TR－A－0125

Equilibrium Point Control of a Monkey Arm Simulator by a Fast Learning Artificial Neural Network

Menashe Dornay and Terence D. Sanger

1991.11.27
(1991.11.22 受付)

ATR視聴覚機構研究所

〒619-02 京都府相楽郡精華町乾谷 ☎07749-5-1411

ATR Auditory and Visual Perception Research Laboratories

Inuidani, Sanpeidani, Seika-cho, Soraku-gun, Kyoto 619-02 Japan

Telephone: +81-7749-5-1411
Facsimile: +81-7749-5-1408
Telex: 5452-516 ATR J

Equilibrium Point Control of a Monkey Arm Simulator by a Fast Learning Artificial Neural Network.

Menashe Dornay [1] and Terence D. Sanger [2]

[1] Cognitive Processes Department, ATR Auditory and Visual Perception Research Laboratories, Sanpeidani, Inuidani, Seika-Cho, Soraku-Gun, Kyoto 619-02, Japan, (FAX Num. 81-7749-5-1408).

[2] Electrical Engineering and Computer Science Department, MIT., E25-534, Cambridge, MA. 02139, U.S.A.

Abstract

A planar 17 muscle model of the monkey's arm based on realistic biomechanical measurements was simulated on a Symbolics Lisp Machine. The simulator implements the equilibrium point hypothesis for the control of arm movements. Given initial and final desired positions, it generates a minimum-jerk desired trajectory of the hand and uses the backdriving algorithm to determine an appropriate sequence of motor commands to the muscles (Flash 1987; Mussa-Ivaldi et al. 1990; Dornay 1991b). These motor commands specify a temporal sequence of stable (attractive) equilibrium positions which lead to the desired hand movement.

A strong disadvantage of the simulator is that it has no memory of previous computations. Determining the desired trajectory using the minimum-jerk model is instantaneous, but the laborious backdriving algorithm is slow, and can take up to one hour for some trajectories. In order to overcome this problem, a fast learning, tree-structured network (Sanger 1991c) was trained to remember the knowledge obtained by the backdriving algorithm. The neural network learned the nonlinear mapping from a 2-dimensional cartesian planar hand position $\{x, y\}$ to a 17-dimensional motor command space $\{u_1, \cdots, u_{17}\}$. Learning 20 training trajectories, each composed of 26 sample points $\{\{x, y\}, \{u_1, \cdots, u_{17}\}\}$ took only 20 minutes on a Sun-4 Sparc workstation. After the learning stage, new, untrained test trajectories as well as the original training trajectories of the hand were given to the neural network as input. The network calculated the required motor commands for these movements. The resulting movements were close to the desired ones for both the training and test cases.

1. Introduction

Hand movements toward stationary targets are common in the motor repertoire of primates (Georgopoulos 1986). In order to control a movement the brain must determine a desired trajectory (hand path and velocity) in visual coordinates, and realize the desired trajectory by giving a *flow of motor commands* to the muscles. Theoretically, finding a trajectory between the initial and desired hand positions is an ill-posed problem because it cannot be solved in a unique way. The experimentally found invariant tendency of primates to generate hand trajectories with roughly straight paths and bell-shaped velocity profiles in unconstrained point to point movements in the horizontal plane in front of the body defines a unique solution to that problem (Morasso 1981). To account for such kinematic features a mathematical model, the *minimum-jerk model* was proposed (Flash and Hogan 1985). The trajectories predicted by the minimum-jerk model are in good agreement with the experimental data for unconstrained movements in front of the body (Flash 1987; Uno et al., 1989). The most useful property of the minimum-jerk model is probably its simplicity. Assuming the movement to start and end with zero velocity and acceleration, the desired hand trajectory is easily computed using the following equation:

$$z(t) = z_0 + \left(z_0 - z_f\right) \times \left(15\pi^4 - 6\pi^5 - 10\pi^3\right) \qquad (1)$$

where z is x or y , (x,y) is the hand position, $\left(x_0, y_0\right)$ is the initial hand position at time $t = 0$, $\left(x_f , y_f\right)$ is the final hand position at $t = t_f$, and $\pi = t/t_f$ (Flash and Hogan 1985).

The equilibrium point hypothesis for the control of movement (EP) suggests that *stable* hand postures are used by the brain to control arm movements (Feldman 1966; Bizzi et al. 1982, 1984). The hypothesis is based on the observation that muscles behave like tunable springs (Rack and Westbury 1969). Furthermore, the muscles acting upon a joint are organized in agonist/antagonist configurations. Thus, at any level of neuromuscular activation, a limb's equilibrium position is achieved when the opposing torques generated by the agonist and antagonist muscles cancel each other. Each motor command vector specifies a unique stable equilibrium position of the arm, with a zero net force at the hand. This equilibrium position corresponds to a limb configuration in which the potential energy stored by the muscles is at a minimum (Mussa-Ivaldi et al. 1990). If the limb is displaced by some external transient perturbation, the elastic neuro-muscular properties generate a torque which restores the equilibrium configuration. According to the EP the brain generates movements by creating a temporal set of stable attractive equilibrium positions called *a virtual trajectory* (Hogan 1984). The location of the attractive equilibrium position along the virtual trajectory changes during the movement, and serves to draw the arm from the initial to the final position. The actual hand trajectory may be different from the virtual trajectory because the real movement is influenced by the dynamic properties of the arm, while the virtual trajectory is only a temporal set of static equilibrium positions. Simulation studies by Flash (1987) have shown that multi-joint arm movements at moderate speed can be generated by choosing a virtual trajectory defined by the minimum-jerk model. This finding dramatically decreases the computational task needed by a brain which uses the EP for controlling a movement. The brain only needs to calculate a set of *static* equilibrium positions on the virtual trajectory, and avoids complex *dynamic* calculations.

If the brain is to use equilibrium point control, it must compute the motor commands to the muscles which specify the equilibrium positions on the virtual trajectory. This is a moderately difficult ill-posed problem. An infinite number of different motor commands for specifying an equilibrium position are possible even in the simplified case where only 2 antagonistic muscles control the torque in each joint. The real case is more complicated because many single and double joint muscles affect the torque produced in each joint. A pseudoinverse algorithm, called *backdriving* (Mussa-Ivaldi et al. 1990), was used as a solution to the problem (Dornay 1990, 1991a, 1991b, 1991c). The goal of the backdriving algorithm is to calculate a change in the motor command to the muscles for moving the location of the equilibrium position during the movement. The backdriving algorithm finds a unique solution by using a local static optimization approach of minimizing the change in the potential energy stored in the muscles during the variation of the motor commands. The algorithm can be conceptualized in two steps. First, a passive displacement from the equilibrium point to the next position is simulated. Second, the motor commands to the muscles are chosen to cancel the resultant hand force, while minimizing the change in potential energy.

The output of the minimum-jerk model is used as the target input to the backdriving algorithm, for specifying the locations of the virtual equilibrium positions, as suggested by Flash (1987). A detailed technical report describing the implementation of the minimum-jerk model and the backdriving algorithm, using a 17-muscle model of the monkey's arm was recently described (Dornay 1991b) and should be consulted for more details. A brief description will be given in sections 2 and 3.

