Artificial Intelligence and Information-Control Systems of Robots -89
I. Plander (Editor)
© Elsevier Science Publishers B.V. (North-Holland), 1989

Performance of Topology-conserving Maps for the Learning of Robot Manipulator Control

Dr. R. Brause, University of Frankfurt, 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 mapping between two high-dimensional spaces by Kohonen [KOH1]. This paper describes the approach and presents the optimal mapping, based on the principle of maximal information gain. It is shown that Kohonens mapping in the 2-dimensional case is optimal in this sense. Furthermore, the principal control error made by the learned mapping is evaluated for the example of the commonly used PUMA robot, the trade-off between storage resources and positional error is discussed and an optimal position encoding resolution is proposed.

1. Introduction

This paper discusses how a topology-conserving mapping stored in memory can be used to learn the control of a robot manipulator. Contrary to the conventional, analytical solutions, learning the manipulator positioning has some major advantages:

a) Since the geometry of the manipulator arm is not explicitely represented but implicitely learned, the control can easily be adapted to tolerate manipulator fabrication variations worn-out joints without special reprogramming and support better positioning accuracy in often used regions of interest.

b) By this method it becomes possible now to control manipulators which have many joints (>3) with not simple (orthogonally or parallel) oriented rotation axes which is analytically very hard or impossible to treat.

c) The calculation for the necessary joint angles for a certain desired cartesian position is done very fast, even in the case when the manipulator joint are not orthogonally or parallel oriented.

2. The problem of inverse kinematics

In the standard control technique of robot manipulators the positioning commands are fed to the servo loop in real time, there is not enough time for the transformation of joint coordinates into cartesian coordinates for servo control purposes. For this reason the servo loop is often implemented in joint coordinates, leaving it to an compiler or interpreter of the list of positioning commands to do the conversion work in advance and to produce the list of joint coordinates. This approach hinders the developement of flexible, mobile robots.

For the transformation of the coordinates of an object in the manipulator end effector ("hand") to the base coordinate system (*kinematic transformation*) Denavit and Hartenberg [DEN] gave a simple mechanism for calculation:

Each segment of a robot manipulator (see e.g fig.1a the PUMA robot) has a transformation matrix which reflects its physical shape and the joint angles; the complete transformation matrix is calculated by successive multiplication of the partial transformation matrices.

Unfortunately, for the inverse transformation (*inverse kinematics*) there do not exist a simple formalism like

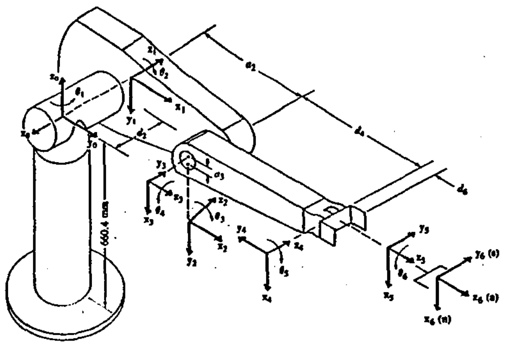

Fig1a The PUMA robot manipulator

this. Instead, we have to work out individual solutions for each manipulator. For arbitrary joint parameters, there exist no standard method to obtain a closed form solution (see [FU],pp.52). To make the closed form solution possible, designers of manipulators are motivated to orient the axes of the joints in parallel or to intersect them by 90 degree. Even then, the closed form solution is not simple and difficult to compute in real time.

As an interesting alternative to analytical calculations, let us regard the mapping of Cartesian coordinates to joint angles by simple look-up lists. This approach has some disadvantages:

- For a good resolution, there is much storage necessary.
- The processing speed is limited by the sequential searching and updating (*learning*) of the list.
- There must be a mechanism for an automatic update of these lists.

The method presented in the next section gives some partial solutions for these problems.

3. Robot control by topology conserving maps

Since the mapping of the inverse kinematics maps in a non-linear way neighbour points in the Cartesian space to neighbour points in the joint space, the required mapping can be termed *topology conserving mapping*.

One of the best known mathematical models which exhibit topographic properties is the one introduced by Kohonen 1982 [KOH1] or [KOH2] and analyzed for instance by Ritter and Schulten [RITT1]. Let us now briefly describe the algorithm.

