8

# Topology Representing Network in Robotics

Kakali Sarkar and Klaus Schulten[1]

with 6 figures

**Synopsis.** We consider the visually guided control of the grasping movements of a highly hysteretic five- joint pneumatic robot arm. For this purpose we apply a modified version of the so-called *topology representing network* algorithm, a vector quantization algorithm that also learns to represent neighborhood relationships. The notion of neighborhood relationships allowed us to average the behavior of neurons which represent similar tasks, both during the training and in generating control signals in the mature state. Based on visual information provided by two cameras, the robot learns to position and orient its end effector properly for the object to be grasped. For simplicity, we consider the grasping of cylindrical objects only. The control is comprised of two stages. In the first stage, the end effector approaches the side of the cylinder facing the robot base; and in the second stage, the end effector grasps the cylinder. Training of the first stage involves a brief episode of supervised learning to prime the network. The control is achieved through a visual feedback loop: for both stages of the motion the system detects the error to target and applies a linear correction. This correction is achieved through a training that yields a vector-quantized representation of a zero-order signal of joint pressures and a first-order correction through Jacobian tensors which relate the error, expressed in terms of camera coordinates, to correct joint pressures. The network is trained satisfactorily after about 300 trial movements, with a residual average error of 1.35 camera pixels. Besides a demonstration of the technical feasibility of control through *topology representing networks*, this chapter provides a tutorial for technical applications of such networks. The algorithm behind a *topology representing network*, its training and employment for task control, is described in complete detail to provide the reader with a comprehensive view of this important class of neural networks in the context of a technical application.

[1]Department of Physics/Beckman Institute, University of Illinois, Urbana, IL 61801, USA.

## 8.1 Introduction

In the early days of research in neurocomputing, networks were seen as devices that were capable of computing logic functions [1]. Such a mechanistic view of neurocomputing became popular mainly because of the fact that computation traditionally was viewed in light of logic gates and switching algebra. However, we have gradually come to know the bottlenecks of the traditional deterministic computer; we observe that the human brain can easily outperform today's supercomputers in tasks where it processes multidimensional analogue data and probabilistic, noisy information. It is now generally believed that an understanding of boolean logic and switching algebra may not enhance our perspective about neuronal information processing in the brain. The quest for a theoretical framework to quantify the underlying computation process has brought computer scientists, physicists, and biologists together. Vigorous research efforts during the last two decades have helped to develop a different perspective about neurocomputing. This interdisciplinary effort has resulted in many promising real-world applications such as speech processing [2], optimization [3], complex control systems [4, 5], and more.

Grasping of objects is one of the most common tasks frequently performed by human beings. Even though this seems to be easy and often spontaneous to most of us, from the control system perspective grasping is complicated. The object to be grasped has to be identified in the environment by its location and by other features. Then the trajectory of the arm movement has to be planned in such a way that it does not collide with any obstacle. Recently, many efforts have been made [6–10] to understand the control mechanism of such complex maneuvers and to make use of these fundamental control techniques to develop viable artificial neural control systems. In this chapter we focus mainly on the control of the execution of grasping motions, assuming an extremely simplified solution for the recognition of the target and the arm's current posture: we provide a set of suitable light-emitting diodes (LEDs) on the arm and the target in an otherwise darkened space.

Nevertheless, the problem of executing motions to grasp a cylinder placed in all possible positions and orientations in a robot's workspace is a difficult one. The motion must involve at least five degrees of freedom and be sufficiently precise. The precision must be achieved for an arm that is subject to random and hysteretic behavior. In fact, in the present case, the controlled arm is driven pneumatically with effectors which are subject to strong hysteresis and oscillations as characterized in [11, 12]. The required control only can be achieved when the network, besides learning the control signals for a sufficiently fine set of arm postures, also learns tensors which allow the arm to linearly correct deviations from the target due to hysteresis and other effects.

The corresponding control problem, in principle, can be formulated in terms of a table look-up algorithm that provides for each target cylinder a table entry which produces the suitable air pressures to move the arm. As was already stated, the entries of the table need to be a set of pressures to move the five degrees of freedom of the arm (see Section 2) as well as a tensor, the Jacobian connecting the deviation from the target, expressed as a vector of five coordinates, to the vector of pressures driving the arm (see Sec. 3). Obviously, such a table look-up program cannot be arbitrarily fine. However, even a coarse grid of, say, 10 points along each coordinate for a five-dimensional space leads to a very large number (100,000) of table entries. Obviously, an optimal choice which, for a given number of entries, produces the smallest error is very desirable. An important ingredient of the criterion stated is the probability distribution of arm postures under normal working conditions. The neural networks used in our study obey such criterion in that they assign their table entries as a result of a training in which arm postures are requested with a frequency distribution which matches that occuring in normal working conditions. In fact, the algorithm allows life-long learning such that the table entries can be continuously adjusted to the work experience.

The problem to optimally assign a finite number of table entries to a continuous space, often of very high dimension, is called the *vector quantization problem*. The neural network algorithm adopted here provides a solution for vector quantization as discussed in [13]. However, there is another important attribute of the control problem that also must be captured by the look-up algorithm in order to be efficient, namely the topology of the control space. This implies that the table entries develop threads between each other which connect entries assigned to arm postures which are very close to each other. These threads serve two purposes, one during training and one after training. The threads can be employed when the table entries are generated, i..e., when the networks are trained. Entries connected through threads contain similar information, and, hence, they can share the improvements to their entries during the training period. The result is a dramatic decrease of the training period since any training episode is shared by many table entries. A particularly important aspect of the sharing of information among table entries is that this feature makes the system much less sensitive to the initial, usually random, entries in the table. In many instances, when table entries are trained separately, convergence to a suitable control program depends on the initial table entries, i.e., the radius of convergence of the training algorithm is not infinite. However, the sharing of table entry updates increases the radius of convergence enormously, as was demonstrated in [14].

The threads between entries are also very beneficial after training, when the system is used to control the arm. The threads allow one to average the control signals (pressures) to the arm over table entries connected through a thread. Such an average improves performance at the early stages of training and can also increase the accuracy of the control: if $N$ units are

