13. LEARNING BALLISTIC MOVEMENTS OF A ROBOT ARM

13.1 Problem and Model Approach

After a sufficiently long training phase, the network described in the preceding chapters can provide the required joint angles for any desired arm position. However, setting the joints to these angles is left to the joint servo motors of the arm. Such motors achieve the target position by changing their torque in opposite direction to any angular deviation from the target joint settings. For slow movements, this is an appropriate strategy because the individual joint movements can then be regarded to good approximation as independent of one another. However, for rapid movements, the inertia of the arm segments leads to a coupling between movements of different joints. For example, the movement of an inner (proximal) joint leads to an acceleration of all outer (more distal) joints and, hence, to the occurrence of torques, which must additionally be overcome by the joint motors. Conversely, the inner joint motors must counterbalance the action of outer joint motors.
In summary, inertial, centrifugal, and gyroscopic forces occur, the interplay of which leads to a complex, nonlinear coupling of all joints. In this situation, a single motor can no longer determine its torque from the present and given joint position alone, but rather its torque must also depend on the movements of all the other joints. Thus, to achieve the desired movement, it is no longer sufficient to take into account the connection between arm position and joint angles alone, *i.e.*, the *kinematics* of the arm. The Newtonian equations of motion of the arm, *i.e.*, its *dynamics*, must be included as well. Although the Newtonian equations can be given in closed form, they become enormously complicated for multi-joint systems. A closed solution is possible only in exceptional cases, and even approximate solutions require a knowledge of the *inertia tensors* of the individual arm segments. In this case, a real-time computation of the arm torques is possible by means of recent

algorithms whose computational effort grows only linearly with the number of arm joints. However, the inertia tensors required for such a computation are frequently known only imprecisely. This is due to the fact that the spatial mass distribution of the arm, which is described by these tensors, in general is very complicated for real systems. Hence, adaptive algorithms with the ability to *learn* these properties of the arm are highly desirable.
equations. the arm are
In this chapter we show how the extended Kohonen algorithm can also be applied to the problem of moving a robot arm by accounting for the arm's dynamics. We choose the task to control a three-link robot arm as introduced in Section 11.1 by means of briefly applied torque pulses at its joints in such a way as to accelerate its end effector to a prescribed velocity. During the remaining time, the arm is to move freely. The relationship between arm configuration, desired velocity, and required torque pulse is to be learned by a network again through trial movements. In contrast to the previous situation, this requires taking into account not only arm kinematics but also arm dynamics, *i.e.*, effects of inertia. Since we have shown in Chapter 12 how a network may learn to compute the transformation from visual information to joint angles, we will not consider this part of the problem here anymore and encode arm configuration directly by joint angles. As before, the arm can move its end effector freely in every spatial direction and reach any point on the working area which is now the planar surface of a table located in front of it. In the simulation, the relationship between end effector motion and joint torques is to be learned for those configurations for which the end effector is located directly above the working area. The configuration of the arm is again specified by its joint angles, expressed in vector notation by $\vec{\theta} = (\theta_1, \theta_2, \theta_3)$. The movement of the arm is effected by three torques $\doteq(d_1, d_2, d_3)$ acting on its joint axes. Let $\mathbf{q}$ denote the position of the end effector in Cartesian coordinates. The equations of motion of the arm are then given by (see for example Brady et al. 1984)

$$d_i(t) = \sum_{j=1}^{3} A(\vec{\theta})_{ij} \text{"} q_j + \sum_{j,k=1}^{3} B(\vec{\theta})_{ijk} \dot{q}_j \dot{q}_k + g_i(\vec{\theta}). \tag{13.1}$$