While both the minimum-jerk model and the backdriving algorithm solve ill-posed problems, no analytical solution exists for the latter. In order to bypass the large amount of computational time needed by the backdriving algorithm, a fast learning, tree structured artificial neural-network (Sanger 1991a, 1991b, 1991c) was trained to remember and generalize the knowledge obtained by the backdriving algorithm. The network algorithm is composed of simple elements with no prior knowledge of the arm properties, and implements a general-purpose learning mechanism.

In this paper we describe control of unconstrained point to point arm movements in the horizontal plane in front of the body, using the minimum-jerk model and the backdriving algorithm as training teachers for a fast learning "neural-like" algorithm. We describe the ability of the neural network to learn the nonlinear mapping between a 2-dimensional hand position to a 17-dimensional motor command space, and to work together with the minimum-jerk model for creating the motor commands needed for arm movements. The combined minimum-jerk model and network algorithm produced a hybrid model which can provide fast control of complex multi-joint systems, after a very short training period (20 minutes on a Sun-4 Sparc workstation).

2. A Planar 2-Joint Arm Model

A stable planar 2-joint model of the arm is shown schematically in Fig. 1. The torso, upperarm, and forearm links are modeled as rigid line segments, representing the bones. The 3 links are interconnected by the shoulder and elbow revolute joints. The relative angles of rotation are $\theta_1 \in [-45°, 90°]$ for the shoulder and $\theta_2 \in [30°, 135°]$ for the elbow. Link length and attachment centers on the bones of the muscles were measured by anatomical dissections in 2 rhesus monkeys (Table 1). Planar projection of the 3-D real structure was done by X-ray. A total of seventeen muscles including shoulder, elbow and two-joint flexors and extensors have been included in the model. The static postural stability of the hand was assured by using a detailed muscle geometry representation, including effective origins and insertions of the muscle attachments to the bones (Fig. 2).

$\boxed{\text{Fig. 1}} \rightarrow$

$\boxed{\begin{array}{c}\text{Table 1}\\ \text{Fig. 2}\end{array}} \rightarrow$

Each muscle is represented as an elastic element whose rest-length and stiffness are regulated by a motor command. A tunable linear length-force curve was attached *to each muscle* (Rack and Westbury 1969; Zeffiro 1986),

$$f = \kappa(u)\left(l(\theta) - l_0(u)\right) = \textit{muscle force} \qquad (2)$$

The motor command u can have any value from zero to one. $\kappa(u)$ is the muscle stiffness, $l(\theta)$ is the length as a function of joint angle, and $l_0(u)$ is the rest-length (length when $f = 0$).

A bigger motor command u will increase κ and decrease l_0, increasing the produced muscle force. Measured volumes and estimated rest-length were used as scaling factors, based on in vivo parameters of the triceps which served as a reference muscle (Zeffiro 1986; Dornay 1991b).

3. Hand Movement From Posture

A variety of arm movements between different positions in the hand workspace were simulated, using the minimum-jerk model for planning the trajectory and the backdriving algorithm for choosing the motor commands (Dornay 1990, 1991a, 1991b, 1991c). The trajectory predicted by the minimum-jerk model was a straight path with 26 intermediate hand-positions defining a bell shaped velocity profile. A motor command to the 17 muscles was specified for each intermediate equilibrium point. The motor command profiles for all the muscles were very smooth in all the tested trajectories. Typical virtual trajectories[1] predicted by the minimum-jerk model and the backdriving algorithms are shown in Fig. 3.

$\boxed{\text{Fig. 3}} \rightarrow$

4. The Neural Network

The neural network is based on a new algorithm for approximating continuous functions in high-dimensional input spaces. The algorithm builds a tree-structured network of variable size which is determined both by the distribution of the input data and by the function to be approximated. Efficient computation in this tree structure takes advantage of the potential for low-order dependencies between the output and the individual dimensions of the input (Sanger 1991a, 1991b, 1991c).

The neural network uses a set of Fourier basis functions. Basis function networks have

[1]As explained in the Introduction, and based on Flash (1987), only virtual static equilibrium trajectories were calculated in this work.

proven to be useful for approximating functions in a variety of different domains (Poggio and Gerosi 1989; Powell 1987; Klopfenstein and Sverdlove 1983). Such networks are represented by equations of the form:

$$\hat{g} = \sum \omega_i \psi_i(x) \tag{3}$$

where $x = (x_1, \cdots, x_p)$ is the input, g is the desired (scalar) output, $\hat{g}$ is the output approximated by the network, and the ω_i's are learned scalar weights. The ψ_i's are multi-dimensional Fourier basis functions. If the input dimension p is large, a prohibitively large number of basis functions is used. This problem is often referred to as the *curse of dimensionality*. The tree algorithm discussed in the next section attempts to avoid this problem by noting that in some regions of the input space, the desired output function can be approximated using only a few dimensions of the input.

5. Network Structure

(A concise summary is provided here. For a complete description, consult Sanger (1991a, 1991b, 1991c)). If the ψ_i's are separable functions we can assume without loss of generality that there exists a finite set of scalar-input functions $\Upsilon = \{\phi_1, \ldots, \phi_N\}$ such that

$$\psi_i(x) = \prod_{j=1}^{p} \phi_{i,j}(x_j) \quad , \quad \phi_{i,j} \in \Upsilon \tag{4}$$

If there are p input dimensions and N possible scalar functions ϕ_n, then there are N^p possible basis functions ψ_i. The tree algorithm for building the network tries to reduce the number of basis functions and coefficients that need to be computed. The network is built up one dimension at a time. Approximation using only the first dimension x_1 gives:

$$\hat{g}^{[1]} = \sum_{n=1}^{N} \alpha_n \phi_n(x_1) \tag{5}$$

where the α_n's are the weights for the one-dimensional basis functions $\phi_n(x_1)$ along this dimension, and the superscript indicates that only one dimension of the input has been used to estimate the scalar output g.

In order to train the weights, we use the LMS learning algorithm (Widrow and Hoff 1960) to reduce the mean-squared output error $E\left[\left(g - \hat{g}^{[1]}\right)^2\right]$ by

$$\Delta\alpha_n = \gamma \left(g - \hat{g}^{[1]}\right) \phi_n(x_1) \tag{6}$$

where γ is a small rate term. Given sufficient input samples x_1, this algorithm will converge until the average value $E\left[\Delta\alpha_n\right] = 0$ for all n. The output $\hat{g}^{[1]}$ will then be the best linear approximation of g based on the values $\phi_n(x_1)$.