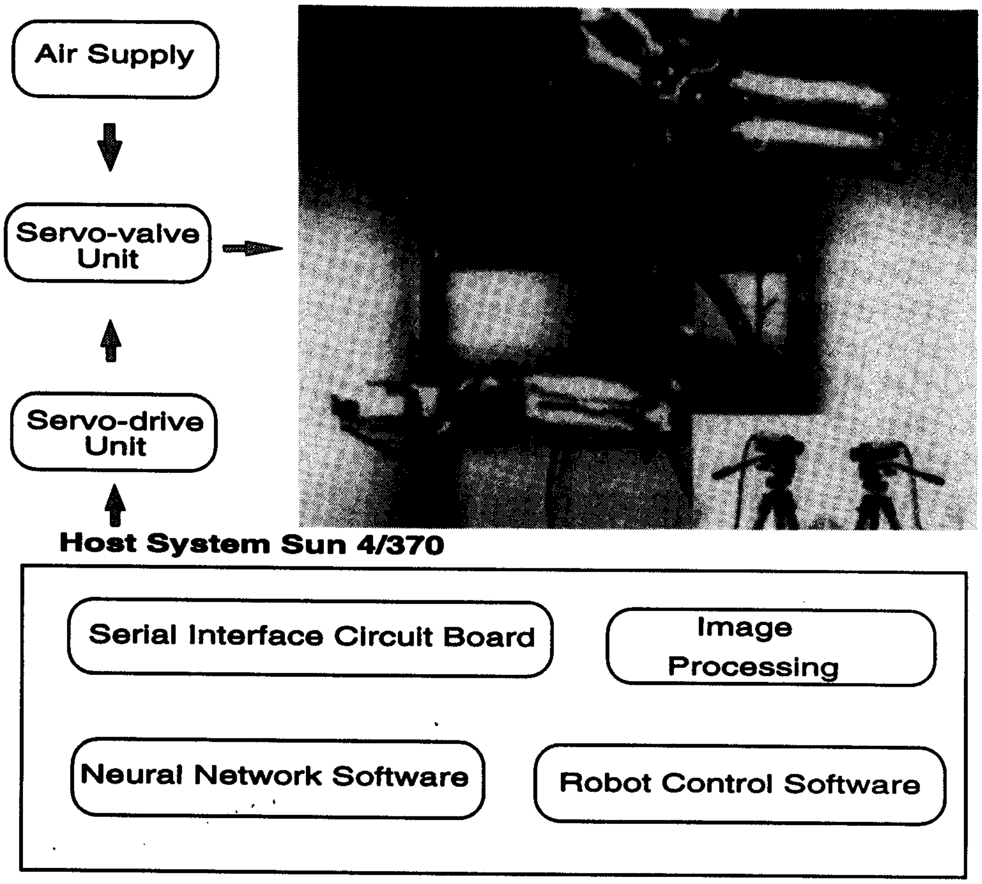

**Fig. 8.1.** Block diagram of the SoftArm robot system [11].

pooled, each with an error $\epsilon$, the error after averaging (assuming, for the sake of simplicity, that the table entries are coded for exactly the same posture) is $\epsilon/\sqrt{N}$.

The threads between the table entries reflect the topology, i.e., neighborhood relationships, of the control space. In the present case, the topology of the control space is obviously that of $\mathbb{R}^5$ since all arm postures required to grasp a cylinder form a manifold embedded in the five-dimensional Euclidean space. In fact, in the algorithm presented below, the threads between the table entries are never actually established. Rather, we use the Euclidean metric to establish a closeness ranking among table entries and use this ranking instead of threads. However, in many cases, a dimension or metric is not obvious and needs to be established while a system is confronted with training tasks. In an early neural network scheme for control based on Kohonen networks [15, 16], such a dimension needed to be specified beforehand. Theses schemes preserved the given dimension (topology) in that they assigned table entries to the task space while keeping the threads, e.g., those representing a two-dimensional grid, intact. Examples addressing the control of robots in computer simulations are found in

[17, 10, 18, 19, 14]. A comprehensive presentation of these networks in a variety of applications, ranging from brain maps to robot control, can be found in [20]. This textbook also discusses at length the statistical mechanical analysis of the convergence properties of the network and fluctuations of the network's table entries. A particularly interesting application of these networks to visual brain maps can be found in [21].

When we attempted to apply neural network algorithms to control real, i.e., not simulated, robot arms, we established that networks with an a priori topology, like generalized Kohonen networks, are not optimal. Instead, we appended the vector quantization scheme described in [13] with Hebbian rules which provided the required threads between table entries. The resulting topology representing the network had been introduced in [22] and discussed at length in [23]. The network has been applied successfully to control an industrial robot with precise response to control signals [24, 25] and also to a pneumatically driven robot [11], the same as the one employed in the present study.

In this chapter we present an extension of our previous work [11] on the control of a pneumatic robot arm by incorporating a control mechanism for the grasping of cylinders of arbitrary orientation. In the following section we first characterize the control problem describing the arm geometry and the ideosyncracies of the pneumatic actuators of the robot arm used. In Sec. 3 we present the topology representing network algorithm employed for control. The section provides all of the algorithmic steps involved in complete detail, but it does not explain the algorithm exhaustively as is done in [23]. However, the detailed presentation of the algorithm in the present contribution might be considered by many readers a better explanation of topology representing networks than any general exposition. In Sec. 4 we demonstrate how the algorithm, after training, performs grasping motions.

## 8.2 Problem Description

The robot–camera system is shown schematically in Fig. 8.1. This system has been described in detail in [11]. The robot contains a pneumatic arm with five joints. At each joint, two or four rubber tubes are connected by chains across sprockets. The rubber tubes are supplied with compressed air from an air compressor. When differential air pressures are supplied to the tubes, differing equilibrium lengths result, which induce a rotation of the joint to a new equilibrium point.

There are five servo drive units for five joints, each of which takes signals from the host computer and sends current output to the servo valve unit. The servo valve unit then converts this electrical signal to pressure information, i.e., it controls the pressures inside the rubber tubes by opening or closing the electrical valves. Two cameras observe the location of the end

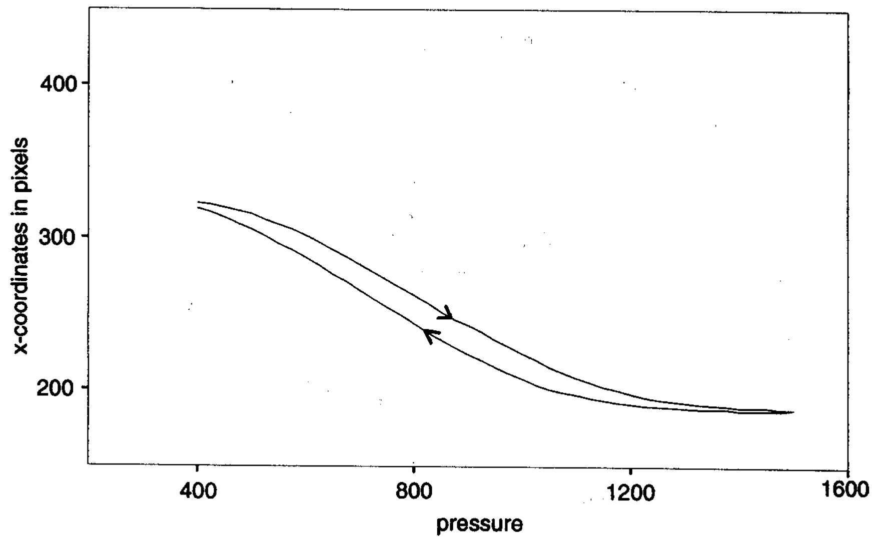

**Fig. 8.2.** Pressure versus position plot for joint 1. Hysteretic behavior of joint 1, of the softarm. The pressure difference in the agonistic and antagonistic tubes of joint 1 was first increased and then decreased.

effector or the cylinder to be grasped and send back the information to the host computer, which then finds the image coordinates in pixels with the help of two parallel image processors.

The servo drive units can be used to control the robot arm in two modes, a pressure-control mode and a position-control mode [11]. The present work has been carried out in the pressure-control mode. The relation between the joint pressures and position is highly nonlinear and also exhibits hysteresis. When the pressure is increasing, the pressure–position relation follows a particular path, but it follows a different path while the pressure is decreasing again. Figure 8.2 shows such type of behavior for joint 1.

The end effector of the robot arm is a two-fingered one and is presented schematically in Fig. 8.3. The movement of the end effector is controlled by the fourth and fifth joints. Each joint produces a motion which is a combination of rotational motions about the axes $XX'$ and $YY'$. Pure rotation about $XX'$ and $YY'$ also can be produced, but each of them is a function of both the fourth and fifth joint pressures.

