Proceedings of the 2000 IEEE
International Conference on Robotics & Automation
San Francisco, CA • April 2000

Modular Neural Net System for Inverse Kinematics Learning

Eimei OYAMA

Robotics Department
Mechanical Engineering Laboratory
Namiki 1-2, Tsukuba Science City
Ibaraki 305-8564 Japan
eimei@mel.go.jp

Susumu TACHI

Faculty of Engineering
The University of Tokyo
Hongo 7-3-1, Bunkyo-ku
Tokyo 113-8656 Japan

Abstract

Inverse kinematics computation using an artificial neural network that learns the inverse kinematics of a robot arm has been employed by many researchers. However, conventional learning methodologies do not pay enough attention to the discontinuity of the inverse kinematics system of typical robot arms with joint limits. The inverse kinematics system of the robot arms is a multi-valued and discontinuous function. Since it is difficult for a well-known multi-layer neural network to approximate such a function, a correct inverse kinematics model for the end-effector's overall position and orientation cannot be obtained by using a single neural network. In order to overcome the discontinuity of the inverse kinematics function, we propose a novel modular neural network system for the inverse kinematics model learning. We also propose the on-line learning and control method for trajectory tracking.

1 Introduction

Often, people who work efficiently in complex and unstructured environments acquire their skills by various learning. Implementation of similar function in robots is necessary in order to create future robots that will work in unstructured environments [1].

The task of calculating all of the joint angles that would result in a specific position/orientation of an end-effector of a robot arm is called the inverse kinematics problem. An inverse kinematics solver using an artificial neural network that learns the inverse kinematics system of a robot arm has been used in many researches; however, many researchers do not pay enough attention to the discontinuity of the inverse kinematics function of typical robot arms with joint limits. The inverse kinematics function of the robot arms, including a human arm with a wrist joint, is a multi-valued function and a discontinuous function. It is difficult for a well-known multi-layer neural network to approximate such a function. A correct inverse kinematics solution for the end-effector's overall position and orientation cannot be obtained by the inverse kinematics model consisting of a single neural network. Therefore a novel modular neural network architecture for the inverse kinematics model learning is necessary.

Jacobs et al. [2] proposed a modular neural network architecture. Gomi and Kawato applied the modular architecture neural networks to the object recognition for manipulating a variety of objects and to inverse dynamics learning [3]. Kawato et al. proposed Multiple Pairs of Forward and Inverse Models as a computational model of the cerebellum [4]. However, the input-output relation of their networks is continuous and the learning method of them is not sufficient for the non-linearity of the kinematics system of the robot arm. Their architecture is not suitable for the inverse kinematics model learning.

The inverse kinematics function decomposes into a finite number of solution branches and each solution branch can be described as the product of an one-to-one inverse map and a set of free parameters describing the redundancy [5]. DeMers et al. proposed the inverse kinematics learning method that a neural network learns each solution branch calculated by the global searches in the joint space [5]. However, the method is a purely off-line learning method and is not applicable for on-line learning, i.e. simultaneous or alternate execution of the robot control and the inverse model learning. Furthermore, the method is not goal-directed.

In this paper, we propose a novel modular neural network architecture for inverse kinematics learning and the on-line incremental learning method for the architecture. We also propose the on-line learning and control method for trajectory tracking. In order to evaluate the proposed architecture, numerical experiments of the inverse kinematics model learning were performed.