If g cannot be well approximated using only x_1, then there will be some residual error $\left(g - \hat{g}^{[1]}\right)$. Although this error will be uncorrelated with $\phi_n(x_1)$ for all n, the variances $E\left[(\Delta\alpha_n)^2\right]$ will be non-zero, indicating that there is pressure to change the weights. Although on average this pressure is zero, for particular instances the output error would be reduced if the weights could be either larger or smaller. In general, the output error will be zero if and only if $E\left[(\Delta\alpha_n)^2\right] = 0$ for all n (Sanger 1991a). We can improve the approximation of g by allowing the weights to vary based on information from the x_2 dimension. Suppose we pick r_1 to be the value of n for which $E\left[(\Delta\alpha_n)^2\right]$ is largest. We now add a term which varies with x_2:

$$\sum_{n=1}^{N} \alpha_{r_1,n} \phi_n(x_2) \tag{7}$$

to the weight α_{r_1} (see figure 4). We use the LMS rule to learn the weights $\alpha_{r_1,n}$:

$$\Delta\alpha_{r_1,n} = \Delta\alpha_{r_1}\ \phi_n(x_2) \tag{8}$$

The approximation at the output is now:

$$\hat{g}^{[2]} = \hat{g}^{[1]} + \phi_{r_1}(x_1) \sum_{n=1}^{N} \alpha_{r_1,n}\ \phi_n(x_2) \tag{9}$$

This procedure can be followed for every value of n at each of the p dimensions. In general, the weight change at any layer in the tree is the error term for the layer below it (see Fig. 4).

$\boxed{\text{Fig. 4}} \rightarrow$

New trees are added as needed below any level node, and weights are trained using a recursive learning rule:

$$\begin{aligned}
\Delta\alpha_0 &= \gamma\,(g - \hat{g}) \\
\Delta\alpha_{r_1} &= \gamma\,(g - \hat{g})\ \phi_{r_1}(x_1) = \Delta\alpha_0\ \phi_{r_1}(x_1) \\
\Delta\alpha_{r_1,r_2} &= \gamma\,(g - \hat{g})\ \phi_{r_1}(x_1)\ \phi_{r_2}(x_2) = \Delta\alpha_{r_1}\ \phi_{r_2}(x_2) \\
&\cdots \\
\Delta\alpha_{r_1,\dots,r_{d+1}} &= \Delta\alpha_{r_1,\dots,r_d}\ \phi_{r_{d+1}}(x_{d+1})
\end{aligned} \tag{10}$$

As the tree is grown, its ability to approximate the desired function g increases. The algorithm was implemented with a time-varying rate term which decreased proportional to t^{-1} from a starting value of 0.005. The basis functions ϕ_n were the Fourier basis. The network used 10 Fourier basis functions ($sin(nx_d)$, $cos(nx_d)$ for $n = 1, \cdots, 5$) for each of the two input dimensions. To learn multiple outputs, separate trees were created for each desired output function g_i, $i = 1, \dots, 17$.

6. Training The Neural Network

The training procedure used 39 minimum-jerk trajectories from the same initial hand position {0.05, 0.33} m. The final positions were on a 0.15 m radius arc drawn from the initial position (see Fig. 5a). Each trajectory consisted of 26 equilibrium positions. Only the odd-numbered trajectories were used for training. Performance of the network was tested on the complete set of 39 trajectories. The network's task was to learn the nonlinear mapping from 2-dimensional cartesian planar hand positions $\{x,y\}$ to the 17-dimensional motor commands space $\{u_1, \cdots, u_{17}\}$.

$\boxed{\text{Fig. 5}} \rightarrow$

The network was trained on 12,000 sample points drawn randomly from the 20 training trajectories. New subtrees were added at fixed intervals every 400 samples. For the first 200 samples of each interval the weights in the new subtree were allowed to converge. During the second 200 sample interval convergence continued, but estimates of the weight variances were accumulated. At the end of the 400 sample interval a single new subtree was added below the leaf node with the highest weight variance. The network was grown to 30 subtrees. The whole training stage took 46 minutes on a Sun Sparc workstation.

7. Evaluation Of Learning

The performance of the network was investigated both at the level of motor command errors estimated by the network itself, and at the level of the predicted trajectories.

Following the training stage, both the train and test trajectories were given as inputs to the neural network. For each input hand trajectory, we used the neural network to calculate approximate muscle activities. Increasing the number of trees and complexity of the network increased the training time while decreasing the motor command errors estimated by the network (Fig. 6). No statistically significant difference could be detected in these motor command errors between the train and test trajectories, and they were pooled and treated as one group. Table 2A shows that increasing the number of trees in the network from 15 to 30 significantly decreased both the average and the variance of the motor command errors estimated by the neural network.

$\boxed{\begin{array}{l}\text{Fig. 6}\\ \text{Table 2}\end{array}} \rightarrow$

The forward arm model simulator (Dornay 1991b) was used to calculate the actual hand trajectories based on the motor commands predicted by the neural network. The accuracy of predictions was tested by comparing the desired input trajectories (Fig. 5a) with the actual trajectories which resulted (Fig. 5b,c). For a given neural network complexity (number of trees) and for a specific trajectory, the hand position errors are calculated by:

$$E = \sum_{i=1}^{K} \sqrt{(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2} \qquad (11)$$

where K is the number of intermediate points defining a trajectory (in our case $K = 26$), $\{\hat{x}_i, \hat{y}_i\}$ represents the actual trajectory predicted by the neural network + forward hand model. $\{x_i, y_i\}$ represents the desired input trajectory given by the minimum-jerk model.

Table 2B shows that increasing the number of trees in the network from 15 to 30, significantly decreased the average and variance of the errors in the hand trajectories. Again, no difference could be detected in the hand position errors between the train and test trajectories, and they were pooled and treated as one group.

8. Discussion

In this paper we have described a neural network implementation for equilibrium point control of arm movements (EP) (Bizzi et al. 1984; Flash 1987; Dornay 1991b). A virtual trajectory of the hand is planned by a *global kinematic optimization* approach (the minimum-jerk model, Flash and Hogan 1985). This virtual trajectory, and the initial motor command to the muscles, serve as inputs to a *local static optimization* approach (the backdriving algorithm, Mussa Ivaldi et al. 1990). The backdriving algorithm calculates a temporal set of motor commands to the muscles, defining a temporal set of static equilibrium positions on the virtual trajectory. This computational approach can create realistic arm movements in primates (Flash 1987), while avoiding complex dynamic calculations suggested by other models (Kawato 1991). A fast learning, general purpose, artificial neural network, with no prior knowledge of the system kinematics statics or dynamics (Sanger 1991a, 1991b, 1991c) was successfully trained to remember the knowledge of the backdriving algorithm. The network learned the nonlinear mapping from a 2-dimensional cartesian hand position $\{x, y\}$ to a 17-dimensional motor command space $\{u_1, \cdots, u_{17}\}$ by supervised learning, in which the backdriving algorithm served as a "teacher". The combined minimum-jerk model and network algorithm produced a hybrid model which can provide fast control of complex multi-joint systems, after a very short training period (20 minutes on a Sun-4 Sparc workstation). Since control of each of the 17 muscles was learned independently, a parallel implementation of the neural network algorithm could reduce the learning time by a factor of 17.

Although the current work employed the EP to avoid calculation of the dynamics, the validity of the hypothesis does not depend on the conclusion of Flash (1987) that no dynamic calculations are needed, but on whether or not a *stable* equilibrium position of the hand is defined at all times by the neuro-musculo-skeletal system (Dornay, 1991c). If biological systems do not behave as suggested by Flash (1987), than complex calculations might be needed for finding a virtual trajectory that will result in a realistic hand movement. This would make the EP less computationally attractive, but not necessarily incorrect. The approach used by us in this study focuses on creating the equilibrium positions on a <u>given</u> virtual trajectory, and is therefore basically correct as long as the use of the equilibrium point control is a valid computational hypothesis.

