The 2015 World Congress on
Aeronautics, Nano, Bio, Robotics, and Energy (ANBRE15)
Incheon, Korea, August 25-28, 2015

Learning of inverse kinematics using neural networks and its application to kinematic control of position-based servo motor

*Fusaomi Nagata[1)], Shota Inoue[2)], Satoru Fujii[3)], Akimasa Otsuka[4)], Keigo Watanabe[5)] and Maki K. Habib[6)]

[1), 2), 3), 4)] *Department of Mechanical Engineering, Faculty of Engineering, Tokyo University of Science, Yamaguchi, Japan*
[5)] *Graduate School of Natural Science and Technology, Okayama University*
[6)] *School of Sciences and Engineering, The American University in Cairo*
[1)] nagata@rs.tus.ac.jp; [4)] otsuka_a@rs.tus.ac.jp
[5)] watanabe@sys.okayama-u.ac.jp; [6)] maki@aucegypt.edu

ABSTRACT

Generally, in making neural networks learn nonlinear relations effectively, it is desired to have proper training set. The training set consists of multiple pairs of an input and an output vectors. Each input vector is introduced to the input layer for forward computation and the paired output training vector is compared with the yielded vector from the output layer. Then, weights between neurons are updated using a back propagation algorithm in backward calculation. The time required for the learning process depends on the number of total weights in the neural network and that of the input-output pairs in the training set. In the proposed learning process, after certain number of iteration, input-output pairs having the worse errors are extracted from the original training set and form a new temporary set. From the following iteration, the temporary training set is applied instead of the original set. In this case, only pairs with worse errors are used for updating the weights until the mean value of errors decreases to a desired level. Once the learning is conducted using the temporary set, the original training set is applied again instead of the temporary set. It is expected by alternately applying the above two types of training sets for iterative learning the convergence time can be efficiently reduced. The effectiveness is demonstrated through simulation experiments using a kinematic model of a leg with four-DOFs. Additionally, when the tip of the leg is controlled kinematically in Cartesian space, an inverse Jacobian matrix is needed. However, the calculation of the inverse Jacobian is not easy. Hence, neural networks based inverse kinematics is used to implement the function of inverse Jacobian to overcome the complexity.

[1)] Professor
[2)] Undergraduate Student
[3)] Undergraduate Student
[4)] Doctor
[5)] Professor
[6)] Professor

1. INTRODUCTION

When designing a serial link structure as shown in Fig. 1 for a multi-legged robot, first of all, its inverse kinematics problem must be solved. Hoang et al. proposed a differential kinematics algorithm to generate omnidirectional walking trajectory of a leg based on backstepping control using Lyapunov stability. Simulation results for walking motion of one leg of the 6LR were shown to prove the effectiveness and applicability of the proposed controller (Hoang 2014). Tejomurtula and Kak proposed a solution of inverse kinematics concerning a simple two link manipulator (Tejomurtula 1999). Duka designed a feedforwad neural network to solve the inverse kinematics problem of a three-link planar manipulator (Duka 2014). Also, Maeda et al. designed a position control scheme for actual robot systems using high dimensional neural networks, in which complex-valued neural network and quaternion neural network learned the inverse kinematics of the robot systems. Two dimensional SCARA robot and three dimensional robot were well controlled using the inverse kinematics (Maeda 2014).

Generally, in making neural networks learn nonlinear relations suitably, desired training set prepared in advance is used. The training set consists of multiple pairs of an input vector and an output one. Each input vector is given to the input layer for forward calculation and the paired output vector is compared with the vector yielded from the output layer. Also, backward calculation means updating the weights using a back propagation algorithm. One cycle consists of one forward calculation and backward one. The time required for the learning process of the neural networks depends on the number of total weights in the neural networks and the one of the input-output pairs in the training set. This paper describes neural networks with efficient weights tuning ability in order to effectively learn the inverse kinematics of a leg kit with multi-DOFs.

In the proposed learning process, after certain number of iteration, input-output pairs having the worse errors are extracted from the original training set and form a new temporary set. Note that an iteration of learning uses all pairs in the training set. Then, from the following iteration, the temporary set is applied instead of the original set. In this case, only pairs with worse errors are used for updating weights until the mean value of errors reduces to a desired level. Once the learning is conducted using the temporary training set, the original set is applied again instead of the temporary set. It is expected by alternately giving the two kinds of training sets the convergence time can be efficiently reduced. The effectiveness is proved through simulation experiments using a kinematic model of a leg with four-DOFs.

Additionally, when the tip of the leg with a serial link structure is controlled kinematically in Cartesian space, an inverse Jacobian matrix is needed to generate velocities in joint space. However, the calculation of the inverse Jacobian is not easy. Hence, neural networks based inverse kinematics is used to implement the function of Jacobian to overcome the complexity.

