PROCEEDINGS OF THE 2ND INTERNATIONAL

IEEE Conference on

Tools for Artificial Intelligence

Hyatt Hotel, Dulles International Airport
Herndon, VA, USA
November 6-9, 1990

Sponsored by:

IEEE Computer Society

IEEE Computer Society Press
Los Alamitos, California

Washington • Brussels • Tokyo

IEEE Proc. Int. Conf. on Tools for Artificial Intelligence
TAI-90, Washington D.C. 1990

Optimal Information Distribution and Performance in Neighbourhood-conserving Maps for Robot Control

Dr. Rüdiger Brause

J.W.Goethe University, FB Informatik VSFT,
Postbox 111932, D - 6000 Frankfurt 11, West-Germany

Abstract

A new programming paradigm for the control of a robot manipulator by learning the mapping between the Cartesian space and the joint space (inverse Kinematic) is discussed. It is based on a neural network model of optimal mappings between two high-dimensional spaces by Kohonen.

This paper describes the approach and presents the optimal mapping, based on the principle of maximal information gain. Furthermore, the principal control error made by the learned mapping is evaluated for the example of the commonly used PUMA robot. By introducing an optimization principle for the distribution of information in the neural network the optimal system parameters, including the number of neurons and the optimal position encoding resolutions, are derived.

1. Introduction

In the standard control technique of robot manipulators the control of the joints is done in joint coordinates, leaving it to an compiler or interpreter of the list of positioning commands to do the conversion of the external, environmental Cartesian coordinates into joint coordinates (*inverse kinematics*) in advance and to produce an executable list of joint coordinates. This approach hinders the developement of flexible, mobile robots.

This paper shows the approach of *learning* the inverse kinematics by using *optimal* topology-conserving mappings and discusses their resource requirements for tolerable positioning errors in the case of a PUMA robot manipulator, shown in figure 1 with a cubic workspace.

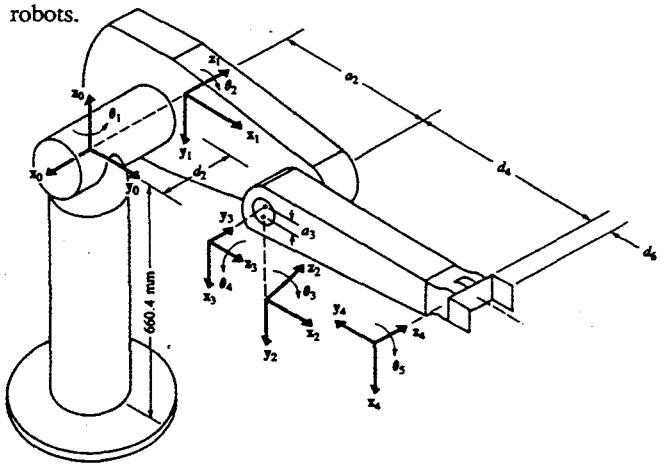
2. Robot control by topology conserving mappings

One of the best known algorithms showing neighbourhood (topology)-conserving properties is the one introduced by Kohonen 1982 [3] or [4] and analyzed for instance by Ritter and Schulten [8]. Let us now briefly describe this algorithm for the case of a 3-dim robot workspace.

The non-linear mapping

Consider as input space $X \subset \Re^3$ the Cartesian space with the input events $\mathbf{x} = (x_1,x_2,x_3)$,and an output space $\{\mathbf{y} = (i,j,k)/ i,j,k \text{ from } 1..n\}$. So the input space is projected on an output space of descrete points $\mathbf{y}$ (*neurons*), determined by 3 natural numbers (indices). To each $\mathbf{y}$ of the output space there corresponds a set $\{\mathbf{x}\}$ of points (*a class*) of the input space. Since it is finite and bounded, the whole set of points $\{\mathbf{y}\}$ can also be ordered by one index k =1..N.

