Biol. Cybern. 69, 345–351 (1993)

Biological Cybernetics
© Springer-Verlag 1993

Solution of the direct and inverse kinematic problems by a common algorithm based on the mean of multiple computations

H. Cruse, U. Steinkühler

Universität Bielefeld, Abteilung für Biologische Kybernetik, Postfach 100131, D-33501 Bielefeld, Germany

Received: 10 June 1992/Accepted in revised form: 30 March 1993

Abstract. For the control of the movement of a multi-joint manipulator a "mental model" which represents the geometrical properties of the arm may prove helpful. Using this model the direct and the inverse kinematic problem could be solved. Here we propose such a model which is based on a recurrent network. It is realized for the example of a three-joint manipulator working in a two-dimensional plane, i.e., for a manipulator with one extra degree of freedom. The system computes the complete set of variables, in our example the three joint angles and the two work-space coordinates of the endpoint of the manipulator. The system finds a stable state and a geometrically correct solution even if only a part of these state variables is given. Thus, the direct and the inverse kinematic problem as well as any mixed problem, including the underconstrained case, can be solved by the network.

1 Introduction

An important problem of motor control is the coordination of movement of a multilimbed manipulator, for example, the human arm. The underlying control algorithm could work in two different domains, or coordinate systems. First, the control system might work in the joint space. This means that the position of the manipulator, in particular the position of its endpoint (end effector) is given in joint-angle coordinates, q_i. Alternatively, the control system might work in hand space or work space coordinates, x_i. This means that world coordinates, for example, of a polar coordinate system are used to describe the manipulator positions. The question as to which of these coordinate systems is used and, in particular, how to distinguish experimentally between both possibilities is extensively discussed by Hollerbach and Atkeson (1987). As sense organs might provide information in one or the other coordinate system (e.g., angle values are provided by joint receptors, whereas world coordinates are provided by visual systems), transformations in both directions might be necessary.

A number of solutions exist for these transformations, the direct kinematic problem (to compute $\mathbf{x}$ from $\mathbf{q}$) or the inverse kinematics problem (computation of $\mathbf{q}$ from $\mathbf{x}$), but these solutions have several drawbacks. First, both transformations are given as separate programs. This means that a supervising system is necessary to decide which of the two programs is required for the particular task. Second, the solutions require a number of calculations, in particular when the manipulator is redundant, i.e., has extra degrees of freedom. Third, when calculating the inverse kinematic problem, the appearance of singularities is possible. This happens for some positions of the manipulator where the determinant of a matrix involved in the transformation becomes zero and the matrix can no longer be inverted. Here we propose a system which does not involve these problems because one and the same algorithm can be used to solve the direct and the inverse kinematic problem. It is based on a massively parallel structure and is not affected by the problems of singularities.

Mussa Ivaldi et al. (1988) pointed out that the inverse kinematic problem could be easily solved by a mechanical model of the manipulator provided with springs to simulate the muscles, in the following way. By simply moving the tip of the (arbitrarily complicated) manipulator in the direction of the position of the intended target point, the joint angles of this mechanical model automatically find their appropriate values. The angle values of this model could then be used to control the real arm. Mussa Ivaldi et al. called this the *passive motion paradigm*. A similar idea was proposed by Hinton (1984). However, how could the properties of such a mechanical model be implemented in a neural network to form the basis of a "mental model" of the arm? Here we propose such a system which represents the geometry of the multi limbed arm. This system contains a neuron for each geometric variable, i.e., the joint angles or the work-space coordinates of the endpoint of the arm. These are called the state variables of the system. As input, an arbitrary combination of a subset of the state variables can be used, and the task of the system is to complete

Correspondence to: H. Cruse

automatically the whole set of state variables. This means that the same system can be used to calculate both the direct or the inverse kinematics, depending on which subset of input variables is chosen. In addition, mixed problems can be solved, as will be shown below. The system will be explained here using the simple example of a three-joint arm working in a plane, but it can be adapted to manipulators of higher dimensions.