The definition of the motor command u used in (2) is another source of ambiguity

(Feldman 1986; Feldman et al. 1990). If we consider a simplified system in which reflex feedback does not exist, such as for deafferented muscles (Bizzi et al. 1982), the motor command must be identified with the α neuronal activity. In intact animals the α neuronal activity specified by the brain can be modified by reflex feedback, and may not remain constant while the motor command specifies an equilibrium position. Therefore the α neuronal activity that the muscles see cannot be the motor command, a constant value of which should specify an equilibrium position (Feldman 1986; Feldman et al. 1990). The motor command according to Feldman (1986) contains a combination of central inputs to the α and γ motoneurons, and to interneurons mediating proprioceptive influences upon the motoneurons. It has been proposed that an important role of reflex feedback is to compensate for sudden reductions in force which may occur in fast movements in the absence of reflexes. This sudden reduction in muscle force is called *muscle yield*, and shows some of the complex nonlinear behavior of muscles (Nichols and Houk 1976; Carter et al. 1990). If we assume that the centrally specified α neural activity defines the location of the equilibrium position, and that the role of reflexes is to compensate for the nonlinear behavior of muscles and to augment the stability of the equilibrium position, we can still use (2) as a simplified linear approximation of the complex motor command suggested by Feldman (1986). In this work we assume that the behavior of the motor command can be roughly estimated according to (2), and use this simplified assumption as an approximation for the behavior of the biological system.

A three layer sequential artificial neural network was recently reported to perform aiming movements of the limb (Massone and Bizzi 1989). The network was trained by supervised learning, and coded sensory motor transformations using an eight muscle static model of the frog leg. The approach adopted by our study is different. Massone and Bizzi's (1989) network produced both the motor commands to the muscles and the hand trajectory. The bell shaped velocity profile was imposed as specific knowledge. This resulted in a network which is quite complex, and very difficult to train. In our approach, we emphasize the generality, simplicity and speed of learning. To do so, the 2 tasks of finding the virtual trajectory and of finding the motor commands to the muscles are separated into 2 independent modules. We also assumed no specific prior knowledge of kinematics, statics or dynamics. The only knowledge imposed on the network was the dimension of the input and output spaces. This resulted in a general and efficient scheme of learning, which the brain may utilize for many other tasks in addition to motor control. Some examples are provided in (Sanger 1991a, 1991b, 1991c).

We do not propose that the learning scheme used in this work represents the whole process of motor learning in biological systems. The task of biological motor learning is much more difficult, because the "teacher" cannot directly show the correct motor commands to the "student" but can show only the desired trajectory. This problem was addressed by Jordan (1990) and termed "*supervised learning with a distal teacher.*" The task presented to the artificial neural network in this work is much simpler, because the problem of converting the error from task oriented coordinates to the motor command space is solved by the biomechanical model of the arm and the backdriving algorithm. Further, biological systems like monkeys or humans can start moving their hands from any position to any position in the workspace, and from various initial hand stiffnesses. The backdriving algorithm incorporated in the biomechanical model exhibits this ability, and the motor commands predicted by it depend both on the desired trajectory and on the initial motor commands. Different sets of motor commands (and hand stiffness) can be produced by the backdriving algorithm for the same trajectory, depending upon the initial motor command at the starting position of the hand. The simple learning employed by the neural network in this work cannot cope with this one-to-many ill-posed problem, and needs a unique mapping between the 2-dimensional hand position input to the 17-dimensional motor command output. For that reason we indeed trained the network starting always with the same initial motor commands to the muscles and from the same initial hand position.

Future extension of this work will teach the network all the knowledge available from the backdriving algorithm. The network will learn the mapping between a given equilibrium position $\boldsymbol{x}_a$ and the motor commands associated with it, and the next desired equilibrium position on the virtual trajectory $\boldsymbol{x}_b$, to a new motor command which will code for $\boldsymbol{x}_b$,

$$\{[\{x_a, y_a\}, \{u_{a,1}, \cdots, u_{a,17}\}, \{x_b, y_b\}] \rightarrow [\{u_{b,1}, \ldots, u_{b,17}\}]\} \qquad (12)$$

Using this learning strategy, hand movements from any initial position to any desired position in the workspace may be generated. The hand stiffness will be specified at all times because each 17-muscle motor command vector specifies both a unique equilibrium hand position and a unique stiffness. The power of the tree-structured network to deal with the higher dimensional input would be even more important for such a study. However, the results presented here demonstrate that a simple artificial neural network can be trained to provide accurate equilibrium point control of a realistic model of the monkey's arm.

Acknowledgments

M. Dornay would like to thanks Drs. E. Bizzi, FA Mussa Ivaldi, J. McIntyre, Dept. Brain & Cog. Sci., MIT, USA, and Drs. M Kawato, K. Nakane, E. Yodogawa, ATR, Japan, for help or support.

T. Sanger was supported by the division of Health Science and Technology, and by NIH grants 5R37AR26710 and 5R01NS09343 to Dr. Emilio Bizzi, Dept. Brain & Cog. Sci., MIT, USA.

REFERENCES

Bizzi E, Accornero N, Chapple W, Hogan N (1982) Arm trajectory formation in monkeys. Exp Brain Res 46:139-143

Bizzi E, Accornero N, Chapple W, Hogan N (1984) Posture control and trajectory formation during arm movements. J Neurosci 4:2738-2744

Carter RR, Crago PE, Keith MW (1990) Stiffness regulation by reflex action in the normal human hand. J Neurophysiol 64:105-118

Dornay M (1990) Control of movement and the postural stability of the monkey's arm. Proc 3rd International Symposium on Bioelectronic and Molecular Electronic Devices, December 18-20, 1990, Kobe, Japan, pp 101-102

Dornay M (1991a) Control of movement, postural stability and muscle angular stiffness - A 17-muscle model of the monkey's arm. Proc IEICE Neuro - Computing Meeting, NC90-106, March 18-19, Tokyo, Japan, pp 225-230

Dornay M (1991b) Static analysis of posture and movement, using a 17-muscle model of the monkey's arm. ATR technical report TR-A-0109, pp 1-30

Dornay M (1991c) Control of movement, postural stability, and muscle angular stiffness. Proc IEEE Int Conf on Systems, Man and Cybernetics, October 13-16, Charlottesville Virginia, USA, pp 1373-1379

Feldman AG (1966) Functional tuning of nervous system with control of movement or maintenance of a steady posture. III. mechanographic analysis of the execution by man of the simplest motor task. Biophysics 11:766-775

Feldman AG (1986) Once more on the equilibrium-point hypothesis (gamma model) for motor control. J of Motor Behavior 18:17-54

Feldman AG, Adamovich SV, Ostry DJ, Flanagan JR (1990) The origin of electromyograms - explanations based on the equilibrium point hypothesis. In: Winters J, Woo S (eds) Multiple muscle systems: biomechanics and movement organizations. Springer-Verlag, in press

Flash T, Hogan N (1985) The coordination of arm movements: an experimentally confirmed mathematical model. J Neurosci 5:1688-1703

Flash T (1987) The control of hand equilibrium trajectories in multi-joint arm movements. Biol Cybern 57:257-274