In the present work, we consider the grasping of cylindrical objects only. In order to grasp such an object, several issues need to be addressed. First, the point of grasping should be very close to the center of mass of the cylinder. If the center of mass is far from the chosen grasping position, the generation of undesirable torques makes it difficult to hold the cylinder. The angle between the axis of symmetry$(ZZ')$ of the cylinder and that of the end effector$(XX')$ is another important factor. The end effector should be placed perpendicular to the symmetry axis of the cylinder. In other words, axis $ZZ'$ should be perpendicular to the plane containing axes $XX'$

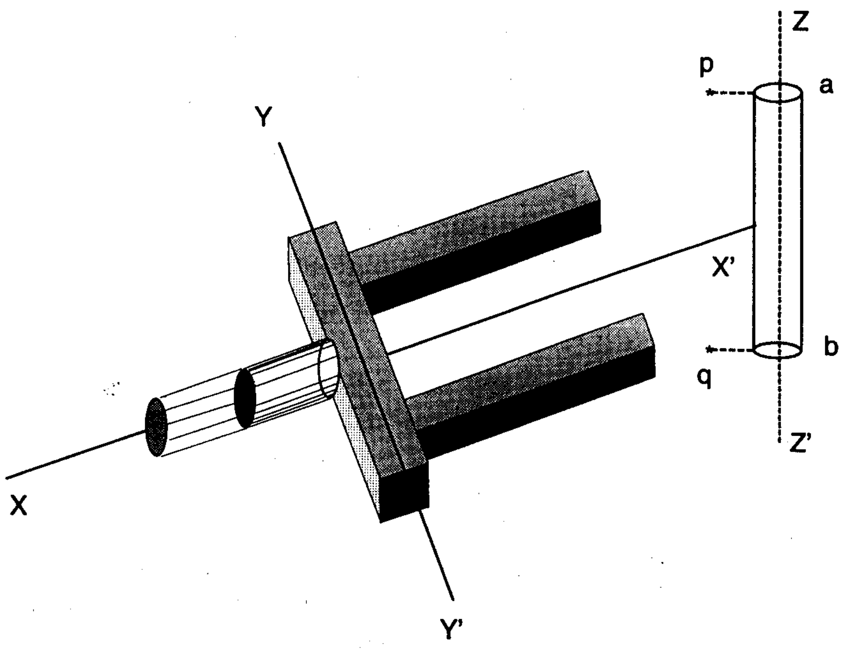

**Fig. 8.3.** A sketch of the end effector (gripper) and the cylinder to be grasped.

and $YY'$. These two aspects have played the role of prime significance in all of our grasping algorithms.

## 8.3 Topology Representing Network Algorithm

The visually controlled motions for grasping cylinders placed in the arm's workspace are carried out in two stages: In the first stage, the arm's gripper is placed in front of the cylinder at a proper orientation as shown in Fig. (8.3); in the second stage, the arm moves toward the center of the cylinder and actually grasps it by closing the gripper's fingers. The training procedures of each stage will be described separately below. Control of the first stage is by far the more difficult problem.

### 8.3.1 Training of First-Stage Motion

The goal of the first stage of the grasping motion is to generate a set of pressures in the arm's tubes which place and orient the gripper in front of the cylinder in a configuration suitable to carry out the second stage of the grasping motion and actually grasp the cylinder. We refer to the suitable configuration reached at the end of the first grasping stage as the *target configuration.* This configuration is realized through application of a set of vectors to the tubes of the arm which are collected in a pressure vector **P**.

The target position for the initial placement of the gripper is determined as follows: As is shown in Fig. 8.3, we fix two lights at the positions $p$ and $q$ such that the line joining $p$ and $q$ is coplanar as well as parallel to the cylindrical axis $ab$. The images of these lights give the representation of the endpoints of another imaginary cylinder of the same size as the original, which, however, is placed at a small distance in front of the original one. The lights appear in the two cameras at points characterized by the coordinates $(u_1, u_2, u_3, u_4)^T$ and $(u_5, u_6, u_7, u_8)^T$. As a result, the position of the target is characterized through an eight-dimensional vector $\mathbf{u}_{target} = (u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8)^T$. The set of all vectors $\mathbf{u}_{target}$ in the robot's workspace form the so-called feature space $V \subset \Re^8$. We seek a training procedure which, for the first stage of the grasping motion, develops a map $\mathbf{u}_{target} \in V \rightarrow \mathbf{P}(u_{target}) \in \mathcal{F}$ which assigns to $\mathbf{u}_{target}$ the proper pressure vector for $\mathbf{P}$, positioning and orienting the gripper in front of the cylinder.

The robot arm is moved through ten tubes which pairwise act in an agonistic–antagonistic manner to rotate the arm's joints. The sum of the two pressures in each agonist–antagonist tube pair determines the stiffness of the motion. In the present study, the total pressure for each joint was kept constant during the operation of the system. As a result, the arm was moved through five independent pressures, one for each joint. The corresponding pressure vector $\mathbf{P}$ is then five-dimensional and the space $\mathcal{F}$ of joint pressures is then embedded in $\Re^5$.

The goal of the training of the $N$ neurons controlling the first stage of the grasping motion is to develop first a set of Voronoi cells covering the feature space $V$ with centers $\mathbf{w}_k \in V$, $k = 1, 2, \ldots N$, and then to develop a map $V \rightarrow \mathcal{F}$. The latter map is established through local affine maps in each of the Voronoi cells, i.e., in the Voronoi cell assigned to neuron $k$, through

$$\mathbf{P}(\mathbf{u}_{target}) = \mathbf{P}_k + A_k \cdot (\mathbf{u} - \mathbf{w}_k), \tag{8.1}$$

where $\mathbf{P}_k$ and $A_k$ are constants (a vector and a tensor) which are acquired through the training.

As was stated earlier, the neurons actually achieve their control through averaging their output $\mathbf{P}(\mathbf{u}_{target})$. The average involves the neurons that have Voronoi cells adjacent to each other in the feature space $V$. To determine the corresponding average, one first needs to determine a ranking among the neurons which describes which neuron's Voronoi cell contains the target vector $\mathbf{u}_{target}$, which Voronoi cell is second closest, third closest, etc. Such ranking is achieved as follows: One determines for each neuron $k$, $k = 1, 2, \ldots N$, the distance

$$\mathbf{D}_k(\mathbf{u}_{target}) = \| \mathbf{u}_{target} - \mathbf{w}_k \| \tag{8.2}$$

and then determines a ranking $k_0, k_1, \ldots k_{N-1}$ such that

$$D_{k_m}(\mathbf{u}_{target}) \leq D_{k_n}(\mathbf{u}_{target}) \qquad \text{for } m < n.$$

One then defines

$$k(r, \mathbf{u}_{target}) = k_r \tag{8.3}$$
$$r(\ell, \mathbf{u}_{target}) = m\,, \quad \text{where} \quad k_m = \ell. \tag{8.4}$$

This ranking can be employed to achieve the desired averaging. We choose for this purpose the functional form

$$\begin{aligned}\overline{\mathbf{P}}(\mathbf{u}_{target}) &= \sum_{k=1}^{N} \alpha(r(k, \mathbf{u}_{target})) \\ &\quad \times \left[\mathbf{P}_{k(r,\mathbf{u}_{target})} + A_{k(r,\mathbf{u}_{target})} \cdot (\mathbf{u}_{target} - \mathbf{w}_{k(r,\mathbf{u}_{target})})\right]\end{aligned} \tag{8.5}$$

with

$$\alpha(r) = e^{-r/10} \tag{8.6}$$

The softarm poses a challenging control problem due to drift in the relationship between pressures applied to the arm's joints and the resulting arm configuration. This drift manifests itself on various time scales; on a very short timescale it is characterized by the hysteretic behavior of the arm shown in Fig. 8.2. On longer time scales a drift arises due to temperature sensitivity and dependence on time of usage of the mechanical characteristics of the arm's tubes. Finally, over the lifetime of the softarm the characteristics of the tubes are subject to wear. The long time changes can be overcome by retraining the arm. In fact, the algorithms for training and control of the arm are essentially identical, such that retraining can be realized during actual usage of the softarm.

