'95 KACC (1995. 10. 23 ~ 25)

A Solution of Inverse Kinematics for Manipulator by Self Organizing Neural Networks

※ Fumiaki TAKEMORI†, Yasuhisa TATSUCHI‡,
Yoshifumi OKUYAMA† *and* Ahmet KANBOLAT†

† Faculty of Engineering, Tottori University
4-101, Koyama-cho Minami, Tottori 680, Japan
‡ Daiichi Kogyo Co.,Ltd., 3-6-4, Nishitenman, Kita-ku, Osaka 530, Japan

Abstract: This paper describes trajectory generation of a robot arm by self-organizing neural networks. These neural networks are based on competitive learning without a teacher and this algorithm which is suitable for problems in which solutions as teaching signal cannot be defined — e.g. inverse dynamics analysis — is adopted to the trajectory generation problem of a robot arm. Utility of unsupervised learning algorithm is confirmed by applying the approximated solution of each joint calculated through learning to an actual robot arm in giving the experiment of tracking for reference trajectory.

Keywords : Self-organizing neural networks, Inverse kinematics, PTP trajectory

1 Introduction

It is convenient to use the well-known *inverse kinematics* analysis to give the trajectory generation of the manipulator. In this method, each joint angle is calculated geometrically by givining only position and its direction of the top of the arm. However, a solution of inverse dynamics problem is intuitive as geometric solution, so inverse dynamics analysis does not have a specific solution. In addition, it has several points that inverse dynamics problem for all mechanisms cannot be solved necessarily, as an example a solution that is impossible to realize mechanically has been calculated.

In this paper, we pay attention to self-organizing neural networks based on competitive learning without a teacher and adapt this algorithm to the trajectory generation problem of a manipulator. It is expected that this method can avoid an ocasionally unrealizable solution of inverse dynamic analysis and this learning method also reduces a large number of calculations peculiar to *unsupervised learning*. We propose an algorithm deriving a solution of inverse dynamics problem for the manipulator trajectory generation. Finally, an utility of unsupervised learning algorithm is confirmed by applying the approximated solution of each joint calculated through learning to actual manipulator in giving the experiment of tracking for the reference trajectory.

2 Self-Organizing Neural Networks

A typical one of the unsupervised learning is Kohonenn self-organizing feature maps.[2-5] The neuron model proposed by Kohonenn is shown in Fig.1 and its behavior is described by the following equations.

$$I_i = \sum_{j=1}^{n} X_j W_{ij},\ O_i = \frac{1}{1+e^{-I_i}} \tag{1}$$

X_j : input
W_{ij} : synaptic weight
O_i : output

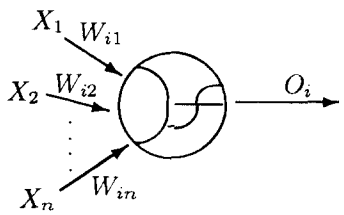

Fig.1 Neuron model

In this model, unlike usual neural network models, the learning is defined as follows so that a neuron itself selectively behaves as a kind of signal.

$$\frac{dW_{ij}}{dt} = \alpha O_i X_i + \beta(O_i) W_{ij},\ \alpha > 0 \tag{2}$$

First term on right-hand side of (2) follows the Hebb's law, while second term exhibits nonlinear damping effect for the output activity O_i and it is possible to stabilize O_i so as to converge to a certain range by setting the constant term in Taylor expansion of function $\beta(O_i)$ to be 0. Thus, to give the second term can make a monotonous change of synaptic weight in equation with first term only changing.

Fig.2 shows a nervous circuit of basic competitive system model.[2] This nervous circuit is composed of n-excitatory synapse E_i and one inhibitory synapse I. The excitatory synapse and the inhibitory synapse recieve exogenous inputs X_i and X_0 respectively. X_0 is ordinarily 0. The main difference between this network and the other is the synaptic connection between input signal and feedback signal of the output.

In this paper, we add the following three conditions to make competitive learning easy.[2]
(1) When all input signal X_i is lower than a constant value $X_{\min}$, no neurons get stimulated.
(2) When input signal is higher than $X_{\min}$, in spite of their number, just one only neuron gets stimulated.
(3) If once a neuron gets stimulated, it keeps this situation not so far as the system is resetted.

2.1 Learning Algorithm

The learning process mentioned above can be given in the following calculation steps.

STEP1 : Give the normalized input pattern to X_i.

STEP2 : We set randomly the initial value of weight W_{ij} from input synapse X_i to neuron j.