2 The principles of the system

We consider a manipulator with n degrees of freedom. These might be provided by rotational joints or, if the length of the limbs can be changed, by translational joints. The system is redundant when, given an m-dimensional work space, n is larger than m. The state of the system can be described by at least $n + m$ variables although it is uniquely defined by any subset of n of these $n + m$ state variables. The task of the system proposed here is to complete the full set of $n + m$ values, given any subset of n values. Later, we will also show that a geometrically possible "true" state can be constructed if less than n values are given for the system.

The primary task of the system is to reconstruct the whole set of state variables even though only a subset of these variables as external input values is available. Principally, such a property is known from the classical Hopfield network (1984). The complete set of variables can be considered as an attractor. In a Hopfield-like recurrent network, the attractor can be reached, even if some of the input variables do not have the appropriate values. Therefore, the basic structure of our system is similar to that of the Hopfield network in that it contains recurrent or feedback connections between the units. However, the Hopfield network is not really appropriate for our task because it permits only a limited number of discrete attractors, whereas our task involves an infinite number of attractors since, within the geometrically possible borderlines, every point of the n-dimensional subspace of the $(n + m)$-dimensional state space should be able to act as an attractor. This can be obtained by introducing the following quantitative and qualitative changes to the basic Hopfield structure. The nonlinear properties of the activation functions, which in principle also occur in the Hopfield network, show a larger variation. They can be simple rectifiers, rectified square root functions, sine, cosine, and arccosine functions; they also can have a nonmonotonic form. As a further nonlinearity, multiplicative interactions between the outputs of two neurons are possible. A major qualitative difference to the Hopfield structure is that in our system the connections between two neurons are allowed to be strongly asymmetrical. Apart from these structural differences, the most significant property of our system is the following. A strong redundancy is introduced into the system as the value of one state variable is not calculated only once, but independently in several different ways. The final value of this variable is then determined by calculating a mean value. This will be summarized here as the mean of multiple computations (MMC).

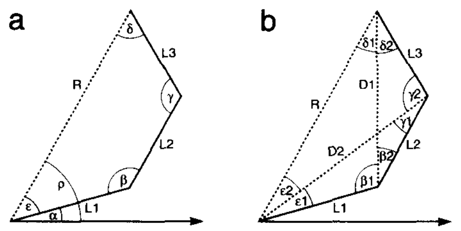

Fig. 1a, b. Definition of the geometric values to describe the position of a three-joint manipulator working in a two-dimensional plane. The values are shown in two different diagrams for clarity

Provided the connections between the units are selected properly, this system is capable of using each geometrically possible point in the $(m + n)$-dimensional state space as an attractor. This means that an arbitrary combination of n variables provided at the input will drive the system to show all $m + n$ variables at the output.

3 Model

A realization of this system will be shown here using a three-joint arm which works in a two-dimensional plane (Fig. 1a). Because $m = 3$ and $n = 2$, this system has one degree of freedom more than necessary; it is thus redundant. The first basic principle of this network is that each important geometrical value, i.e., each state variable, is not only calculated one way. Rather, different equations to calculate this value are solved, and the corresponding analog solutions are then implemented into the network. In the case of the joint angles it is helpful not only to calculate them as such, but also to calculate precursors which are defined parts of a joint angle. This leads to 17 internal state variables. Each corresponds to an output unit in the network (the units in the horizontal row in Fig. 2). The geometrical meaning of these variables is shown in Fig. 1. The multiple calculations of a variable will be explained using the angle β as an example. The following six equations are used to calculate β:

$$\beta = \arccos\,[(L1^2 + L2^2 - D2^2)/(2L1\,L2)] \tag{1}$$

$$\beta = \beta 1 + \beta 2 \tag{2}$$

$$\beta = \pi - (\varepsilon 1 + \gamma 1) \tag{3}$$

$$\beta = \pi - (\varepsilon - \varepsilon 2 + \gamma - \gamma 2) \tag{4}$$

$$\beta = \pi - (\varepsilon 1 + \gamma - \gamma 2) \tag{5}$$

$$\beta = \pi - (\varepsilon - \varepsilon 2 + \gamma 1) \tag{6}$$

As these equations show, the analog calculation requires a network which contains several nonlinearities. In Fig. 2 each set of these equations is symbolized by a black rectangle. The number of equations used for each variable is symbolized by the number of arrows connecting the black rectangle and the circular unit below. All