2. LEG WITH FOUR-DOFS AND ITS FORWARD KINEMATICS

Generally, multi-legged robot has multiple legs with a serial link structure. Figure 2 shows an example of a leg with four-DOFs, which is used for graduation study of undergraduate students (Nagata 2013). Five coordinate systems $\Sigma_k - x_k y_k z_k (0 \leq k \leq 4)$ are assigned at each joint. The position of the arm tip in base frame $\Sigma_0 - x_0 y_0 z_0$ is defined with $\mathrm{x} = [x\ y\ z]^T$. Table 1 tabulates the Denavit-Hartenberg (DH) notation extracted from the leg with four joints. a is the link length which is the distance between two adjacent z-axes measured along x-axis, α is the link twist angle between two adjacent z-axes measured around x-axis, d is the link offset which is the distance between two adjacent x-axes measured along z-axis and θ is the joint angle between two adjacent x-axes measured around z-axis. Actually, Fig. 2 shows the initial pose of the leg with the angles of $\boldsymbol{\theta} = [\theta_1\ \theta_2\ \theta_3\ \theta_4]^{\mathrm{T}} = [0\ 0\ 0\ 0]^{\mathrm{T}}$. The homogeneous transform ${}^{k-1}\mathrm{T}_k$ using the four parameters is written by

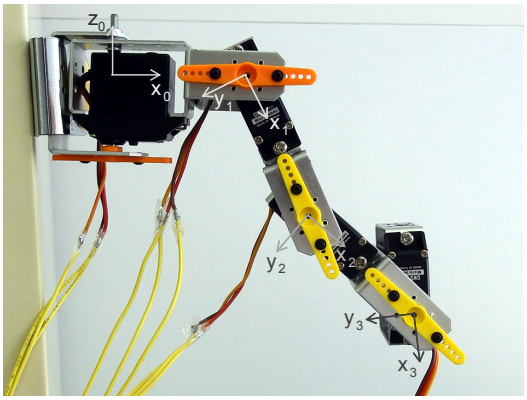

Fig. 1 A serial link structure with four-DOFs.

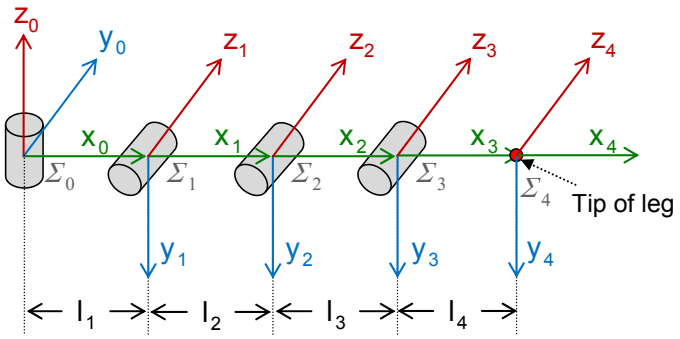

Fig. 2 Kinematics model of the leg with four-DOFs.

Table 1 Denabit-Hartenberg notation designed for the leg with four joints.

k	a	α	d	θ
1	l_1	$-\frac{\pi}{2}$	0	θ_1
2	l_2	0	0	θ_2
3	l_3	0	0	θ_3
4	l_4	0	0	θ_4

$$^{k-1}T_k = Rot(z,\theta)Trans(0,0,d)Trans(a,0,0)Rot(x,\alpha) = \begin{pmatrix} C_\theta & -S_\theta C_\alpha & S_\theta S_\alpha & aC_\theta \\ S_\theta & C_\theta C_\alpha & -C_\theta S_\alpha & aS_\theta \\ 0 & S_\alpha & C_\alpha & d \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1)$$

so that, $^0\boldsymbol{T}_4$ is obtained by

$$^0T_4 = {}^0T_1\,{}^1T_2\,{}^2T_3\,{}^3T_4 = \begin{pmatrix} R_{11} & R_{12} & -S_{\theta 1} & x \\ R_{21} & R_{22} & C_{\theta 1} & y \\ R_{31} & R_{32} & 0 & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2)$$

where

$$R_{11} = C_{\theta 4}(C_{\theta 1}C_{\theta 2}C_{\theta 3} - C_{\theta 1}S_{\theta 2}S_{\theta 3}) + S_{\theta 4}(C_{\theta 1}S_{\theta 2}C_{\theta 3} - C_{\theta 1}S_{\theta 3}S_{\theta 4}) \quad (3)$$

$$R_{12} = S_{\theta 4}(C_{\theta 1}S_{\theta 2}S_{\theta 3} - C_{\theta 1}C_{\theta 2}C_{\theta 3}) - C_{\theta 4}(C_{\theta 1}S_{\theta 2}C_{\theta 3} + C_{\theta 1}C_{\theta 2}S_{\theta 3}) \quad (4)$$

$$R_{21} = C_{\theta 4}(S_{\theta 1}C_{\theta 2}C_{\theta 3} - S_{\theta 1}S_{\theta 2}S_{\theta 3}) - S_{\theta 4}(S_{\theta 1}S_{\theta 2}C_{\theta 3} + S_{\theta 1}C_{\theta 2}S_{\theta 3}) \quad (5)$$

