a second end effector location $\mathbf{v}_1$.

Taking $\mathbf{u}+(\mathbf{u}-\mathbf{v}_0)$ as the input for the first corrective movement corresponds to reaching for a *virtual* target which was shifted from the real target location $\mathbf{u}$ by the residual distance $\mathbf{u} - \mathbf{v}_0$ the end effector still has to move. Reaching for this virtually modified target location leads to a correction of the previous positioning movement. After the first corrective movement the residual error becomes $\mathbf{u} - \mathbf{v}_1$ and the virtual target is shifted to $\mathbf{u}+(\mathbf{u}-\mathbf{v}_0)+(\mathbf{u}-\mathbf{v}_1)$ which is the input for the second corrective movement, etc. The described feedback loop can be performed several times. We interrupt the positioning process after the fourth corrective movement and continue the training by presenting the target object at another location.

III. The Architecture of the Neural Network

Figure 3 shows the architecture of the neural network employed. The network consists of three linear output units l, $l = 1, 2, 3$, one for each joint of the robot arm (only one output unit, depicted as a square, is shown in Fig. 3). Each output unit has N input lines, each of which receives the same input, either the current target location $\mathbf{u}$ or the current virtual target location $\mathbf{u} + \mathbf{u} - \mathbf{v}_0 + ... + \mathbf{u} - \mathbf{v}_{i-1}$. To each

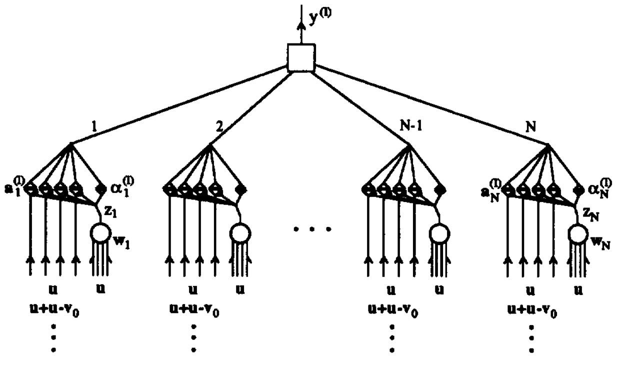

Fig. 3: The architecture of the neural network. The neural units $n = 1, ..., N$ which are depicted as circles gate the input of the output units $l = 1, 2, 3$, only one of which is shown (denoted as square). The input at each of the N input lines of output unit l are either the image coordinates $\mathbf{u}$ of the current target location or the image coordinates $\mathbf{u} + \mathbf{u} - \mathbf{v}_0 + ... + \mathbf{u} - \mathbf{v}_{i-1}$ of the current virtual target location. The weights $\mathbf{a}_n^{(l)} \in \Re^4$ and $\alpha_n^{(l)} \in \Re$ at each input line define a linear mapping from target locations to joint angles. The output $y^{(l)}$ is given by the sum over the contributions from all input lines, i.e., all linear mappings.

input line a "gating" unit n, $n = 1, ..., N$ is connected (depicted as circle in Fig. 3). The larger the output z_n of gating unit n, the more the connected input line contributes to the output signal $y^{(l)}$ of output unit l. As we will see below, the output z_n is large for gating units n with their weight vector $\mathbf{w}_n$ close to the target coordinates $\mathbf{u}$.

The output units function as sigma-pi units [12], and at each of their N input lines they perform a weighted sum

$$\begin{gathered}\alpha_n^{(l)} z_n + \mathbf{a}_n^{(l)} \cdot (z_n \mathbf{u}) \\ l = 1, 2, 3 \quad n = 1, ..., N\end{gathered} \tag{1}$$

over the current input signal $\mathbf{u}$ which has been gated by z_n, plus z_n itself. The dot in (1) denotes the dot product of two vectors. The weights at input line n of output unit l are denoted by the scalar $\alpha_n^{(l)}$ and by the four-dimensional vector $\mathbf{a}_n^{(l)}$. At each of the N input lines the weights $(\alpha_n^{(l)}, \mathbf{a}_n^{(l)})$ determine a linear mapping from target locations to joint angles. The final output $y^{(l)}$ of output unit l is given by the sum

$$y^{(l)} = \sum_{n=1}^{N} \alpha_n^{(l)} z_n + \mathbf{a}_n^{(l)} \cdot (z_n \mathbf{u}) \qquad (2)$$

over the contributions from all input lines, i.e., all linear mappings.

