International Journal of Fuzzy Logic and Intelligent Systems, vol. 8, no. 3, September 2008 pp. 185-191

Experimental Studies of Real-Time Decentralized Neural Network Control for an X-Y Table Robot

Hyun Taek Cho, Sung Su Kim and Seul Jung*

Intelligent Systems and Emotional Engineering(ISEE) Lab, BK21 Mechatronics Group
Chungnam National University, Daejeon, Korea 305-764

Abstract

In this paper, experimental studies of a neural network (NN) control technique for non-model based position control of the x-y table robot are presented. Decentralized neural networks are used to control each axis of the x-y table robot separately. For an each neural network compensator, an inverse control technique is used. The neural network control technique called the reference compensation technique (RCT) is conceptually different from the existing neural controllers in that the NN controller compensates for uncertainties in the dynamical system by modifying desired trajectories. The back-propagation learning algorithm is developed in a real time DSP board for on-line learning. Practical real time position control experiments are conducted on the x-y table robot. Experimental results of using neural networks show more excellent position tracking than that of when PD controllers are used only.

Key words : x-y table, neural network controller, RCT

1. Introduction

Recently, intelligence becomes one of the key issues in the controller design of dynamical systems. Two main intelligent tools are known as neural network and fuzzy logic.

Specially, neural networks have been used in many applications such as pattern recognition of images and voices, control of nonlinear systems, prediction of forecast and stock markets, and so on. Specially, in the nonlinear system control area, neural networks have become one of the powerful nonlinear controllers that perform quite well to satisfy given specifications. Their powerful characteristics such as a nonlinear mapping capability, adaptation and learning capabilities are very useful for controlling complicated systems which are very difficult to model and control the dynamics of those due to their highly nonlinear behaviors. Their ultimate goal is to minimize the position tracking errors by compensating for uncertainties caused from unknown system dynamics.

Another important aspect in neural network control is an on-line control capability. On-line control when learning and control happens at the same time is one of many important issues within the learning control system area. One simple learning structure is the direct inverse control [1,2]. Although this scheme is simple, it is very sensitive to the stability. As a modified scheme, the pre-filter type neural network structure has been proposed[3]. To have the on-line neural network learning control capability, the back-propagation algorithm should be satisfied with obtaining the proper gradient functions. The difficulty of driving gradient functions yields a variety of off-line learning control schemes [2].

In this framework, a few on-line learning algorithms for neural network control systems have been proposed. One famous learning algorithm is the feedback-error learning structure proposed by [4]. It is an inverse control structure that identifies the inverse of the system dynamics at the end of a learning process. Another kind of on-line learning schemes is to tune PID controller gains adaptively. When the system dynamics are time-varying, pre-determined controllers' gains result in poor performances in tracking [5,6].

The controller gains should be modified appropriately with respect to system variations. Ultimately, they have the same goals to minimize errors, but the objective functions are set differently and the corresponding back-propagation algorithms are differently derived. Some theoretical foundations of neural network control have been analyzed based on the Lyapunov stability of position and weight error bounds with respect to bounded uncertainties, and obtained good results in some convergence ranges [7-11].

Here we are using a similar learning structure to the feedback error learning structure, but the difference comes when the compensation happens at the different location [4]. The presented inverse control technique called the reference compensation technique (RCT) is conceptually different from the existing inverse controllers. A neural network does control the inverse of the plant dynamics by doing that the NN controller compensates for uncertainties in the dynamical system

Manuscript received Jan. 17, 2008; revised Jun. 10, 2008
*Corresponding author
This research was financially supported by the Ministry of Education, Science Technology (MEST) and Korea Science and Engineering Foundation(KOSEF).

by modifying desired trajectories. This provides us with a great advantage of handling the plant that is controlled by prefixed controller gains resulting in poor performances due to system dynamic changes or other nonlinear effects. Without modifying the predetermined controller gains, the neural network can compensate for uncertainties by closing another outer loop.