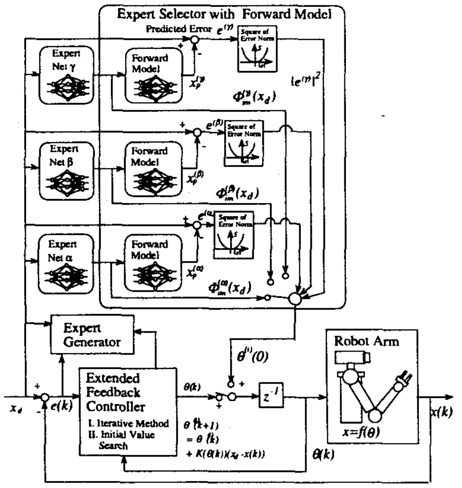


Figure 1 Inverse Kinematics Solver with Modular Architecture Networks

2 Modular Neural Net System

Let $\boldsymbol{\theta}$ be the $m \times 1$ joint angle vector and $\boldsymbol{x}$ be the $n \times 1$ position/orientation vector of a robot arm. The relationship between $\boldsymbol{\theta}$ and $\boldsymbol{x}$ is described by $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{\theta})$. $\boldsymbol{f}$ is a C^1 class function. Let $\boldsymbol{J}(\boldsymbol{\theta})$ be the Jacobian of the robot arm, defined as $\boldsymbol{J}(\boldsymbol{\theta}) = \partial \boldsymbol{f}(\boldsymbol{\theta})/\partial \boldsymbol{\theta}$. When a desired hand position/orientation vector $\boldsymbol{x}_d$ is given, an inverse kinematics problem that calculates the joint angle vector $\boldsymbol{\theta}_d$ satisfying the equation $\boldsymbol{x}_d = \boldsymbol{f}(\boldsymbol{\theta}_d)$ is considered.

In this paper, a function $\boldsymbol{g}(\boldsymbol{x})$ that satisfies $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{g}(\boldsymbol{x}))$ is called an inverse kinematics system of $\boldsymbol{f}(\boldsymbol{\theta})$. The acquired model of the inverse kinematics system $\boldsymbol{g}(\boldsymbol{x})$ in the robot controller is called an inverse kinematics model. Let $\boldsymbol{\Phi}_{im}(\boldsymbol{x})$ be the output of the inverse kinematics model.

Although the inverse kinematics function of the robot arms is a multi-valued and discontinuous function, the inverse kinematics function can be constructed by the appropriate synthesis of continuous functions [5]. We propose a novel modular neural network system that can learn a precise inverse kinematics model by the appropriate switching of multiple neural networks [6].

2.1 Configuration of Proposed Inverse Kinematics Solver

Figure 1 shows the configuration of the inverse kinematics solver with the modular architecture networks for inverse kinematics model learning. Each expert network in Figure 1 approximates the continuous region of the inverse kinematics function. The expert selector selects one appropriate expert according to the desired position/orientation of the end-effector of the arm, as described in Section 2.2. The extended feedback controller calculates the inverse kinematics solution based on the output of the selected expert by using the output error feedback. When no precise solution is obtained, the controller performs a kind of global search, as shown in Section 2.3. The expert generator generates a new expert network based on the inverse kinematics solution.

2.2 Configuration of the Expert and Selection by the Forward Model

In order to cover the overall work space, each expert has its representative posture. The representative posture is the inverse kinematics solution obtained in the global searches by the extended feedback controller when the expert is generated. Let $\boldsymbol{\theta}_r^{(i)}$ be the representative posture of the *i-th* expert and $\boldsymbol{x}_r^{(i)}$ be the end-effector position/orientation that corresponds to $\boldsymbol{\theta}_r^{(i)}$. Let $\boldsymbol{\Phi}_{im}^{(i)}(\boldsymbol{x})$ be the output of *i-th* expert when the input of the expert is $\boldsymbol{x}$. Each expert is trained to satisfy the following equation:

$$\boldsymbol{x}_r^{(i)} = \boldsymbol{f}(\boldsymbol{\Phi}_{im}^{(i)}(\boldsymbol{x}_r^{(i)}))). \tag{1}$$

By changing the bias parameters of the output layer of the neural network, the above equation can easily be satisfied. Each expert approximates the continuous region of the inverse kinematics function in which the reaching motion can move the end-effector smoothly from its representative posture.

The predicted position/orientation error is used as the performance index of the expert. When the desired end-effector position $\boldsymbol{x}_d$ is given, the *i-th* expert calculates the output $\boldsymbol{\Phi}_{im}^{(i)}(\boldsymbol{x}_d)$. The expert selector selects an expert with the smallest predicted error among all experts. Let $\boldsymbol{\Phi}_{fm}(\boldsymbol{\theta})$ be the output of the forward kinematics model and $\boldsymbol{\Phi}_{im}^{(i)}(\boldsymbol{x}_d)$ be the output of the *i-th* expert. The predicted error of the *i-th* expert p_i is calculated as $p_i = |\boldsymbol{x}_d - \boldsymbol{\Phi}_{fm}(\boldsymbol{\Phi}_{im}^{(i)}(\boldsymbol{x}_d))|$.

Let $\boldsymbol{\Phi}'_{fm}(\boldsymbol{\theta})$ be the desired output for $\boldsymbol{\Phi}_{fm}(\boldsymbol{\theta})$. The learning of the forward kinematics model is conducted as $\boldsymbol{\Phi}'_{fm}(\boldsymbol{\theta}) = \boldsymbol{x} = \boldsymbol{f}(\boldsymbol{\theta})$.