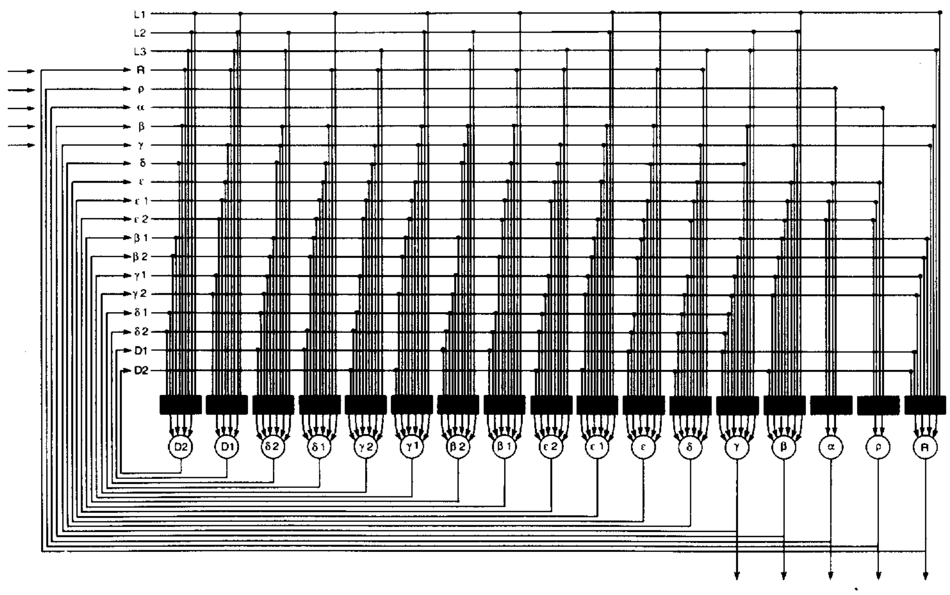

Fig. 2. Basic structure of the network used to model the geometric properties of the three-joint arm shown in Fig. 1. Each black rectangle represents a set of equations which calculate the value shown in the circle below. The *circles* stand for calculation of the arithmetic mean

these calculations are easily possible if all variables are presented at the input level (Fig. 2, left vertical column). With the appropriate hardware these calculations can all be performed in parallel and thus very fast. As Fig. 2 shows, not all possible connections occur. The connections, in particular, are asymmetrical.

The multiple computation of the internal variables introduces a redundancy into the system which has to be removed in order to obtain unique values. This is done by calculating a mean value of the different solutions presented by the system for each variable. Several possibilities exist. Here we used the simple arithmetic mean. In some cases a variable cannot be calculated, either because no solution of the corresponding equation exists (e.g., when a square root with negative argument occurs) or when an internal value enters ranges where the limited accuracy of the calculating system produces errors too large. In the first case the argument of the square root function is set to zero. To cope with the second problem, all equations corresponding to (1) are extended as follows: if one of the values of the denominator, in this example $L1$ or $L2$, becomes smaller than 0.1 then (1) is replaced by (2). Thirdly, if the argument of the arccosine function is > 1 or < -1, it is replaced by 1 or -1, respectively. The output of these mean calculations, which are symbolized in Fig. 2 as circles, is fed back to the input layer (the left-hand column in Fig. 2) and used for the calculation of the next cycle. A value can arrive at the input layer either via the feedback line, or it can be provided from outside, i.e., as an actual input value. In the system itself the internal feedback channel is suppressed when an external signal appears. The input layer in Fig. 2 also offers the possibility to change the values $L1$, $L2$, and $L3$, which describe the length of the limbs. In the simulations shown here, these values are held constant at a length of 1, but this does not affect the general validity of the results.

