Inverse Kinematics Learning for Robotic Arms with Fewer Degrees of Freedom by Modular Neural Network Systems

Eimei Oyama
National Institute of Advanced Industrial Science and Technology (AIST)
1-2-1 Namiki, Tsukuba Science City
Ibaraki 305-8564 Japan
Email: eimei.oyama@aist.go.jp

Taro Maeda
Human and Information Science Laboratory
NTT Communication Science Laboratories
3-1 Morinosato Wakamiya, Atsugi-shi,
Kanagawa 243-0198 Japan

John Q. Gan and Eric M. Rosales
Department of Computer Science
University of Essex
Colchester CO4 3SQ, UK

Karl F. MacDorman
Department of Adaptive Machine Systems
Osaka University
1-1 Yamadaoka Suita
Osaka 565-0871 Japan

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

Arvin Agah
Department of Electrical Engineering and Computer Science
The University of Kansas
Lawrence, KS 66045 USA

***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, the inverse kinematics system of typical robot arms with joint limits is a multi-valued and discontinuous function. Since it is difficult for a well-known multi-layer neural network to approximate this kind of function, a correct inverse kinematics model cannot be obtained by using a single neural network. To overcome the difficulties of inverse kinematics learning, we proposed a novel modular neural network system that consists of a number of experts, with each expert approximating a continuous part of the inverse kinematics function. The proposed system selects one appropriate expert whose output minimizes the expected position/orientation error of the end-effector of the arm. The system can learn a precise inverse kinematics model of a robotic arm with equal or more degrees of freedom than that of its end-effector. However, there are robotic arms with fewer degrees of freedom. The system cannot learn a precise inverse kinematics model of this kind of arm. To overcome this, we adopted a modified Gauss-Newton method for finding the least-squares solution. Through the modifications presented in this paper, the improved modular neural network system can obtain a precise inverse kinematics model of a general robotic arm.**

***Index Terms*— Learning control systems, Manipulator kinematics, Neural networks, Gauss-Newton method**

I. Introduction

People who work efficiently in complex, unstructured environments acquire their skills through various kinds of learning. It may be necessary to implement similar abilities in robots to enable them to work in the same kinds of environments [1][2].

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 much research [3][4]; however, many researchers did not pay enough attention to the fact that the inverse kinematics function of a typical robot arm with joint limits is a multi-valued, discontinuous function. Although given enough hidden units multi-layer neural networks are in theory universal approximators, it is difficult to train them to approximate these kinds of discontinuous functions with currently popular algorithms.

Jacobs et al. [5][6] proposed a modular neural network architecture that has been used by many researchers. However, the typical input-output mapping of their networks is described as a single-valued function. Their learning methods would not be suitable for the inverse kinematics learning of a general robotic arm.

The inverse kinematics function can be decomposed into a finite number of inverse kinematics solution branches. DeMers et al. proposed an inverse kinematics learning method in which a neural network learns each solution branch [7][8]. However, the method is purely off-line and is not applicable for on-line learning, that is, the simultaneous or alternate execution of robot control and inverse model learning. Furthermore, the method is not goal-directed. There is no direct way to train the learner to output a joint angle vector corresponding to a given desired end-effector position/orientation.

We proposed a novel modular neural network architecture for inverse kinematics learning based on DeMers' method [9][10]. The proposed modular neural network system consists of a number of experts, implemented using artificial neural networks. Each expert approximates a continuous region of the inverse kinematics function. The proposed modular neural network system selects one expert whose output minimizes the expected position/orientation error of the end-effector of the arm. The proposed system can learn a precise inverse kinematics model of a robotic arm with equal or more degrees of freedom than that of its end-effector.

However, there are robotic arms with fewer degrees of freedom. The system is not applicable to these robotic arms,

because it uses Resolved Motion Rate Control (RMRC) [11] to find the inverse kinematics solutions and RMRC is only applicable for a robotic arm with equal or greater degrees of freedom than that of its end-effector.

To obtain a precise inverse kinematics model of a robot with fewer degrees of freedom, some modifications are necessary. We modified the Gauss-Newton method for finding the joint angle vector trajectory from the initial posture of the arm to the given desired end-effector position/orientation. By the modifications proposed in this paper, the improved modular neural network system can obtain a precise inverse kinematics model of a general robotic arm. Numerical experiments of the inverse kinematics learning were performed in order to evaluate the performance of the improved system.