The hysteretic properties of the softarm require that one linearly corrects the arm posture to reduce the error $d = ||\mathbf{x} - \mathbf{x}_{target}||$, where $\mathbf{x}$ characterizes the current arm posture and $\mathbf{x}_{target}$ is the desired posture. As was specified above, and for the second-stage gripper movement further below, the posture is characterized by certain vectors of camera coordinates such that $d$ is measured in units of camera pixels. The corrections of arm postures seek to reduce the error $d$ below a tolerance

$$\text{tol}(t) = 0.1 + 100 \cdot e^{-t/120}\,\text{pixels}\,. \tag{8.7}$$

Here $t$ counts the number of training steps. The tolerance is chosen large at the beginning of the training and reduces towards a small final value.

Obviously, one cannot enforce an overall accuracy of less than a camera pixel. In fact, the remaining final average error measures a little less than a pixel for each network, and a little over one pixel for the two networks controlling stage-one and stage-two movements combined (see Sec. 4). To reduce the error $d$ below the tolerance [Eq. (8.7)] usually requires several linear correction steps. Accordingly, the control system linearly corrects the arm posture repeatedly until the tolerance is met. In the course of the training, when the tolerance is already at a small value, e.g., after 200 training steps, the system typically requires eight correction moves, whereas it requires only about two to three such moves after training is completed.

The final result of a training procedure is optimal quantities $\mathbf{w}_k$ and $\mathbf{P}_k$, $A_k$ for all $N$ neurons $k$. At the beginning of the training, these quantities need to be assigned initial values. In many cases [10, 14], the initial values of quantities to be acquired are chosen randomly. However, such choices lead to long learning periods that are particularly unfavorable in cases where "real-world" systems are trained. In the present case, the robot arm requires about 30 s for a single training step, a period that can lead to long overall training times. Furthermore, the radius of convergence of a training procedure [14] might not be infinite, such that some initial assignments, will not lead to convergence. Averaging as in Eq. (8.5) increases the radius of convergence [14], but the radius need not necessarily become infinite. A finite radius of convergence would require that the initial values of $\mathbf{w}_k$ and $\mathbf{P}_k$, $A_k$ be chosen closer to the correct values. For this reason and, in particular, to speed up the overall training period, we acquired initial values in a supervised learning scheme. The learning was continued, after a brief phase, in an unsupervised form. For the sake of a more systematic exposition of the training schemes chosen, it is more suitable to present first the unsupervised learning scheme adopted here and then the supervised scheme, even though the schemes were applied in the opposite order.

## Unsupervised Learning Scheme

The unsupervised learning scheme consists of several hundred training steps, each of which results in an update of the quantities $\mathbf{w}_k$ and $\mathbf{P}_k$, $A_k$. The values of these quantities before the learning step are defined as $\mathbf{w}_k^{old}$ and $\mathbf{P}_k^{old}$, $A_k^{old}$ and after the learning step as $\mathbf{w}_k^{new}$ and $\mathbf{P}_k^{new}$, $A_k^{new}$.

We now outline how any particular step proceeds. The learning steps are numbered $t = 1, 2, \ldots$, and each learning step consists of ten substeps.

1. A cylinder is placed in a new, usually randomly chosen position in the workspace of the arm. To ascertain that the cylinder is actually placed in the workspace, one often adopts a "split brain" procedure [24], having the robot itself position the cylinder, but then "forgetting" the control signals (joint pressures in the present case). The cameras detect the cylinder and provide the vector $(v_1, \ldots, v_8)^T$ characterizing the cylinder position. For the following we define

$$\mathbf{v}_{target} = (v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8)^T \, . \tag{8.8}$$

Actually, the position $\mathbf{v}_{target}$ used for the stage-one motion does not coincide with the cylinder position, but rather is a position between the robot base and the cylinder, close to the cylinder as defined above.

2. The closeness ranking $k(r, \mathbf{v}_{target})$ of the neurons and its inverse $r(k, \mathbf{v}_{target})$ is determined, as described in Eqs. (8.3) and (8.4) above: $k(0)$ is the index of the neuron with its $\mathbf{w}_k^{old}$ closest to $\mathbf{v}_{target}$, $k(1)$ is the index of the neuron with its $\mathbf{w}_k^{old}$ second closest to $\mathbf{v}_{target}$, etc. Conversely, $r(119)$ is the rank of the neuron with index 119, i.e., $r(119) = 5$ implies that the particular neuron 119 has its $\mathbf{w}_{119}^{old}$ sixth closest to $\mathbf{v}_{target}$.

3. The vectors (weights) $\mathbf{w}_k^{old}$ are updated according to

$$\mathbf{w}_k^{new} = \mathbf{w}_k^{old} + \gamma_w\left(r(k, \mathbf{v}_{target}), t\right) \cdot \left(\mathbf{v}_{target} - \mathbf{w}_k^{old}\right) . \tag{8.9}$$

$\gamma_w$ is a function that decays exponentially with the number $t$ of the learning step as well as with the closeness rank $r(k, \mathbf{v}_{target})$

$$\gamma_w(r, t) = \epsilon \cdot e^{-r/\sigma} e^{-t/\lambda} \tag{8.10}$$

with $\epsilon = 0.7$, $\sigma = 5$, and $\lambda = 100$.

4. The pressure that is supposed to move the robot arm toward the target $\mathbf{v}_{target}$ then is determined according to the averaging procedure [Eq. (8.5)]

$$\begin{aligned}\overline{\mathbf{P}}(\mathbf{v}_{target}) \quad &= \quad \sum_{k=1}^{N} \alpha\left[(k, \mathbf{v}_{target}]\right] \\ &\quad \times \left[\mathbf{P}_{k(r,\mathbf{v}_{target})} + A_{k(r,\mathbf{v}_{target})} \cdot \left(\mathbf{v} - \mathbf{w}_{k(r,\mathbf{v}_{target})}\right)\right] .\end{aligned} \tag{8.11}$$

5. The pressure [Eq. (8.11)] is applied to the robot arm's tubes and the robot moves its gripper. The resulting gripper configuration is detected by the cameras and the vector of camera coordinates $\mathbf{v}_i \in V$ is supplied. This motion was termed in our previous studies [11] the *coarse movement* of the arm.

6. The values $\mathbf{P}_k^{old}$ then are updated according to

$$\mathbf{P}_k^{new} = \mathbf{P}_k^{old} + \gamma_p\left(r(k), t\right) \cdot \left[\overline{\mathbf{P}}(\mathbf{v}_{target}) - \mathbf{P}_k^{old} - A_k(\mathbf{v}_i - \mathbf{w}_k)\right] , \tag{8.12}$$

where $\overline{\mathbf{P}}(\mathbf{v}_{target})$ is the pressure determined in substep 4 and

$$\gamma_p(r, t) = \epsilon' \cdot e^{-r/\sigma} e^{-t/\lambda} \tag{8.13}$$

with $\epsilon' = 0.8$.

7. The system now determines an improved vector of pressures which attempt to correct the remaining differences between $\mathbf{v}_{target}$ and $\mathbf{v}_i$:

$$\overline{\mathbf{P}}_{fine} \quad = \quad \overline{\mathbf{P}}(\mathbf{v}_{target}) + \sum_{r=0}^{S} \alpha(r) \left[A_{k(r)} \cdot (\mathbf{u} - \mathbf{v}_i)\right] , \tag{8.14}$$

where $\overline{\mathbf{P}}(\mathbf{v}_{target})$ is again the pressure determined in substep 4 and $\alpha(r)$ is given in Eq. (8.6).

8. The pressure $\overline{\mathbf{P}}_{fine}$ is applied to the arm's tubes and the robot arm assumes a new gripper position. This position is detected by the cameras and corresponding camera coordinates $\mathbf{v}_f$ are supplied. This motion had been termed *fine movement* in our previous studies [11].