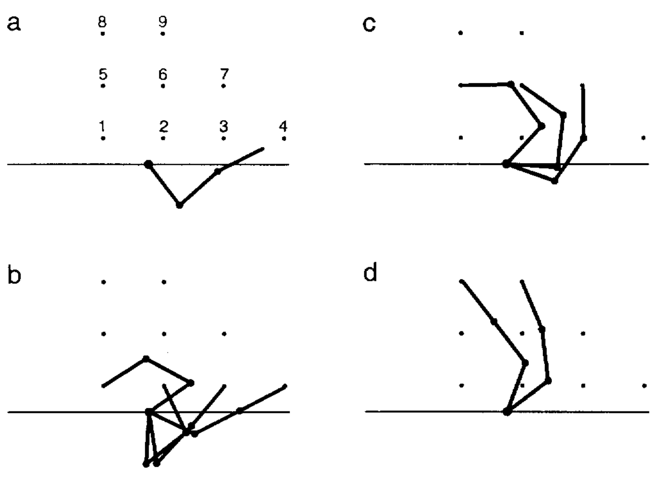

Fig. 3. a Arrangement of the nine target points and the starting position of the arm; **b–d** positions of the arm when the target points are reached

4 Results

The first question that arises is whether such a recurrent system has stable solutions. This is experimentally tested for the following situations. Nine target points, which were distributed over a large part of the work space were selected and numbered 1 to 9 (Fig. 3a). For each endpoint an arbitrary, geometrically possible, arm position was chosen which was then defined by the values of the joint angles α, β, and γ. These arm positions are shown in Fig. 3. The arm started at the position shown in Fig. 3a. Then the three angle values representing point 1 were used as input to the network, and the relaxation of the system was observed. After 30 time steps, the angles of point 2 were given as input. This was repeated for every point until point 9 was reached. Figure 4a–c shows the corresponding angle values over time. Figure 4d,e shows

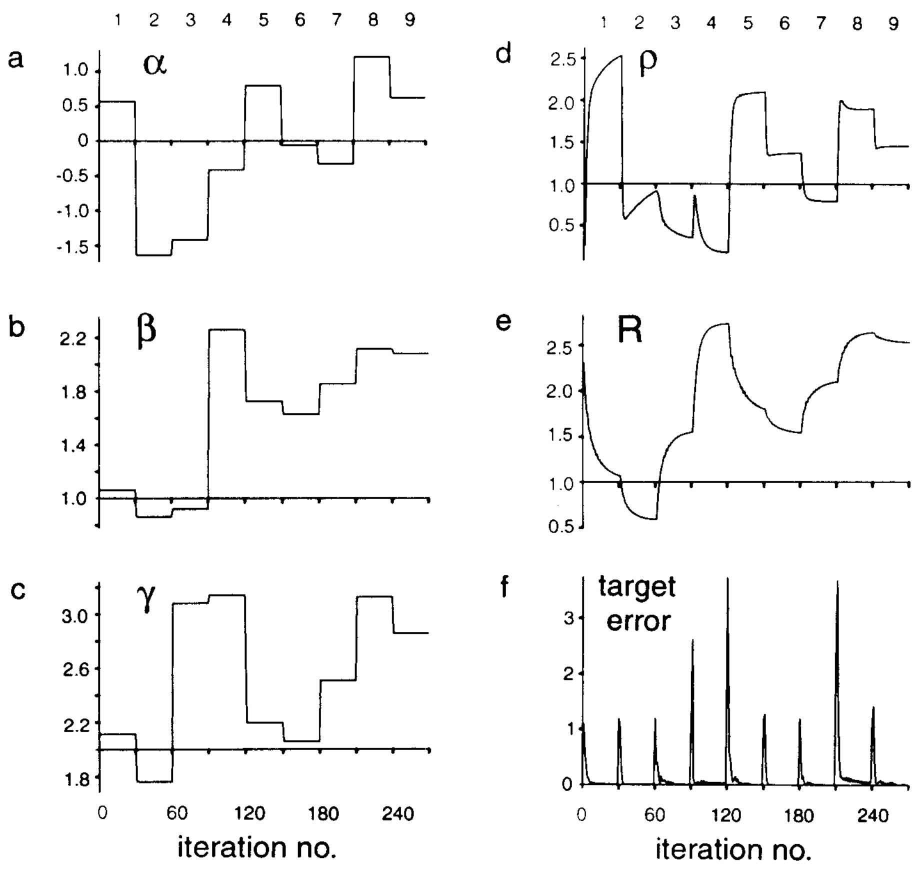

Fig. 4a–f. Solving the direct kinematic problem. **a–c** The angle values which are given as input to the system. **d–f** The temporal development of angle ρ (**d**), length R (**e**), and of the target error (**f**). Abscissa in this and the following figures is the number of iteration cycles. The numbers above this and the following figures indicate the corresponding target points as defined in Fig. 3. The angle values in this and the following figures are given in radians