As an extension of our previous researches on the neural network control, the decentralized neural network control scheme is newly proposed [12, 13]. Focusing on experimental studies, tracking performances between PD controllers and neural networks are compared. Different from multi-joint robot manipulators, the x-y table robot has the structural characteristics of which x and y axes are very much decoupled. Assuming that the x-y table robot moves slowly, coupling effects of two axes are further minimized and the control structure can be decoupled. However, although the x-y table robot is a likely decoupled system, coupled nonlinear uncertainties are still present. To solve this problem, each axis is controlled separately with a separate neural network controller. To study an on-line control of neural networks, the cost effective x-y table robot was built as a test-bed. Since the cost of building the x-y table is quite effective, the accuracy of position control depends on the structure of the robot and the resolution of positional sensors such as optical encoders.

To achieve on-line learning, a faster calculation of a neural network learning algorithm is required. The DSP board is used for a calculation of the back-propagation algorithm. Interfacing hardware between the DSP system and motor drivers for the x-y table robot was implemented.

2. PD Control of x-y Table Robot

The dynamic equation of an x-y table robot in the Cartesian space coordinates is given by

$$D(z)\ddot{z}+C(z,\dot{z})\dot{z}+F_f(\dot{z})=F \tag{1}$$

where the vectors $z,\dot{z},\ddot{z}$ are the 2 x 1 displacement such that $z=[xy]^T$, velocity, and acceleration of an each axis of the x-y table robot, respectively; $D(z)$ is a 2 x 2 symmetric positive definite inertia matrix; $C(z,\dot{z})\dot{z}$ is a 2 x 1 vector of Coriolis and centrifugal torques and F is a 2 x 1 vector of actuator torques; $F_f(\dot{z})$ is a 2 x 1 vector representing Coulomb friction and viscous friction forces : $F_f(\dot{z})=k_1\,\mathrm{sgn}(\dot{z})+k_2\dot{z}$.

In most practical cases, since it is difficult to know the robot dynamic model exactly, it is advised to implement non-model based control schemes. The PD control compensation for a robot dynamic control has been studied in many papers because of its simplicity and the guaranteed stability [14]. The desired PD control law is composed of position and velocity displacements such as

$$F=K_D\dot{e}+K_Pe, \tag{2}$$

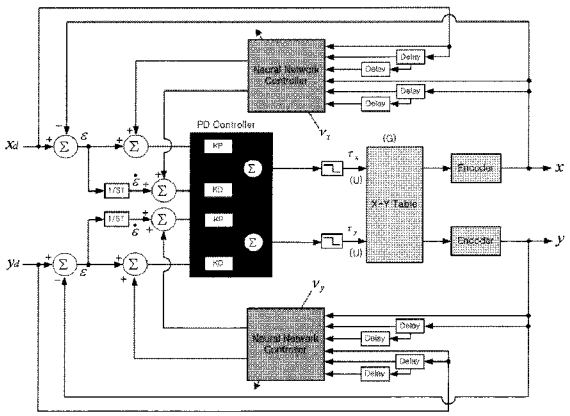


Fig. 1. Control Block Diagram

where $\dot{e}=\dot{z}_d-\dot{z}$, $e=z_d-z$, and K_D,K_P are PD controller gains.

Combining equation (1) and (2), we have a closed loop error equation as follows

$$D\ddot{e}+K_D\dot{e}+K_Pe=D\ddot{z}_d+C\dot{z}+F_f. \tag{3}$$

Multiplying both sides by D^{-1} yields the second order error equation as

$$\ddot{e}+D^{-1}K_D\dot{e}+D^{-1}K_Pe=D^{-1}(D\ddot{z}_d+C\dot{z}+F_f). \tag{4}$$