Let every point $\mathbf{y}$ (neuron) weight the input by one weight per input component, i.e. by a weight vector or *class prototype* $\mathbf{w} = (w_1,w_2,w_3)$ from X. Then the mapping of the sensor space



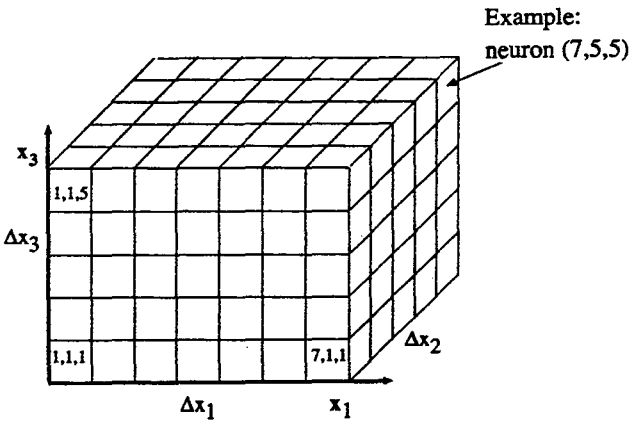


Fig.1 The PUMA robot manipulator [2] and a cubic workspace

(perhaps deformed by sensor characteristics) to the Cartesian space is done by $\mathbf{x} \mapsto y_c=(i,j,k)$ with

$$|\mathbf{x}-\mathbf{w}_c| = \min_k |\mathbf{x}-\mathbf{w}_k| \qquad (2.1)$$

This input-output mapping defines a neighbourhood of points $\mathbf{x}$ around every $\mathbf{w}_c$ to be mapped to the neuron y_c. The whole input space is devided into classes of disjoint point sets, one for each workspace cell of figure 1.

The following stochastic learning step for the weights has topology-conserving capabilities (see [5]) and basically implements a gradient search for the least mean squared error (LMSE):

> In the (t+1)-th iteration step, change the weight vector $\mathbf{w}_k$ for all neurons y_k which are in the neighbourhood of y_c

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \gamma(t+1)\, h(t+1,c,k)\, [\mathbf{x}(t+1) - \mathbf{w}_k(t)] \qquad (2.2)$$

This is accomplished by the *neighbourhood function*

$$h(t, c, k) = \begin{cases} 1 & \text{if } y_k \text{ is in the neighbourhood } N_c(t) \text{ of } y_c \\ 0 & \text{else} \end{cases}$$

and the conditions for the *learning rate* $\gamma(t)$

$$\lim_{t\to\infty} \gamma(t) = 0, \quad \sum_{t=1}^{\infty} \gamma(t) > \infty, \quad \sum_{t=1}^{\infty} \gamma(t)^2 < \infty \qquad (2.3)$$

e.g. $\gamma(t) := 1/t$

The neighbourhood function h(.) can be varied; for instance Ritter and Schulten [9] assumed h(.) to be a Gaußian-shaped function, e.g. $h(t,c,k) := \exp(-(y_c-y_k)^2 / 2\sigma(t)^2)$, instead of a step function used by Kohonen [5]. In both cases, the neighbourhood is made smaller with increasing t by decreasing the step-width or the standard deviation σ of the Gaußian distribution.

The difference of this stochastic algorithm, minimizing the least mean square error (LSME), to the classic ones (see e.g. [10]), lies in the definition of a neighbourhood for the learning process. In the classic case, either all weights (*class prototypes*) are updated (which cause fluctuations in one part of the mapping to pass to other, more distant parts) or only one weight (the selected class prototype) is updated, resulting in a poor convergence of the weights of rare selected neurons. In figure 2a a sequence of converging states of the mapping of a set of 2-dim inputs to a 2-dim neural network is shown. In the rectangle of the 2-dim input space the set of weight vectors $\{\mathbf{w}\}$ is drawn, each one connected with its nearest 4 neural neighbours; thus forming a 2-dim grid. The neural network itself is not shown.