Consider an sensor input space $X \subset \Re^3$ with the input events, characterized by data tupels $\mathbf{x} = (x_1, x_2, x_3)$ and an output space $\{\mathbf{y} = (y_1, y_2, y_3)\}$ with y_i a natural number. So the input space is projected on an output space of descrete points $\mathbf{y}$ (*neurons*), determined by 3 indices (grid space, see fig.5a). So to each $\mathbf{y}$ of the output space there corresponds a set $\{\mathbf{x}\}$ of points (*a class*) of the input space.

Let every point $\mathbf{y}$ (neuron) weight the input by one weight per input component, i.e. by a weight vector $\mathbf{w} = (w_1, w_2, w_3)$ from X.

Suppose, the input events $\mathbf{x}$ (t), t=1..n occur sequentially.

Then the mapping of the sensor space (perhaps deformed by sensor characteristics) to the Cartesian grid space is done by $\mathbf{x} \mapsto \mathbf{y}_{ijk}=(i,j,k)$ with the global neuron selection rule

$$|\mathbf{x}-\mathbf{w}_{ijk}| = \min_{i'j'k'} |\mathbf{x}-\mathbf{w}_{i'j'k'}| \qquad (3.1)$$

To each Cartesian position $\mathbf{y}_{ijk}=(i,j,k)$ there corresponds by a non-linear mapping a joint coordinate position Θ_{ijk} which also should be learned.

For a fast, parallel operation a decentralized version of the algorithm should be used which is also described in [KOH3].

Let $\mathbf{x}_R$ denote the real position, measured after the movement by some external or internal sensor in Cartesian coordinates; i.e. joint sensor coordinates are transformed prior by the kinematic transformation.

The **learning algorithm** for the inverse kinematics contains therefore two learning rules at time step t+1:

a) For the mapping (input space→Cartesian space) take

$$\mathbf{w}_{i'j'k'}(t+1) = \mathbf{w}_{i'j'k'}(t) - \gamma(t+1)\, h(t+1, i,j,k, i',j',k')\, [\mathbf{x}(t+1) - \mathbf{w}_{ijk}(t)] \qquad (3.2)$$

b) For the learning of the proper joint angles Θ_{ijk} corresponding to position (i,j,k) take

$$\Theta_{i'j'k'}(t+1) = \Theta_{i'j'k'}(t) - \gamma(t+1)\, h(t+1,i,j,k,i',j',k')\, [\mathbf{x}(t+1) - \mathbf{x}_R(t)] \qquad (3.3)$$

The learning rate $\gamma(t)$ has to fullfill the standard needs of stochastic approximation; 1/t is a good choice.

The neighbourhood function h(.) can be varied; for instance Ritter, Martinetz and Schulten [RITT3] assumed h(.) to be a Gaußian-shaped function instead of a step function. 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. [TOU]), 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.

Since we map a real-valued position $\mathbf{x}$ to an indexed position (i,j,k) with a certain Θ_{ijk}, we get a positional error (see section 5.1). To reduce this resolution error, we approximate the true position $\Theta_{true}(\mathbf{x})$ by the sum of the coarse resolution value Θ_{ijk} and the first term of a Taylor expansion, a linear approximation $\Delta\Theta = \mathbf{A}\ (\mathbf{x}-\mathbf{w}_{ijk})$ for the rest:

$$\Theta(\mathbf{x}) = \Theta_{ijk} + \Delta\Theta = \Theta_{ijk} + \mathbf{A}_{ijk}\,(\mathbf{x}-\mathbf{w}_{ijk}) \qquad (3.4)$$

Certainly, the matrix $\mathbf{A}_{ijk}$ is a good approximation only for a small section of the output space and is therefore different for different positions (i,j,k). Following Ritter, Martinetz and Schulten [RITT3], we first make a coarse positioning, get the sensored, real position $\mathbf{x}_I$, and then make the fine movement with (3.4) and measure finally the resulting position $\mathbf{x}_R$.

The learning of the set of parameters for coarse and fine movement replace the rule (3.3) by a learning rule for the general parameter vector $\mathbf{u}_{ijk}(n)$:

$$\mathbf{u}_{ijk}(t+1) = \mathbf{u}_{ijk}(t) - h(.)g(t+1)[\mathbf{u}_{ijk}^*(t+1)-\mathbf{u}_{ijk}(t)] \qquad (3.5)$$

with the neighbourhood-function h(.) of (3.4) and the (t+1)th estimation $\mathbf{u}_{ijk}^*$ of $\mathbf{u}_{ijk}$.