2.3 Extended Feedback Controller

The conventional on-line inverse model learning methods, such as Forward and Inverse Modeling proposed by Jordan [7] and Feedback Error Learning proposed by Kawato, are based on the local information of the forward system near the output of the inverse model. The desired output signal provided by these methods is not always in the direction that finally reaches the correct solution of the inverse problem. An extended feedback controller avoids that drawback by employing a global search technique based on the multiple starts of the iterative procedure [8][9]. When the iterative procedure from the output of the selected expert cannot reach a correct solution, the extended feedback controller repeats the iterative procedure from a number of initial values until a correct solution is obtained.

The proposed inverse kinematics solver solves an inverse kinematics problem according to the following procedural steps:

(1) When the desired end-effector position $\boldsymbol{x}_d$ is given, the expert selector selects the expert with the minimum predicted error among all the experts. The extended feedback controller moves the arm to the posture that corresponds to the output of the expert and then improves the end-effector position/orientation by using the output error feedback, as described in Section 2.4.

(2) When no precise inverse kinematics solution is obtained in step (1), the other expert which predicted error is lower than an appropriate threshold r_{eim} is selected in increasing order of the predicted error and the iterative improvement procedure by the output error feedback is conducted.

(3) When no solution is obtained in steps (1) and (2), an expert is randomly selected and a reaching motion from the representative posture of the selected expert is conducted. The above procedure is repeated until the reaching motion is successfully conducted or all the experts are tested.

(4) If a precise solution is obtained in the above procedural steps, the solution is used as the desired output signal for the expert, as shown in 2.4.

(5) When no solution is obtained in the above procedural steps, the controller starts a type of global search. The controller repeats the initial joint angle vector generation by using a uniform random number generator and the reaching motion from the generated posture, as described in 2.4, until a precise solution is obtained. When a precise solution is obtained, a new expert is generated and the solution is used as the representative posture $\boldsymbol{\theta}_r$ of the expert.

2.4 Reaching Motion

An illustration of the reaching motion, which is a kind of iterative improvement procedure, follows.

Let $\boldsymbol{\theta}(0)$ be the initial posture of the iterative procedure, which is the output of the selected expert $\boldsymbol{\Phi}^{(i)}(\boldsymbol{x}_d)$; the representative posture of the selected expert $\boldsymbol{\theta}_r^{(i)}$; or the randomly generated posture.

Let $\boldsymbol{x}_s$ be the initial end-effector position which is defined as $\boldsymbol{x}_s = \boldsymbol{f}(\boldsymbol{\theta}(0))$. The extended feedback controller conducts a reaching motion from $\boldsymbol{x}_s$ to $\boldsymbol{x}_d$ by using Resolved Motion Rate Control (RMRC) [10]. The reaching motion is conducted as the tracking control to the following desired trajectory of the end-effector $\boldsymbol{x}_d(k)(k = 0, 1, \dots, T+1)$ described as follows. Let T be an integer that satisfies the following inequality:

$$T - 1 \leq \frac{|\boldsymbol{x}_d - \boldsymbol{x}_s|}{r_{st}} < T \qquad (2)$$

The desired trajectory $\boldsymbol{x}_d(k)$ is a straight line from $\boldsymbol{x}_s = \boldsymbol{f}(\boldsymbol{\theta}(0))$ to $\boldsymbol{x}_d$ which is calculated as follows:

$$\boldsymbol{x}_d(k) = \begin{cases} (1 - \frac{k}{T})\boldsymbol{x}_s + \frac{k}{T}\boldsymbol{x}_d & (0 \leq k < T) \\ \boldsymbol{x}_d & (k \geq T) \end{cases} \qquad (3)$$

When the orientation is represented by the Direction Cosine Matrix or the Quaternion, the components of $\boldsymbol{x}_d(k)$, which represents the orientation, must be normalized.

We assume that a precise end-effector position feedback controller is already obtained by learning [11]. Otherwise, we assume that the controller can accurately estimate the coordinate transformation by the observation of the robot arm movement and the numerical differentiation technique [9].