As we can see, the random chosen initial values of the weight vector (first picture with iteration count 0) are properly adapted reflecting the ordered, 2-dim topology of the input distribution (last picture, after 100000 iterations).

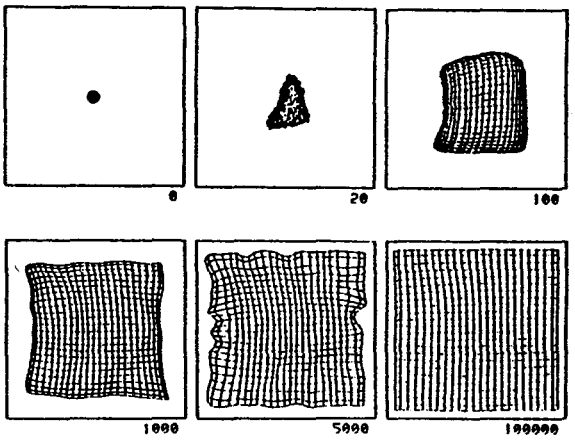


Fig 2a learning of a 2-dim topographic mapping (from [5])

Robotics

To use this non-linear mapping for robot control, we will try to replace the analytic inverse kinematics by the "Kohonen mapping" defined in (2.1) for the three joint coordinates of the point on the palm of the robot manipulator.

To each Cartesian position $y_c=(i,j,k)$ there corresponds by the non-linear mapping a joint coordinate position $\Theta_c=(\theta_1,\theta_2,\theta_3)$ which should be learned.

Denoting $\mathbf{u}:=\Theta_c$ we get the stochastic approximation learning rule in the neighbourhood h(.) by

$$\mathbf{u}_c(t+1) = \mathbf{u}_c(t) + h(.)\gamma(t+1)[\mathbf{u}_c^*(t+1)-\mathbf{u}_c(t)] \qquad (2.4)$$

with the $(t+1)^{th}$ estimation $\mathbf{u}_c^*$ of $\mathbf{u}_c$.

3. Optimal mappings and maximal information gain

Let us consider a mapping as it is defined in equation (2.1). Since sets of points of the input space are mapped to single points in the output space, there is certainly less information in the input than in the output pattern. A plausible principle of a good mapping is to transmit as much information from the input to the output as possible (*maximal information gain principle*). This optimality criterion was proposed by Linsker [6], who suggested that this might be a fundamental principle for the organization of biological neural systems.

Knowing the input pattern $\mathbf{x}$, the Shannon information gain from the N output points $\mathbf{w}_i$ is

$$I_{trans} = I_{out} - I_{out/inp} = -\ln[P(\mathbf{w}_i)] + \ln[P(\mathbf{w}_i/\mathbf{x})]$$

The average transmitted information for all inputs and outputs is with the expectation operation $\langle f(\mathbf{w}_i)\rangle := \Sigma_{\mathbf{w}_i} P(\mathbf{w}_i)\, f(\mathbf{w}_i)$

$$\langle I_{trans}\rangle_{\mathbf{w}_i,\mathbf{x}} = \langle I_{out}\rangle_{\mathbf{w}_i,\mathbf{x}} - \langle I_{out/inp}\rangle_{\mathbf{w}_i,\mathbf{x}}$$
$$= -\Sigma_i P(\mathbf{w}_i)\ln[P(\mathbf{w}_i)] - \Sigma_{\mathbf{x}} P(\mathbf{x}) \Sigma_i P(\mathbf{w}_i/\mathbf{x})\ln[P(\mathbf{w}_i/\mathbf{x})]$$

The average transmitted information $\langle I_{trans} \rangle$ is maximized when

$$\langle I_{out} \rangle_{w_i,x} \stackrel{!}{=} \max \qquad (3.1)$$

and

$$\langle I_{out/inp} \rangle_{w_i,x} \stackrel{!}{=} \min \qquad (3.2)$$

It is easy to see [1] by variation analysis that (3.1) is satisfied when $P(\mathbf{w}_i) = P(\mathbf{w}_j) = 1/N$ for all i and j.
Furthermore, if every input pattern $\mathbf{x}$ is only assigned to one appropriate class $\mathbf{y}_i$, we have $\langle I_{out/inp} \rangle = 0$ which satisfies condition (3.2). This means, that also for the maximal average information transmission the condition $P(\mathbf{w}_j) = 1/N$ is sufficient.