$$R_{22} = S_{\theta 4}(S_{\theta 1}S_{\theta 2}S_{\theta 3} - S_{\theta 1}C_{\theta 2}C_{\theta 3}) - C_{\theta 4}(S_{\theta 1}S_{\theta 2}C_{\theta 3} + S_{\theta 1}C_{\theta 2}S_{\theta 3}) \quad (6)$$

$$R_{31} = S_{\theta 4}(S_{\theta 2}S_{\theta 3} - C_{\theta 2}C_{\theta 3}) - C_{\theta 4}(S_{\theta 2}C_{\theta 3} + C_{\theta 2}S_{\theta 3}) \quad (7)$$

$$R_{32} = S_{\theta 4}(S_{\theta 2}C_{\theta 3} + C_{\theta 2}S_{\theta 3}) + C_{\theta 4}(S_{\theta 2}S_{\theta 3} - C_{\theta 2}C_{\theta 3}) \quad (8)$$

$$\begin{aligned} x = {} & l_1C_{\theta 1} + l_2C_{\theta 1}C_{\theta 2} + l_3C_{\theta 1}(C_{\theta 2}C_{\theta 3} - S_{\theta 2}S_{\theta 3}) \\ & + l_4C_{\theta 1}(C_{\theta 2}C_{\theta 3}C_{\theta 4} - S_{\theta 2}S_{\theta 3}C_{\theta 4} - S_{\theta 2}C_{\theta 3}S_{\theta 4} - C_{\theta 2}S_{\theta 3}S_{\theta 4}) \end{aligned} \quad (9)$$

$$\begin{aligned} y = {} & l_1S_{\theta 1} + l_2S_{\theta 1}C_{\theta 2} + l_3S_{\theta 1}(C_{\theta 2}C_{\theta 3} - S_{\theta 2}S_{\theta 3}) \\ & + l_4S_{\theta 1}(C_{\theta 2}C_{\theta 3}C_{\theta 4} - S_{\theta 2}S_{\theta 3}C_{\theta 4} - S_{\theta 2}C_{\theta 3}S_{\theta 4} - C_{\theta 2}S_{\theta 3}S_{\theta 4}) \end{aligned} \quad (10)$$

$$z = -l_2S_{\theta 2} - l_3(S_{\theta 2}C_{\theta 3} + C_{\theta 2}S_{\theta 3}) - l_4C_{\theta 4}(S_{\theta 2}C_{\theta 3} + C_{\theta 2}S_{\theta 3}) + l_4S_{\theta 4}(S_{\theta 2}S_{\theta 3} - C_{\theta 2}C_{\theta 3}) \quad (11)$$

3. DESIGN OF NEURAL NETWORK-BASED INVERSE KINEMATICS

For example, an original mobile robot can be designed with the leg module shown in Fig. 1. In this paper, Eq. (2) for $\theta \in \Re^4 \rightarrow \mathrm{x} \in \Re^3$ is called the forward kinematics $\mathrm{x} = fkine\ (\theta)$ and can be analytically calculated. On the contrary $\mathrm{x} \in \Re^3 \rightarrow \theta \in \Re^4$ is to be the inverse kinematics $\theta = ikine\,(\mathrm{x})$. It is not easy but complex to obtain the analytical solutions of the inverse kinematics. In this section, the mapping of inverse kinematics is tried to be acquired in the neural networks shown in Fig. 3. The number of the hidden layers and that of neurons are not important for researching the weights tuning method proposed in the next section. This means that they are out of evaluation, so that two hidden layers and 30 neurons were tentatively set. After here, the neural networks are called the NN.

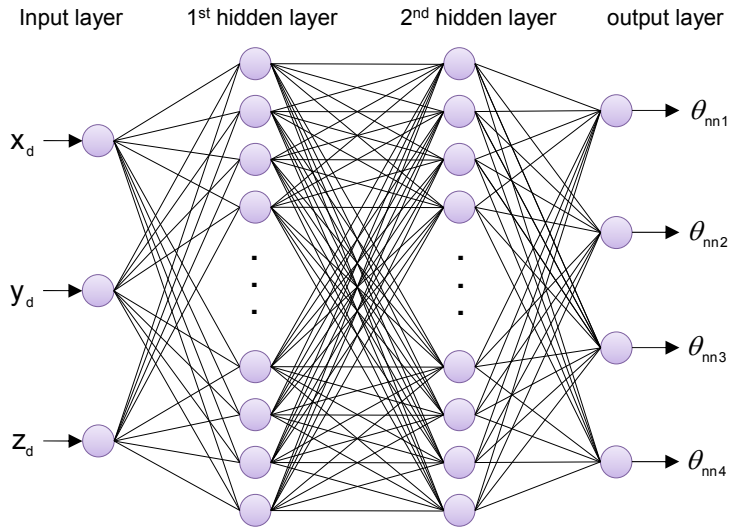

Fig. 3 Neural networks with three inputs of $\mathrm{x} \in \Re^3$ and four outputs $\theta_{nn} \in \Re^4$.