The receptive field of a gating unit n is determined by its four-dimensional weight vector $\mathbf{w}_n$. In the neural network model presented, the output z_n of gating unit n is given by its "neighborhood-rank" to the target location $\mathbf{u}$, i.e.,

$$z_n(\mathbf{u}) = e^{-k_n/\lambda}$$

with k_n as the number of gating units m which have their weight vector $\mathbf{w}_m$ closer to $\mathbf{u}$ than gating unit n, i.e., with $\|\mathbf{u} - \mathbf{w}_m\| < \|\mathbf{u} - \mathbf{w}_n\|$. z_n is unity for $k_n = 0$ and decreases monotonically for increasing k_n with a characteristic decay constant λ. Hence, z_n is largest for the gating unit with its weight vector $\mathbf{w}_n$ closest to the target location $\mathbf{u}$, second largest for the gating unit with its weight vector second closest to the target location, etc.

Employing a Hebb-rule with a linear decay term leads to an update rule for the weight vectors $\mathbf{w}_n$, $n = 1, ..., N$, of the form

$$\begin{aligned} \Delta \mathbf{w}_n &\propto z_n \mathbf{u} - z_n \mathbf{w}_n \\ &\propto e^{-k_n/\lambda} (\mathbf{u} - \mathbf{w}_n) \end{aligned}$$

which is the adaptation step of the "neural-gas" network proposed in [8]. Instead of the neighborhood within an external lattice as in Kohonen's feature map algorithm, it is the *neighborhood rank* of the weight vector $\mathbf{w}_n$ to the target location $\mathbf{u}$ which determines the size of the adaptation step. Starting with a large λ which slowly decreases to small values, the receptive fields of the gating units, determined by $\mathbf{w}_n$, become distributed over the relevant parts of the task space, i.e., those parts where input signals were located in the past. Compared to Kohonen's feature map which was employed in [5, 6] the "neural-gas" network adjusts the weights $\mathbf{w}_n$ more quickly to the relevant regions of the input space without the need of prespecifying the topology of an external lattice, which would require *a priori* knowledge about the topological structure of the task space [8].

To perform the first, open-loop controlled positioning movement, each of the three linear output units l receives the current target location $\mathbf{u}$ at each of its N input lines and generates (see eq.(2))

$$\begin{aligned} y^{(l)}(t_0) &= \sum_{n=1}^{N} \alpha_n^{(l)} z_n + \mathbf{a}_n^{(l)} \cdot (z_n \mathbf{u}) \\ &= \sum_{n=1}^{N} z_n \left(\alpha_n^{(l)} + \mathbf{a}_n^{(l)} \cdot \mathbf{u} \right). \end{aligned} \qquad (3)$$

as output value. Hence, the output is given by a superposition of linear mappings, with the largest contribution from the linear mapping which is associated with the gating unit with its $\mathbf{w}_n$ closest to $\mathbf{u}$ ($k_n = 0$), the second largest contribution from the linear mapping associated with the gating unit with its $\mathbf{w}_n$ second closest to $\mathbf{u}$ ($k_n = 1$), etc. Compared to [5, 6], we now employ a "soft" instead of a "winner-take-all" rule for selecting the linear mapping most specialized for generating the currently required output.

After the gross positioning a number of feedback-guided corrective movements are performed. For the i-th of these feedback-guided corrective movements instead of the target location $\mathbf{u}$ the corresponding virtual target location $\mathbf{u} + \mathbf{u} - \mathbf{v}_0 + ... + \mathbf{u} - \mathbf{v}_{i-1}$ is taken as input for generating the output, yielding

$$\begin{aligned} y^{(l)}(t_i) &= \sum_{n=1}^{N} z_n (\alpha_n^{(l)} + \mathbf{a}_n^{(l)} \cdot (\mathbf{u} + \mathbf{u} - \mathbf{v}_0 + \\ &\qquad ... + \mathbf{u} - \mathbf{v}_{i-1})) \\ &= y^{(l)}(t_{i-1}) + \sum_{n=1}^{N} z_n \mathbf{a}_n^{(l)} \cdot (\mathbf{u} - \mathbf{v}_{i-1}) \end{aligned} \qquad (4)$$

for performing the i-th movement step. Through adjustments of $\alpha_n^{(l)}$ and $\mathbf{a}_n^{(l)}$ (3) has to adapt to the globally nonlinear transformation $y^{(l)}(\mathbf{u})$ from pixel coordinates $\mathbf{u}$ of a target location to the corresponding angle at joint l. In addition, as we can see from (4), $\sum_n z_n \mathbf{a}_n^{(l)}$ has to adapt to the Jacobian matrix of $y^{(l)}(\mathbf{u})$ at $\mathbf{u}$. In (3) the Jacobian matrix leads to a significantly higher precision of the open-loop movement than a zero order approximation through $\alpha_n^{(l)}$ alone, and in (4) it enables the robot to perform the feedback-guided corrective movements.