$\mathbf{A}(\vec{\theta})$ and $\hat{\mathbf{B}}(\vec{\theta})$ are configuration-dependent matrices which describe the dynamical properties of the arm. The term $\mathbf{g}(\vec{\theta})$ takes the contribution of gravity into account. If the end effector is initially at rest, a briefly applied

torque pulse

$$\dot{(}t) = \vec{\tau} \cdot \delta(t) \tag{13.2}$$

thus imparts to it the velocity $\mathbf{v} = \dot{\mathbf{q}}$ satisfying

$$\vec{\tau} = \mathbf{A}(\vec{\theta})\mathbf{v}. \tag{13.3}$$

Here $\vec{\tau} = (\tau_1, \tau_2, \tau_3)$ denotes three torques acting on the three joints of the robot arm. In particular, the coefficients B_{ijk} and g_i do not influence the velocity attained immediately after the torque pulse (the change in $\mathbf{v}$ during the subsequent force-free motion, which is affected by B_{ijk} and g_i, is not included here). Motions generated by the brief torque pulses described by (13.3) are termed *ballistic movements*. Equation (13.3) describes a relationship between configuration $\vec{\theta}$, torque amplitude $\vec{\tau}$, and resulting end effector velocity $\mathbf{v}$ in a form similar to that of Eq.(143). Hence, the learning algorithm developed in Chapter 11 is again applicable. As before, we make use of a lattice and define a vector $\mathbf{w_r}$ and a matrix $\mathbf{A_r}$ for each lattice site $\mathbf{r}$. Just as in Chapter 11, each $\mathbf{w_r}$ specifies an arm configuration, but this time in terms of joint angles. Thus, $\mathbf{w_r}$ is now a three-component vector.
In the course of the learning phase, each lattice site $\mathbf{r}$ becomes responsible for a small subregion of the arm's configuration space, the subregion extending about the configuration defined by the joint angles $\mathbf{w_r}$. The matrix $\mathbf{A_r}$ should converge to the transformation matrix which, according to (13.3) connects the desired end effector velocity $\mathbf{v}$ and the required torque amplitudes $\vec{\tau}$ in this subregion.
The training phase of the robot consists again of a sequence of trial movements. For each trial, the starting configuration $\vec{\theta}$ is obtained by requiring the end effector to be at some randomly chosen position within the working area. From there, the end effector is to be moved with a prescribed velocity $\mathbf{u}$ (also chosen at random during the learning phase). On the basis of the initial configuration $\vec{\theta}$, the system selects that transformation matrix $\mathbf{A_s}$, for which $\|\mathbf{w_s} - \vec{\theta}\| = \min_{\mathbf{r}} \|\mathbf{w_r} - \vec{\theta}\|$. On the basis of the prescribed velocity $\mathbf{u}$, it then performs the movement resulting from the torque amplitude

$$\vec{\tau} = \mathbf{A_s}\mathbf{u}. \tag{13.4}$$

From the end effector velocity $\mathbf{v}$ actually obtained, an improved estimate

$$\mathbf{A}^* = \mathbf{A_s} + \frac{\epsilon'}{\|\mathbf{v}\|^2}(\vec{\tau} - \mathbf{A_s}\mathbf{v})\mathbf{v}^T \tag{13.5}$$

is derived for $\mathbf{A_s}$, and, taking into account the input quantity $\vec{\theta}$, the learning steps

$$\mathbf{w}_\mathbf{r}^{(\text{new})} = \mathbf{w}_\mathbf{r}^{(\text{old})} + \epsilon h_{\mathbf{rs}}(\vec{\theta} - \mathbf{w}_\mathbf{r}^{(\text{old})}) \tag{13.6}$$

$$\mathbf{A}_\mathbf{r}^{(\text{new})} = \mathbf{A}_\mathbf{r}^{(\text{old})} + h'_{\mathbf{rs}}(\mathbf{A}^* - \mathbf{A}_\mathbf{r}^{(\text{old})}) \tag{13.7}$$

are carried out for the variables $\mathbf{w_r}$, $\mathbf{A_r}$, respectively. Subsequently, the next trial movement is executed.

13.2 A Simulation

In the following simulation $h_{\mathbf{rs}}$ and $h'_{\mathbf{rs}}$ were again chosen as Gaussians, and $\sigma(t)$, $\sigma'(t)$, $\epsilon(t)$ and $\epsilon'(t)$ were all of the familiar form $x(t) = x_i \cdot (x_f/x_i)^{t/t_{max}}$. The motion of the robot arm was simulated on a computer, using a dynamics simulation algorithm as suggested by Walker and Orin (1982). The mass distribution was assumed to consist of three unit point masses located at the middle and front joints, and at the end of the arm. Let us consider a Cartesian coordinate system whose origin is located at the base of the arm and whose xy-plane coincides with the plane of the working surface. The x and y-axes run parallel to the short and long edges of the working surface, respectively. For each trial movement, the desired velocity was chosen as a random vector with an isotropic distribution of direction and its length a random value uniformly distributed between 0 and 1.
The network consisted of a planar, rectangular 15×24 lattice of 360 neural units. A random initial state was generated in the following way: For each lattice site $\mathbf{r}$, an end effector position on the working surface was selected at random, and $\mathbf{w_r}$ was set to the corresponding joint angles. For this position, the correct transformation matrix $\mathbf{A}$ was computed. The individual elements of $\mathbf{A_r}$ were then calculated from the elements of $\mathbf{A}$ by superposition of random errors according to