STEP3 : By the distance between input pattern X_i and vector of synaptic weight W_{ij}, find neuron j located at the center of bubble so that D_j becomes minimum.

$$D_j = \sum_{i=1}^{n}(X_i - W_{ij})^2 \qquad (3)$$

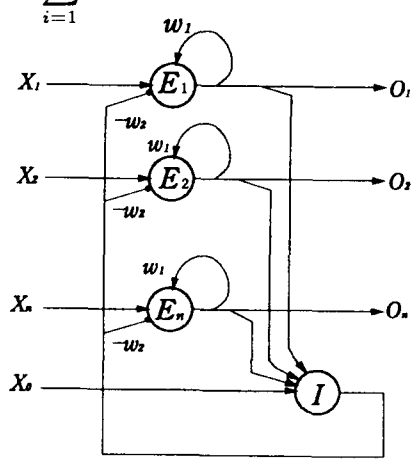

Fig.2 Competitive-system model

STEP4 : Collectively adjust neurons neighbouring the neuron j selected in **STEP3** by

$$W_{ij}(t+1) = W_{ij}(t) + \alpha(t)(X_i(t) - W_{ij}(t)), \qquad (4)$$

$$j \in N_j(t). \qquad (5)$$

Both $\alpha(t)$ and $N_j(t)$ are experimentally given to decay in time, $\alpha(t)$ takes the value of 0~1 and $N_j(t)$ is a class of distance functions, which considers that the region of neurons located on a two-dimensional plane becomes narrow with passing of time.

3 Application to Robot Arms

Robot arms used in the experiment are RRRR-structures of which waist, shoulder, elbow, and wrist joint have rotative joints as respectively shown in Fig.3. The top of the wrist joint has hold-type "end-effector". DC motor as actuator is installed in each joint and PD compensator is used to control it.

In this paper, trajectory generation of each joint is given by a neural network so that the end-effector can draw linear trajectory referred to as "PTP trajectory". In PTP trajectory, it is assumed that the end-effector moves in a linear motion with constant velocity and if once position and direction of the end-effector at start point and at final point are given, it is easy to compute principal axis vector direction (this means $\theta_1+\theta_2+\theta_3$ in Fig.3) at an arbitrary point on linear trajectory. Thus, as shown in Fig.6, giving the state of end-effector as input signal to the neural network model, each joint angle is computed by competitive learning.

3.1 Simulation and Experiment

In order to normalize input signals applied to neural network, Coordinates of actual robot arms are normalized as follows

$$L_0 = 0.35,\ L_1 = 0.21,\ L_2 = 0.16,\ L_3 = 0.25,$$

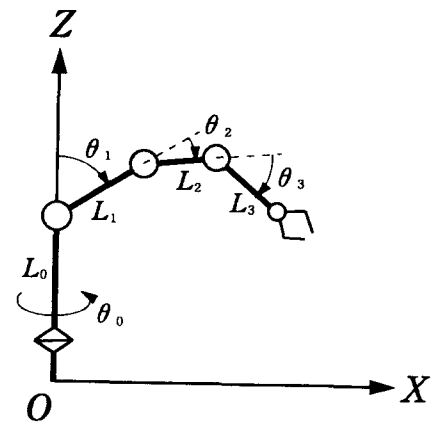

Fig.3 Configration of arm

while the normalized output is modified to angle value by

$$\theta_i = 360(O_i - 0.5). \tag{6}$$

Now, we try to make a plan of PTP trajectory as follows;

start point: $P_s(x, z)$=(0.0, 1.0)
final point: $P_f(x, z)$=(0.485, 0.321)
final principal axis vector: ϕ_f=135[deg]

For simply, here, it is assumed that trajectory generation is carried out on a two-dimensional plane i.e., θ_0 does not rotate. Number of neuron is 10×10 and input signals multiplied by joint number (×3) are prepared. Trajectory computing algorithm is as follows.

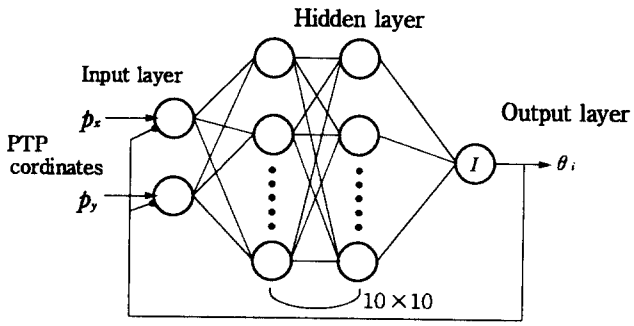

Fig.4 Competitive learning

STEP1 : Give reference coordinates $P^a(x^a_{pi}, z^a_{pi})$ as input pattern X^a_i, i.e.