3.1 Neural Networks Leaned with Randomly Prepared Training Set

Figure 4 shows training set $\mathrm{x}_j \in \Re^3\ (1 \le \mathrm{j} \le 100)$ composed of 100 samples representing the relations $\mathrm{x}_j = fkine\ (\theta_j)$ between $\theta_j \in \Re^4$ in joint space and $\mathrm{x}_j \in \Re^3$ in Cartesian space, which are randomly generated by using the forward kinematics. The implicit relation of inverse kinematics that Fig. 4 has, is tried to be learned with the NN illustrated in Fig. 3. The NN has four layers where the input layer has three units of x_j and the output layer has four units of θ_j. The first and second hidden layers have thirty units, respectively.

After passing enough learning process using the training set, the performance of the trained NN was evaluated using the training test set along the rectangle path in Fig. 4. Figure 5 shows the result, in which markers '+' are the training test set

$x_{dj} \in \Re^3 \ (1 \le j \le 120)$ for the input layer of the NN. Also 'x' are the output of forward kinematics fkine (θ_{nnj}). Note that θ_{nnj} is the output from the NN in response to x_{dj}. It is recognized from the results that the NN learned with the training set shown in Fig. 4 performs a passable generalization for inverse kinematics but some small errors are observed.

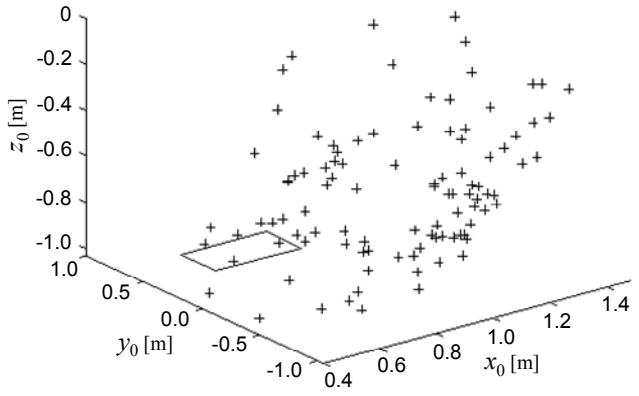

Fig. 4 Training set $x_j \in \Re^3$ representing the relation $x_j = fkine\ (\theta_j)$ between $\theta_j \in \Re^4$ in joint space and $x_j \in \Re^3$, in Cartesian space, which are prepared randomly.

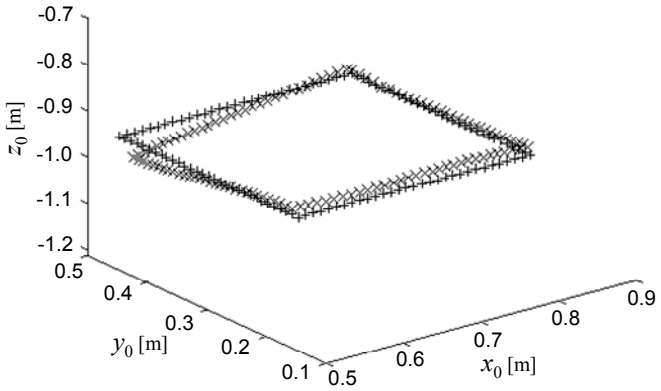

Fig. 5 Markers '+' are the training test set x_{dj} for the NN. Also 'x' are the output of forward kinematics fkine (θ_{nnj}). Note that θ_{nnj} is the output from the forward calculation of the NN when x_{dj} is given to the input layer.

3.2 Neural Networks Leaned with Regularly Prepared Training Set

In this subsection, the performance of the NN learned with regularly prepared training set (60 pairs of x_j and θ_j) is evaluated. Figure 6 shows the training set representing a path consisting of $x_j \ (1 \le j \le 60)$, i.e., the sampled number for training set is 60. The NN shown in Fig. 3 was learned with the set until the error satisfactorily converged. Figure 7 shows the performance result of the trained NN. Marker '+' in the

upper figure are training test set x_{dj} $(1 \le j \le 300)$ for the NN. Also, 'x' in the lower figure are the output of forward kinematics fkine (θ_{nnj}). Note that θ_{nnj} is the output from the forward calculation of the NN when x_{dj} $(1 \le j \le 300)$ is given to the input layer. It is recognized from the result that the NN learned with the training set shown in Fig. 6 performs a desirable generalization for inverse kinematics.

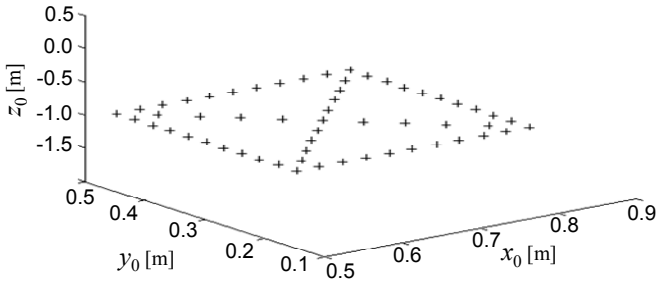

Fig. 6 Training set representing the relation between $\theta_j \in \Re^4$ and $x_j \in \Re^3$, which are prepared regularly.