Let $\boldsymbol{J}^{+}(\boldsymbol{\theta})$ be the pseudo-inverse matrix (Moore-Penrose generalized inverse matrix) of $\boldsymbol{J}(\boldsymbol{\theta})$ which is calculated as $\boldsymbol{J}^{+}(\boldsymbol{\theta}) = \boldsymbol{J}^T(\boldsymbol{\theta})(\boldsymbol{J}(\boldsymbol{\theta})\boldsymbol{J}^T(\boldsymbol{\theta}))^{-1}$. $\boldsymbol{J}^{+}(\boldsymbol{\theta})$ is used as the coordinate transformation gain of the output error feedback. Let $\boldsymbol{\theta}(k)$ be an approximate inverse kinematics solution at step k. When r_{st} is small enough, $\boldsymbol{\theta}(k)$ can be calculated as follows:

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) + \boldsymbol{J}^{+}(\boldsymbol{\theta}(k))(\boldsymbol{x}_d(k+1) - \boldsymbol{f}(\boldsymbol{\theta}(k))). \quad (4)$$

If a precise solution $\boldsymbol{\theta}(k)$, which end-effector position error norm $|\boldsymbol{x}_d(k) - \boldsymbol{f}(\boldsymbol{\theta}(k))|$ is lower than an appropriate threshold r_e, is obtained, the solution can be used for the desired output for the inverse kinematics model as follows:

$$\Phi'_{im}(x_d(k)) = \theta(k). \quad (5)$$

When the controller cannot find a precise solution because of the singularity of Jacobian or the joint limits, the reaching motion is regarded as a failure.

2.5 Discontinuity Check

The initial status of the experts sometimes causes a situation where one expert approximates multiple solution branches of the inverse kinematics function. In the case that one region contacts the other region in the workspace coordinates, the result is that the expert tries to approximate a discontinuous function. This situation should be avoided. When the matrix norm of $\partial\Phi^{(i)}_{im}(x)/\partial x|_{x=x_d}$ is larger than an appropriate threshold r_{jix}, a new expert with a new representative posture θ_r corresponding to x_d is generated to approximate the region near to x_d. $\partial\Phi^{(i)}_{im}(x)/\partial x|_{x=x_d}$ is approximately calculated by using numerical differentiation and checked when the *i-th* expert is selected according to step (1) in Section 2.3.

3 Tracking Control

In this section, the tracking control to desired end-effector trajectories is considered. When the desired end-effector position/orientation trajectory $x_d(k)(k = 0, 1, 2, \ldots, T_d)$ is given, the procedure of the tracking control by the proposed solver is as follows:

(1) Let $x_d(k_r)$ be the representative points of the desired trajectory $x_d(k)$. In this paper, $k_r = 0, T_d$ is used. First, the solver tries to find an expert the output of which can reach $x_d(0)$ precisely as the procedure described in Section 2.4 and the predicted error of which on the representative points $x_d(k_r)$ is the smallest among all experts.

(2) We assume that the *i-th* expert which can reach $x_d(0)$ is selected and the precise inverse kinematics solution $\theta(0)$ is obtained. The tracking control to $x_d(k)$ is conducted according to the following control low.

$$\Delta\theta_{fb}(0) = \theta(0) - \Phi^{(i)}_{im}(x_d(0)) \quad (6)$$

$$\theta(k+1) = \Phi^{(i)}_{im}(x_d(k+1)) + \Delta\theta_{fb}(k+1) \quad (7)$$

$$\Delta\theta_{fb}(k+1) = \alpha\Delta\theta_{fb}(k) + J^{+}(\theta(k))e(k) \quad (8)$$

$$e(k) = x_d(k) - f(\theta(k)) \quad (9)$$
$$0 < \alpha \leq 1.$$

In this paper, α is set at 0.9.

(3) When the solver cannot calculate $\theta(k)$ that can realize $x_d(k)$ due to the singularity of $J(\theta(k))$ or the joint limits, the solver give up the tracking control.

During the tracking control, the learning of the selected expert is conducted as $\Phi'^{(i)}_{im}(x_d(k)) = \theta(k)$ when $\theta(k)$ is precise enough.

Since the proposed method in this paper does not use the redundancy efficiently, there is no guarantee that the learning controller can trace all the trajectories that the arm can trace by using appropriate trajectory planing techniques.