Georgopoulos A (1986) On Reaching. Annual Review of Neuroscience 9:147-170

Hogan N (1984) An organizing principle for a class of voluntary movements. J Neurosci 4:2745-2754

Jordan MI (1990) Motor learning and the degrees of freedom problem. In: Jeannerod(ed) Attention and performance XIII. Hillsdake Lawrence Erlbaum Associates, pp 796-836

Kawato M (1991) Optimization and learning in neural networks for formation and control of coordinated movement. In: Mayer D (ed) Attention and performance XIV. Hillsdale, New jersy: Lawrence Erlbaum, in press

Klopfenstein RW, Sverdlove R (1983) Approximation by uniformly spaced gaussian functions. In: Chui CK, Schumaker LL, Ward JD (eds) Approximation Theory, vol IV. Academic Press, pp 575-580

Massone L, Bizzi E (1989) A neural network model for limb trajectory formation. Biol Cybern 61:417-425

Morasso P (1981) Spatial control of arm movements. Exp Brain Res 42:223-227

Mussa-Ivaldi FA, Hogan N, Bizzi E (1985) Neural, mechanical and geometric factors subserving arm posture in humans. J Neurosci 5:2732-2743

Mussa-Ivaldi FA, Morasso P, Hogan N, Bizzi E (1990) Network models of motor systems with many degrees of freedom. In: Fraser MD (ed) Advances in control networks and large scale parallel distributed processing models. Albex Publ Corp, in press

Nichols TR, Houk JC (1976) Improvement in linearity and regulation of stiffness that results from action of stretch reflex. J Neurophysiol 39:119-142

Poggio T, Girosi F (1989) A theory of networks for approximation and learning. MIT AI Memo 1140

Powell MJD (1987) Radial basis functions for multivariable interpolation. In: Mason JC, Cox MG (eds) Algorithm for Approximation. Clarendon Press, Oxford, pp 143-167

Rack PMH, Westbury DR (1969) The effects of length and stimulus rate on tensionin isometric cat soleus muscle. J Physiol 204:443-460

Sanger TD (1991a) A tree-structured adaptive network for function approximation in high-dimensional spaces. IEEE Trans Neural Networks 2:285-293

Sanger TD (1991b) Basis function trees as a generalization of local variable selection methods for function approximation. In: Lippmann RP, Moody JE, Touretzky DS (eds) Advances in neural information processing systems 3, Morgan Kaufmann, Proc NIPS'90, Denver Co., pp 700-706

Sanger TD (1991c) A tree-structured algorithm for reducing computation in networks with separable basis functions. Neural Computation 2:67-78

Uno Y, Kawato M, Suzuki R (1989) Formation and control of optimal trajectory in human multi joint arm movement - minimum torque-change model. Biol Cybern 61:89-101

Widrow B, Hoff ME (1960) Adaptive switching circuits. In: IRE WESCON Conv Record, Part 4, pp 96-104

Zeffiro TA (1986) Motor adaptations to alterations in limb mechanics. Ph.D. Thesis, Dept. of Brain and Cognitive Sciences, MIT, USA

Figure Legends

Fig. 1. A stable planar 2-joint arm model. The postural stability of the arm is represented by the hand stiffness **K**. A negative-definite **K** is a necessary and sufficient condition for hand stability. **a:** When the hand is displaced from a stable equilibrium position, an elastic restoring force is observed. **b:** Plotting the restoring force magnitudes following 0.0001 m displacements around a stable equilibrium position creates a stiffness ellipse. **c:** Typical directions and relative magnitudes of the restoring forces at 0.0001 m displacements from the equilibrium position of the hand. Due to non-linearity, the directions of the restoring forces are not necessarily towards the equilibrium position. The stiffness data were obtained from the monkey arm simulator (Dornay 1991b). The stiffness representation was proposed by Mussa-Ivaldi et al. (1985).

Fig. 2. The geometric model of the shoulder flexor muscle Pectoralis Major Sternalis is shown as a typical example. The muscle is modeled as an elastic line attached to different links. It originates from the bone represented by the torso link at **1**. Its insertion to the bone which is represented by the upperarm link is at **4**. The muscle is constrained by connective tissues represented by the effective origin **2** and effective insertion **3**. The joint angle affects whether the muscle is wrapped around its pulley at the joint (**a**) or whether it is unwrapped (**b**). Note that the moment arm of the muscle is constant when it is wrapped, and depends on the joint angle when it is not wrapped. The effect of the geometric model on the postural stability of the hand is discussed in detail in Dornay (1991b, 1991c).

Fig. 3. Equilibrium trajectories. **a:** Two typical minimum-jerk paths, from **1** to **2** and from **2** to **3**. The backdriving algorithm calculated the motor commands for defining each

intermediate point on the paths as a stable equilibrium position. Six stable stiffness ellipses are drawn on the paths, showing that the equilibrium positions are attractive. **b:** The hand and joint trajectories, and **c:** the motor commands, for the path from 1 to 2. Note the correspondence between the motor commands and the joint trajectories. **U** = motor command, **S** = shoulder, **E** = elbow, **D** = double joint muscle, **e** = extensor, **f** = flexor. The serial muscle numbers refer to Table 1.

Fig. 4. Two-layer tree constructed by the algorithm. The x 's are inputs, ϕ 's are basis functions, and α 's are weights. See section 5 for further explanation. (Reprinted from Sanger 1991c by permission of MIT press).

Fig. 5. The training procedure was based on creating trajectories from the same initial cartesian hand position {0.05, 0.33} m. The final positions were on a 0.15 m radius arc from the initial position. The 39 desired trajectories (**a**) were created by the minimum-jerk model. Odd-numbered trajectories were used for training. The network learned to predict the motor commands which produced the desired trajectories. The testing trajectories were not provided during training. After the learning stage, the trained and test trajectories were provided as input. The network produced 17 motor commands for each of 26 positions on each input trajectory. The motor commands were used to produce the hand trajectories by the forward arm model, and then the produced trajectories were plotted in (**b**) and (**c**). The accuracy of predictions is tested by comparing the desired trajectories (**a**) with the predicted ones (**b**), (**c**).