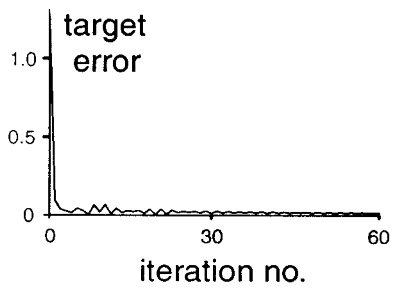

Fig. 5. The temporal development of the target error (× 0.5) when approaching point 4, shown for a longer time than in Fig. 4f

the values of angle ρ and the length of R. As can be seen, in most cases a stable value is reached before the 30th time step. This can be seen even more clearly in Fig. 4f, which shows the geometric distance between the tip of the arm, defined by the endpoint coordinates (R, ρ) and the target point (target error). Thus, this figure illustrates the relaxation behavior of the system. The target is considered to be reached when the error is smaller than 0.01 length units. In some cases the criterion is met after only 10 time steps. In other cases it takes somewhat longer than 30 steps. The worst case was found for point 4. The development of the target error in this case is shown in Fig. 5 on a longer time scale. In all cases investigated the system solves the problem of direct kinematics. When using another set of angle values for these 9 target points, the behavior of the system was not found to be essentially different.

To test the behavior of the system when solving the inverse kinematic problem, the same target points and the same starting position, as shown for Fig. 3, were used. In this case only the coordinates of the target points are given as input. This means that the system is underconstrained; it has one extra degree of freedom. Figure 6 shows the behavior in the same way as was done above for the direct kinematics. Figure 6a,b shows the input values of the polar coordinates ρ and R over time. Figure 6c–e shows the development of the three joint angles, and Fig. 6f that of the target error. In this and the following figures the target error shows the distance between target point and the endpoint coordinates of the arm, calculated from the actual values of α, β, and γ. Again, in most cases the target is reached in less than 30 time steps. In one case (point 4) the system needed about 60 steps to converge. This is shown in Fig. 7 on a longer time scale.

In the example of Figs. 6 and 7 the arm started in a somewhat extreme position as the angle of the hand joint has a negative value (see Fig. 3a). When starting from a more "comfortable" position (Fig. 8a), the system relaxes into different positions for each target point and the relaxation is a little faster in some cases (Fig. 8b).