9. The system employs the remaining error between $\mathbf{v}_f$ and $\mathbf{v}_{target}$ to update the tensors $A_k$ according to

$$A_k^{new} = A_k^{old} + \gamma_j(r,t) \cdot (\Delta\mathbf{P} - A_{k(r)}^{old}\Delta\mathbf{v}).\Delta\mathbf{v}^T\|\Delta\mathbf{v}\|^{-2}, \qquad (8.15)$$

where

$$\gamma_j(r,t) = \epsilon'' \cdot e^{-r/\sigma}e^{-t/\lambda} \qquad (8.16)$$

with $\epsilon'' = 0.01$ and where we defined $\Delta\mathbf{P} = \overline{\mathbf{P}}_{fine} - \overline{\mathbf{P}}(\mathbf{v}_{target})$, $\overline{\mathbf{P}}(\mathbf{v}_{target})$ as again being the pressure vector of substep 4, and $\Delta\mathbf{v} = \mathbf{v}_f - \mathbf{v}_i$.

10. The system determines the error $d = \|\mathbf{v}_f - \mathbf{v}_{target}\|$ between the present gripper position and the target position. In the case where $d$ exceeds the tolerance [Eq. (8.7)], another correction move is executed and, accordingly, the system carries out steps 7–9 again; otherwise, the system goes to the next step. In the case where steps 7–9 are executed once more, one first redefines $\mathbf{P}_k^{new} \rightarrow \mathbf{P}_k^{old}$ and $A_k^{new} \rightarrow A_k^{old}$.

11. The unsupervised learning scheme either terminates when a set number of steps has been executed or starts another round of substeps, beginning with substep 1 above.

## Supervised Learning Scheme

The supervised learning scheme described now was employed to obtain better starting values for the quantities $\mathbf{w}_k$ and $\mathbf{P}_k$, $A_k$, which specify how the neurons $k$, $k = 1, 2, \ldots N$ control the initial stage of the grasping motion. The supervised learning scheme defines a sequence of target camera coordinates $\mathbf{v}_{target}$ by actually moving the gripper to the respective configuration and communicating the respective pressures to the learning scheme. The procedure, applied in the first $n_{sup} = 50$ steps of the learning scheme, is as follows:

1. A random pressure vector $\mathbf{P}_{target}$ is chosen.

2. $\mathbf{P}_{target}$ is applied to the tubes of the arm and the arm moves to a new position. The gripper configuration is detected by the cameras and the corresponding camera coordinates $\mathbf{v}_{target}$ are supplied.

3. The closeness ranking $k(r, \mathbf{v}_{target})$, $r(k, \mathbf{v}_{target})$ of the neurons is determined as in the unsupervised scheme.

4. The vectors (weights) $\mathbf{w}_k^{old}$ are updated, as in the unsupervised scheme, according to

$$\mathbf{w}_k^{new} = \mathbf{w}_k^{old} + \gamma_w\left(r(k, \mathbf{v}_{target}), t\right) \cdot (\mathbf{u} - \mathbf{w}_k^{old}), \qquad (8.17)$$

where $\gamma_w$ is as defined in Eqs. (8.3) and (8.4).

5. The pressure vectors $\mathbf{P}_k^{old}$ are updated according to

$$\begin{aligned} \mathbf{P}_k^{new} &= \mathbf{P}_k^{old} + \gamma_p\left(r(k, \mathbf{v}_{target}), t\right) \qquad (8.18) \\ &\quad \times \left[\mathbf{P}_{target} - \mathbf{P}_k^{old} - A_k(\mathbf{v}_{target} - \mathbf{w}_k)\right], \end{aligned}$$

where $\gamma_p(r, t)$ is as defined in Eq. (8.9).

6. The system then determines a pressure vector

$$\begin{aligned} \overline{\mathbf{P}}(\mathbf{v}_{target}) &= \sum_{k=1}^{N} \alpha\left[r(k, \mathbf{v}_{target})\right] \cdot \left[\mathbf{P}_{k(r,\mathbf{v}_{target})} \right. \qquad (8.19) \\ &\quad \left. + A_{k(r,\mathbf{v}_{target})} \cdot (\mathbf{v}_{target} - \mathbf{w}_{k(r)})\right]. \end{aligned}$$

7. This pressure is applied to the arm's tubes, and, as a result, the arm moves its gripper to a new position.

8. The cameras detect the new gripper position and supply the corresponding camera coordinates $\mathbf{v}_i$.

9. The system now determines an improved vector of pressures which attempt to correct the remaining differences between $\mathbf{v}_{target}$ and $\mathbf{v}_i$:

$$\overline{\mathbf{P}}_{fine} = \overline{\mathbf{P}}(\mathbf{v}_{target}) + \sum_{r=0}^{S} \alpha(r) \cdot \left(A_{k(r)} \cdot (\mathbf{v}_{target} - \mathbf{v}_i)\right). \qquad (8.20)$$

10. The pressure $\overline{\mathbf{P}}_{fine}$ is applied to the arm's tubes and the gripper moves to a new position.

11. The cameras detect the new gripper position and supply the corresponding camera coordinates $\mathbf{v}_f$.

12. The system then updates the tensors $A_k^{old}$ according to

$$\begin{aligned} A_k^{new} &= A_k^{old} + \gamma_j(r(k, \mathbf{v}_{target}), t) \qquad (8.21) \\ &\quad \times \left[(\mathbf{P}_{target} - \overline{\mathbf{P}}_{fine}(\mathbf{v}_{target}) - A_{k(r)}^{old}(\mathbf{v}_{target} - \mathbf{v}_f))\right] \\ &\quad \times (\mathbf{v}_{target} - \mathbf{v}_f)^T \left\|\mathbf{v}_{target} - \mathbf{v}_f\right\|^{-2}. \end{aligned}$$

Note that both expressions updating $\mathbf{P}_k^{old}$ and $A_k^{old}$, i.e., Eqs. (8.18) and (8.21), include $\mathbf{P}_{target}$, i.e., knowledge of the pressure which would have guided the arm, except for hysteretic effects, exactly to the target gripper position characterized by $\mathbf{v}_{target}$.

13. The system determines the error $d = ||\mathbf{v}_f - \mathbf{v}_{target}||$ between the present gripper position and the target position. In the case where $d$ exceeds the tolerance [Eq. (8.7)], another correction move is executed, and, accordingly, the system carries out steps 9–12 again; otherwise, the system goes to the next step. In the case where steps 9–12 are executed once more, one redefines first $\mathbf{P}_k^{new} \rightarrow \mathbf{P}_k^{old}$ and $A_k^{new} \rightarrow A_k^{old}$.

14. In the case where $n_{sup}$ training steps have been completed, the system terminates; otherwise, it begins another round of substeps beginning with substep 1 above.

## 8.3.2 Training of Final Grasping of the Cylinder — Second Stage of Movement

After the gripper has been placed and oriented properly in front of the cylinder (see Fig. 8.3) in the first stage of the movement, the gripper needs to be translated toward the cylinder until the fingers of the gripper enclose the cylinder sufficiently, i.e., until the center of the gripper coincides with the center of the cylinder. This translation is referred to as the *second stage* of the gripper movement. Since this movement does not require rotation of the gripper, only three degrees of freedom are active in the second stage of the movement. This considerably simplifies the control problem which requires, hence, a lower resolution of the neural network representation such that 200 neurons suffice.

The algorithm employed here for control and training of stage-two movement has been described in [11]; for the sake of completeness and consistency of notation, we review the algorithm below.

The aim of the algorithm is to guide the center of the gripper $\mathbf{g}$ to the center of the cylinder. The latter is characterized through two sets of camera coordinates, $(c_1, c_2)$ and $(c_3, c_4)$, corresponding to the image of the gripper center in the left and in the right camera, respectively. For the control of stage-two movement, the map

$$\mathbf{c} \rightarrow \mathbf{p} \tag{8.22}$$

is required, where $\mathbf{c} = (c_1, c_2, c_3, c_4)$ is a four-dimensional vector and $\mathbf{p}$ defines the set of pressures to translationally move the gripper. Since the last two joints of the five-jointed softarm are involved in gripper rotation, they are not required for the second-stage movement and only three pressures need to be specified. Accordingly, the map to be determined is $\Re^4 \rightarrow \Re^3$.