$$(\mathbf{A_r})_{ij} = \mathbf{A}_{ij} + \alpha \|\mathbf{A}\| \cdot \eta. \tag{13.8}$$

Here, $\eta \in [-1, 1]$ is a uniformly distributed random variable, and α is a parameter measuring the deviation of the initial matrices $\mathbf{A_r}$ from their correct values. The simulation data were $\alpha = 0.25$, $\epsilon_i = 0.8$, $\epsilon_f = 0.02$, $\epsilon'_i = 1$, $\epsilon'_f = 0.5$, $\sigma_i = \sigma'_i = 3$, $\sigma_f = \sigma'_f = 0.2$ and $t_{max} = 10,000$.
The initial state of the lattice is shown in Fig. 13.1. To illustrate the correspondence between lattice sites and arm configurations, in Fig. 13.1a a

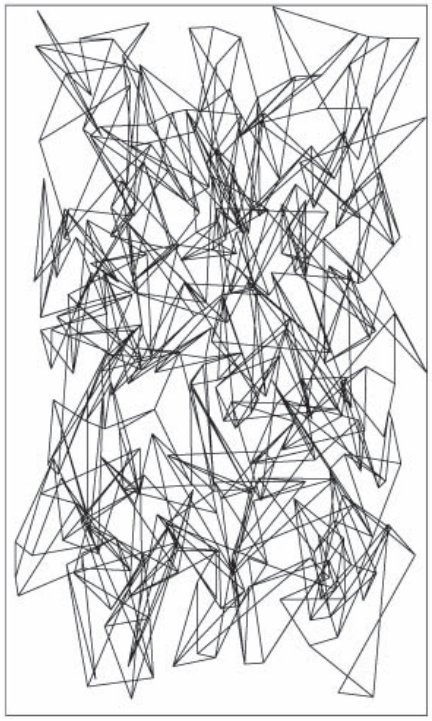

Abb. 13.1: Assignment of end effector positions to lattice sites at the beginning of the simulation.

Abb. 13.2: The reaction of the end effector to two test movements along the horizontal x-direction and vertical y-direction for the end effector positions of Fig. 13.1a.

perpendicular view of the working surface is shown. For each of the 360 neural units the end effector position pertaining to the arm configuration associated with that site is marked and connected in the familiar way with those other end effector locations that pertain to neighboring neural units. Since a similar illustration of the matrices $\mathbf{A_r}$ is not directly possible, in Fig. 13.1b we instead show the reaction of the end effector to test movements. For each of the end effector positions of the 360 neural units, the reaction of the end effector to two different target movements with velocities in the x- and in the y-direction is shown. Initially, these reactions only show a small correlation with the desired velocities, due to the considerable errors

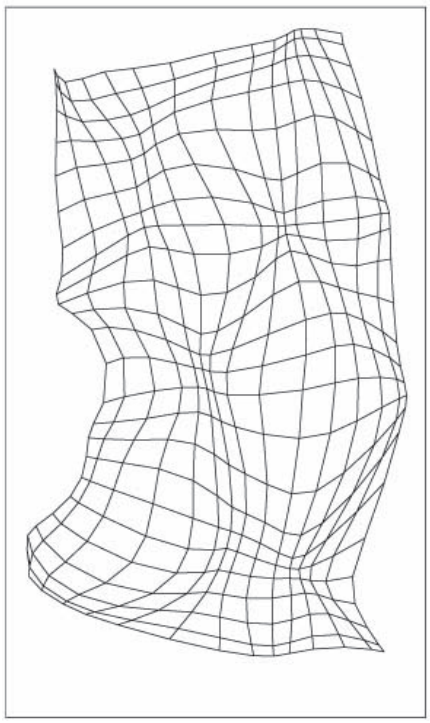

Abb. 13.3: As in Fig. 13.1a, but after 500 trial movements. By this time, a recognizable order has already emerged.