4 Simulations

4.1 Inverse Kinematics Learning of a 2-DOF Arm

We performed simulations of the inverse kinematics model learning for a 2-DOF arm moving in the 2-DOF plane. The relationship between the joint angle vector $\theta = (\theta_1, \theta_2)^T$ and the end-effector position vector $x = (x, y)^T$ is as follows:

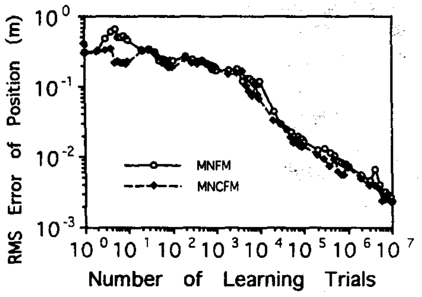
$$\begin{aligned} x &= x_0 + L_1\cos(\theta_1) + L_2\cos(\theta_1+\theta_2) \quad (10)\\ y &= y_0 + L_1\sin(\theta_1) + L_2\sin(\theta_1+\theta_2). \end{aligned}$$

L_1 is $0.30m$, L_2 is $0.25m$, the range of θ_1 is $[30°, 150°]$, and the range of θ_2 is $[-150°, 150°]$.

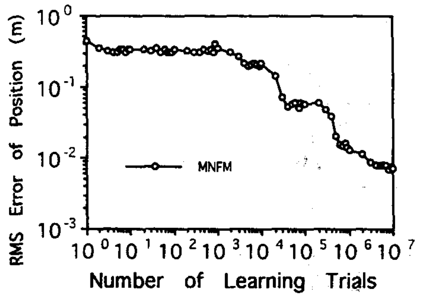
In the simulations, joint angle vectors were generated by using a uniform random number generator, and the end-effector positions that correspond to the generated vectors were used as the desired end-effector positions. In order to evaluate the performance of the solver, 1,000 desired end-effector positions were generated for the estimation of the root mean square (RMS) error of the end-effector position $e = x_d - f(\Phi_{im}(x_d))$.

A 4-layered neural network was used for the simulations. The 1st layer, i.e., the input layer, and the 4th layer, i.e., the output layer, consisted of linear neurons. The 2nd and the 3rd layers had 15 neurons each. The back-propagation method was utilized for the learning. Before the learning, the inverse kinematics solver had one expert the representative posture of which was $(0.0, 0.0)^T$.

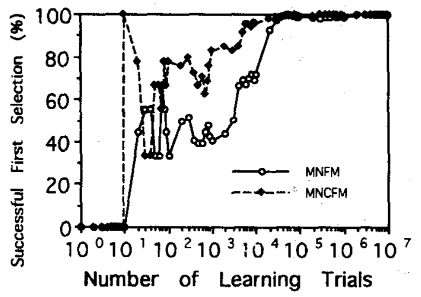
For comparison, the solver with the complete forward model which outputs $f(\theta)$ without error was



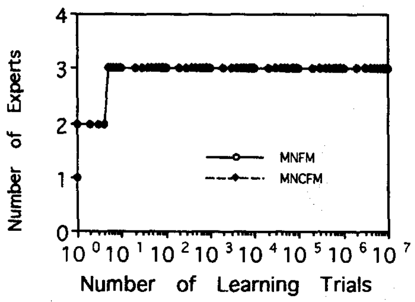
(a) RMS position error



(b) RMS position error of forward model

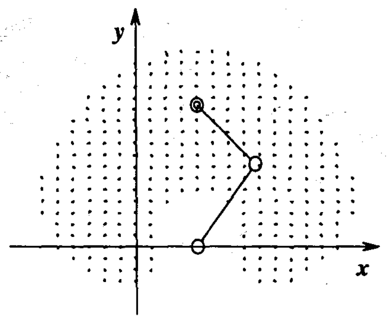


(c) Percentage of successful first selection

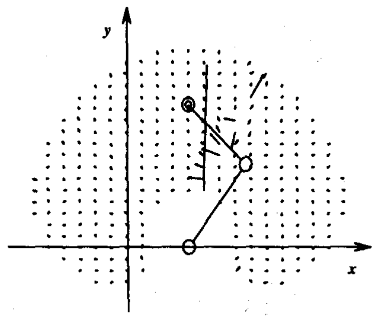


(d) Number of experts

Figure 2 Performance Change of Inverse Model by Learning



(a) Proposed Modular Neural Networks



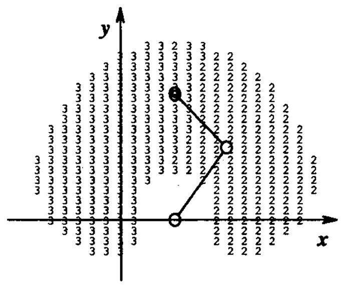
(b) Single Neural Network

Figure 3 Error Vectors of Inverse Kinematics Model

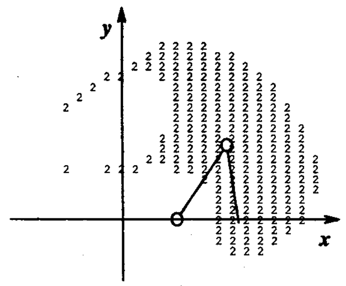
also tested. Hereafter, MNCFM indicates the Modular Neural network system with a Complete Forward Model. MNFM indicates the Modular Neural network system with a Forward kinematics Model that consists of a neural network with no previous learning. r_{eim} was $0.3m$, r_e was $0.005m$, r_{st} was $0.02m$, and r_{jix} was 10^2.