The embedding spaces $\Re^4$ and $\Re^3$ define a (Euclidean) metric $||\cdots||$ that will be employed.

The strategy of the present neural network approach, as outlined in [11], is to represent the relevant three-dimensional manifold $\Omega$ of gripper centers $\mathbf{c} \in \Re^4$ through vector quantization involving $n$ neurons, where $n = 200$. The neurons labeled $\ell, \ell = 1, 2, \dots n$ are to be assigned positions $\omega_\ell \in \Re^4$, which represent the manifold $\Omega$ of possible gripper centers. To each of the neurons we also assign a pressure vector $p_\ell \in \Re^3$ and $3 \times 4$-tensor $a_\ell$. The latter are to be chosen to establish affine maps

$$\mathbf{p}(\mathbf{c}) = \mathbf{p}_\ell + a_\ell \cdot (\mathbf{c} - \omega_\ell), \tag{8.23}$$

which optimally approximate the exact map [Eq. (8.22)] in the Voronoi cell of neuron $\ell$ in the manifold $\Omega$, i.e., in the space of all gripper centers $\mathbf{c}$ with $||\mathbf{c} - \omega_\ell|| \le ||\mathbf{c} - \omega_m||$, $m = 1, 2, \dots n$.

In order to determine the pressure that guides the gripper to the cylinder center $\mathbf{c}_{target}$ in stage two of the movement, one determines, in analogy to the case of stage-one movements, the closeness ranking $\ell(r, \mathbf{c}_{target})$ and, inversely, $r'(\ell, \mathbf{c}_{target})$. As in the case of a stage-one movement, the pressures supplied to the robot arm are actually averages of the pressures [Eq. (8.23)] contributed by neurons of neighboring Voronoi cells. The corresponding averages for the control of stage-two movements are given by

$$\begin{aligned}\overline{\mathbf{p}}(\mathbf{c}_{target}) &= \sum_{\ell=1}^{n} \alpha(r'(\ell, \mathbf{c}_{target})) \\ &\quad \times \left[\mathbf{p}_{\ell(r,\mathbf{c}_{target})} + a_{\ell(r,\mathbf{c}_{target})} \cdot (\mathbf{c}_{target} - \omega_{\ell(r,\mathbf{c}_{target})})\right],\end{aligned} \tag{8.24}$$

where $\alpha(r)$ is as defined in Eq. (8.6).

The final result of the training procedure is optimal quantities $\omega_\ell$ and $\mathbf{p}_\ell$, $a_\ell$ for all $n$ neurons $\ell$. At the beginning of the training procedure these quantities are assigned random values. Stage-two movement control does not require supervised learning to improve the initial values and cuts down the training period; the reason for this is that the three-dimensional posture control of a robot arm with averaging of control signals converges rapidly with an infinite convergence radius, as is demonstrated in [14].

## Learning Scheme

The unsupervised learning scheme consists of several hundred training steps, each of which results in an update of the quantities $\omega_\ell$ and $\mathbf{p}_\ell$, $a_\ell$. The quantities before the learning step are defined as $\omega_\ell^{old}$ and $\mathbf{p}_\ell^{old}$, $a_\ell^{old}$, and after the learning step $\omega_\ell^{new}$ and $\mathbf{p}_\ell^{new}$, $a_\ell^{new}$. We now outline how any particular step proceeds. The learning steps are numbered $t = 1, 2, \dots,$ and each learning step consists of nine substeps.

1. A target position $\mathbf{c}_{target}$ is chosen randomly to operate the robot in a "split brain" fashion: a random set of pressures $(p_1, p_2, p_3)$ is applied

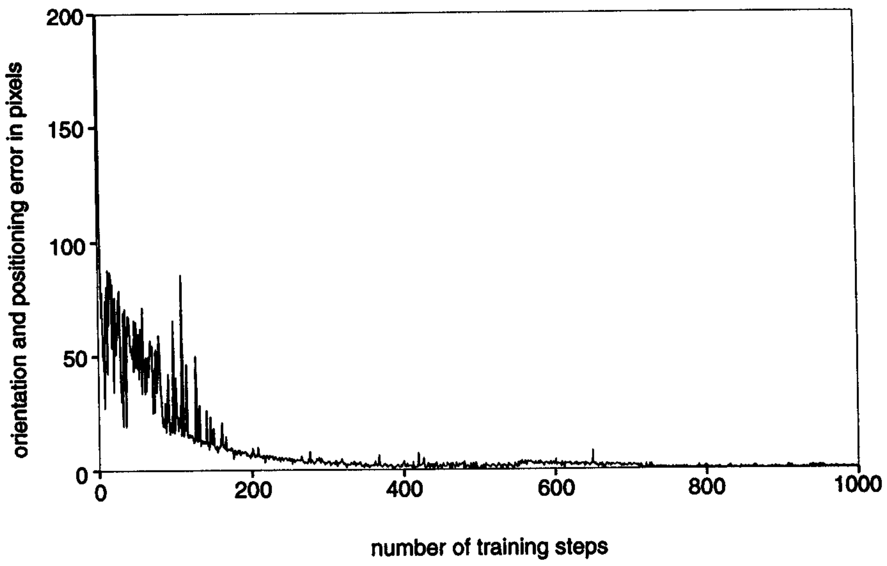

**Fig. 8.4.** Positioning and orientation error versus number of steps. This figure shows the learning curve for the network controlling the first stage of the gripper movement.

to the tubes of the first three joints of the softarm. The arm moves to a corresponding position. This position is detected through the cameras and communicated to the system in the form of the four-dimensional vector $\mathbf{c}_{target}$. This procedure ascertains that the chosen positions $\mathbf{c}_{target}$ actually belong to the workspace of the arm.

2. The closeness ranking $\ell(r, \mathbf{c}_{target})$ and its inverse $r(\ell, \mathbf{c}_{target})$ are established.

3. The values $\omega_\ell^{old}$ are updated using the expression

$$\omega_\ell^{new} = \omega_\ell^{old} + \gamma_w(r(\ell, \mathbf{c}_{target}), t) \cdot (\mathbf{c}_{target} - \omega_\ell^{old}) . \qquad (8.25)$$

Here $\gamma_w(r,t)$ is chosen as

$$\gamma_w(r,t) = e^{-r/\sigma_2}\, e^{-\sqrt{t}/9}, \qquad (8.26)$$

where $\sigma_2 = 5$.

4. The pressure vector $\overline{\mathbf{p}}(\mathbf{c}_{target})$, which is supposed to move the gripper center toward $\mathbf{c}_{target}$, then is determined according to the averaging procedure in Eq. (8.24).

5. This pressure is applied to the tubes of the robot arm and the arm moves the gripper. The resulting position of the gripper center is detected by the cameras and the vector $\mathbf{c}_i$ of camera coordinates is supplied.

6. The values $\mathbf{p}_\ell^{old}$ are then updated according to
$$\begin{aligned}\mathbf{p}_\ell^{new} &= \mathbf{p}_\ell^{old} + \gamma'_p(r(\ell, \mathbf{c}_{target}), t) \\ &\times \left[\overline{\mathbf{p}}(\mathbf{c}_{target}) - \mathbf{p}_\ell^{old} - a_\ell^{old}(\mathbf{c}_i - \omega_\ell^{old})\right],\end{aligned} \tag{8.27}$$
where $\overline{\mathbf{p}}(\mathbf{c}_{target})$ is the pressure vector determined in substep 4 and where
$$\gamma'_p(r(\ell, \mathbf{c}_{target}), t) = \epsilon'' \cdot e^{-r/\sigma_2} e^{-\sqrt{t}/9} \tag{8.28}$$
with $\epsilon'' = 0.8$ and $\sigma_2 = 5$.