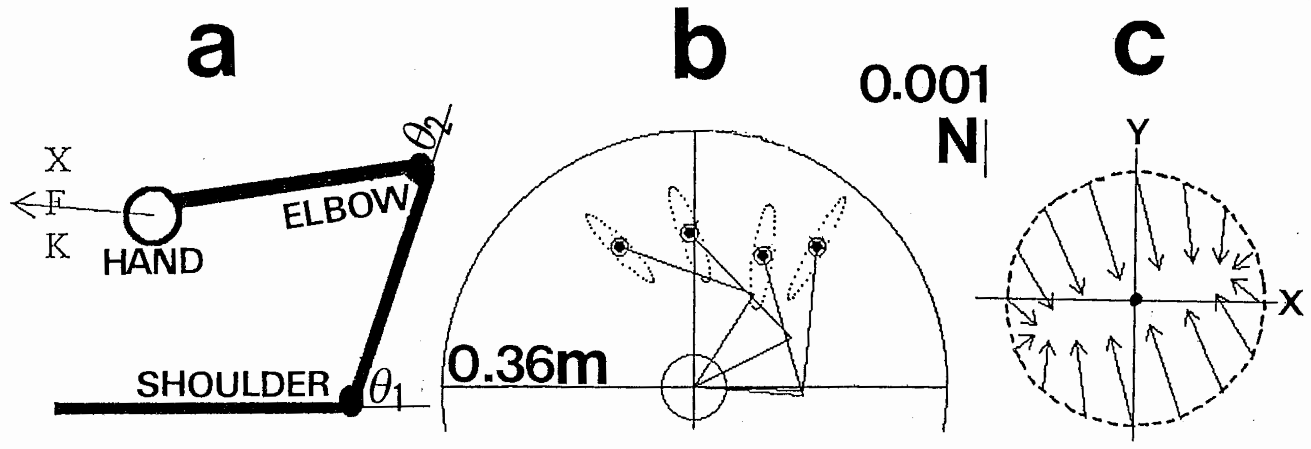
Fig. 6. Effect of the neural network complexity on the training time and on the motor command errors. More trees represent a more complex network. The average of the motor

command errors was pooled over the 39 trajectories, and is shown for each number of trees. ● = Training time; ○ = Motor command error.

Table Legends

Table 1. The dissected muscles.

Table 2. Analysis of network performance. Increasing the number of trees in the network from 15 to 30: (**A**) significantly decreased both the average and variance of the motor command errors, and (**B**) significantly decreased both the average and variance of the hand trajectory errors. Since the variances were unequal, appropriate ttest for unequal variances were used both in (**A**) and (**B**). No significant difference could be detected in the motor command or hand trajectory errors between the train and test trajectories, and they were pooled and treated as one group (N = 39 trajectories, for both **A** and **B**).



$$\mathbf{K} = \frac{\partial \mathbf{F}}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial F_x}{\partial x} & \frac{\partial F_x}{\partial y} \\ \frac{\partial F_y}{\partial x} & \frac{\partial F_y}{\partial y} \end{bmatrix} = \textit{hand stiff, ness} \qquad \mathbf{F} = \begin{bmatrix} F_x \\ F_y \end{bmatrix} = \textit{hand force}, \qquad \mathbf{X} = \begin{bmatrix} x \\ y \end{bmatrix} = \textit{hand posi tion}$$

Fig. 1

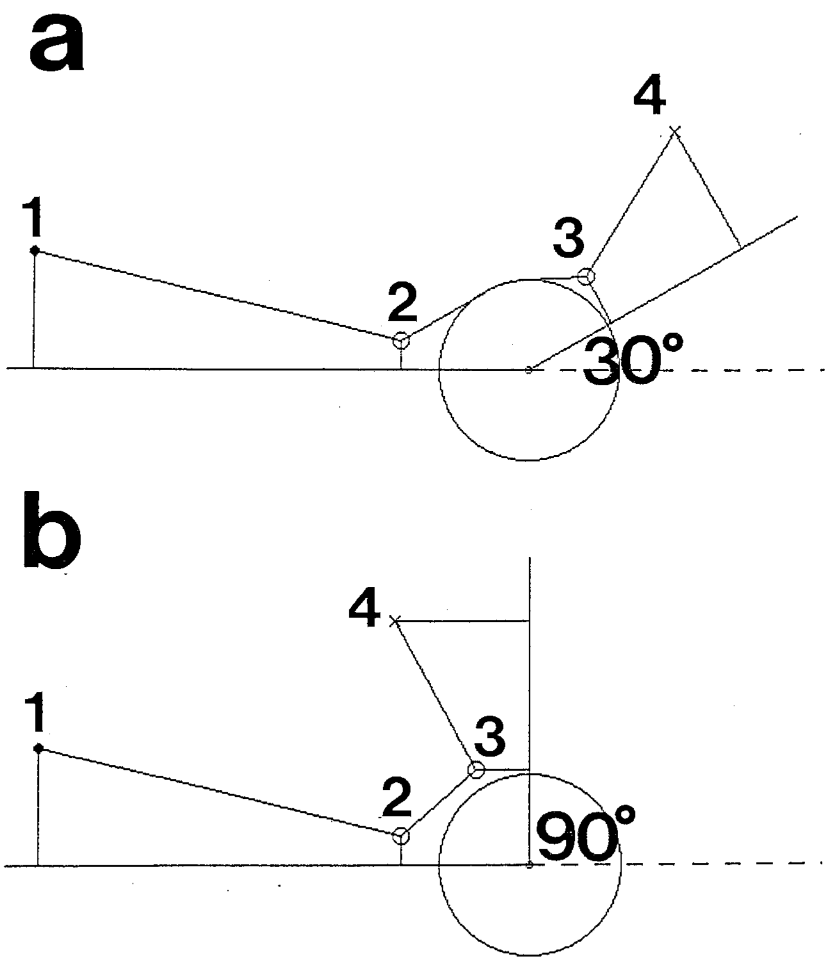
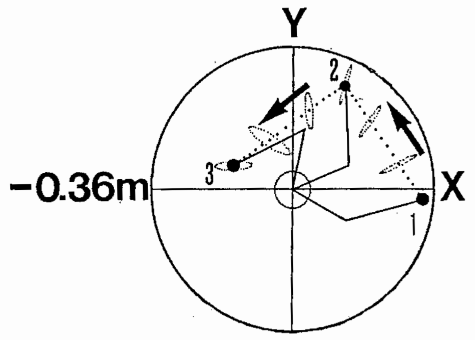
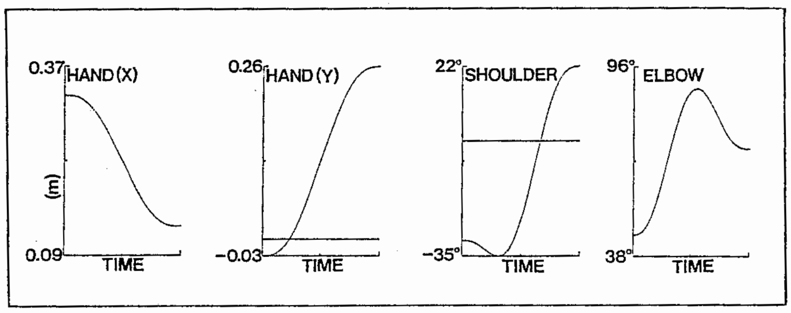