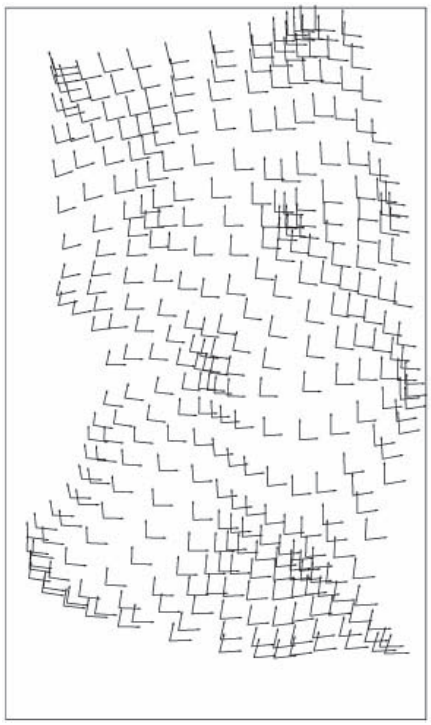

Abb. 13.4: An improved agreement with the target movements is also visible in the test movements of the end effector.

in the matrices $\mathbf{A_r}$ (see Fig. 13.1b). Figure 13.2 shows the state of the robot after 500 trial movements. At this stage, a recognizable, lattice-type correspondence between end effector positions and neural units has emerged, and the actual velocities resulting for the test movements point approximately in the x- and y-directions. Finally, Fig. 13.3 shows the result after 10,000 trial movements. In Fig. 13.3a, a regular mapping between lattice sites and end effector positions can be recognized in the working surface. The test movements are now carried out with good accuracy (Fig. 13.3b).
The representation chosen can only visualize the reaction to test movements that lie in the plane of the working surface. Therefore, for the developmental stages of Fig. 13.2 and Fig. 13.3, the Euclidean matrix norm $e_{\mathbf{r}} := \|\mathbf{A_r} -$

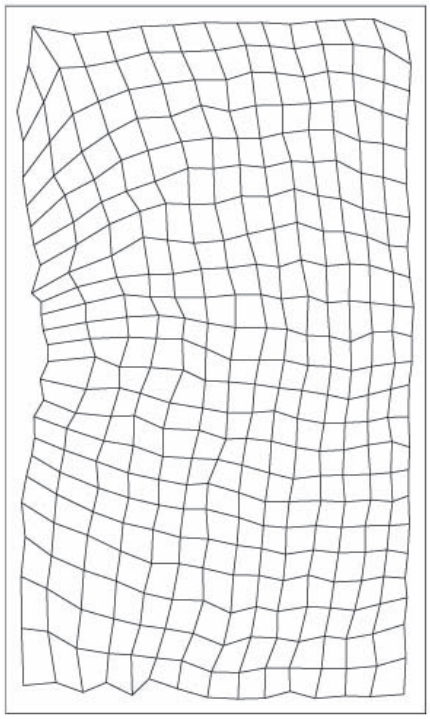

Abb. 13.5: The final result after 10,000 trial movements shows the formation of a good correspondence between lattice sites and end effector positions of the working area.

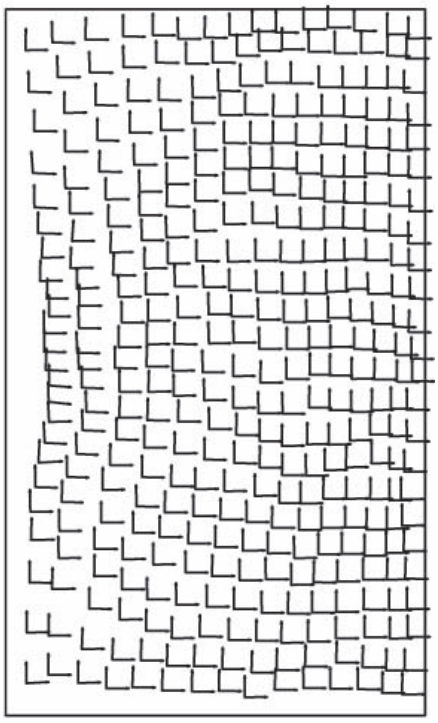

Abb. 13.6: Now the corresponding test movements of the end effector agree well with the target movements.