What does this mean for the *density of the classes* (number of classes per input space area unit, also called *magnification factor* $M(\mathbf{x})$) in the input space ?
It can be shown [1], that the condition above implies $M(\mathbf{x}) \sim p(\mathbf{x})$. In other words, for the topology conserving mapping which preserves the maximum of information *the point density of the class prototypes must approximate the probability distribution of the input patterns*.
It should be noted that this is contrary to the findings of Linsker himself in [7], who argued that in optimal topology-conserving maps the often referenced classes should become bigger in the space, not smaller.
For the algorithm of section 2, Ritter and Schulten [8] found that $M(\mathbf{x}) \sim p(\mathbf{x})$ is not generally true in the n-dim case. For the linear, 1-dim case they found $M(\mathbf{x}) \sim p(\mathbf{x})^{2/3}$, contrary to Kohonen [4]. For the 2-dim (complex) case they also found $M(\mathbf{x}) \sim p(\mathbf{x})$. Therefore, at least for the 2-dim case, Kohonens mapping can be termed *optimal*.

For robot control the optimality criterion above is quite instructive to interprete. If we have regions of the action space where the action occur very often, this region should be better controlled and should have therefore a better resolution to minimize the average control error.

4. The linear approximation

The positioning algorithm presented in section 2 is far too rough. Since we map a real-valued position $\mathbf{x}$ to an indexed position $\mathbf{y}_c$=(i,j,k) with a certain Θ_c, we get a positional error: For a cubic workspace with the edge-length of 70 cm and N=1000 neurons we have an error of $7x3^{1/2}$=12.12 cm which is much too high for normal robot operation. To reduce this resolution error, we approximate the true position $\Theta_{true}(\mathbf{x})$ by the sum of the coarse resolution value Θ_c and a linear approximation $\Delta\Theta = \mathbf{A}\,(\mathbf{x}-\mathbf{w})$, the first term of a Taylor expansion:

$$\Theta(\mathbf{x}) = \Theta_c + \Delta\Theta = \Theta_c + \mathbf{A}_c\,(\mathbf{x}-\mathbf{w}_c) \qquad (4.1)$$

Certainly, the matrix $\mathbf{A}_c$ is a good approximation only for a small section of the output space and is therefore different for different positions (i,j,k). With the redefinition $\mathbf{u}_c := (\theta_1,\theta_2,\theta_3,A_{11}, \ldots ,A_{33})_c^T$ we can learn both Θ_c and $\mathbf{A}_c$ in (2.4). The new estimations of Θ_c and of $\mathbf{A}_c$ are obtained by using the measured error $(\mathbf{x}-\mathbf{x}_F)$ of the final position $\mathbf{x}_F$ to the desired position $\mathbf{x}$ in the linear approximation

$$\Theta_c^* = \Theta_c + \mathbf{A}_c(\mathbf{x}-\mathbf{x}_F) \qquad (4.2)$$

and $(\mathbf{A}_c^*)_{ij} := [\theta_i(\mathbf{x}_F+d\mathbf{x}) - \theta_i(\mathbf{x}_F)]/dx_j$
$= [\mathbf{A}\,d\mathbf{x}]_i / dx_j \approx [\mathbf{A}(\mathbf{x}-\mathbf{x}_F)]_i / (\mathbf{x}-\mathbf{x}_F)_j$

which uses the fact that $\mathbf{A}$ is the first derivation in the first term of the Taylor expansion. A more complex, but faster converging estimation of $\mathbf{A}_c^*$ is developed in [9] which uses the value $\mathbf{x}_I$ of an intermediate positioning as additional information.