Clearly, it is easy to control the x-y table robot since the inertia matrix D can be approximately a diagonal matrix. On the left side of equation (4), an independent axis control can be applied. With the help of high feedback gains, we can achieve a better tracking performance as long as the closed loop system is stable since coupling effects due to the other axis motion are minimal compared with those of other robot manipulators [15]. When the x-y table robot moves slowly, the effects of Coriolis and centrifugal forces become also minimal. Then an x axis control equation becomes a SISO system and can be decoupled as follows:

$$\ddot{e}_x+\frac{k_{dx}}{d_x}\dot{e}_x+\frac{k_{px}}{d_x}e_x=\frac{1}{d_x}(d_x\ddot{x}_d+f_{fx}), \tag{5}$$

where d_x is an x axis inertia component of D and f_{fx} is the friction component in x axis. In the similar way, a y axis control equation can be represented as a SISO system.

However, in the practical system, there should be a saturation limit of motor torque for protecting motors from being driven with a large load torque. Therefore, the controller gains are practically limited. In addition, nonlinear uncertainties such as frictions, backlashes, and belt tension are present to effect system performances. So, the proposed idea is to use a neural network as an auxiliary controller to improve the control

performance of a standard PD controller by canceling out those uncertainties.

3. Neural Network Compensation Technique

A robot control scheme is presented here as depicted in Figure 1. The basic control concept of this scheme is that the NN controller acts as an inverse of the robot dynamics. Compensation is done at the trajectory level and those compensating signals are amplified through PD controller gains. If we have a PD controller given in (2), neural network outputs are added to the reference trajectory $z_d, \dot{z}_d$.

We have the following control law:

$$F = K_D(\dot{e} + \Phi_d) + K_P(e + \Phi_p), \tag{6}$$

where Φ_d, Φ_p are 2 x 1 output vectors of two distinguished neural networks. Combining (1) and (6) yields

$$K_D\dot{e} + K_P e = D\ddot{z} + C\dot{z} + F_f - (K_D\Phi_d + K_P\Phi_p). \tag{7}$$

Denoting $\Delta = D\ddot{z} + C\dot{z} + F_f$ and $\Psi = K_D\Phi_d + K_P\Phi_p$ yields the closed loop dynamic equations as follows:

$$K_D\dot{e} + K_P e = \Delta - \Psi. \tag{8}$$

At convergence, when performance specifications are satisfied, the ideal outputs of neural networks become compensating signals of uncertain terms.

$$\Delta \cong \Psi \tag{9}$$

So, we can achieve that equation (8) becomes zero. This means that the neural network identifies the inverse dynamics of the system.

For an x axis control, since we implemented the system based on the assumption that two axes are separately controllable, neural network outputs become at the convergence

$$k_{dx}\phi_{dx} + k_{px}\phi_{px} \cong d_x\ddot{x} + c\dot{x} + f_{fx}, \tag{10}$$

where $\phi_{dx}\phi_{px}$ are neural network outputs of an x axis.

For y axis, it can be represented in a similar way.

4. Learning Algorithms

Our goal is on-line learning and control to minimize positional errors of the dynamical system, specially the x-y table robot within a certain range of accuracies without a pre-learning step. To develop the back-propagation algorithm, there are two methods to be considered: off-line learning and on-line learning. These methods result from how to form the objective functions to be minimized. The ultimate goal of both methods is same, however their implementations are different.

Here we implement the on-line learning algorithm based on the modified system error. Careful selection of a training signal is required to satisfy the gradient function in the back-propagation algorithm.

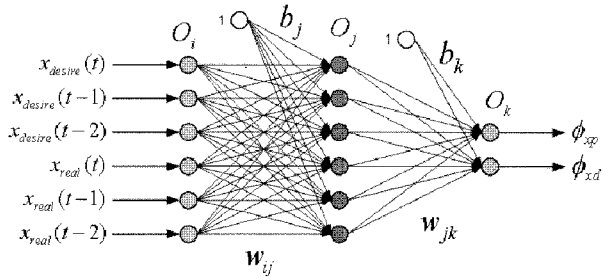


Fig. 2 Neural Network Structure for an X Axis

Since the x-y table system is a likely decoupled system, it is better to use two neural networks separately for x and y axis. A centralized neural network has been used for highly nonlinear robot manipulators [16]. In the paper [16], effects of number of hidden units, learning rate, and momentum coefficient have been investigated by extensive simulation studies. Another reason of using decoupled neural network structure is to reduce the number of weights in neural network. For a centralized neural network, the number of inputs becomes double. This yields the increment of the number of hidden units and total number of weights.