Figure 2 shows the progress of the inverse kinematics model learning. Figure 2(a) shows the RMS error of the end-effector position $\sqrt{E[e^T e]}$ by using the inverse kinematics model. It can be seen that the RMS error decreases and the precision of the inverse model becomes higher as the number of trials increases. The RMS error became $1.50 \times 10^{-3} m$ after 5×10^7 learning trials. The precision of MNCFM is better than that of MNFM. However, there is not so much difference between MNFM and MNCFM. Figure 2(b) shows the RMS error of the forward kinematics model $\sqrt{E[e_{fm}^T e_{fm}]}$. e_{fm} is the error vector of the forward kinematics model defined as $e_{fm} = f(\theta) - \Phi_{fm}(\theta)$. Figure 2(c) shows the percentage of the trials in which the posture generated by the first selected expert can successfully reach the desired position. After 10^5

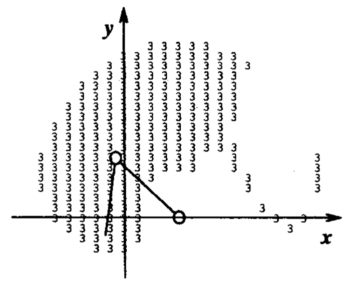
learning trials, the first selection was always correct. The percentage of MNCFM is better than that of MNFM. Figure 2(d) shows the number of the experts which constructs the inverse kinematics model. The number of the experts which constructs the inverse kinematics model became 3 after 4 times learning trials.



(a)Selected experts



(b)Active region of second expert



(c)Active region of third expert

Figure 4 Configuration of Learned Inverse Kinematics Model

Figure 3 shows the position error vector of one example of an inverse kinematics model acquired by the proposed method. We were able to obtain a precise inverse kinematics model of the overall points that the robot arm can reach. The simulations of the inverse kinematics model learning, consisting of a single neural networks, by using the Forward and Inverse Modeling, were also performed. The learning was performed from 10 different initial states of the neural network. The minimum RMS error after 10^9 learning trials was $1.20 \times 10^{-2}m$. An inverse model which consists of a single neural networks learned by using Forward and Inverse Modeling cannot obtain a precise inverse kinematics model of the arm as shown in Figure 3(b). There are some regions where it is far from precise, which is caused by the discontinuity of the inverse kinematics function.

Figure 4 illustrates how the expert is selected. Because the representative posture of the first expert was $(0.0, 0.0)^T$ and the Jacobian of the posture is singular, the expert is rarely used. The second expert and the third expert covers almost all region. Figure 4(b) shows the region where the predicted output error of the second expert is lower than $0.01m$. The graphics of the robot arm in Figure 4(b) shows the representative posture of the second expert. Figure 4(c) shows the region where the predicted output error of the third expert is lower than $0.01m$.

4.2 Inverse Kinematics Learning of a 7-DOF Arm

We considered the inverse kinematics model learning of a 7-DOF arm (Mitsubishi Heavy Industries, Ltd.'s "PA-10"). The configuration of the arm is illustrated in Figure 5. The simulations of the inverse kinematics learning as described in Section 4.1 were conducted.

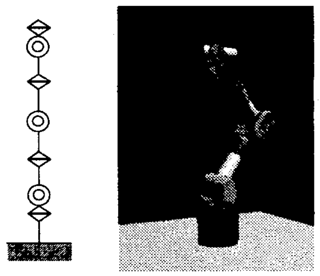
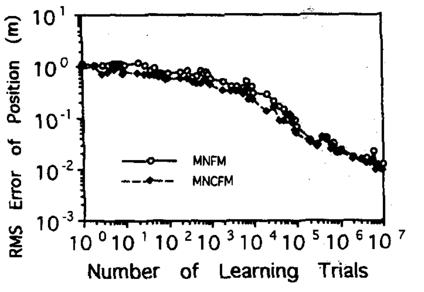


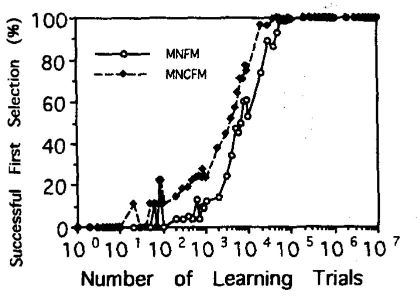
Figure 5 Configuration of 7-DOF Robot Arm

The 2nd and the 3rd layers of the forward kinematics model and the experts had 25 neurons each. 16,384 desired end-effector positions were generated by using a uniform random number generator for the evaluation.