Nevertheless, on principle there rests a positioning error due to the linear approximation for a non-linear function. Let us compute this error for a linear path in the cubic workspace of a PUMA robot (see fig. 4a).

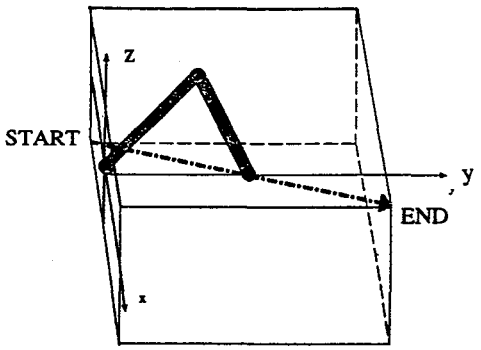


Fig. 4a A linear path in the workspace

Let us assume that the position events are equally distributed in the workspace, the algorithm with the estimation for the joint coordinates of $\mathbf{w}_c$ has converged to the true value. Then Θ_c is the true inverse kinematic transformation at $\mathbf{w}_c$; the matrix $\mathbf{A}_c$ has converged, too, and is identical to the first derivate of $\Theta_{true}(\mathbf{x})$ at $\mathbf{w}_c$. Knowing the analytical solution for the PUMA robot [2] we can compute the maximal positioning error of the approximation (see fig.4b) for each neuron $\mathbf{y}_c$ in the linear path [1].

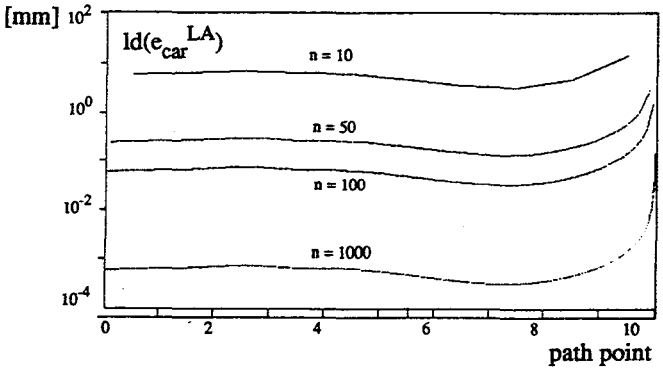


Fig. 4b The absolute Cartesian positional error as a function of the neuron number

When we regard figure 4b, we notice that the functions for n=10,100,1000 seem to be the same in one figure, only shifted for a certain, constant amount. Thus, the logarithm of the error of the linear approximation $\lg(e^{LA})$ should be linear in the decimal logarithm of n :

$$\lg(e^{LA}) \sim -\lg(n) \quad \text{or} \quad \lg(e^{LA}) = a + b \lg(n), \; b<0 \qquad (4.3)$$

This gives us an expression for the error of the linear approximation

$$e^{LA} = C\, n^b \qquad \text{with } C := 10^a \qquad (4.4)$$

which fits the data of the simulation very well [1].
The approximation error e^{LA} can be seen as a kind of resolution error or error of lacking information of the network due to the finite, limited number of neurons.

5. Optimal information distribution and performance

Let us evaluate now the relation between the information storage size and the maximal error made by the linear approximated position. By this evaluation, we hope to get some hints how to choose optimally the neural network parameter *n* and the resolutions (number of bits) r_w,r_θ and r_A of the variables w_c, $\Theta_c$and A_c which determine the mapping in equations (2.1), (2.2), (2.4) and (4.1). Since the number of neurons and the resolution r=ld(2^r)=ld(number of possible states) :=information(neuronal variable) represent a certain distribution of information, the question for an optimal system configuration becomes the question for an *optimal information distribution.* Let us first regard the conventional approach with constant resolutions.