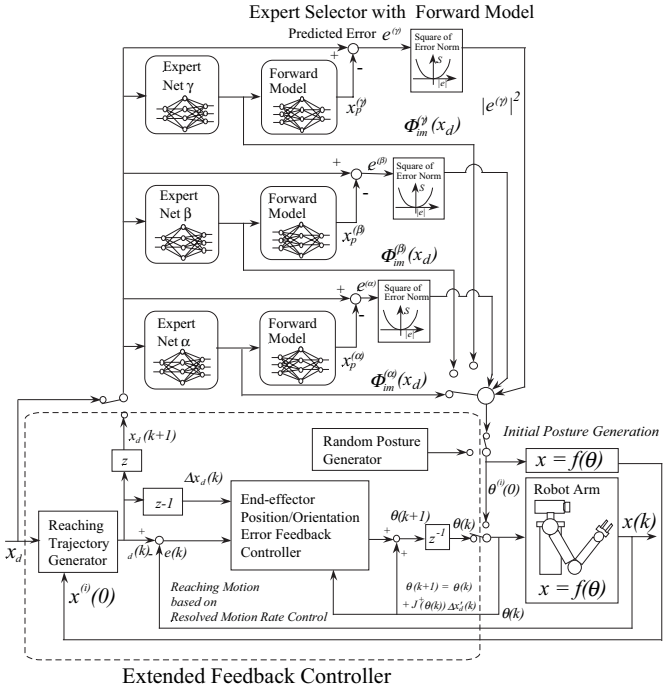


Fig. 1. Inverse kinematics solver with modular neural net system

II. Modular Neural Network System

We next review the original modular net system for inverse kinematics learning before presenting its modifications.

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 end-effector position/orientation vector $\boldsymbol{x}_d$ is given, an inverse kinematics computation 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 function 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 $\boldsymbol{g}(\boldsymbol{x})$ is usually a multi-valued and discontinuous function, the inverse kinematics function can be constructed by the appropriate synthesis of continuous functions [8][9][10].

A. Configuration of the proposed inverse kinematics solver

Fig. 1 shows the configuration of the improved inverse kinematics solver with the modular architecture networks for inverse kinematics learning. Each expert network in Fig. 1 approximates a continuous region of the inverse kinematics function. The forward models in Fig. 1 approximate the forward kinematics function of the robot arm. They have the same input-output relationship. The performance index of each expert is the predicted end-effector position/orientation error calculated by using the forward model. The expert selector chooses one appropriate expert by using the expected performances of the experts.

The extended feedback controller consists of the reaching trajectory generator, the end-effector position/orientation error feedback controller, and the random posture generator as shown in Fig. 1. The extended feedback controller calculates inverse kinematics solutions by a kind of iterative improvement method as described in Section II-C. When no precise solution is obtained by using the output of the selected expert as the initial value of the calculation, the controller performs a global search, which is also described in Section II-C. The proposed modular network system generates a new expert when no expert can generate an initial value that finally reaches an inverse kinematics solution for a given desired end-effector position/orientation.

B. Configuration of the expert

Each expert has its representative posture. 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 the *i-th* expert when the input of the expert is $\boldsymbol{x}$. Each expert is trained to satisfy the following equation:

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

The above condition can be easily satisfied by changing the bias parameters of the output layer of the expert. Let $\boldsymbol{\Phi}_{im}^{\prime(i)}(\boldsymbol{x})$ be the desired output for the *i-th* expert. To maintain the above condition, the expert is periodically trained by setting the desired output as follows:

$$\boldsymbol{\Phi}_{im}^{\prime(i)}(\boldsymbol{x}_r^{(i)}) = \boldsymbol{\theta}_r^{(i)}. \tag{2}$$

Each expert approximates a continuous region of the inverse kinematics function in which the reaching motion can move the end-effector smoothly from its representative posture.

The proposed modular net system uses the predicted position/orientation error as the performance index of the expert. The expert selector chooses the expert with the minimum predicted error. Let $\boldsymbol{\Phi}_{fm}(\boldsymbol{\theta})$ be the output of the forward

kinematics model. The predicted error of the *i-th* expert p_i is calculated as follows:

$$p_i = ||\boldsymbol{x}_d - \boldsymbol{\Phi}_{fm}(\boldsymbol{\Phi}_{im}^{(i)}(\boldsymbol{x}_d))||. \quad (3)$$

C. Extended feedback controller

Conventional on-line inverse model learning methods, such as forward and inverse modeling proposed by Jordan [4] and feedback error learning proposed by Kawato [12], 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 method.

When a desired end-effector position/orientation $\boldsymbol{x}_d$ is given, the expert selector chooses the expert with the minimum predicted error among all the experts. When the predicted error of the selected expert is lower than an appropriate threshold r_{eim}, the extended feedback controller conducts a reaching motion from the posture corresponding to the output of the expert to $\boldsymbol{x}_d$ by using a newly developed iterative improvement technique, as described in Section III. When the predicted error of the selected expert is larger than an appropriate threshold r_{eim}, the extended feedback controller conducts a reaching motion from the representative posture of the selected expert.

When the controller cannot find a precise solution because of the singularity of the Jacobian or the joint limits, the reaching motion is regarded as a failure and the reaching motions from the following three kinds of initial postures are conducted until a precise solution is obtained.