Figure 6(a) shows the change of the RMS error of the end-effector position. Figure 6(b) the percentage of the successful first selection. After 10^7 learning trials, 5 or 6 experts were generated. After 10^7 learning trials, the RMS end-effector position error became $1.20 \times 10^{-2}m$.

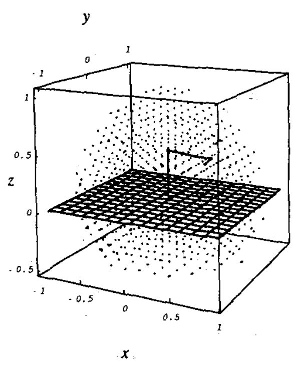


(a)RMS position error

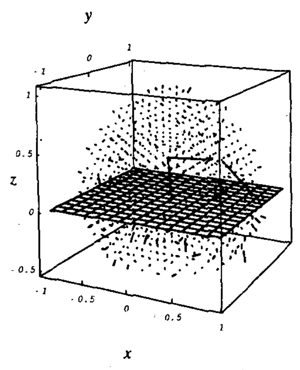


(b)Percentage of successful first selection

Figure 6 Performance Change of Proposed Inverse Kinematics Solver



(a) Proposed Modular Neural Networks



(b) Single Neural Network

Figure 7 Position Error Vector of Inverse Kinematics Model of 7-DOF Arm

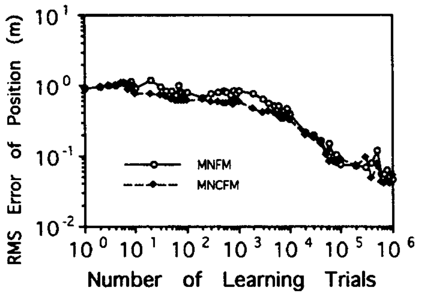
Figure 7(a) shows the position error vectors of the proposed inverse kinematics model. Figure 7(b) shows those of the inverse kinematics model which consists of single neural network learned by forward and inverse modeling.

We concluded that the proposed method succeeded in the inverse kinematics model learning of a 7-DOF arm. However, the computation time of one kinematics solution by the learned solver was almost $1.0ms$ (Intel Pentium II 450 MHz). The time is quite large compared to that of an analytical closed-form inverse kinematics solution. Since it is not easy to obtain an analytical closed-form inverse kinematics solution of a redundant arm and the characteristics of the arm is not always constant, the proposed method is still useful. The improvement of the learning is necessary. Almost 2.0×10^6 times learning trials were necessary to decrease the RMS error lower than $2.0 \times 10^{-2}m$. The on-line learning with a real robot is actually impossible. The precise forward kinematics model of the robot arm must be obtained.

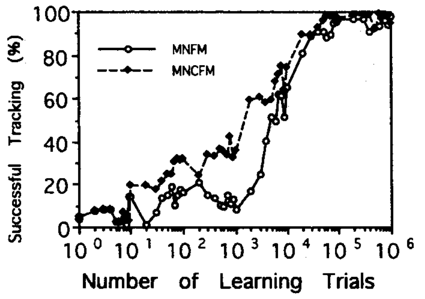
4.3 Learning during Tracking Control

The simulations of the inverse kinematics learning during the tracking control were performed using the 7-DOF arm used in Section 4.2.