Constant position resolutions

Assuming a workspace of X_1=X_2=X_3= 71.7 cm length a stored number of 12 bit resolution gives us an resolution increment (error) of 0.175 mm; a 10 bit resolution gives only 0.7 mm resolution.
Since our system is specified for each "neuron" by 3 weights of w_c, 3 joint coordinates Θ_c and 9 matrix coefficients of A_c we have for $N = n_1 n_2 n_3 = n^3$ neurons with the same resolution of $r = r_w = r_\theta = r_A$ bits in each number (storage element) and a necessary storage of

$$s = n^3 (3r_w+3r_\theta+9r_A) \quad \text{Bits} \qquad (5.1)$$

or $n^3 (3+3+9) = 15\, n^3$ storage elements (SE)

With the network parameter *n* we get the following table of information storage requirements.

As we can see, a good spacial control resolution (high n) is closely related to high storage requirements. Since the necessary storage is a function of the order $O(n^3)$ of the spacial control resolution, the practical application of the algorithm is limited in the present implementation stage by the storage requirements.

n neurons N	*10* 10^3	*50* $1.25\ 10^5$	*100* 10^6	*1000* 10^9
number of SE	$1.5\ 10^4$	$1.87\ 10^6$	$1.5\ 10^7$	$1.5\ 10^{10}$
3SE = 4byte (10bit res.)	20kB	2.5MB	20MB	20GB
2SE = 4byte (12bit res.)	30kB	3.74MB	30MB	30GB

Fig. 5a Resolutions and storage requirements

It should be noted that this calculation is independant whether the algorithm is implemented in VLSI hardware by neuron-like structures or merely simulated on a conventional computer system.

Optimal information distribution

Let us assume a fault-free transformation of joint angles to Cartesian coordinates by the robot manipulator mechanics. Then the Cartesian error can always directly be calculated by the known direct kinematics when the error in the joints are given. The overall maximal positioning error e^{MAX} is determined by the superposition of the two independant sources of error, the error e^{LA} of the linear approximation and the error e^{RES} of the information distribution (resolutions r_w,r_θ and r_A and number N(n) of neurons) in the neural network:

$$e^{MAX} = |\, e^{LA} + e^{RES} |\, \qquad (5.2)$$

For a certain change Δs in the information storage distribution the error will change by

$$\Delta e^{MAX}(s) = \frac{d}{ds} e^{MAX}(s)\ \Delta s \qquad (5.3)$$

$$= [\, \frac{\partial}{\partial n} e^{MAX}(n) \frac{\partial}{\partial s} n(s) + \frac{\partial}{\partial r_w} e^{MAX}(r_w) \frac{\partial}{\partial s} r_w(s)$$
$$+ \frac{\partial}{\partial r_\theta} e^{MAX}(r_\theta) \frac{\partial}{\partial s} r_\theta(s) + \frac{\partial}{\partial r_A} e^{MAX}(r_A) \frac{\partial}{\partial s} r_A(s)\,]\ \Delta s$$

If all terms of the sum are equal, no information redistribution can diminish the error any more. We can therefore define:

> In an *optimal information distribution* a small (*virtual*) change in the distribution (a change of n, r_w,r_θ or r_A) neither increases nor decreases the positioning error.

This leads us to a system of three equations with the four variables n, r_w, r_θ and r_A. In [1] this is solved, getting three variables as a function of the forth. By additionally using the storage equation $s=n^3 3(r_w+r_\theta+3r_A)$ we can finally calculate the optimal information distribution parameters n, r_w, r_θ and r_A, shown in figure 5b for a given overall information storage, and the maximal joint positioning error $e^{MAX}(s_{opt})$ using those optimal parameters.