$\mathbf{A}^{exact}(\mathbf{w_r})\|$ of the deviation from the exact transformation matrix is given in Fig. 13.4 for each lattice site $\mathbf{r}$ as a height above the end effector position in the working surface corresponding to $\mathbf{w_r}$. Hence, an "error surface" above the working surface is created, whose height at each point is a measure of the discrepancy between desired and actual movement, averaged over all spatial directions. Figure 13.4 shows that the remaining errors are inhomogeneously distributed and are largest for those configurations in which the end effector is located near the base of the arm. This is due to the singular character of the transformation between torque amplitude and velocity for positions close

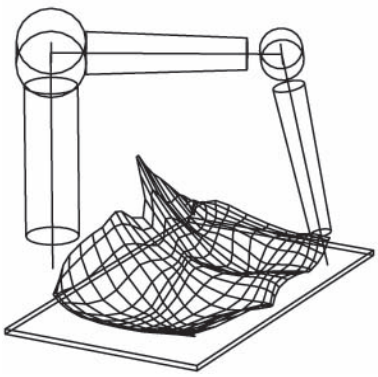

Abb. 13.7: "Error surface" above the working surface after 500 trial movements. At this time, the errors are still relatively large.

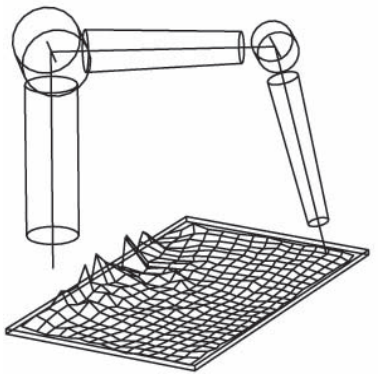

Abb. 13.8: After 10,000 trial movements, no significant errors remain except for end effector positions near the base of the arm.

to the base,[1] near which a convergence of the procedure will take a larger number of trial movements.

During our discussion of models of oculo-motor control and visuo-motor coordination in Chapters 9, 11, and 12 we repeatedly came to see the important role of neighborhood cooperation between neurons for the success of learning the output mapping. Also in the present case, neighborhood cooperation has a positive effect on the convergence of the learning algorithm. This can be illustrated by repeating the simulation as before, except that neighborhood cooperation is suppressed for the learning steps of $\mathbf{A_s}$. This is achieved by setting the parameters characterizing the range of $h'_{\mathbf{rs}}$ to values $\sigma'_i = 0$ and $\sigma'_f = 0$ (implying $h'_{\mathbf{rs}} = \delta_{\mathbf{rs}}$). Figure 13.5 illustrates the limited learning success by showing the reaction to the two test movements in the x- and y-direction. A closer inspection shows that $\mathbf{A_r}$ converges to the correct transformation only for those neural units with sufficiently "good" initial random values of $\mathbf{A_r}$. The remaining units do not achieve convergence, even if further learning steps are allowed.

[1] The singularity is analogous to the one in the transformation between joint angles and effector positions encountered in Sect. 11.4.

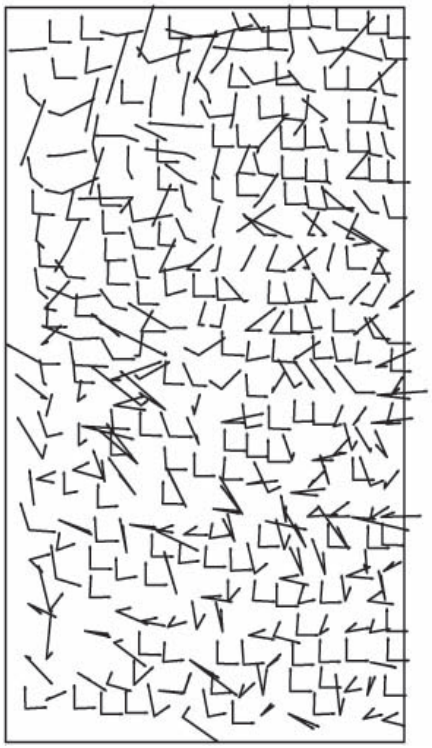

Abb. 13.9: Result of the same simulation as in Fig. 13.1-13.3, obtained after 10,000 learning steps, but without lateral interaction between the array variables $\mathbf{A_r}$, i.e., with $h'_{\mathbf{rs}} = \delta_{\mathbf{rs}}$. In this case, the desired convergence is only achieved for a fraction of all end effector configurations. This illustrates the important contribution of lateral interaction to a robust convergence behavior of the system.