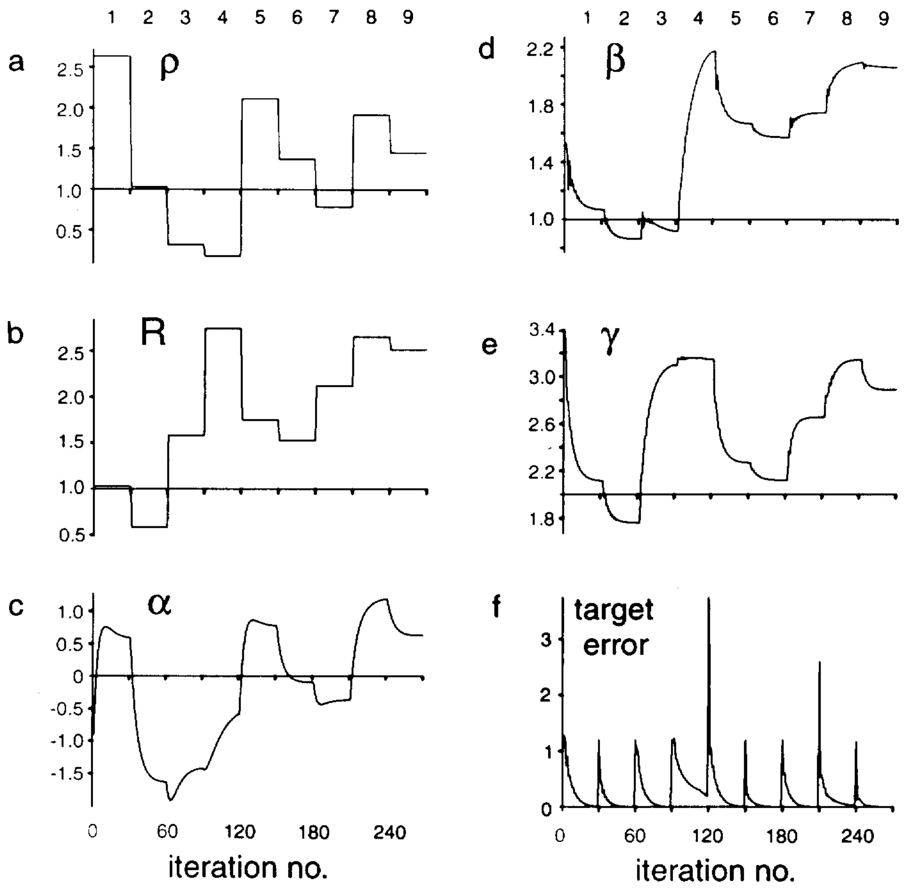

Fig. 6a–f. Solving the inverse kinematic problem. **a, b** The polar coordinates of the target points which are given to the system. **c–f** The development of angle α (**c**), angle β (**d**), angle γ (**e**), and the target error (**f**)

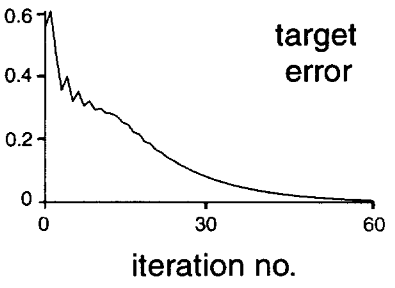

Fig. 7. The temporal development of the target error ($\times 0.5$) when approaching point 4, shown for a longer time than in Fig. 6f

Figure 9 shows the behavior of the system when the two target-point coordinates plus one joint angle, in this case angle α, are used as input. Because the same positions as above are taken, only the temporal development of the angles β and γ and the target error is shown, which is again very similar to the earlier results. Figure 10 shows the results when one target coordinate plus two other angles, β and γ, are used as input. In this case the relaxation needs a little more time for points 1–3, but relatively less time for point 4.

5 Discussion

The MMC system shown here is based on a type of network which can be applied when (1) the variables of the system depend on each other, (2) these dependencies can be described quantitatively, and (3) more variables can be defined than are necessary to describe the state of the system. The results have shown that the MMC system proposed here can be used to calculate the direct and the inverse kinematics. The system calculates a complete set of state variables, in our example the three joint angles and the endpoint coordinates when only three of these five variables are given. The system can therefore be considered to correct errors or, in other words, to complete missing information. MMC even finds a stable and geometrically correct solution when a smaller number of input values is given (underconstrained system). In this case, the system finds a solution which is influenced by the earlier state of the system (this is only implicitly shown in Fig. 8). Even for large step sizes at the input the system usually needs less than 30 cycles to find a stable solution. In no case was it found to need more than 60 cycles to approach the target point. The exact relaxation time seems to depend on different factors which could be the position of the target point within the work space, the input parameters selected, or, when calculating the

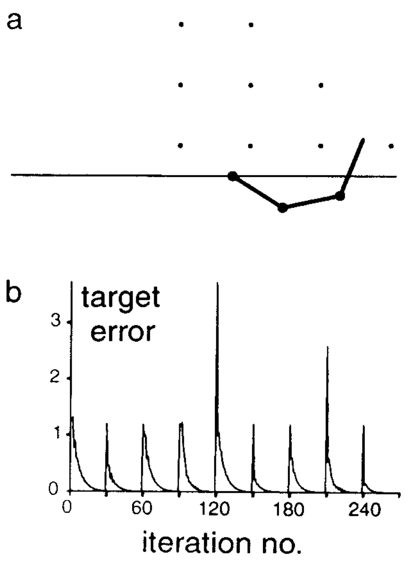

Fig. 8a, b. Solving the inverse kinematic problem. **a** The starting position of the arm. **b** The development of the target error. All arm positions are generally different from those shown in Fig. 3.