IV. The Learning Rules for the Output Units

Usually the deviations $\mathbf{u}-\mathbf{v}_i$ of the end effector from the target are small after the open-loop movement and the subsequent corrections. Therefore, we can take the small change $y^{(l)}(t_i)-y^{(l)}(t_{i-1})$ generated by a corrective movement together with the corresponding difference $\mathbf{v}_i - \mathbf{v}_{i-1}$ in the end effector's camera coordinates to obtain an improvement of the Jacobian $\sum_n z_n \mathbf{a}_n^{(l)}$ and through this an adaptation step for the $\mathbf{a}_n^{(l)}$s. Minimizing the quadratic cost function

$$C_1 = \frac{1}{2}\sum_{l=1}^{3}[y^{(l)}(t_i) - y^{(l)}(t_{i-1}) - \sum_{n=1}^{N} z_n \mathbf{a}_n^{(l)} \cdot (\mathbf{v}_i - \mathbf{v}_{i-1})]^2$$

via gradient descent yields for the change of $\mathbf{a}_n^{(l)}$ with each corrective movement

$$\Delta \mathbf{a}_n^{(l)}(t_i) = \epsilon' \cdot z_n[y^{(l)}(t_i) - y^{(l)}(t_{i-1}) - \sum_{n=1}^{N} z_n \mathbf{a}_n^{(l)} \cdot (\mathbf{v}_i - \mathbf{v}_{i-1})] \cdot (\mathbf{v}_i - \mathbf{v}_{i-1}).$$

If we denote the input $\mathbf{u} + (\mathbf{u} - \mathbf{v}_0) + (\mathbf{u} - \mathbf{v}_1) + ...$ after having been gated by z_n with

$$\mathbf{x}_n(t_0) = z_n\mathbf{u}, \quad \mathbf{x}_n(t_1) = z_n(\mathbf{u}+\mathbf{u}-\mathbf{v}_0),$$
$$\mathbf{x}_n(t_i) = z_n(\mathbf{u}+\mathbf{u}-\mathbf{v}_0 + ... + \mathbf{u}-\mathbf{v}_{i-1}),$$

then we obtain the adaptation rule

$$\Delta \mathbf{a}_n^{(l)}(t_i) = -\epsilon' \cdot \left[y^{(l)}(t_{i+1}) - y^{(l)}(t_i)\right] \cdot [\mathbf{x}_n(t_{i+1}) - 2\mathbf{x}_n(t_i) + \mathbf{x}_n(t_{i-1})]. \quad (5)$$

To derive an adaptation rule for $\alpha_n^{(l)}$ we have to remember that a change in $\alpha_n^{(l)}$ only effects the gross positioning and not the corrective movements. The corrective movements are determined solely by the $\mathbf{a}_n^{(l)}$s. Since the $\mathbf{a}_n^{(l)}$s are given already through (5), an improvement of the gross positionings can only be achieved by adjusting the $\alpha_n^{(l)}$s. A cost function the minimization of which yields adaptation rules for the $\alpha_n^{(l)}$s and leads to improved gross positionings is given by

$$C_2 = \frac{1}{2}\sum_{l=1}^{3}\left[y^{(l)}(t_i) - y^{(l)}(t_{i-1})\right]^2 = \frac{1}{2}\sum_{l=1}^{3}\left[y^{(l)}(t_i) - \sum_{n=1}^{N} z_n\alpha_n^{(l)} + \mathbf{a}_n^{(l)}\mathbf{x}_n(t_{i-1})\right]^2.$$

This cost function is determined by the difference $y^{(l)}(t_i) - y^{(l)}(t_{i-1})$ in the output of the network for two subsequent movement steps. A minimization of this difference corresponds to minimizing the size of the corrective movement steps, which can only be achieved through improved gross positioning movements. Minimizing the quadratic cost function C_2 via gradient descent with respect to $\alpha_n^{(l)}$ yields

$$\Delta\alpha_n^{(l)}(t_i) = \epsilon' \cdot \left[y^{(l)}(t_i) - y^{(l)}(t_{i-1})\right] \cdot z_n \quad (6)$$

as adjustment for the weights $\alpha_n^{(l)}$ at time step t_i.

If we neglect the discreteness of the time steps t_i, both adaptation rules (5) and (6) can be formulated by using time derivatives of the presynaptic excitations $\mathbf{x}$, z_n and postsynaptic excitation $y^{(l)}$ of each output unit l, $l = 1,2,3$, yielding

$$\begin{aligned}\dot{\mathbf{a}}_n^{(l)} &\propto -\dot{y}^{(l)} \cdot \ddot{\mathbf{x}}_n \\ \dot{\alpha}_n^{(l)} &\propto \dot{y}^{(l)} \cdot z_n.\end{aligned}$$

Similar Hebbian-like adaptation rules using time derivatives of the pre- and postsynaptic excitation have been proposed for modelling classical conditioning (see, e.g., [13]).