(1) The posture corresponding to the output of the other expert with the predicted error smaller than r_{eim}
(2) The representative postures of the randomly selected experts
(3) Randomly generated postures

When a precise solution is obtained from a randomly generated posture, a new expert is generated and the solution is used as the representative posture $\boldsymbol{\theta}_r$ of the expert. If a precise solution is obtained in the above steps, the solution is used as the desired output signal for the expert.

III. Updated Reaching Motion Control

A. Reaching motion for equal or more degrees of freedom

Before presenting the updated method, the previous method is presented.

Let $\boldsymbol{\theta}(0)$ be the initial posture of the iterative method, 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. The extended feedback controller conducts a reaching motion from $\boldsymbol{x}_s = \boldsymbol{f}(\boldsymbol{\theta}(0))$ to $\boldsymbol{x}_d$.

We assume that a precise end-effector position/orientation feedback controller is already obtained by numerical differentiation techniques or by learning [13]. We used a resolved motion rate control (RMRC) [11] to realize the reaching motion. The desired trajectory of the end-effector position/orientation is on a straight line which connects $\boldsymbol{x}_s$ to $\boldsymbol{x}_d$. The previous reaching motion is conducted as the tracking control to the following desired trajectory of the end-effector $\boldsymbol{x}_d(k)(k = 0, 1, \ldots, T+1)$ described as follows:

Let T be an integer that satisfies $T-1 \leq ||\boldsymbol{x}_d - \boldsymbol{x}_s||/r_{st} < T$, where r_{st} represents step size. 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:

$$\boldsymbol{x}_d(k) = \begin{cases} (1 - \frac{k}{T})\boldsymbol{x_s} + \frac{k}{T}\boldsymbol{x}_d & (0 \leq k < T) \\ \boldsymbol{x}_d & (k \geq T). \end{cases} \quad (4)$$

When the orientation is represented by the direction cosine matrix or the quaternion, the components of $\boldsymbol{x}_d(k)$ must be normalized.

Let $\boldsymbol{J}^+(\boldsymbol{\theta})$ be the pseudo-inverse matrix (Moore-Penrose generalized inverse matrix) of $\boldsymbol{J}(\boldsymbol{\theta})$, which is calculated as

$$\boldsymbol{J}^+(\boldsymbol{\theta}) = \boldsymbol{J}^T(\boldsymbol{\theta})(\boldsymbol{J}(\boldsymbol{\theta})\boldsymbol{J}^T(\boldsymbol{\theta}))^{-1}. \quad (5)$$

$\boldsymbol{J}^+(\boldsymbol{\theta})$ 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 and $\boldsymbol{\Delta\theta}(k)$ be the change of $\boldsymbol{\theta}(k)$. The trajectory of the joint angle vector $\boldsymbol{\theta}(k)$ is calculated as follows:

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) + \boldsymbol{\Delta\theta}(k) \quad (6)$$

$$\boldsymbol{\Delta\theta}(k) = \boldsymbol{J}^+(\boldsymbol{\theta}(k))(\boldsymbol{x}_d(k+1) - \boldsymbol{f}(\boldsymbol{\theta}(k))). \quad (7)$$

When n is smaller than or equal to m and r_{st} is small enough for the non-linearity of the forward kinematics function $\boldsymbol{f}(\boldsymbol{\theta})$, $\boldsymbol{\theta}(k+1)$ can satisfy the following equation:

$$\boldsymbol{x}_d(k+1) \approx \boldsymbol{f}(\boldsymbol{\theta}(k+1)). \quad (8)$$

If a precise solution $\boldsymbol{\theta}(k)$, from which end-effector position/orientation error norm $||\boldsymbol{x}_d(k) - \boldsymbol{f}(\boldsymbol{\theta}(k))||$ is lower than an appropriate threshold r_e, is obtained, the solution is used for training the selected expert as follows:

$$\boldsymbol{\Phi}_{im}^{\prime(i)}(\boldsymbol{x}_d(k)) = \boldsymbol{\theta}(k). \quad (9)$$

B. Updated reaching motion

$\boldsymbol{J}^+(\boldsymbol{\theta})$ defined in Equation (5) cannot be calculated when n is greater than m. Usually, an arm with fewer degrees of freedom cannot realize a given desired trajectory of the end-effector position/orientation. The reaching motion is not achieved by tracking a straight line from $\boldsymbol{x}_s = \boldsymbol{f}(\boldsymbol{\theta}(0))$ to $\boldsymbol{x}_d$ as in Section III-A. A slight but important modification of the reaching motion is necessary.