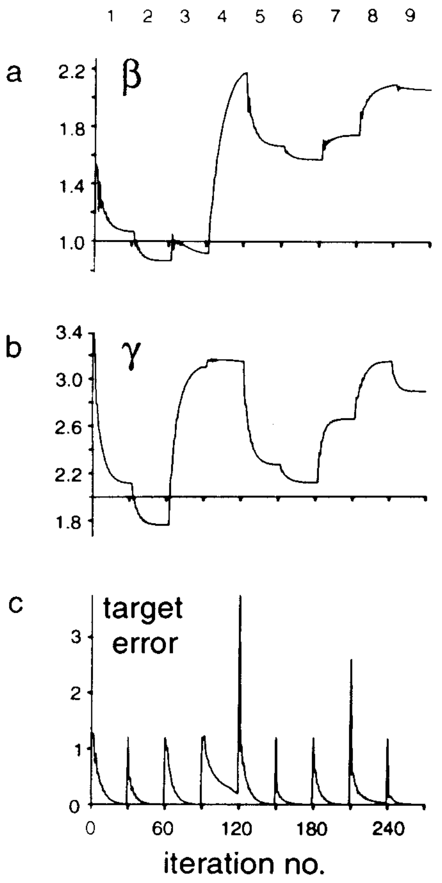

Fig. 9a–c. Solving a mixed problem. As input to the system the values of both polar coordinates and the angle α are used. These values can be found in Fig. 6a, b and Fig. 4a, respectively. Development of angle β (**a**), of angle γ (**b**), and of the target error (**c**)

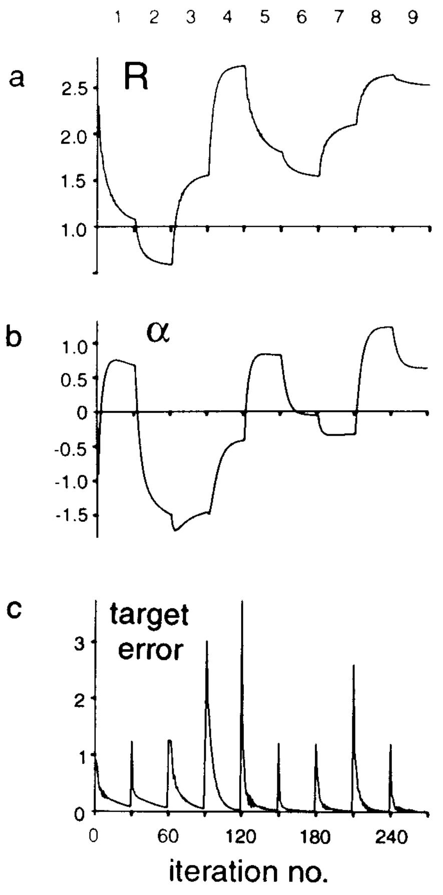

Fig. 10. Solving a mixed problem. As input to the system one polar coordinate (ρ) and two joint angles, β and γ, are used. These values can be found in Figs. 6a, 4b, and 4c, respectively. Development of the second polar coordinate (R) (**a**), of angle α (**b**), and of the target error (**c**)

underconstrained inverse kinematic problem, the starting position of the arm. These dependencies have to be further investigated. In our experiments we never found the network not to converge. Work to prove the general convergence properties of the system is in progress.

After a change in the input signals, the system needs some time for relaxation. During this time the state variables may not represent a geometrically possible configuration. Two possibilities exist to use the system for direct control of a robot arm. Either the output values have to be frozen during this "refractory period" until the internal system has relaxed – one might speculate that this corresponds to the well-known psychological refractory period – or the actual values for α, β, and γ must be used to control the arm. Oscillations occurring in these values might be overcome by using either a low-pass filter or, more simply, only the output values of every second time step.