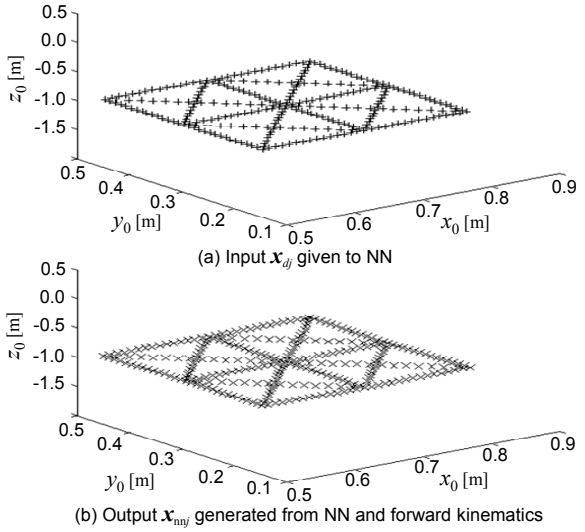

Fig. 7 Marker '+' in the upper figure are the training test set x_{dj} for NN. Also 'x' in the lower figure are the output x_j of forward kinematics fkine (θ_{nnj}). Note that θ_{nnj} is the output from the forward calculation of the NN when x_{dj} is given to the input layer.

4. EFFICIENT LEARNING ABILITY

4.1 In Case of Fundamental Learning Method

When the back propagation algorithm is used for adjusting weights in neural networks, it is serious problem that much time is required for satisfactory convergence. Here, an efficient training method is introduced by using the case shown in Fig. 6. First

of all, error E_i for criterion at the i-th learning procedure was defined by

$$E_i = \bar{e} = \frac{1}{60}\sum_{j=1}^{60} e_j \tag{12}$$

where e_j is the error in case that the j-th sample, i.e., a pair of x_j and θ_j in training set is given to input and output layers, which is obtained by

$$e_j = \sqrt{\sum_{k=1}^{4}(\theta_{jk} - \theta_{nnjk})^2} \tag{13}$$

where $\theta_j \in \Re^4$ and $\theta_{nnj} \in \Re^4$ are the training vector for the output layer and the actual output from the NN, respectively. $k(1 \le k \le 4)$ is the number of each joint. j $(1 \le j \le 60)$ is the sampled number in the training set. The learning, i.e., the tuning of weights, was continued until the following condition is satisfied.

$$E_i < E_d \tag{14}$$

where E_d is the maximum allowable error set in advance.

In the conventional learning method used in section 3.2, all samples in the training set, i.e., sixty pairs, were sequentially given for updating weights. Figure 8 shows the learning result in case of section 3.2, in which sixty times of weights updating by using a back propagation algorithm were conducted in one learning procedure on the horizontal axis. In other words, one learning procedure means one iteration using sixty pairs. Also note that one weights updating procedure consists of a pair of a forward propagation to calculate the error and a back propagation to update all weights based on the error. It was confirmed that totally 22,810 × 60 = 1,368,600 times of weights updating were conducted and the total calculation time was 8,351 seconds. In this case, the desired maximum allowable error E_d was set to 0.02. The specification of CPU was Intel(R) Core(TM) i3 560 3.33 GHz.

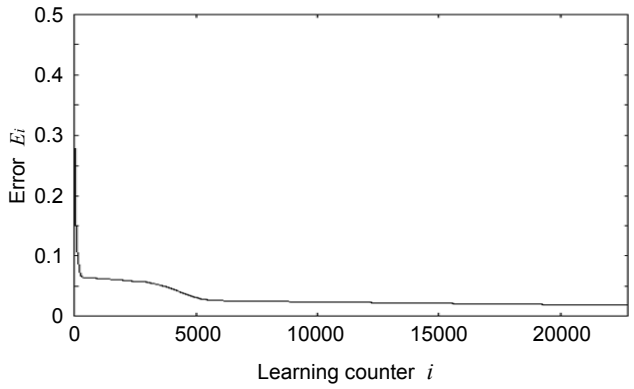

Fig. 8 An example of learning result using a conventional method, in which the desired maximum allowable error E_d is set to 0.02.