Let $\boldsymbol{J}^*(\boldsymbol{\theta})$ be the generalized inverse matrix for finding a least-squares solution, which is calculated as follows:

$$\boldsymbol{J}^*(\boldsymbol{\theta}) = (\boldsymbol{J}^T(\boldsymbol{\theta})\boldsymbol{J}(\boldsymbol{\theta}))^{-1}\boldsymbol{J}^T(\boldsymbol{\theta}). \quad (10)$$

The real calculation is conducted by using Singular Value Decomposition (SVD) of $\boldsymbol{J}(\boldsymbol{\theta})$. By using $\boldsymbol{J}^*(\boldsymbol{\theta})$, the update vector of the Gauss-Newton method is calculated as follows:

$$\boldsymbol{\Delta\theta}_G(k) = \boldsymbol{J}^*(\boldsymbol{\theta}(k))(\boldsymbol{\theta}(k))(\boldsymbol{x}_d - \boldsymbol{f}(\boldsymbol{\theta}(k))). \quad (11)$$

The above $\Delta\boldsymbol{\theta}_G(k)$ is sometimes too large for controlling the robotic arm. Let $r_{\theta st}$ be the step size parameter to keep $||\Delta\boldsymbol{\theta}(k)||$ small enough. $\Delta\boldsymbol{\theta}(k)$ is calculated as follows:

$$\Delta\boldsymbol{\theta}(k) = \begin{cases} \Delta\boldsymbol{\theta}_G(k) & (||\Delta\boldsymbol{\theta}_G(k)|| < r_{\theta st}) \\ \frac{r_{\theta st}}{||\Delta\boldsymbol{\theta}_G(k)||}\Delta\boldsymbol{\theta}_G(k) & (||\Delta\boldsymbol{\theta}_G(k)|| \geq r_{\theta st}). \end{cases} \quad (12)$$

$\boldsymbol{\theta}(k)$ is a unique inverse kinematics solution for $\boldsymbol{x}(k) = \boldsymbol{f}(\boldsymbol{\theta}(k))$ in most cases. Instead of Equation (9), the learning of the selected expert learning is conducted as follows:

$$\boldsymbol{\Phi}'^{(i)}_{im}(\boldsymbol{x}(k)) = \boldsymbol{\theta}(k). \quad (13)$$

Let $r_{\theta min}$ be an appropriate positive number defining the minimum size of $\Delta\boldsymbol{\theta}(k)$. The reaching motion is regarded as successfully finished when the following inequalities are established:

$$||\boldsymbol{x}_d - \boldsymbol{f}(\boldsymbol{\theta}(k))|| < r_e \quad (14)$$

$$||\Delta\boldsymbol{\theta}(k)|| < r_{\theta min} \quad (15)$$

IV. Simulations

The modular neural network systems selected the expert with minimum error by using the complete forward kinematics model $\boldsymbol{f}(\boldsymbol{\theta})$. We performed simulations of learning the inverse kinematics model of a five degree-of-freedom (DoF) arm, and for simplicity tested the modular neural network systems in these simulations.

A. A novel learner

Such learning algorithms as the backpropagation of error that are typically applied to multi-layer neural networks are not fast. To accelerate the learning speed, we adopted a novel learner as the expert of the proposed modular network system and a novel learning method based on the Kalman filter [14]. The learner is an ensemble of linear neural networks. We call the learner *collection of linear learning units* (CLLU). The learner is described in Appendix A.

B. Learning the inverse kinematics of a 5-DoF arm

We conducted simulations of learning the inverse kinematics model of a 5-DoF arm (Pioneer 2 Robotic Arm) [15] for the overall end-effector position and orientation. The configuration of the arm is illustrated in Fig. 2. Let $\boldsymbol{x} = (\phi_1, \phi_2, \phi_3, x, y, z)^T$ be the end-effector position/orientation vector. $(\phi_1, \phi_2, \phi_3)^T$ is the Euler angle vector and $(x, y, z)^T$ is the position vector. Let $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)^T$ be the joint angle vector. The learning of the inverse kinematics model $\boldsymbol{\Phi}_{im}(\boldsymbol{x})$ that transforms $\boldsymbol{x}$ to $\boldsymbol{\theta}$ was conducted.

For comparison, the modular neural network system was also tested under the condition that each expert was represented by a multi-layer neural network trained by the backpropagation of error. Hereafter, MLN indicates this system, whilst CLLU indicates the modular neural network system with a collection of linear learning units as experts.