$$X^a_i := \begin{bmatrix} x^a_{pi} \\ z^a_{pi} \end{bmatrix} \tag{7}$$

$$a = \begin{cases} 0 : & \text{PTP coordinates} \\ 1 : & \text{wrist-joint coordinates} \end{cases}$$

STEP2 : Assume that synaptic weight W^a_{ij} is the coordinates of the top of the robot arm, i.e.

$$W^a_{ij} := \begin{bmatrix} x^a_{ij} \\ z^a_{ij} \end{bmatrix} = \begin{bmatrix} \sum_{l=1}^{3-a} \left(L_l \sin \left(\sum_{k=1}^{l} \theta_k \right) \right) \\ \sum_{l=0}^{3-a} \left(L_l \cos \left(\sum_{k=1}^{l} \theta_k \right) \right) \end{bmatrix} \tag{8}$$

and give random value of θ_k as input synapse.

STEP3 : From Eq.(6), find the neuron j so that the square of distance between reference point and the top of the robot arm, i.e.

$$D_j = \sum_{i=1}^{n} \left(\begin{bmatrix} x^a_{pi} \\ z^a_{pi} \end{bmatrix} - \begin{bmatrix} x^a_{ij} \\ z^a_{ij} \end{bmatrix} \right)^2 \tag{9}$$

becomes minimum.[7]

STEP4 : For $a = 0$; adjust $\theta_{3i}(t)$ by the following equation in order to conform principal axis vector $\phi(t)$ to true one $\phi_i(t)$.

$$\theta_{3i}(t+1) = \theta_{3i}(t) + \alpha(t)(\phi_i(t) - \phi(t)) \tag{10}$$

$\alpha(t)$ and $N_j(t)$ then take the values shown in Table 1.

For $a = 1$; Complete computation.

STEP5 : Repeat **STEP1** $\sim$ **4** for a=0 and 1.

Table 1

Learning iteration	$N_j(t)$	$\alpha(t)$
$0 \sim 1.0 \times 10^2$	20	0.35
$1.0 \times 10^2 \sim 5.0 \times 10^2$	16	0.25
$5.0 \times 10^2 \sim 1.0 \times 10^3$	11	0.20
$1.0 \times 10^3 \sim 2.0 \times 10^3$	7	0.15
$2.0 \times 10^3 \sim 5.0 \times 10^3$	3	0.1
$5.0 \times 10^3 \sim 1.0 \times 10^4$	0	0.1
$1.0 \times 10^4 \sim$	0	0.05

3.2 Learning Results

Fig.6 shows the learning process of PTP trajectory generation using a workstation. Each trajectory data applied to actual robot arms must be rearranged monotonously because in simulation they are outputted randomly. These curves are shown as circle-points in Fig.6.

In the experiment of tracking for reference trajectory, we construct PD control system, and if we apply circle-points curves as reference trajectories to robot arm, then, as a result, solid curves in Fig.7 can be obtained.

Output data learned by unsuitable parameters shows that there are obviously dense parts and sparse parts in the drawn trajectory as shown in Fig.6, which leads to rapid changes in the angle data and causes the blurring of robot arm in actual experiment and interferes with ideal trajectory formation. This results from that output data tends to deflect to first output position when neuron value increases from one direction only. But making some increase directions of neuron uniform is able to avoid this tendency. Thus, to tune parameters suitably is important, however, it is not easy to find its regularity. Therefore, in tuning the parameters, we must still depend on trial-and-error.

4 Conclusion

In this paper, it was confirmed that PTP trajectory of robot arm could be computed by neural networks using self-organizing feature maps. This method can avoid

an occasional unrealizable solution of inverse dynamics analysis. In this learning method, however, although it is possible to reduce the number of calculations peculiar to "supervised learning", there are some "unsupervised learning" points for which parameter values making the synaptic weight decay have to be selected by trial-and-error.