Fig. 2

a


Y
X
-0.36m
1
2
3

b


0.37
HAND (X)
(m)
0.09
TIME
0.26
HAND (Y)
-0.03
TIME
22°
SHOULDER
-35°
TIME
96°
ELBOW
38°
TIME

c

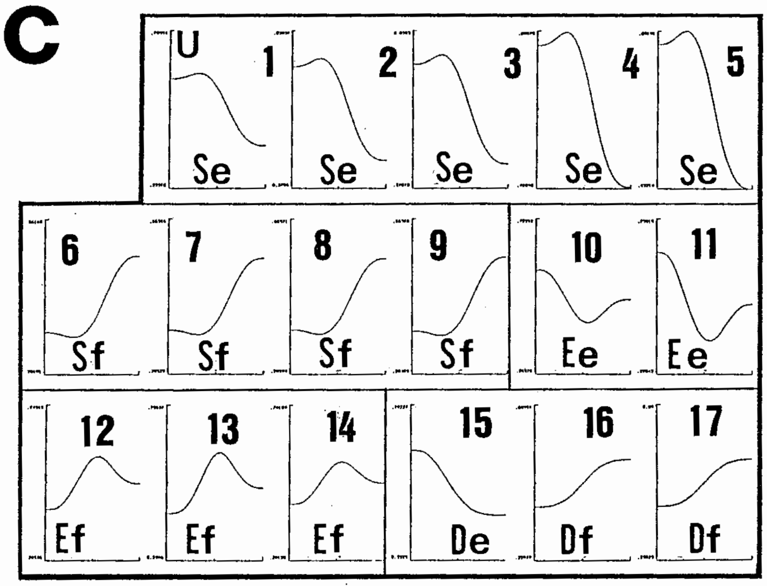

U
1
Se
2
Se
3
Se
4
Se
5
Se
6
Sf
7
Sf
8
Sf
9
Sf
10
Ee
11
Ee
12
Ef
13
Ef
14
Ef
15
De
16
Df
17
Df

Fig 3

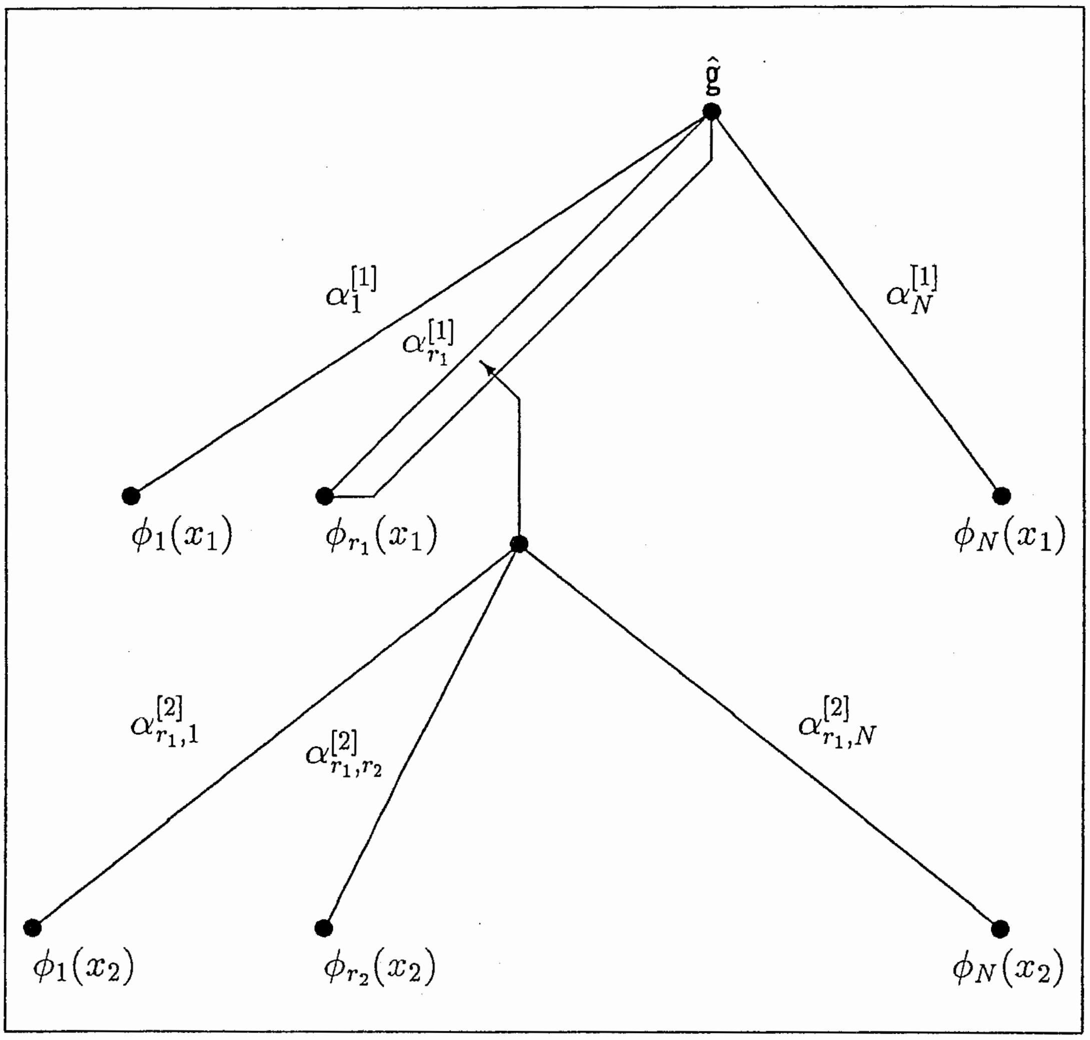

$\hat{g}$
$\alpha_1^{[1]}$
$\alpha_{r_1}^{[1]}$
$\alpha_N^{[1]}$
$\phi_1(x_1)$
$\phi_{r_1}(x_1)$
$\phi_N(x_1)$
$\alpha_{r_1,1}^{[2]}$
$\alpha_{r_1,r_2}^{[2]}$
$\alpha_{r_1,N}^{[2]}$
$\phi_1(x_2)$
$\phi_{r_2}(x_2)$
$\phi_N(x_2)$