The multi-layer feed-forward neural network (FFNN) and the back-propagation(BP) updating algorithm are used. They are composed of an input buffer, a non-linear hidden layer and an output layer as shown in Figure 2. Inputs are selected as delays of desired trajectories and actual trajectories not only to give dynamic characteristics to neural networks, but also to prevent neural networks from generating abruptly changed signals at the output nodes. The nonlinear functions used for hidden units and output units are *tanh(x)* function that is bounded by ± 1. The weight updating law minimizes the objective function $E(e,\dot{e})$ for each axis which is a function of position tracking errors. Defining $v_x = k_{dx}\dot{e}_x + k_{px}e_x$ in (8) as a training signal of the x axis, the neural network controller simplifies (8) as follows:

$$v_x = \delta_x - \phi_x, \tag{11}$$

where δ_x is an x axis dynamic element of the dynamic equation of the x-y table, Δ in (8) and $\phi_x = k_{dx}\phi_{dx} + k_{px}\phi_{px}$. In the similar way, for the y axis

$$v_y = \delta_y - \phi_y, \tag{12}$$

where δ_y is a y axis dynamic element of the dynamic equation of the x-y table, Δ in (8) and $\phi_y = k_{dy}\phi_{dy} + k_{py}\phi_{py}$. Namely, the

unknown dynamic equation of the x-y table, Δ can be decoupled into unknown dynamic equations of δ_x and δ_y.

The weight updating law minimizes the objective function E which is a quadratic function of the training signal v where v is a control input from a PD controller for each axis.

The objective function for both neural networks is defined as

$$E = \frac{1}{2}v^2, \tag{13}$$

where v is either v_x or v_y. Differentiating the objective function with respect to the weight w and applying a chain rule yields the gradient of E as follows:

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial v}\frac{\partial v}{\partial w} = v\frac{\partial v}{\partial w}. \tag{14}$$

A problem here is to find the partial derivative of $\frac{\partial v}{\partial w}$. Since $\frac{\partial v}{\partial w} = -\frac{\partial \phi}{\partial w}$ making use of (11) or (12) the gradient becomes

$$\frac{\partial E}{\partial w} = -v\frac{\partial \phi}{\partial w} = -v[k_d\frac{\partial \phi_d}{\partial w} + k_p\frac{\partial \phi_p}{\partial w}], \tag{15}$$

where ϕ_d and ϕ_p are outputs of neural networks. The gradient $\frac{\partial \phi_d}{\partial w}, \frac{\partial \phi_p}{\partial w}$ can be obtained easily. The BP update rule for the weights with a momentum term is given as

$$\Delta w(t) = -\eta v\frac{\partial \phi}{\partial w} + \alpha \Delta w(t-1). \tag{16}$$

The stability analysis has been well organized in the paper [7,8]

5. Experimental Studies

5.1 Experimental setups

The experimental setup for the x-y table robot system is shown in Figure 3. The overall system setup is composed of three parts: an x-y table robot, the DSP system, and an interface PC. Each axis is actuated by a 5.2:1 reduction geared type DC motor to generate a faster torque. The DC motors are driven by their own drivers commanded from the DSP board mounted in the PC. The DSP board is used to compute the neural network learning algorithm in real time for two decentralized neural compensators, which require a huge calculation time. The overall sampling time is about 1 KHz. Sensing encoder signals, calculating position errors, processing the back propagation algorithm, and generating torques can be done within one sample time.

The torque from DC motors is transferred through timing belts. Since the x axis is placed on the y axis with a guide by a rail, the movement in the y axis direction is effected by the whole x axis frame movement. Backlash due to loose tension of the timing belt and imbalance caused from an actuation by one side motor are considered as unknown uncertainties.