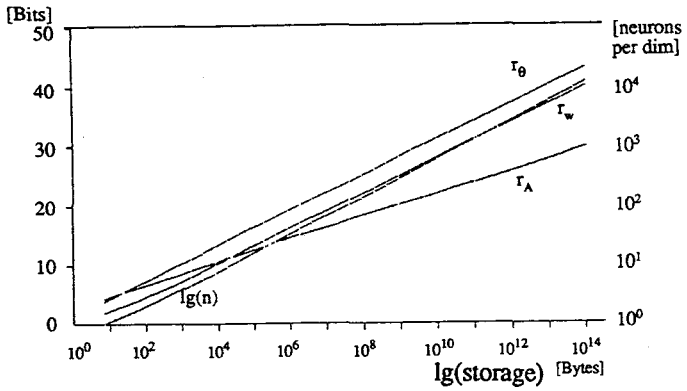


Fig. 5b The optimal information distribution parameters

The corresponding maximal Cartesian error is plotted in figure 5c for the point P in the linear path (cf. fig. 4b). For comparison, in the same plot the error using optimized n, but an optimal equal resolution $r_w=r_\theta=r_A=: r$ is additionally shown.

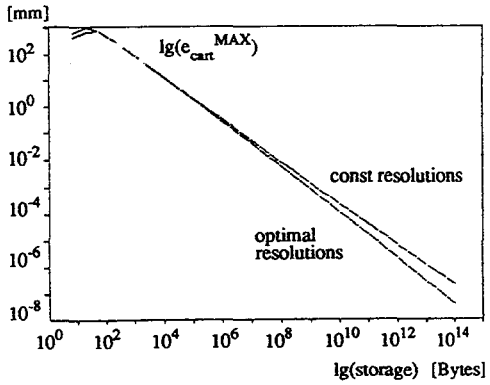


Fig. 5c The maximal Cartesian error at optimal information distribution

We can see, that there is also a difference between the optimized distribution of information and the non-optimized, constant one for the same number of neurons. In the case of the Cartesian error at 1.13 10^{14} Bytes storage, the error due to non-optimized resolutions is 5.6 times greater than the optimized one!

Nevertheless, if we regard the configuration with an Cartesian error of 0.201 mm, a value which is in the range of normal mechanical inaccuracy and therefore more important for practical applications, the necessary 1.9 MB of storage memory is contained in 39.6^3 neurons with the information of r = 16.4 Bits in each variable. The optimal configuration (r_w=17, r_θ=20.1, r_A=15 Bits) gives an error of 0.164 mm, only 18% less than the non-optimized one! Therefore, if the software problem of using floating point calculations with different numbers of bits is considered, it seems not advisable for practical simulations and for microprocessor control of robots to use different resolutions for the different storage variables w_c, Θ_c and A_c.

6. Discussion

This paper shows how a neighbourhood-conserving mapping can be used for the control of robot manipulators.

An optimal mapping in the sense of maximal information transmission results when the algorithm provides a magnification factor of the pattern space tesselation equal to the probability density of the pattern distribution.

Additionally, using a principle of optimal information distribution, the paper showed how a given, maximal positioning error implies the number of neurons, the information per neuron and the overall storage requirements.

This approach of implementing the inverse kinematics by a table of stored function values and learning them by executing the positioning task in contrast to the use of analytical functions reveals some interesting properties:

- ♣ The inverse control is *very fast* because it is based on a memory mapping and not on analytical calculations using transcendental functions.
- ♣ There are *no analytical solutions* necessary. This provides an easy control even of multi-jointmanipulators with worn-out joints or unconventional architectures such as non-orthogonal segment axes.
- ♣ This in turn enables the user of the robot to change the robot architecture and tailor it to its needs without the necessity for time-consuming queries to the origin factory. The user can *adapt the robot directly to his needs* without deeper knowledge of positioning analysis and control software.
- ♣ The learning algorithm provides coarse positioning resolution in rare-used regions and *fine resolution at often used locations* as pick-ups etc., including obstacle avoiding.
- ♣ *Movement restrictions* such as minimal energy, minimal deviation of an expected angle can easily be introduced by changing the learning equation (2.4) appropriately.