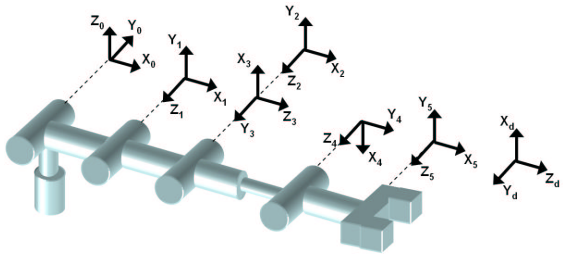


Fig. 2. Configuration of Power 2 Robotic Arm

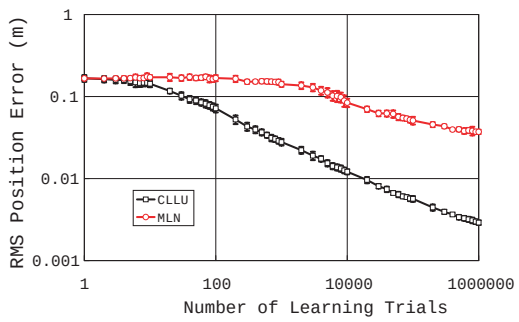
In the simulations, joint angle vectors were generated by using a uniform random number generator, and the end-effector positions that correspond to the generated vectors were used as the desired end-effector positions/orientation $\boldsymbol{x}_d$. To evaluate the performance of the proposed neural net system, 3,200,000 desired end-effector positions/orientations were generated for the estimation of the root mean square (RMS) error of the end-effector position and orientation. r_{eim} was set at $0.33\ m$, r_e was set at $0.002\ m$ and r_{st} was set at $0.05\ rad$.

The parameters concerning CLLU are defined in Appendix A and set as follows: P_0 was set at 10^4. Q and R were set at 0.1 and 0.01, respectively. $r_{e\theta}$ was set at 0.0001. $T_{e\theta}$ was set at 1. α was set at 2. r_{min} was set at $1/2^6$. By changing the above parameters, especially, R, $r_{e\theta}$ and $T_{e\theta}$, the precision of the learned inverse kinematics model and the memory consumption can be changed.

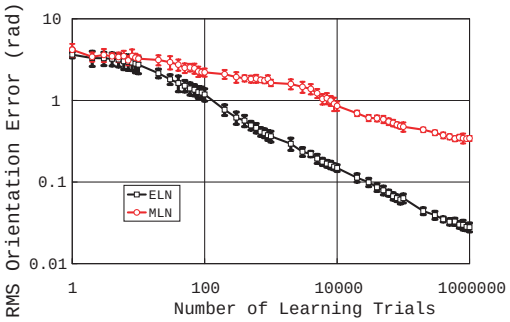
A 4-layered neural network was used for MLN. The first layer (input layer) and the fourth layer (output layer) of the experts consisted of linear neurons. The second and the third layers each consisted of 30 nonlinear neurons. The backpropagation method was used for training. The learning rate for the experts was set at 0.005. The momentum parameter was set at 0.5.

Ten sets of 1,000,000 learning trials for the inverse kinematics learning by MLN and CLLU were conducted. Fig. 3(a) shows the means and the standard deviations of the learning curves in terms of the RMS error in the end-effector position controlled by CLLU and MLN, respectively. The vertical bars represent the standard deviations. Fig. 3(b) shows the means and the standard deviations of the learning curves in terms of the RMS error in the end-effector orientation in a similar manner. Fig. 3(c) shows the means and the standard deviations of the percentage of the trials in which the posture generated by the first selected expert can successfully reach the desired position/orientation. Fig. 3(d) shows the means and the standard deviations of the number of experts needed in CLLU and MLN to model the inverse kinematics.

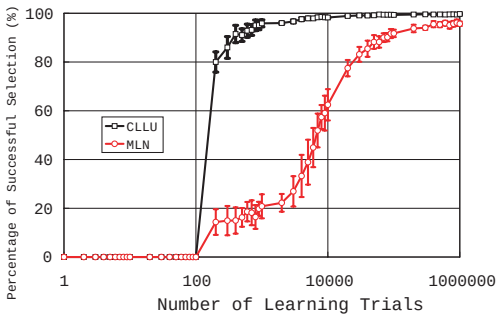
It can be seen that the RMS position error and the RMS orientation error decrease and the precision of the inverse model increases with the number of trials. The RMS position error of CLLU became smaller than $1.0 \times 10^{-2} m$ at about 2.0×10^4 learning trials. The RMS orientation error of CLLU became smaller than $1.0 \times 10^{-1} rad$ before 4.0×10^4 learning



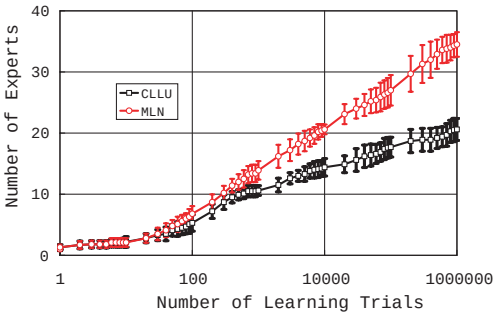
(a) RMS position error



(b) RMS orientation error



(c) Percentage of successful first selection



(d) Number of experts

Fig. 3. Performance change of proposed inverse kinematics solver

trials. The RMS position error of MLN was still larger than $1.0 \times 10^{-2} m$ and the RMS orientation error of MLN was still larger than $3.0 \times 10^{-1} rad$ after 1.0×10^6 learning trials. 4.0×10^4 learning trials by CLLU take 152 seconds using the Intel Xeon 2.0 GHz (512KB cache/FSB400) and Intel C++ Compiler. 10^5 learning trials by CLLU take 310 seconds. 10^6 learning trials by MLN take $18,464$ seconds. The learning speed of the CLLU is much faster than that of MLN.