Fig. 4

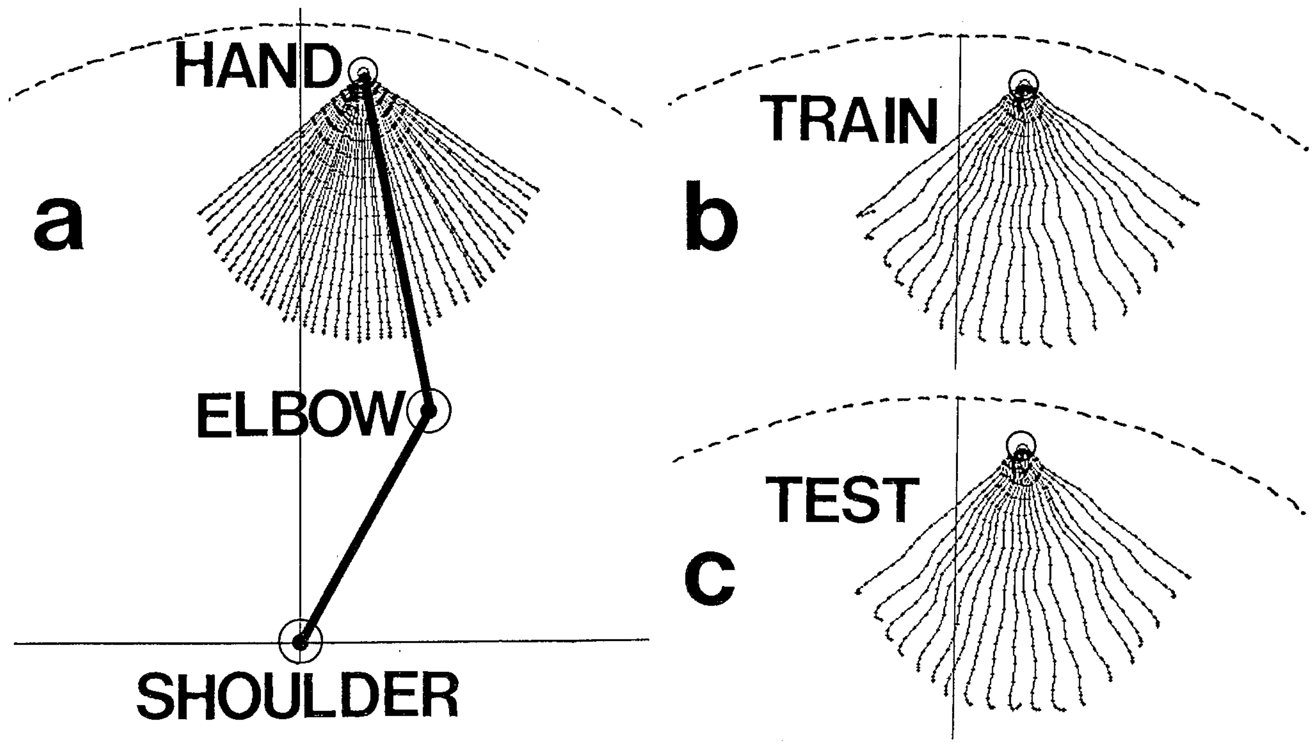


Fig. 5

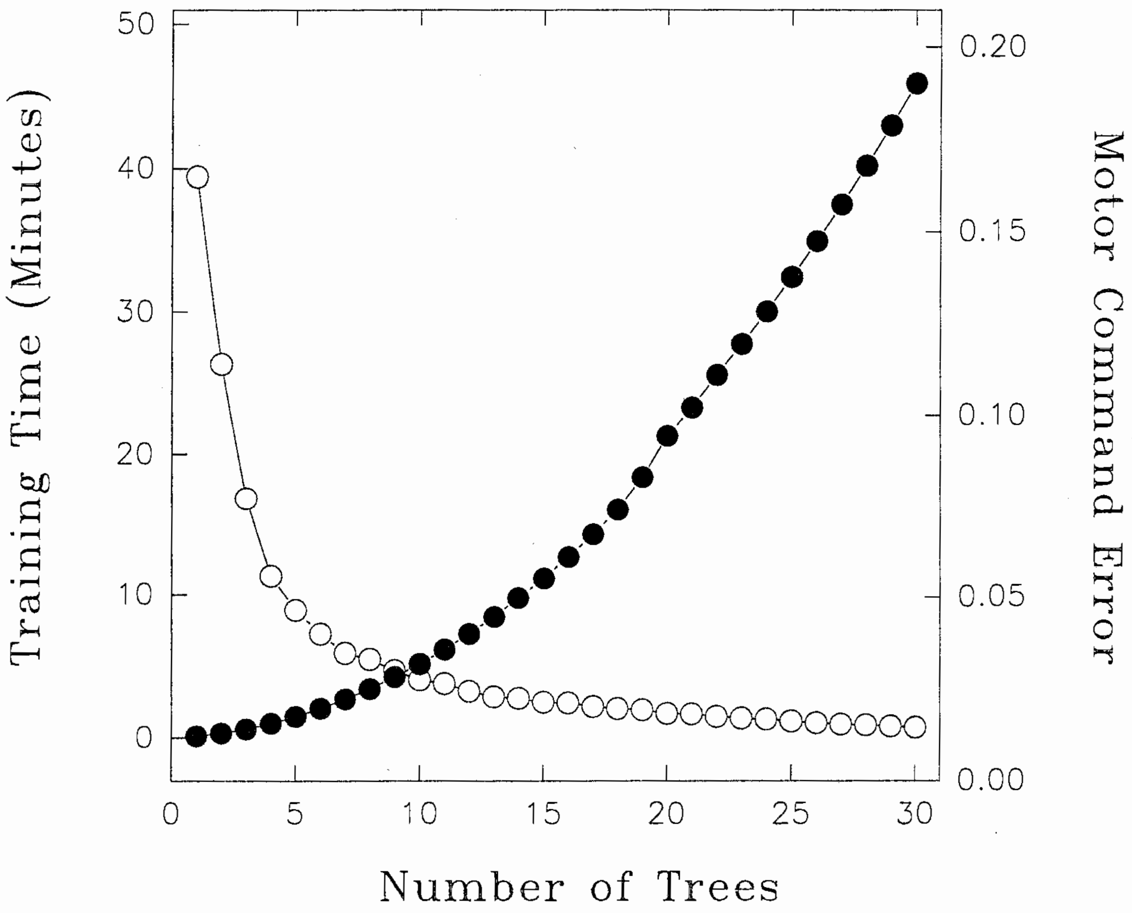

50
40
30
20
10
0
Training Time (Minutes)
0.20
0.15
0.10
0.05
0.00
Motor Command Error
0
5
10
15
20
25
30
Number of Trees

Fig. 6

TABLE 1, THE DISSECTED MUSCLES.

#	Muscle	Origin	Insertion	Function
1	Latissimus Dorsi	Vertebrae	Humerus	Shoulder Extensor
2	Posterior Deltoid	Scapula	Humerus	Shoulder Extensor
3	Teres-Major	Scapula	Humerus	Shoulder Extensor
4	Teres-Minor	Scapula	Humerus	Shoulder Extensor
5	Infra-Spinatus	Scapula	Humerus	Shoulder Extensor
6	Pectoralis Major Capsularis	Clavicula	Humerus	Shoulder Flexor
7	Pectoralis Major Sternalis	Sternum	Humerus	Shoulder Flexor
8	Anterior Deltoid	Clavicula	Humerus	Shoulder Flexor
9	Coraco Brachialis	Scapula	Humerus	Shoulder Flexor
10	Triceps Lateralis	Humerus	Ulna	Elbow Extensor
11	Triceps Medialis	Humerus	Ulna	Elbow Extensor
12	Brachialis	Humerus	Ulna	Elbow Flexor
13	Brachio-Radialis	Humerus	Radius	Elbow Flexor
14	Pronator Teres	Humerus	Radius	Elbow Flexor
15	Triceps Longus	Scapula	Ulna	2-Joint Extensor
16	Biceps Brevis	Scapula	Radius	2-Joint Flexor
17	Biceps Longus	Scapula	Radius	2-Joint Flexor

TABLE 2, ANALYSIS OF NETWORK PERFORMANCE

TASK		15 TREES (I)		30 TREES (II)		STATISTICAL ANALYSIS
		Average	Variance	Average	Variance	
A	Motor command errors estimated by the neural - network	$21 * 10^{-3}$ (100%)	$26 * 10^{-6}$ (100%)	$15 * 10^{-3}$ (71%)	$80 * 10^{-7}$ (31%)	Average: (II) < (I) ttest: $p < 0.001$ Variance: (II) < (I) ftest: $p < 0.001$
B	Hand position errors for the predicted motor commands (cm)	8.0 (100%)	14 (100%)	5.4 (68%)	4.7 (34%)	Average: (II) < (I) ttest: $p < 0.002$ Variance: (II) < (I) ftest: $p < 0.001$