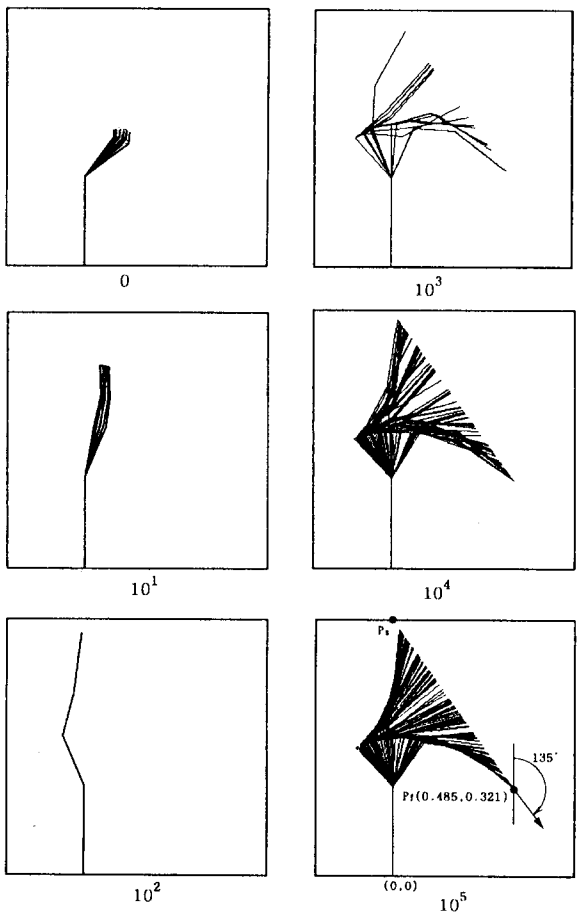

Fig.5 Learning process

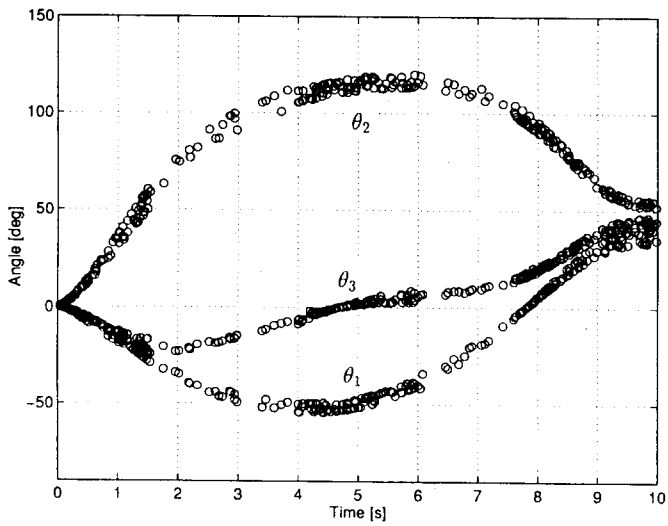

Fig.6 Calculated trajectories

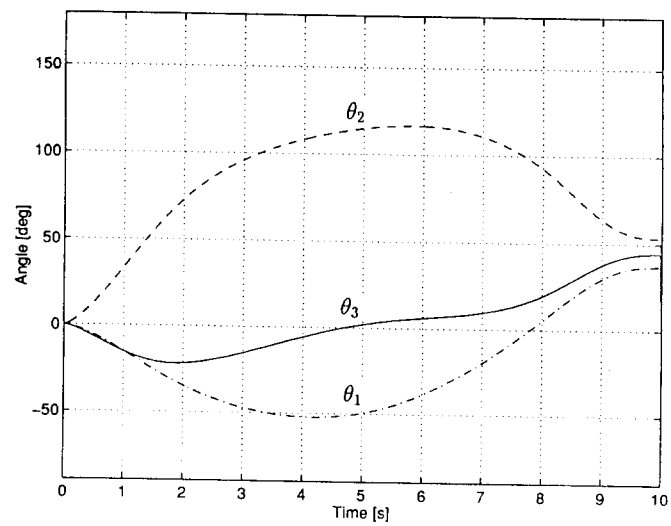

Fig.7 Experimental result

References

[1] Y.Tatsuchi : A Study on Trajectory Generation of Vertical Multi-Joint Type Robot Arms by Neural Network (In Japanese), Thesis for M.Sc. Engineering Division, Tottori University (1994)

[2] K.Nakano : An Introduction to Neurocomputing (In Japanese), Corona Publishing Co. (1991)

[3] R.P.Lippman : An Introduction to Computing with Neural Nets, IEEE ASSP Magazine, pp.4/22 (April,1987)

[4] S.Murai et. al : Self-Organizing Learning of Neural Networks by Using a Temporal Correlation (In Japanese), The 38th SCI, pp.129/130 (1994)

[5] Y.Anzai : Recognition And Learning (In Japanese), Iwanami Publishing Co. (1991)

[6] J.J.Craig : Robotics, pp.93/95,209, Kyohritsu Publishing Co.(1991)

[7] S.Oda et. al : Path Planning of the Manipulator Using Hopfield Network, (In Japanese), The 38th SCI, pp.401/402 (1994)