V. The Performance

For the training of the robot system we present the target object at 4,000 different locations within the work space of the robot, and for each target location the robot performs one gross positioning and four subsequent, feedback-guided corrective movements. The network consists of $N = 300$ gating units. The weights $\mathbf{w}_n$, $\alpha_n^{(l)}$, $\mathbf{a}_n^{(l)}$ of the network are initialized at random.

Figure 4 shows the average positioning error dependent on the number of presented training targets. The error decreases rapidly in the beginning, and after only 500 training steps the deviation from the

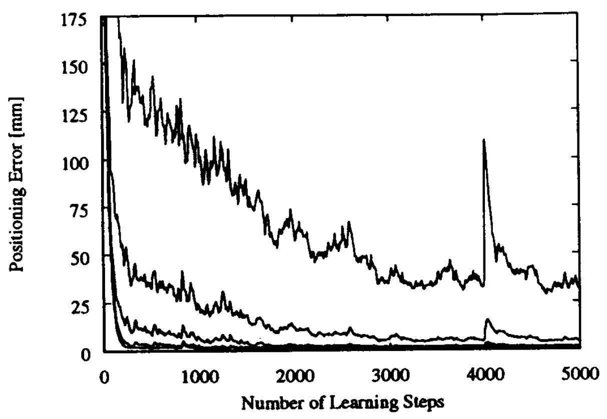

Fig. 4: The average positioning error with the number of training steps. The five graphs show the error after the first, open loop, positioning and after each of the four subsequent corrective movements. After 4,000 learning steps we increased the last arm segment of the robot by 100 mm to test the network's adaptive capabilities.

target after the last corrective movement reaches its minimum value of 1.3 mm which cannot be further reduced because of the finite resolution of the cameras. Since the work space in front of the robot has a volume of about 1 m^3, the error relative to the range of the arm is about 0.13%. The positioning error of the open-loop movement reaches its minimal value of about 4 cm, i.e., 4%, after 4,000 training steps. At this point we increase the last arm segment of the robot by 100 mm to test the network's adaptive capabilities. As we can see in Fig. 4, the residual error after the last corrective movement is not affected by this change, and with 500 further training steps the error of the open-loop movement has also regained its previous value.

Acknowledgment

We would like to thank Jörg Walter who implemented the robot and vision software and Narendra Ahuja who provided the PUMA robot for this project. This work has been supported by the Beckman Institute for Advanced Science and Technology of the University of Illinois at Urbana-Champaign, by the Carver Charitable Trust, and by a fellowship of the Volkswagen Foundation to T.M.

References

[1] Kawato M, Furukawa K, Suzuki R (1987) A hierarchical neural-network model for control and learning of voluntary movements. Biological Cybernetics, 57:169-185.

[2] Kuperstein M, Rubinstein J (1989) Implementation of an Adaptive Neural Controller for Sensory-Motor Coordination. IEEE Control Systems Magazine, Vol.9, No.3, pp 25-30.

[3] Mel BW (1987) A Robot that Learns by Doing. AIP Proceedings 1987, Neural Information Processing System Conf., Denver, CO, 1987.

[4] 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.

[5] Ritter H, Martinetz T, Schulten K (1989) Topology Conserving Maps for Learning Visuomotor-Coordination. Neural Networks, 2:159–168.

[6] Martinetz T, Ritter H, Schulten K (1990) Three-dimensional neural net for learning visuomotor-coordination of a robot arm. IEEE-Transactions on Neural Networks 1(1):131-136.

[7] Kohonen T (1984) Self-organization and associative memory. Springer Series in Information Sciences 8, Heidelberg.

[8] Martinetz T, Schulten K (1991) A "Neural-Gas" network learns topologies. Proceedings of the ICANN-91, Helsinki, pp 397-402, Elesevier, Amsterdam.

[9] Hebb D (1949) Organization of Behavior. Wiley, New York.

[10] Walter J, Martinetz T, Schulten K (1991) Industrial robot learns visuomotor coordination by means of "neural-gas" network. Proceedings of the ICANN-91, Helsinki, pp 357-364, Elesevier, Amsterdam.

[11] Keele SW (1981) Behavioral analysis of movement. In: Brooks VB (ed) Handbook of physiology, the nervous system, vol II. Motor control, Part 2. American Physiol Society, Bethesda, Md, pp 1391-1414.

[12] Feldman JA, Ballard DH (1982) Connectionist models and their properties. Cognitive Science, 1982, 6:205-254.

[13] Tesauro G (1990) Neural models of classical conditioning. In: Connectionist Modeling and Brain Function, eds. Hanson SJ, Olson CR, MIT Press, Cambridge, pp 74-104.