The general parameter vector contains 12 components: the 3 joint coordinates of Θ_{ijk} and the 9 matrix coefficients A_{ij}

$$\mathbf{u}_{ijk} = (\theta_1,\theta_2,\theta_3,A_{11}, \dots ,A_{33})^T_{ijk}$$

The new estimation of Θ_{ijk} is obtained by using the measured error $(\mathbf{x}-\mathbf{x}_F)$

$$\Theta_{ijk}^* = \Theta_{ijk} + \mathbf{A}_{ijk}(\mathbf{x}-\mathbf{x}_F) \qquad (3.6)$$

The new estimation of $\mathbf{A}_{ijk}$ uses both the measured positions $\mathbf{x}_I$ and $\mathbf{x}_F$:

$$\mathbf{A}_{ijk}^* = \mathbf{A}_{ijk} + \mathbf{A}_{ijk}(\,(\mathbf{x}-\mathbf{x}_F) - (\mathbf{w}_{ijk}-\mathbf{x}_I)\,)\,(\mathbf{x}_F-\mathbf{x}_I)^T / |(\mathbf{x}_F-\mathbf{x}_I)|^2 \qquad (3.7)$$

4. Optimal topology conserving mappings

Let us now consider the characteristics of an *optimal* mapping.

This leads us to the question : *optimal - in what sense?*

Let us consider a mapping as it was introduced in the previous section. 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. One plausible principle of a good mapping is to transmit as much information from the input to the output as possible (*maximal information principle*). This optimality criterion was recently proposed by Linsker [LIN1], 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

$$<f(w_i)> := \Sigma_{w_i} P(w_i)\, f(w_i)$$

$$<I_{trans}>_{w_i,x} = <I_{out}>_{w_i,x} - <I_{out/inp}>_{w_i,x}$$
$$= - \Sigma_i P(w_i) \ln[P(w_i)] - \Sigma_x P(x) \Sigma_i P(w_i/x) \ln[P(w_i/x)]$$

The average transmitted information $<I_{trans}>$ is maximized when

$$<I_{out}>_{w_i,x} \overset{!}{=} \max \qquad (4.1)$$

$$<I_{out/inp}>_{w_i,x} \overset{!}{=} \min \qquad (4.2)$$

It is easy to see by variation analysis (e.g. in [BRA], appendix A) that (4.1) is satisfied when

$$P(w_i) = P(w_j) = 1/N \quad \text{for all } i,j \qquad (4.3)$$

For the demand (4.2) we know that the values for $P(w_i/x)$ must be very unequal to yield a minimum. This is fulfilled for a tesselation of the output space without specifying *how* it was obtained, as for instance by a mapping like the one of equation (3.1). Specifically, every input pattern x is only assigned to one appropriate class y_i.

Then we have

$$\text{forall } x \text{ of } w_i \quad P(w_i/x) \ln[P(w_i/x)] = 1 \ln[1] = 0$$
$$\text{forall } x \text{ not of } w_i \quad P(w_i/x) \ln[P(w_i/x)] = \lim_{P\to 0} P \ln[P] =$$
$$= \lim_{P\to 0} \frac{(\ln[P])'}{(1/P)'} = \lim_{P\to 0} -P = 0$$

and therefore

$$<I_{out/inp}> = 0$$

This means, that for a maximal average information transmission it necessary and sufficient to have $P(w_i) = 1/N$. What does this mean for the density of the classes (also called *magnification factor*) in the input space ?

The class density is identical to the point density of the class prototypes w_i. In the optimal mapping every class has the same occurence probability 1/N and therefore the number of classes K in a certain area ΔA of the input space is

$$K := \frac{\text{probability mass of the whole area } \Delta A}{\text{average probability of one class}}$$
$$= \int_{\Delta A} p(x)\,dx \;\; / \;\; 1/N$$

With the number of classes per area $K/\Delta A$ the class density or *magnification factor* M(x) becomes

$$M(x) = \lim_{\Delta A\to 0} K/\Delta A = \lim_{\Delta A\to 0} N/\Delta A \int_{\Delta A} p(x)dx = N\, p(x) \qquad (4.4)$$

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 [LIN2], who stated that in optimal topology-conserving maps the often referenced classes should become bigger in the space, not smaller.

For robot control this demand 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.