7. The system now determines an improved vector of pressures which attempt to correct the remaining differences between $\mathbf{c}_{target}$ and $\mathbf{c}_i$:
$$\overline{\mathbf{p}}_{fine} = \overline{\mathbf{p}}(\mathbf{c}_{target}) + \sum_{r=1}^{n} \alpha(r'(\ell, \mathbf{c}_{target})) \cdot a_{\ell(r)} \cdot (\mathbf{c}_{target} - \mathbf{c}_i), \tag{8.29}$$
where $\overline{\mathbf{p}}(\mathbf{c}_{target})$ is again the pressure vector determined in substep 4 and where $\alpha(r)$ is as defined in Eq. (8.6).

8. The pressure $\overline{\mathbf{p}}_{fine}$ is applied to the arm's tubes and the arm assumes a new gripper position. This position is detected by the cameras and the corresponding camera coordinates $\mathbf{c}_f$ are supplied.

9. The system employs the remaining error between $\mathbf{c}_{target}$ and $\mathbf{c}_f$ to update the tensors $a_\ell^{old}$:
$$a_k^{new} = a_k^{old} + \epsilon''' e^{-r/\sigma} \cdot a_k^{old}(\mathbf{c}_{target} - \mathbf{c}_f)\Delta\mathbf{c}^T \|\Delta\mathbf{c}\|^{-2} \tag{8.30}$$
with $\epsilon''' = 0.01$, $\sigma = 5$, and $\Delta\mathbf{c} = \mathbf{c}_f - \mathbf{c}_i$.

10. The system determines the error $d = \|\mathbf{c}_f - \mathbf{c}_{target}\|$ between the present gripper position and the target position. In the case where $d$ exceeds the tolerance [Eq. (8.7)], another correction move is executed and, accordingly, the system carries out steps 7–9 again; otherwise, the system goes to the next step. In the case where steps 7–9 are repeated, one first redefines $\mathbf{p}_\ell^{new} \rightarrow \mathbf{p}_\ell^{old}$ and $a_\ell^{new} \rightarrow a_\ell^{old}$.

11. The learning scheme either terminates when a set number of steps have been executed or starts another round of substeps, beginning with substep 1 above.

## 8.4 Experimental Results and Discussion

### 8.4.1 Robot Performance

Target locations for the training were selected by moving the end effector to a position that was chosen by supplying random pressures to the joints.

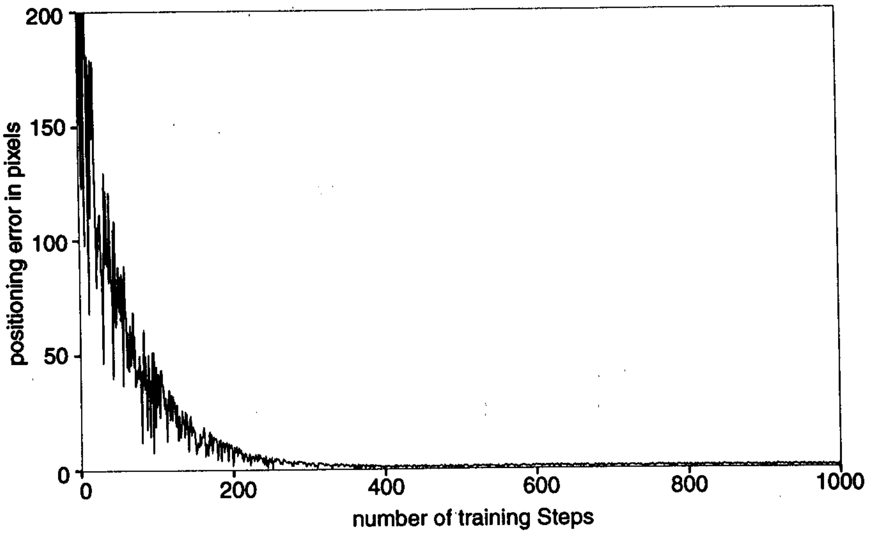

**Fig. 8.5.** Positioning error of the end effector for the neural network controlling the second stage of gripper movement.

Maximum and minimum pressures for each joint were stated such that the robot arm picked target positions within a workspace of size 375 mm $\times$ 750 mm $\times$ 750 mm.

The camera viewed the resulting position and orientation of two lights that were fixed at positions $p$ and $q$ (Fig. 8.3) and sent the corresponding $\mathbf{v}_{target}$ to the system.

In each learning step, after the target location $\mathbf{v}_{target}$ was chosen, the robot arm went to a particular arbitrarily chosen position from where it tried to reach the target location $\mathbf{v}_{target}$ using one coarse movement and several fine movements.

All of the weights $\mathbf{w}_k$, pressures $\mathbf{P}_k$, and Jacobians $A_k$ initially were assigned randomly. The initial $n_{sup} = 50$ learning steps followed the supervised procedure, introduced in Sec. 3, in which the knowledge of the pressures $\mathbf{P}_{target}$ corresponding to the target positions $\mathbf{v}_{target}$ were provided. After the first 50 steps, the robot started to learn in an unsupervised mode, i.e., the pressures $\mathbf{P}_{target}$ no longer were provided. Each trial, on average, took 30 s to complete. Two networks were trained separately in this way. One network, consisting of 1000 neurons, was employed for stage-one movements which positioned and oriented the gripper in front of the cylinder. For $S$, introduced in Eqs. (8.14) and (8.20), a value of 400 was chosen. The robot learned a set of five pressures $\mathbf{P}_k$ and a set of $5 \times 8$ Jacobian matrices. A smaller network of 200 neurons was employed for second-stage movements leading to grasping. In the later case, only three joints were used, and here the robot learned a set of $3 \times 4$ Jacobian matrices in an unsupervised way, as was already described in [11]. The tolerance level for

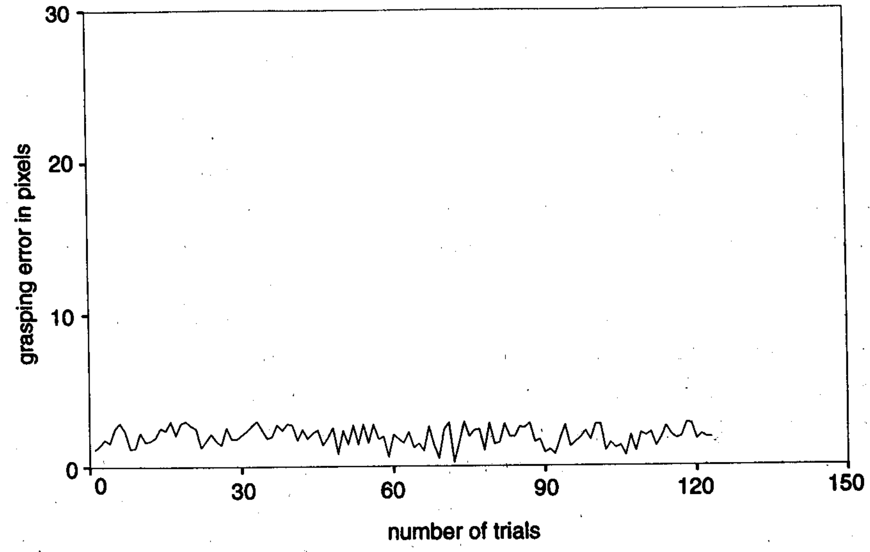

**Fig. 8.6.** Grasping error versus number of trials; the figure here shows the combined error for both of the networks.

error ($\mathbf{u}_{target} - \mathbf{v}_f$) for each learning step was an exponential function of time [Eq. (8.7)].