It should be noted that CLLU requires more memory than MLN. While MLN requires less than $2MB$ memory, CLLU requires $412MB$ memory in 10^6 learning trials when all the real numbers in the programs were stored as 64-bit floating-point data.

V. Conclusions

In this paper, we proposed some modifications of the modular neural network system for learning the inverse kinematics of robotic arms with fewer degrees of freedom. We confirmed the performance of the proposed system by numerical experiments. By the proposed improvement, the modular neural net system is applicable to general least-squares problems. Although the proposed architecture still has a number of limitations, we believe that the architecture can be the basic model of the practical inverse kinematics solver with a learning function. We are implementing more efficient learners, such as locally weighted regression (LWR) [16], locally weighted projection regression (LWPR) [17], and the normalized Gaussian network (NGnet) using the on-line expectation-maximization (EM) algorithm [18]. Although these learners require more computation, they require much less memory and relatively fewer training data. The results of using these learners will be reported in the 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] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233-242, 1999. [why isn't the title capitalized, as others are?]

[3] M. Kuperstein,"Neural Model of Adaptive Hand-Eye Coordination for Single Postures," *Science*, vol. 239, pp. 1308-1311, 1988.

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

[5] R. A. Jacobs, M. I. Jordan, S. J. Nowlan and G. E. Hinton,"Adaptive Mixtures of Local Experts," *Neural Computation*, vol. 3, pp. 79-87,1991.

[6] 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.

[7] D. DeMers and K. Kreutz-Delgado, "Learning Global Direct Inverse Kinematics," *Advances in Neural Information Processing Systems 4*, pp. 589-594, 1992.

[8] 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.

[9] E. Oyama and S. Tachi, "Inverse Kinematics Model Learning by Modular Architecture Neural Networks," *Proc. of International Joint Conference on Neural Networks '99*, 1999.
[10] E. Oyama and S. Tachi, "Modular Neural Net System for Inverse Kinematics Learning," *Proc. of International Conference on Robotics and Automation 2000*, pp. 3239-3246, 2000.
[11] D. E. Whitney, "Resolved Motion Rate Control of Manipulators and Human Prostheses," *IEEE Trans. on Man-Machine System*, vol. 10, no. 2, pp. 47-53, 1969.
[12] M. Kawato, K. Furukawa and R. Suzuki, "A Hierarchical Neural-network Model for Control and Learning of Voluntary Movement," *Biological Cybernetics*,vol.57, pp.169-185, 1987
[13] E. Oyama, A. Agah, T. Maeda, S. Tachi and K. F. MacDorman, "Coordinate Transformation Learning of Human Visual Feedback Controller based on Disturbance Noise and Feedback Error Signal," *Proc. of International Conference on Robotics and Automation 2001*, pp. 4186-4193, 2001.
[14] E. Oyama and T. Maeda, "Inverse Kinematics Learning by Modular Neural Network System which uses Specialized Neural Networks," *Proc. of International Symposium on Measurement and Control in Robotics 2003*, pp. 275-280, 2003.
[15] J. Q. Gan, E. Oyama, E. Rosales, and H. Hu, "A Complete Analytical Solution to the Inverse Kinematics of the Pioneer 2 Robotic Arm," Robotica, will appear.
[16] S. Schaal, C. G. Atkeson and S. Vijayakumar,"Real-Time Robot Learning with Locally weighted [?] Statistical Learning," *Proc. of IEEE International Conference on Robotics and Automation*, pp.288-293, 2000.
[17] S. Vijayakumar and S. Schaal, "First and Efficient Incremental Learning for High-dimensional Movement Systems," *Proc. of IEEE International Conference on Robotics and Automation*, pp. 1894-1899, 2000.
[18] S. Sato and S. Ishii, "On-line EM Algorithm for the Normalized Gaussian Network," Neural Computation, vol. 12, no. 2, 2000.

Appendix A Novel Learner Composed of Linear Learning Units

A. A linear learning unit

In the small region around a point of the position/orientation vector space of the end-effector, the continuous part of the inverse kinematics function $\boldsymbol{f}^{-1}(\boldsymbol{x})$ can be approximated as follows:

$$\boldsymbol{f}^{-1}(\boldsymbol{x}) \approx \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}.$$

where $\boldsymbol{A}$ is an $m \times n$ matrix and $\boldsymbol{b}$ is an $m \times 1$ vector. Let us consider the learning of $\boldsymbol{f}^{-1}(\boldsymbol{x})$ around $\boldsymbol{x}$ by changing $\boldsymbol{A}$ and $\boldsymbol{b}$. We use the Kalman filter algorithm for the learning.