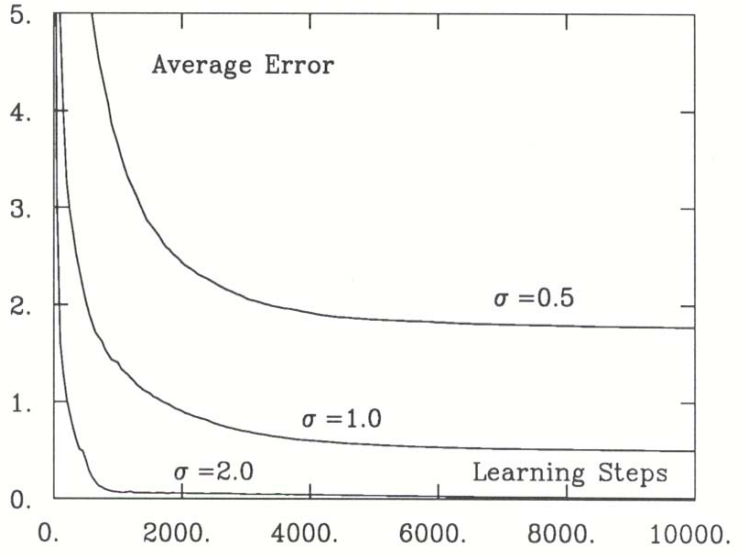

Abb. 13.10: Improvement of the convergence behavior by lateral interaction between the lattice sites. For three different initial ranges σ'_i of the lateral interaction $h'_{\mathbf{rs}}$, the diagram shows the decrease, as a function of the number of

The essential role of neighborhood-based cooperative learning is also evident in Fig. 13.6 where we present the average error $\langle \|\mathbf{v} - \mathbf{u}\| \rangle$ between the target movement $\mathbf{u}$ and the actual movement $\mathbf{v}$ performed as a function of the number of learning steps carried out. The average was taken over all lattice points and over isotropically distributed, unit target velocities $\mathbf{u}$. The three curves correspond to three simulations with the same disordered initial state but distinct initial ranges $\sigma_i' = 0.5$, $\sigma_i' = 1.0$, and $\sigma_i' = 2.0$ for the lateral interaction $h_{\mathbf{rs}}'$. The initial state was generated according to Eq. (13.8) with a value $\alpha = 2$, *i.e.*, the deviations of the initial matrices from their correct values were significantly higher than in the simulations of Figs. 13.1-13.4. The remaining simulation data were chosen as before. In the case of the long-range interaction $\sigma_i' = 2$ the error decreases fastest and the system achieves a very small residual error. In the case of shorter ranges $\sigma_i' = 1$, $\sigma_i' = 0.5$, the decay of the error during training is slowed down and only some of the matrices $\mathbf{A_r}$ manage to converge to their correct values. The residual errors are correspondingly larger, the smaller of the two occurring for the longer of the two ranges.

The procedure described here is not restricted to the learning of ballistic movements. Another conceivable application would be to learn in this manner the relationship between joint torques and the force exerted by the end effector. This would be of interest for movements in which the end effector is guided in its motion by contact with a surface and in which the contact is to be maintained with a specified contact force ("compliant motions"). Similarly, it would be possible to learn configuration-dependent joint torques compensating for the influence of gravity on the arm, thus eliminating one of the main factors responsible for changes in the end effector velocity during force-free, ballistic phases of the motion.

This concludes our investigation of the capabilities of Kohonen's model and its extensions by means of computer simulations. In the subsequent chapters, we will take a closer look at important mathematical aspects of the model and analyze some of its properties that became evident in the simulations.