As in Eq. (8.7), in the initial stages the tolerance was set to a high level, and as the network became mature it became lower and lower. Both of the networks took 400 steps to reduce the error for both the positioning and orientation below 3 pixels. Figures 8.4 and 8.5 show error levels for both of the networks after 1000 learning steps. For a mature network, three fine movements were sufficient to reduce the error below the tolerance level.

## 8.4.2 Combination of Two Networks for Grasping

After the training was completed, the mature networks were tested for grasping a cylinder. The combined network, trained first by the supervised and then by the unsupervised algorithm, was used to place the robot gripper in front of the actual cylinder by sending visual inputs from two lights at positions $p$ and $q$ (Fig. 8.3). After this initial positioning, the visual inputs were changed to the images of the center of line $ab$. The network consisting of 200 neurons then became activated and the gripper approached that center slowly by small movements. The results for the two networks then were combined and are shown in Fig. 8.6. Figures 8.5 and 8.6 demonstrate that the network is satisfactorily trained after only about 300 trial movements, with a residual average error of 1.35 camera pixels.

### 8.4.3 Discussion

Control of positioning and grasping movements of robot arms often has been addressed in the literature, in particular by researchers in control theory and artificial intelligence [26]. The major problem with the control theory and the artificial intelligence approaches is that they both depend on the domain knowledge and, therefore, require cumbersome efforts to design the control system. Moreover, these approaches are not robust when one deals with real life, e.g., hysteretic, robots. In this work we have taken a different approach which is based on our understanding of the map-generating mechanism in human brains [21]. Our previous effort to control the positioning of the end effector of a pneumatic robot [11] was successful but limited to a restricted set of target configurations. In the present study we allow arbitrary orientations of a target cylinder to be grasped and thereby have made the problem of grasping control more difficult to accomplish. Nevertheless, the *topology representing network* algorithm along with supervised tuning accomplished control of grasping after only a modest number (300) of training episodes. Presently, we extend this study to network architectures that closely resemble biological motor pathways, in particular those that involve cortical as well as cerebellar components. We also employ a more sophisticated method for visual recognition of target and arm posture.

*Acknowledgments.* We would like to thank Ted Hesselroth for the algorithms of Sec. 3.2, Hillol Kargupta for helpful discussions about the supervised algorithm, Joerg Walter for the vision system code, and Volker Ehrlich and Benno Puetz for help with Fig. 8.1. The authors express their gratitude to the Carver Charitable Trust for support. Funds for the robot system were provided by the Beckman Institute through the Capital Development Board of the University of Illinois. The computations were carried out in the National Institutes for Health Resource for Concurrent Biological Computings (grant 1 P41 RR05969*01).

## References

[1] W. Mc Culloch, W. Pitts (1943) A logical calculus of the ideas immanent in the nervous activity. *Bull. Math. Biophys.*, 5:115–133

[2] O. Ghitza (1987) Robastness against noise: The role of timing-synchrony measurement. *Proc. Int. Conf. on Acoustics Speech and Signal Processing, ICASSP-87, Dallas*, April 1987

[3] D.W. Tank, J.J. Hopfield (1986) Simple neural optimization networks: An A/D converter, signal decision circuit and a linear programming circuit. *IEEE Trans. Circuits Syst.* CAS-33:533–541

[4] K. Furuta, M. Sampei (1988) Path control of a three-dimensional linear motional mechanical system using laser. *IEEE Trans. Indust. Electron.* 35(1):52–59

[5] L.E. Weiss, A.C. Sanderson, C.P. Neuman (1987) Dynamic sensor-based control of robots with visual feedback. *IEEE J. Robotics Automat.* **RA-3**(5):404–417

[6] M. Kuperstein (1987) Neural model of adaptive hand-eye coordination for single postures. *Science* **239**:1301–1311

[7] M. Kuperstein (1987) Adaptive visual-motor coordination in multijoint robots using parallel architecture. *IEEE Int. Automat. Robotics (Raleigh, NC)*, 1596–1602

[8] J. A. Walter, T. M. Martinetz, K. Schulten (1991) Industrial robot learns visuomotor coordination by means of "neural-gas" network. In: *Proc. Int. Conf. Artificial Neural Networks, Helsinki, 1991* (Elsevier, Amsterdam)

[9] H. Miyamoto, M. Kawato, T. Setoyama, R. Suzuki (1988) Feedback-error-learning neural network for trajectory control of a robotic manipulator. *Neural Networks* **1**:251–265

[10] T. Martinetz, H. Ritter, K. Schulten (1990) Three-dimensional neural net for learning visuo-motor coordination of a robot arm. *IEEE Trans. Neural Networks* **1**:131–136

[11] T. Hesselroth, K. Sarkar, K. Schulten, P.P. van der Smagt (in press) Neural network control of a pneumatic robot arm. *IEEE Trans. Syst., Man Cybernet.*

[12] P. van der Smagt, K. Schulten (1993) Control of pneumatic robot arm dynamics by a neural network. In: *Proc. World Congress on Neural Networks, Portland, OR, July 11–15, Vol. 3*, pp. 180–183

[13] T. Martinetz, S. Berkovich, K. Schulten (1993) "Neural gas" for vector quantization and its application to time series prediction. *IEEE Trans. Neural Networks* **4**:558–569

[14] T. Martinetz, K. Schulten (1993) A neural network for robot control: Cooperation between neurons as a requirement for learning. *Comput. Electr. Engrg.* **19**:315–332

[15] T. Kohonen (1982) Analysis of a simple self-organizing process. *Biol. Cybern.* **44**:135–140

[16] T. Kohonen (1982) Self-organized formation of topologically correct feature maps. *Biol. Cybern.* **43**:59–69

[17] H. Ritter, T. Martinetz, K. Schulten (1989) Topology-conserving maps for learning visuo-motor coordination. *Neural Networks* **2**:159–168

[18] T. Martinetz, H. Ritter, K. Schulten (1990) Learning of visuo-motor coordinaation of a robot arm with redundant degrees of freedom. In: *Parallel Processing in Neural Systems and Computers*, R. Eckmiller, G. Hartmann, G. Hauske (Eds.) (Elsevier, Amsterdam), pp. 431–434

[19] T. Martinetz, K. Schulten (1990) Hierarchical neural net for learning control of a robot's arm and gripper. In: *International Joint Conference on Neural Networks, San Diego, CA, Vol. 2* (Institute of Electrical and Electronics Engineers, New York), pp. 747–752

[20] H. Ritter, T. Martinetz, K. Schulten (1992) *Neural Computation and Self-Organizing Maps* (revised English Edition) (Addison-Wesley, Reading, MA)

[21] K. Obermayer, G.G. Blasdel, K. Schulten (1992) Statistical mechanical analysis of self-organization and pattern formation during the development of visual maps. *Phys. Rev. A* **45**:7568–7589

[22] T. Martinetz, K. Schulten (1991) A neural gas network learns topologies. In: *Artificial Neural Networks*, T. Kohonen, O. Simula, J. Kangas (Eds.) (Elsevier, Amsterdam), pp. 397–402

[23] T. Martinetz, K. Schulten (1996) Topology representing networks. *Neural Networks* 7:507–522

[24] J. A. Walter, K. Schulten (1993) Implementation of self-organizing neural networks for visuo-motor control of an industrial robot. *IEEE Trans. Neural Networks* 4:86–95

[25] T. Martinetz, K. Schulten (1993) A neural network with hebbian-like adaptation rules learning visuo-motor coordination of a PUMA robot. In: *Proc. IEEE Int. Conf. Neural Networks (ICNN-93), San Francisco*, pp. 820–825

[26] P. H. Winston (1984) *Artificial Intelligence* (Addison-Wesley, Reading, MA)