To simplify the description, the following variables are introduced:

$$\boldsymbol{X} = (\boldsymbol{x}^T, 1)^T$$
$$\hat{\boldsymbol{A}} = (\boldsymbol{A}, \boldsymbol{b}).$$

Let $\boldsymbol{P}$ be the covariance matrix of each row vector of $\hat{\boldsymbol{A}}$. $\boldsymbol{P}$ is an $(n+1) \times (n+1)$ positive definite symmetric matrix. Let Q be a scalar parameter describing the rate of the change of each component of $\boldsymbol{A}$ and $\boldsymbol{b}$.

Before learning, we do not know the value of $\hat{\boldsymbol{A}}$. Therefore, the initial value of $\boldsymbol{P}$ is set as follows:

$$\boldsymbol{P}(0) = P_0 \boldsymbol{I}_{n+1}$$

where P_0 is a very large positive real number and $\boldsymbol{I}_{n+1}$ is a $n+1 \times n+1$ identity matrix.

Periodically, $\boldsymbol{P}$ is updated as follows:

$$\boldsymbol{P}(+) = \boldsymbol{P}(-) + Q\boldsymbol{I}_{n+1}$$

where $\boldsymbol{P}(-)$ is the value of $\boldsymbol{P}$ before the update and $\boldsymbol{P}(+)$ is the value of P after the update.

Let $\boldsymbol{\theta}' = \boldsymbol{f}^{-1}(\boldsymbol{x})$ be the desired output for the learner and R be a scalar parameter describing the covariance of the following error vector defined as:

$$\boldsymbol{e}_\theta = \boldsymbol{\theta}' - \hat{\boldsymbol{A}}\boldsymbol{X}.$$

Since we assume that $\boldsymbol{\theta}'$ contains no noise, R mainly corresponds to the non-linearity of $\boldsymbol{f}^{-1}(\boldsymbol{x})$.

When $\boldsymbol{x}$ and $\boldsymbol{\theta}' = \boldsymbol{f}^{-1}(\boldsymbol{x})$ is given, $\hat{\boldsymbol{A}}$ can be updated effectively by using the Kalman filter algorithm. The Kalman gain is calculated as follows:

$$\boldsymbol{K} = \frac{\boldsymbol{P}(-)\boldsymbol{X}}{(\boldsymbol{X}^T\boldsymbol{P}(-)\boldsymbol{X} + R)}.$$

Let $\hat{\boldsymbol{A}}(-)$ be the value of $\hat{\boldsymbol{A}}$ before learning and $\hat{\boldsymbol{A}}(+)$ be the value of $\hat{\boldsymbol{A}}$ after learning. $\hat{\boldsymbol{A}}$ and $\boldsymbol{P}$ are updated as follows:

$$\hat{\boldsymbol{A}}(+) = \hat{\boldsymbol{A}}(-) + \boldsymbol{K}\boldsymbol{e}_\theta^T$$
$$\boldsymbol{P}(+) = \boldsymbol{P}(-) - \boldsymbol{K}\boldsymbol{X}^T\boldsymbol{P}(-).$$

After $n+1$ updates of $\hat{\boldsymbol{A}}$ and $\boldsymbol{P}$, $\hat{\boldsymbol{A}}$ converges to a relatively precise value. If the position/orientation vector space is divided into a set of small regions, the above learning algorithm is useful in each small region.

B. Division of the input vector space

We use a novel learner that consists of a collection of the above linear learning units. Let $\boldsymbol{x}_{min}$ and $\boldsymbol{x}_{max}$ be the lower and upper limits for the input vector $\boldsymbol{x}$.

Let us consider the *j-th* unit that has variables such as $\boldsymbol{x}_{min}^{(j)}$, $\boldsymbol{x}_{max}^{(j)}$, $\hat{\boldsymbol{A}}^{(j)}$ and $\boldsymbol{P}^{(j)}$. $\boldsymbol{x}_{min}^{(j)}$ and $\boldsymbol{x}_{max}^{(j)}$ represent the limits of the input region of the unit. The input region of the unit is a $n-D$ hyper-rectangular parallelepiped. When the input $\boldsymbol{x}$ satisfies the following inequalities

$$x_{min,i}^{(j)} \leq x_i \leq x_{max,i}^{(j)} (i = 1, 2, \ldots, n),$$

the output of the unit is calculated as

$$\boldsymbol{\theta}^{(j)} = \hat{\boldsymbol{A}}^{(j)}\boldsymbol{X}.$$

$\boldsymbol{\theta}^{(j)}$ is used as the output of the whole units as

$$\boldsymbol{\theta} = \boldsymbol{\theta}^{(j)}.$$

Let us consider the following updated error of the *j-th* unit.