4.2 In Case of Proposed Efficient Learning Method

On the other hand, in our proposed method, a new training set consisting of pairs of input x_l and output θ_l, that have not been well trained, is extracted from the original training set. The condition of the extraction is given by

$$e_l > \bar{e} + e_{sd} \tag{15}$$

where

$$e_l = \sqrt{\sum_{k=1}^{4} (\theta_{lk} - \theta_{nnlk})^2} \tag{16}$$

$$e_{sd} = \sqrt{\frac{1}{60} \sum_{j=1}^{60} (\bar{e} - e_j)^2} \tag{17}$$

where e_{sd} is the standard deviation of e_j $(1 \le j \le 60)$. The pairs of x_l and θ_l satisfying Eq. (15) are extracted and form a new training set with less number of pairs than that of the original training set, i.e., 60. When this new training set is applied instead of the original set, the error $\tilde{E}_i$ at the i-th learning procedure, i.e., iteration, is calculated by

$$\tilde{E}_i = \frac{1}{m} \sum_{l=1}^{m} e_l \tag{18}$$

where m is the number of the pairs extracted from the original set based on Eq. (15). In our proposed method, E_i and $\tilde{E}_i$ are alternately used for the evaluation of convergence at the i-th learning procedure. Figure 9 illustrates the concept of the proposed efficient learning process, in which the original training set and the extracted one are alternately applied for 200 times of iterative learning, respectively. The vertical axis means the number of the pairs in the training set.

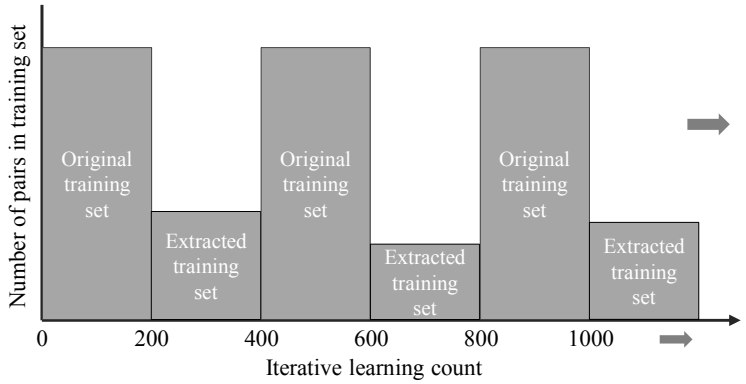

Fig. 9 Image of the proposed learning process, in which original training set and extracted one are alternately applied to the NN for 200 times of iterative learning, respectively.

Then, the proposed method was applied to the same problem explained in section 3.2 to evaluate the effectiveness. Figure 10 shows the learning result until E_i reached to 0.02, in which sequential 100 times of E_i and 1,000 times of $\tilde{E}_i$ are used alternately and repeatedly. It is observed that totally 650,220 times of weights updating were conducted and the calculation time largely decreased to 4,096 seconds.

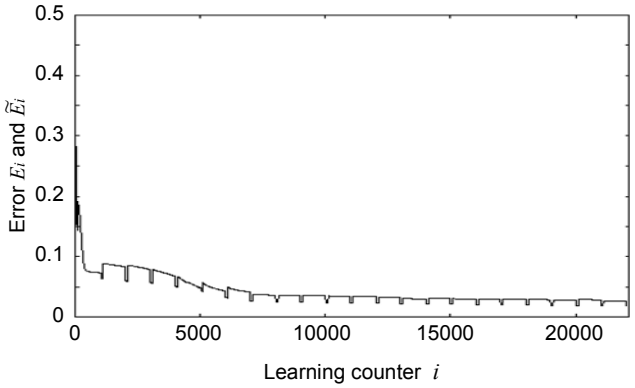

Fig. 10 Learning result using the proposed method, in which original training set and extracted one are alternately applied to the NN for 100 and 1000 times of iterative learning, respectively.

5. KINEMTIC CONTROL OF THE LEG

5.1 Jacobian of the Leg

When the tip of the leg is controlled kinematically in Cartesian space, Jacobian matrix is needed. By differentiating Eqs. (9), (10) and (11) by time, the following relation is obtained.

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = \begin{pmatrix} J_{11} & J_{12} & J_{13} & J_{14} \\ J_{21} & J_{22} & J_{23} & J_{24} \\ 0 & J_{32} & J_{33} & J_{34} \end{pmatrix} \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \end{pmatrix} \tag{19}$$

where

$$\begin{aligned} J_{11} = & -l_1 S_{\theta 1} - l_2 S_{\theta 1} S_{\theta 2} - l_3 S_{\theta 1} (C_{\theta 2} C_{\theta 3} - S_{\theta 2} S_{\theta 3}) \\ & - l_4 (S_{\theta 1} C_{\theta 2} C_{\theta 3} C_{\theta 4} - S_{\theta 1} S_{\theta 2} S_{\theta 3} C_{\theta 4} - S_{\theta 1} S_{\theta 2} C_{\theta 3} S_{\theta 4} - S_{\theta 1} C_{\theta 2} S_{\theta 3} S_{\theta 4}) \end{aligned} \tag{20}$$

$$\begin{aligned} J_{12} = & -l_2 C_{\theta 1} S_{\theta 2} - l_3 C_{\theta 1} (S_{\theta 2} C_{\theta 3} + C_{\theta 2} S_{\theta 3}) \\ & - l_4 (C_{\theta 1} S_{\theta 2} C_{\theta 3} C_{\theta 4} + C_{\theta 1} C_{\theta 2} S_{\theta 3} C_{\theta 4} + C_{\theta 1} C_{\theta 2} C_{\theta 3} S_{\theta 4} - C_{\theta 1} S_{\theta 2} S_{\theta 3} S_{\theta 4}) \end{aligned} \tag{21}$$