Nevertheless, the method of learning a mapping provides also some problems.

- ● One of it consists of the *time overhead* for the updating algorithm. It must be underlined that the algorithm presented in section 3 is essentially a sequential one since it uses a global decision (2.1) for searching the neuron with the minimal distance. Using the 40^3 neurons of the example gives us the necessity of searching 64000 neurons for the minimal distance which takes a long time in systems with a non-uniform positioning distribution. On the other hand, if we replace the sequential search by direct transformation of the input pattern to the neural weight position (neuron index) we

can not use the problem-specific resolution of the neuronal grid but have to use regular distances.

It should be noted that the algorithm can be parallelized as it was shown by Kohonen in [5]. This feature can be exploited by multiprocessor systems or, more effective, by neural chips which model each neuron by a separate hardware unit, thus representing a fast, adequate hardware base for the parallel algorithm.

- Another problem is the *high amount of storage* necessary for the function table. As it was shown in this paper, an optimized storage approach can overcome this problem and reduce the storage amount for reasonable positioning errors to the modest request of less than 2 MBytes.

Additionally, some problems of robot manipulator control should be mentioned which still rest to be solved:

- The neural positioning represents only an approach for the *low level primitives* which are used by higher layers such as trajectory generation which in turn is used by movement generation. The low level approach is completely isolated in respect to the higher level functions and is not applicable to them.

- The neural positioning is only learned for a *fixed workspace*. If the workspace changes by an affine transformation, i.e. a translation, a rotation or a scaling, the mapping is no longer valid and must be relearned: there are no movement primitives which are conserved and facilitate the relearning.
  The topology-conserving memory mapping can be regarded as a special case of an associative memory, with all its adjacent problems.

- *Time sequences* of positionings can not be used on other start positions as the original one in contrast to human beings who can repeat the same learned movement on different start positions: There is no "abstract", position independant coding of a movement.

In summary, the topology-conserving memory mapping can be regarded as an interesting, new approach for the problem of inverse kinematics which promises good results in practical applications. Nevertheless, there rest some important problems to be solved for a satisfactory theory of robot movement control.

References

[1] R.Brause, Performance and storage requirements of topology-conserving maps for robot manipulator control, Fachbereich Informatik report 5/89, University of Frankfurt, West Germany 1989

[2] K.S.Fu, R.C. Gonzales, C.S.G. Lee, Robotics, McGraw Hill, 1987

[3] T. Kohonen, Self-organized Formation of Topologically Correct Feature Maps, Biological Cybernetics, 1982, Vol 43, pp 59-69

[4] T. Kohonen, Clustering, Taxonomy and topological Maps of Patterns, IEEE Proc. 6th Int. Conf. Pattern Recognition, Oct. 1982, pp.114-128

[5] T. Kohonen, Self Organization and Associative Memory, Springer Verlag 1984

[6] R.Linsker, Self-Organization in a Perceptual Network, IEEE Computer, March 1988, pp.105-117

[7] R.Linsker, Towards an Organizing Principle for a layered Perceptual Network, in D. Anderson (ed), Neural Information Processing Systems, Amer. Inst. of Physics (NY), 1988

[8] H.Ritter, K.Schulten, On the Stationary State of Kohonen's Self-Organizing Sensory Mapping, Biological Cybernetics, 1986, Vol 54, pp.99-106

[9] H.Ritter, T.Martinetz, K.Schulten, Topology-Conserving Maps for Learning Visuo-Motor-Coordination, Neural Networks, Vol 2/3 , pp. 159-167, Pergamon Press 1989, New York

[10] J.S. Tou, Gonzales, Pattern Recognition Principles, Addison-Wesley 1974