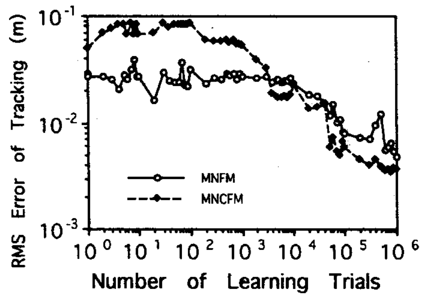
A desired trajectory $\boldsymbol{x}_d(k)$ was generated as follows. First, an initial joint angle vector $\boldsymbol{\theta}_s$ was generated by using a uniform random number generator. The end-effector position $\boldsymbol{x}_s$ that corresponded to $\boldsymbol{\theta}_s$ was regarded as the initial desired end-effector position. Second, the desired total change of the hand position $\Delta\boldsymbol{x}_{dt}$ was generated by using a normal random number generator, with the standard deviation of $0.2m$. The final desired hand position was calculated as $\boldsymbol{x}_e = \boldsymbol{x}_s + \Delta\boldsymbol{x}_{dt}$. $\boldsymbol{x}_d(k) = (1 - k/T_d)\boldsymbol{x}_s + (k/T_d)\boldsymbol{x}_e$ was used as the desired trajectory. T_d was fixed at 20. All of the trajectories can be realized by RMRC. 16,384 desired trajectories were generated for the evaluation of the inverse kinematics solver. The trials of the tracking control defined in Equations (7) and (8) to the generated trajectories were performed with the inverse kinematics model learning as described in Section 3. When the hand position error became larger than $0.05m$, the trial was regarded as a failure and the tracking control was given up.



(a)RMS position error of inverse kinematics model



(b)Percentage of successful tracking control trials



(c)RMS position error in tracking control trials
Figure 8 Performance Change of Tracking Control

Figure 8 shows the progress of the proposed learning method. 4 or 5 experts were generated after 10^6 times learning trials. Figure 8(a) shows the RMS error of the end-effector position by using the inverse kinematics solution as calculated by the learned inverse kinematics model. Figure 8(b) shows the percentage of the successful tracking trials when the desired trajectories for the evaluation were used. The percentage increased to more than 98 % after 10^6 times learning trials. The percentage of the inverse kinematics model consisting of a single neural networks learned by the Forward and Inverse Modeling was lower than 95 %. Figure 8(c) shows the RMS error of the end-effector position in the tracking trials. By the error feedback, the tracking error is much smaller than the RMS error of the inverse kinematics model. The proposed method can learn the precise tracking control function.

5 Conclusions

We proposed a novel modular neural network architecture for the inverse kinematics model learning and confirmed the performance of the proposed system by numerical experiments. The improvement for faster learning and the utilization of the redundant degrees of freedom will be reported in near future.

References

[1] C. G. Atkeson and S. Schaal, "Learning tasks from a single demonstration," *Proc. of IEEE International Conference on Robotics and Automation*, Vol.2, pp.1706-1712, 1997.

[2] R. A. Jacobs and M. I. Jordan," Learning Piecewise Control Strategies in a Modular Neural Network Architecture," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol.23, pp.337-345, 1993.

[3] H. Gomi and M. Kawato,"Recognition of Manipulated Objects by Motor Learning with Modular Architecture Networks," *Neural Networks*, Vol.6, pp.485-497, 1993.

[4] M. Kawato, "Multiple Paired Forward-Inverse Models in the Cerebellum," *Proc. of The Fifth International Conference on Neural Information Processing (ICONIP'98)*, Vol.3, pp.1177-1180, 1998

[5] D. DeMers and K. Kreutz-Delgado, "Solving Inverse Kinematics for Redundant Manipulators," in *Neural Systems for Robotics*, O. Omidvar and P. v. d. Smagt ed., Academic Press, 1997.

[6] E. Oyama and S. Tachi, "Inverse Kinematics Model Learning by Modular Architecture Neural Networks", *Proceedings of International Joint Conference on Neural Networks '99*, 1999.

[7] M. I. Jordan, "Supervised Learning and Systems with Excess Degrees of Freedom," *COINS Technical Report,88-27*,pp.1-41,1988.

[8] A. W. Moore,"Fast, Robust Adaptive Control by Learning only Forward Models," *Advances in Neural Information Processing Systems 4*, pp.571-578, 1992.

[9] E. Oyama and S. Tachi, "Inverse Model Learning by Using Extended Feedback System," *Proc. of the fifth International Symposium on Measurement and Control in Robotics(ISMCR'95)*, pp.291-296, 1995.

[10] D. E. Whitney, "Resolved Motion Rate Control of Manipulators and Human Prostheses," *IEEE Trans. on Man-MachineSystem*, Vol.10, No.2, pp.47-53, 1969.

[11] E. Oyama and S. Tachi, "Coordinate Transformation Learning of Hand Position Feedback Controller by Using Change of Position Error Norm," *Advances in Neural Information Processing Systems 11*, pp. 1038-1044, 1999.