$$J_{13} = -l_3 C_{\theta 1}(S_{\theta 2}C_{\theta 3} + C_{\theta 2}S_{\theta 3}) - l_4(C_{\theta 1}S_{\theta 2}C_{\theta 3}C_{\theta 4} + C_{\theta 1}C_{\theta 2}S_{\theta 3}C_{\theta 4} + C_{\theta 1}C_{\theta 2}C_{\theta 3}S_{\theta 4} - C_{\theta 1}S_{\theta 2}S_{\theta 3}S_{\theta 4}) \quad (22)$$

$$J_{14} = -l_4(C_{\theta 1}S_{\theta 2}C_{\theta 3}C_{\theta 4} + C_{\theta 1}C_{\theta 2}S_{\theta 3}C_{\theta 4} + C_{\theta 1}C_{\theta 2}C_{\theta 3}S_{\theta 4} - C_{\theta 1}S_{\theta 2}S_{\theta 3}S_{\theta 4}) \quad (23)$$

$$J_{21} = l_1 C_{\theta 1} + l_2 C_{\theta 1}C_{\theta 2} + l_3 C_{\theta 1}(C_{\theta 2}C_{\theta 3} - S_{\theta 2}S_{\theta 3}) + l_4(C_{\theta 1}C_{\theta 2}C_{\theta 3}C_{\theta 4} - C_{\theta 1}S_{\theta 2}S_{\theta 3}C_{\theta 4} - C_{\theta 1}S_{\theta 2}C_{\theta 3}S_{\theta 4} - C_{\theta 1}C_{\theta 2}S_{\theta 3}S_{\theta 4}) \quad (24)$$

$$J_{22} = -l_2 S_{\theta 1}S_{\theta 2} - l_3 S_{\theta 1}(S_{\theta 2}C_{\theta 3} + C_{\theta 2}S_{\theta 3}) - l_4(S_{\theta 1}S_{\theta 2}C_{\theta 3}C_{\theta 4} + S_{\theta 1}C_{\theta 2}S_{\theta 3}C_{\theta 4} + S_{\theta 1}C_{\theta 2}C_{\theta 3}S_{\theta 4} - S_{\theta 1}S_{\theta 2}S_{\theta 3}S_{\theta 4}) \quad (25)$$

$$J_{23} = -l_3 S_{\theta 1}(S_{\theta 2}C_{\theta 3} + C_{\theta 2}S_{\theta 3}) - l_4(S_{\theta 1}S_{\theta 2}C_{\theta 3}C_{\theta 4} + S_{\theta 1}C_{\theta 2}S_{\theta 3}C_{\theta 4} + S_{\theta 1}C_{\theta 2}C_{\theta 3}S_{\theta 4} - S_{\theta 1}S_{\theta 2}S_{\theta 3}S_{\theta 4}) \quad (26)$$

$$J_{24} = -l_4(S_{\theta 1}S_{\theta 2}C_{\theta 3}C_{\theta 4} + S_{\theta 1}C_{\theta 2}S_{\theta 3}C_{\theta 4} + S_{\theta 1}C_{\theta 2}C_{\theta 3}S_{\theta 4} - S_{\theta 1}S_{\theta 2}S_{\theta 3}S_{\theta 4}) \quad (27)$$

$$J_{32} = -l_2 C_{\theta 2} - l_3(C_{\theta 2}C_{\theta 3} - S_{\theta 2}S_{\theta 3}) + l_4(S_{\theta 2}C_{\theta 3}S_{\theta 4} + C_{\theta 2}S_{\theta 3}S_{\theta 4} - C_{\theta 2}C_{\theta 3}C_{\theta 4} + S_{\theta 2}S_{\theta 3}C_{\theta 4}) \quad (28)$$

$$J_{33} = -l_3(C_{\theta 2}C_{\theta 3} - S_{\theta 2}S_{\theta 3}) + l_4(S_{\theta 2}C_{\theta 3}S_{\theta 4} + C_{\theta 2}S_{\theta 3}S_{\theta 4} - C_{\theta 2}C_{\theta 3}C_{\theta 4} + S_{\theta 2}S_{\theta 3}C_{\theta 4}) \quad (29)$$

$$J_{34} = -l_4(C_{\theta 2}C_{\theta 3}C_{\theta 4} - S_{\theta 2}S_{\theta 3}C_{\theta 4} - S_{\theta 2}C_{\theta 3}S_{\theta 4} - C_{\theta 2}S_{\theta 3}S_{\theta 4}) \quad (30)$$