Is this demand satisfied for the topology-conserving maps of Kohonen ?

As we know from equation (3.1) and the considerations before, equation (4.2) is satisfied.

Additionally, Kohonen found in [KOH2] for a one-dimensional array of class prototypes that their point density converge to the input distribution. Contrary to this, Ritter and Schulten found by calculating the n-dimensional case [RITT1] that this is not true, but that the magnification factor is proportional to $p(x)^{2/3}$, for the 2-dim (complex) case they also found $M(x) \sim p(x)$. Therefore, at least for the 2-dim case, Kohonens mapping fulfills equation (4.4) and so (4.3) and (4.1) and can be termed *optimal*.

5. Error analysis of the non-linear mapping

Since we map an infinite set of real-valued input events to a fixed number of discrete positions (i,j,k), we have a positional error. Certainly, the more positions (i,j,k) we have, the smaller the error will be; but there is always a principal error. Even with a linear approximation (see section 3) the resulting error will be smaller, but not zero. Therefore, two questions arise:

- *What is the principal error we make by using the topology-conserving mapping ?*
- *What is principal error we make by using the linear approximation ?*

For these considerations we focus our analysis on the stationary state, i.e. the mapping is learned (has converged) and do not change any more. Furthermore, let us assume that the input events are equal distributed in the Cartesian space, i.e. we do not have areas of special interest (cf. section 4).

In practical applications it is more important to know the maximal possible error than the average error. So we will focus our investigations on the maximal eror of the learned mapping.

5.1 The error of the topology-conserving mapping

Let us consider an Cartesian space , shown in figure 5a. It is devided into regular cubes of edge lengthes Δx_1, Δx_2 and Δx_3.

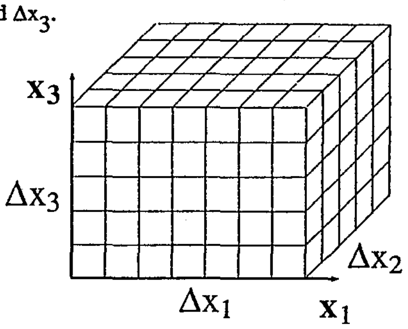

Fig. 5a The output space (i,j,k)

If the working space has the edge-lengths X_1,X_2 and X_3 then the space contains $N=n_1n_2n_3$ cubes (or neurons) with $n_i:=X_i/\Delta x_i$. The maximal deviation of the correct positioning occurs obviously on the boarder of the the classes.The *maximal Cartesian positioning error* in a regular grid is therefore

$$e_{cart}^{max} = |\Delta x/2| = 1/2(\Delta x_1^2+\Delta x_2^2+\Delta x_3^2)^{1/2} \qquad (5.1)$$

Example: For a cubic workspace with the edgelength of 70 cm and N=1000 neurons you have an error of 7 $x3^{1/2}$=12.12 cm which is much too high for normal robot operation.

5.2 The error of the linear approximation

The error of the coarse movement of section 5.1 may be corrected by an additional fine movement as it was introduced in equation (3.4). Let us now compare the error we make by this linear approximation of the exact analytical solution for the inverse kinematic problem.

Since the analytical solution is different for different types of robots, let us regard the commonly used PUMA manipulator type as shown in figure 2a. It has three joint angles θ_1,θ_2,θ_3 and the constant length a_2,a_3,d_2,d_4 for the arm movement and another triple joints θ_4,θ_5,θ_6 for the hand movement (which do not concern us for the moment).

Let us regard the error of the arm movement, i.e. positioning the hand, characterized by the vector **p** from the base to the intersection of the last three joint axes.

According to [FU], pp.63, the position $\mathbf{p} = (p_1,p_2,p_3)^T$ of the manipulator hand in (shifted) world coordinates has three inverse kinematic functions as analytical solutions:

$$\begin{aligned}\theta_1 &= \tan^{-1}(f_1(p_1,p_2,d_2)) &&:= t_1(\mathbf{p}) \quad |\theta_1| \le \pi\\ \theta_2 &= \tan^{-1}(f_2(p_1,p_2,p_3,\ a_2,a_3,d_2d_4)) &&:= t_2(\mathbf{p}) \quad |\theta_2| \le \pi\\ \theta_3 &= \tan^{-1}(f_3(p_1,p_2,p_3,\ a_2,a_3,d_2d_4)) &&:= t_3(\mathbf{p}) \quad |\theta_3| \le \pi\end{aligned} \qquad (5.1)$$