$$\boldsymbol{e}_\theta^{(j)}(+) = \boldsymbol{\theta}' - \hat{\boldsymbol{A}}^{(j)}(+)\boldsymbol{X}$$

As the learning proceeds, the norm of $\boldsymbol{P}^{(j)}$ becomes small and the change of $\hat{\boldsymbol{A}}^{(j)}$ becomes small. When the input region

of the *j-th* unit is too large, the learning based on the Kalman filter cannot keep the inequality defined as

$$|\boldsymbol{e}_{\theta}^{(j)}(+)| < r_{e\theta}$$

where $r_{e\theta}$ is an appropriate positive threshold. When the above inequality breaks down an appropriate number of times $T_{e\theta}$, the *j-th* unit is divided. As $r_{e\theta}$ decreases, the possibility of the division of the unit increases and the memory consumption increases.

The conceptual diagram of the calculation of the learner is illustrated in Fig. A-1.

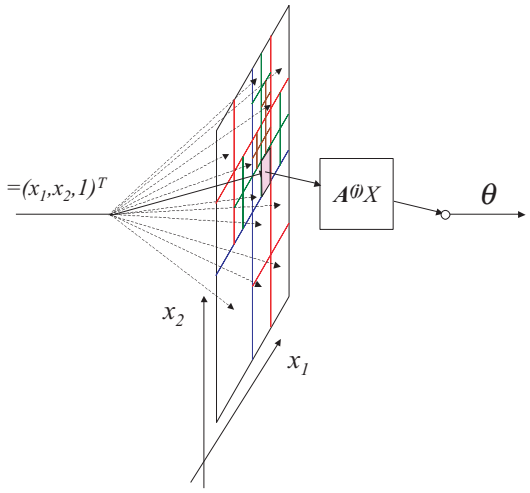


Fig. A-1 Calculation of the proposed learner

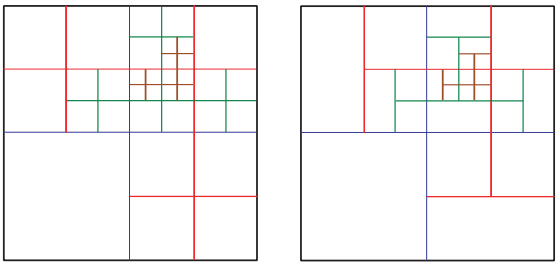
In the previous work [14], the *j-th* unit is divided into 2^n hyper-rectangular parallelepipeds. The length of the side of the generated units is half that of the *j-th* unit. The *j-th* unit is divided into 2 hyper-rectangular parallelepipeds. The following scaled lengths of the sides of the parallelepiped are calculated:

$$r_i = \frac{x_{max,i}^{(j)} - x_{min,i}^{(j)}}{x_{max,i} - x_{min,i}} (i = 1, ..., n)$$

The axis corresponding to the maximum scaled length is selected for the division. The length of the selected side of the generated units is half that of the *j-th* unit. The length of the other side of the generated units is the same as that of the *j-th* unit.

Fig. A-2 shows two examples of the divided input space when the input dimension n is 2. Fig. A-2(a) shows an example of the divided input space by the previous division method [14] and Fig. A-2(b) shows that by the method used in this paper. The modification of the division method can reduce the memory consumption. However, the number of divisions must be greater than that of the previous method to realize the same precision. The parameters concerning the divisions, R, $r_{e\theta}$, and $T_{e\theta}$ should be selected carefully. It should be noted that the division method is still under development.

To suppress memory consumption, the minimum size of units is defined. If the scaled length of the side of a unit is smaller than the minimum size r_{min}, the unit is not divided even if the unit cannot learn a given desired output.



(a) Previous method (b) Modified method

Fig. A-2 Division of Input Space (2 Dimensional Case)

$\boldsymbol{P}^{(l)}$ of *l-st* generated unit is initialized as follows:

$$\boldsymbol{P}^{(l)} = \alpha \boldsymbol{P}^{(j)}$$

where α is an appropriate real number, larger than 1. The initial value of $\hat{\boldsymbol{A}}^{(l)}$ of the *l-st* generated unit is set at $\hat{\boldsymbol{A}}^{(j)}$. The generated units are trained instead of the *j-th* unit.

One drawback of the proposed learner is the discontinuity of the input-output relationship of the learner. However, the discontinuity is not a problem when a desired position/orientation vector is given discretely. Furthermore, a number of interpolation techniques developed for robotic control is useful when a desired trajectory of the position/orientation vector is given. The other drawback of the learner is the large memory consumption when the input dimension is large. However, the maximum input dimension of the inverse kinematics problem is 6 or 7 in many cases. We believe further improvement of the learner can overcome these drawbacks.