Here, for example, a constraint condition is considered. The condition is that the orientation of the leg tip is fixed to the ground as given by

$$\theta_2 + \theta_3 + \theta_4 = c \ (c = \text{const.}) \quad (31)$$

so that

$$\dot{\theta}_2 + \dot{\theta}_3 + \dot{\theta}_4 = 0 \quad (32)$$

By including Eq. (32) into Eq. (19),

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ 0 \end{pmatrix} = \begin{pmatrix} J_{11} & J_{12} & J_{13} & J_{14} \\ J_{21} & J_{22} & J_{23} & J_{24} \\ 0 & J_{32} & J_{33} & J_{34} \\ 0 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \end{pmatrix} \quad (33)$$

is obtained, in which Jacobian $J(\theta)$ is defined as

$$J(\theta) = \begin{pmatrix} J_{11} & J_{12} & J_{13} & J_{14} \\ J_{21} & J_{22} & J_{23} & J_{24} \\ 0 & J_{32} & J_{33} & J_{34} \\ 0 & 1 & 1 & 1 \end{pmatrix} \tag{34}$$

In order to conduct the kinematic control, desired relative position Δx should be designed in Cartesian space and be transformed into relative angle $\Delta\theta$ in joint space with $J^{-1}(\theta)$. However, since the calculation of $J^{-1}(\theta)$ is considerably complicated, an easy method to generate $\Delta\theta$ with the NN shown in Fig. 3 is introduced.

5.2 Kinematic Control without Inverse Jacobian $J^{-1}(\theta)$

It is assumed that $\theta(k)$ and $x(k)$ are the position vectors in joint and Cartesian spaces, respectively, so that

$$x(k) = \text{fkine}\ \{\theta(k)\} \tag{35}$$

where k denotes the discrete time. The desired movement $\Delta x_d(k)$ in Cartesian space is written by

$$\Delta x_d(k) = x_d(k) - x_d(k-1) \tag{36}$$

To conduct the above motion, the following movement in joint space can be generated using the trained NN.

$$\Delta\theta_d(k) = \text{ikine}_{\_nn}\{\mathbf{x}_d(k)\} - \text{ikine}_{\_nn}\{\mathbf{x}_d(k-1)\} \tag{37}$$

where the function $\theta_{nn}(k) = \text{ikine}_{\_nn}\{\mathbf{x}_d(k)\}$ means the mapping of inverse kinematics through the NN shown in Fig. 3. Through the proposed procedure, the desired movement $\Delta\theta_d(k)$ for kinematic control can be generated without using $J^{-1}(\theta)$.

6. CONCLUSIONS

In this paper, learning of inverse kinematics using neural networks with efficient weights tuning ability has been described for a serial link structure. Generally, in making neural networks learn a relation among multi inputs and outputs, a desired training set prepared in advance is used. The training set consists of multiple pairs of an input and an output vectors. Each input vector is introduced to the input layer for forward computation and the paired output vector is compared with the yielded vector from the output layer in terns. The time required for the learning process of the neural networks depends on the number of total weights in the neural networks and that of the

input-output pairs in the training set.

This paper has introduced neural networks with efficient weights tuning ability in order to effectively learn the inverse kinematics of a leg kit with multi-DOFs. In the proposed learning process, input-output pairs, which have the worse errors in learning process, are extracted from the original training set and form a new temporary training set. From the following iteration of learning, the temporary training set is applied instead of the original one. That means only pairs with worse errors are used for updating weights until the mean value of errors decrease to a desired level. Once the learning process is conducted using the temporary training set, the original training set is applied again instead of the temporary set. By repeatedly using these two training sets alternately, the convergence time could be efficiently reduced. The effectiveness was proved through simulation experiments using a kinematic model of a leg with four-DOFs.

Additionally, when the tip of the leg with a serial link structure is controlled kinematically in Cartesian space, an inverse Jacobian matrix is needed to generate velocities in joint space. However, the calculation of the inverse Jacobian is not easy. Hence, neural networks based inverse kinematics was introduced to overcome the complexity.

REFERENCES

Duka, A.V. (2014), “Neural network based inverse kinematics solution for trajectory tracking of a robotic arm,” *Procedia Technology*, **12**, 20–27.

Hoang, G, Min, J.H., Lee, G.M., Jun, B.H., Kim H.K. and Kim, S.B. (2014), “Omni-directional walking control for a six-legged robot using differential kinematics algorithm,” *Procs. of 2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)*, 1163–1168.

Maeda, Y, Fujiwara, T. and Ito, H. (2014), “Robot Control Using High Dimensional Neural Networks,” *Procs. of SICE Annual Conference 2014 -International Conference on Instrumentation, Control, Information Technology and System Integration-*, 738–743.

Nagata, F., Otsuka, A., Sakakibara, K., Watanabe, K., Habib, M.K. (2013), “Experiment Systems Using Three Types of Motors for Biomimetic Machine Research,” *Procs. of SICE Annual Conference 2013 -International Conference on Instrumentation, Control, Information Technology and System Integration-*, 2711–2717.

Tejomurtula, S. and Kak, S. (1999), “Inverse kinematics in robotics using neural networks,” *Information Sciences*, **116**, 147–164.