The linear approximations in (3.4) are determined by the matrix $\mathbf{A}_{ijk}$, the first derivate of the analytical solution in a Taylor expansion. In the stable, convergent state which we discuss we can approximate this by

$$A_{lm} = [t_l\,(w_{ijk}+dx_m/2) - t_l\,(w_{ijk}-dx_m/2)]\,/\,dx_m \qquad (5.2)$$

With the knowledge of equations (5.1) and (5.2) we can calculate the maximal position error in the joint space, and with the ordinary kinematic transformation (see [FU],p63) also the maximal Cartesian position error. This error e^{LA} is shown in figure 5b for a linear path through a cubic workspace with a length of 717mm. Parameter is the number n of neurons per grid dimension.

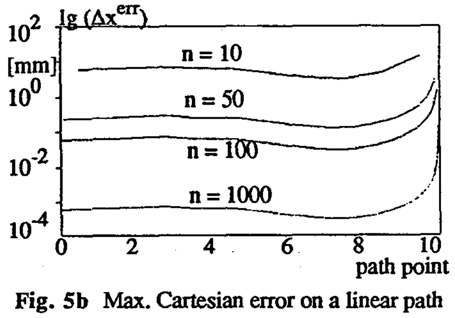

Fig. 5b Max. Cartesian error on a linear path

When we regard the figure 5b, 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 logarithm of n :

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

This gives the function

$$e^{LA}(n) = C\,n^b \qquad \text{with } C := 10^a \qquad (5.3)$$

For one point *P* of the linear path (the local maximum of the joint error in the first part of the path) for the joint error the constants in the simulation were a = 0.30390655 and b = - 2.656719975 .

6. Performance and storage size

The real time performance of the robot control algorithm by topology-conserving maps is quite good, because the non-linear mapping is done essentially by fast lookup-tables (which are learned), distributed over special parallel processing elements (the "neurons") and not by real-time calculations of one sequentially working microprocessor.

Nevertheless, even when the learning overhead is small (the learning of all parameters is done and the mapping is stable), we have to pay a price for the fast control: the price of a big storage size. The better the positioning resolution is, the more storage for the lookup-tables is required: Each "neuron" has 3 weights of w_{ijk}, 3 joint coordinates Θ_{ijk_3} and 9 matrix coefficients of A_{ijk}. With $N = n_1n_2n_3 = n^3$ neurons and the resolutions of r_w, r_θ, r_A bits we have a necessary storage of

$$s = n^3\,(3r_w+3r_\theta+9r_A) \quad \text{Bits} \qquad (6.1)$$

Example: Consider a resolution of 12 Bits for all storage variables. For n=10 we have only 30kByte storage, but for n=1000 we need 30 GigaByte !

Additionally, when we take into account that the same resolution do not has the same influence on the errors in the different variables, a good choice of the storage distribution over the different resolutions becomes crucial for the performance of the mapping algorithm.

In our neural network control system we have two kinds of errors due to resolutions

- the coordinate resolution error (maximal 1/2 digitalization increment) due to the digitalization process of the real values which represent a coordinate in the joint or Cartesian space
- the neural network resolution error due to the linear approximation of the joint angles

The overall maximal joint positioning error is determined by the superposition of the two independant sources of error: the approximation error and the resolution error:

$$e^{MAX} = |\,e^{LA} + e^{RES}| \qquad (6.2)$$

When we add some storage increment Δs the error will change by

$$\Delta e^{MAX}(s) = d/ds\, e^{MAX}(s) \quad \Delta s \qquad (6.3)$$

Let us assume that we take some storage amount from one kind of variable and put it to another one, i.e we change two resolution parameters without changing the storage requirements.