An important problem in classical systems calculating the inverse kinematics is the possible occurrence of singularities. These arise when for some special positions a division by zero becomes necessary. Of course, this can also happen in our system, but this would only mean that one of the different parallel computations provides no significant value. Due to the redundancy of the network this does not significantly affect the behavior of the whole system. Generally, the highly parallel structure makes the system rather independent of errors occurring within the system.

As mentioned, MMC can be used to solve the direct kinematic and the inverse kinematic problem. An immediate change from one task to the other is possible simply by changing the corresponding input values. Furthermore, mixed problems can be solved, too, in the sense that an arbitrary combination of state variables can be prescribed. For example, only one of the two world coordinates might be fixed, and the other could be left to the discretion of the system. Or it is possible that one specific joint angle is clamped to a fixed value or prevented from exceeding a given value. One might also extend the external conditions such that a given joint is not supposed to move beyond a given position in the work space, for example, because of the presence of an obstacle. Furthermore, the whole system could be connected to another network in order to supervise the first system with a general aim, for example, to observe the minimum cost principle (Cruse 1986). This means that for a given end effector position, the arm should adopt a position in which each joint stays as near as possible to an optimal (minimum cost) angle value.

Morasso et al. (1989) and Morasso and Sanguineti (1991) proposed a network to control redundant motor systems, based on the idea that a mental model of the motor system is used to calculate a virtual movement, with the resulting variables of the model then being used to control the actual motor system. The model of these authors differs from ours in the following respects: their model solves the problem of redundancy by relying on the elastic properties of the muscles and ligaments. Our model is purely geometric, but there are pros and cons. Morasso et al.'s model directly provides a signal to control the muscles, whereas our model only provides angle values. On the other hand, in our system we can easily fix some arbitrary variables and, thus, can also solve the direct kinematic problem. Morasso et al.'s model chooses one specific way to solve the redundancy problem, namely taking into account the elasticity of the muscles and ligaments. Our system works on a higher level of abstraction and, therefore, opens the possibility to introduce different strategies to control the redundancy, such as the minimum cost principle. The main difference, however, is the fact that for the calculation within the network we introduce a highly redundant computation and subsequently a mean value operation on the different results, what we call the method of the MMC.

Finally, it should be mentioned that this principle could also be applied to other tasks where the formation of an internal kinematic model might be helpful. An immediate application is the following. The example shown in Fig. 1 might be interpreted as a standing upright human body, with the angle α describing the position of the ankle joint, angle β that of the knee, and angle γ that of the hip joint. The end effector might describe the position of the center of gravity of the body. If one uses cartesian coordinates (x, y) as world coordinates instead of the polar coordinates used in Fig. 1, the system can be used to control body height (y) and body displacement (x). Thus, equilibrium control and control of leg movement are not separate but can be solved by the same system.

Acknowledgements. This work was supported by BMFT grant No. 01 IN 104 B/1. We would like to express our thanks to Prof. Dr. J. Dean for helpful discussions and to A. Baker for proofreading the English manuscript.

References

Cruse H (1986) Constraints for joint angle control of the human arm. Biol Cybern 54:125–132

Hinton G (1984) Parallel computations for controlling an arm. J Motor Behav 16:171–194

Hollerbach JM, Atkeson CG (1987) Deducing planning variables from experimental arm trajectories. Pitfalls and possibilities. Biol Cybern 56:279–292

Hopfield JJ (1984) Neurons with graded response have collective computational properties like those of two-state neurons. Proc Natl Acad Sci USA 81:3088–3092

Morasso P, Sanguineti V (1991) Neurocomputing concepts in motor control. In: Paillard J (ed) Brain and space. Oxford University Press, Oxford, pp 404–432

Morasso P, Mussa Ivaldi FA, Vercelli G, Zaccaria R (1989) A connectionist formulation of motor planning. In: Pfeifer R, Schreter Z, Fogelman-Soulie F, Steels L (eds) Connectionism in perspective, Elsevier, Amsterdam, pp 413–420

Mussa Ivalid FA, Morasso P, Zaccaria R (1988) Kinematic networks – a distributed model for representing and regularizing motor redundancy. Biol Cybern 60:1–16