Let $h(s-\Delta s)$ and $g(s+\Delta s)$ be the errors of the two kind of variables after the change in the storage configuration. The error $e^{MAX}(s)$ will then change to $\underline{e}^{MAX}(s)$ by the first order approximation

$$e^{MAX}(g(s),h(s)) + \left[\frac{\partial e^{MAX}}{\partial g}(g)\frac{\partial g(s)}{\partial s} - \frac{\partial e^{MAX}}{\partial h}(h)\frac{\partial h(s)}{\partial s}\right] \Delta s$$

reduce storage in h

$$= e^{MAX}(g(s),h(s-\Delta s)) - \left[\frac{\partial e^{MAX}}{\partial g}(h)\ \frac{\partial g(s)}{\partial s}\right] \Delta s$$

add the storage to g

$$= e^{MAX}(g(s+\Delta s),h(s-\Delta s)) =: \underline{e}^{MAX}(s)$$

The error will therefore diminish when

$$\frac{\partial}{\partial g} e^{MAX}(g)\ \frac{\partial}{\partial s} g(s) \quad < \quad \frac{\partial}{\partial h} e^{MAX}(h)\ \frac{\partial}{\partial s} h(s)$$

If the two derivates are equal, no storage rearrangement can diminish the error any more. The storage configuration can therefore be termed *optimal*.

This idea can be applied to the multi-variable case. For the developement of (6.3) we have

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

For an optimal storage configuration all terms have equal values. This leads us to a system of three equations with the four variables n, r_w, r_θ and r_A. In [BRA] this is solved, getting three variables as a function of the forth one. By additionally using the storage equation (6.1) we finally can calculate the maximal joint positioning error $e^{MAX}(s_{opt})$ and, due to the known kinematic transformation, the maximal Cartesian positioning error $e_{Cart}^{MAX}(s_{opt})$ as a function of the optimal storage requirement s_{opt}. This is plotted in figure 6.2a.

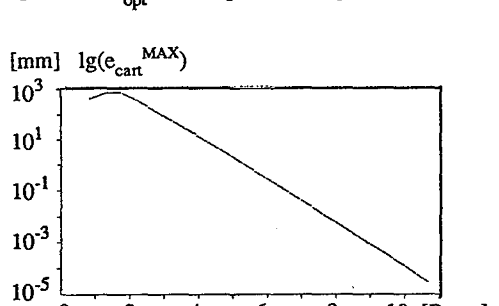

Fig.6.2a The maximal Cartesian positioning error at optimal storage distribution

7. Conclusion

Robot control by topology conserving mappings represents an interesting approach for a non-analytical solution to the problem of inverse kinematic of robot manipulators.

It was shown that an optimal learning algorithm preserves the probability distribution of the input space. Due to the nature of this learning process, locations which are more frequented will have a higher control resolution. Thus, the learning algorithm adapts the mapping to the focus of interest.

Additionally, in contrast to the analytical solution, the learning enables the adaption to worn-out or new manipulator geometrics which are difficult or impossible to get analytically. By the use of a appropriate learning rule of stochastic approximation also constraints for the movement (minimal energy etc) can be introduced.

The error analysis showed that by using an optimal storage distribution strategy the algorithm has only modest storage requirements.

Thus, as soon as fast multiprocessor systems or special neural chips are available, a decentralized version of the algorithm represents a very powerful, fast and flexible base of robot control.

Adress of the author:
Dr. R.Brause
J.W. Goethe University, FB20, VSFT,
Postfach 111932
D- 6000 Frankfurt 11, West Germany

References

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

[DEN] J. Denavit, R.S. Hartenberg, A Kinematic Notation for Lower-Pair Mechanismen Based on Matrices, Journal Appl. Mech., 1955, Vol 77, pp.215-221

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

[KOH1] T. Kohonen, Self-organized Formation of Topologically Correct Feature Maps, Biological Cybernetics, 1982, Vol 43, pp 59-69

[KOH2] T. Kohonen, Clustering, Taxonomy and topological Maps of Patterns, IEEE Proc. 6th Int. Conf. Pattern Recognition, Oct. 1982, pp.114-128

[KOH3] T. Kohonen, Self Organization and Associative Memory, Springer Verlag 1984

[LIN1] R.Linsker, Self-Organization in a Perceptual Network, IEEE Computer, March 1988, pp.105-117

[LIN2] 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

[RITT1] H.Ritter, K.Schulten, On the Stationary State of Kohonen's Self-Organizing Sensory Mapping, Biological Cybernetics, 1986, Vol 54, pp.99-106

[RITT2] H.Ritter, K.Schulten, Convergence Proporties of Kohonen's Topology Conserving Maps, Biological Cybernetics (in press)

[RITT3] H.Ritter, T. Martinetz, K. Schulten, Topology-Conserving Maps for Learning Visuomotor-Coordination, private communication

[TOU] Tou, Gonzales, Pattern Recognition Principles, Addison-Wesley 1974