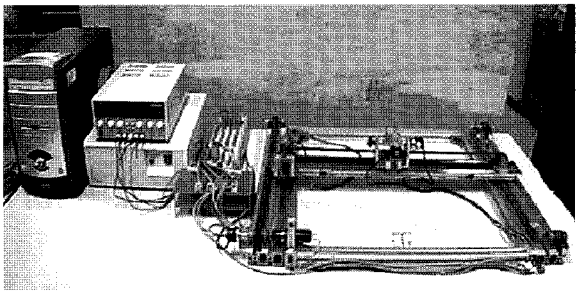


Fig. 3. Experimental Setups

5.2 Experimental results

For the experiment, the x-y table is commanded to move the circular trajectory with a 0.25m diameter. In order for the robot to complete a circular trajectory, it takes about 6.5 seconds. First, a tracking control performance with a PD controller only is tested. The PD controller gains are selected as $K_P = 5I$ for a low gain for both axes and $k_{dx} = 1$ for the x axis, $k_{dy} = 2$ for the y axis. The reason for different setting of controller gains is that the y axis is heavier than the x axis. The position tracking errors are clearly shown in Figure 4. The circular tracking offset error is due to less torques generated from the low PD controller. And the irregular shape of circle tracking is due to other dynamical effects.

Next, the neural controller with a pre-specified PD gain controller is used to draw the same circular trajectory. With the help of a fast computing power of the DSP system, on line learning is possible without a pre-learning step. For the neural network controllers, the learning rates are optimized as 0.005 for low gains. The neural network has a 6-6-2 two layered structure. As shown in Figure 2, it has 6 inputs, 6 hidden units, and 2 output units.

Positional delayed data are used as inputs. The number of hidden units is optimized by trial and error process. We did not see any difference in performance, even worse when more than 6 hidden units are used. The effects of the number of hidden units, initial weight values, and input data type of neural network on the position tracking performance of the robot manipulators have been reported in [16]. The momentum terms are selected as 0.3 for both neural networks in all cases.

The tracking result of using neural networks shown in Figure 5 is excellent compared with that of Figure 4. The plot shown in Figure 5 is after the convergence of 2 trials of repeating the same circular trajectory. The corresponding tracking plot for each axis is shown in Figure 6. Since the plot scale is somewhat big, tracking errors cannot be distinguished from the figures.

One of neural compensating signals along with outputs of PD controllers is plotted in Figure 7. Those controllers' output

signals are captured just before they are fed into PWM motor drivers from the DSP board. The outputs ϕ_{px}, ϕ_{py} from neural networks are plotted which compensate at positional trajectories. The neural network output signals are bounded by ± 1. The output of the y axis neural network control shows more oscillatory behavior than that of the x axis neural network controller. This is because the movement of y axis has to control x axis as well. Another interesting point is that PD controller outputs are immediately minimized as neural network outputs become larger. This is because neural networks take over the control and become dominant in a very short time. The interesting point is that the shape of NN compensating signals in Figure 7 is similar to the desired sinusoidal function.

Another experiment of tracking the letter N trajectory as shown in Figure 8 has been conducted. The overall traveling time for writing the letter N takes 20 seconds. In this case, quite high P gain just before motor saturation condition is used. The proportional gain values are 600 and 700 for x axis and y axis, respectively. The derivative gains 0.6 and 0.8 for x axis and y axis, respectively, which is relatively small. Tracking performances of PD controllers for x and y axis are shown in Figures 9 and 10, respectively. Even though tracking errors are quite small compared with those of when low PD gains are used, we see that offset errors in x and y axis tracking. The offset error of y axis tracking is relatively larger. The performances of using neural network controllers are shown in Figures 11 and 12. We clearly see that tracking errors are further minimized to zero. Overshoots are observed at turning points in both cases.

During the experiments, we have found that there is an accuracy limit of sensing data from optical encoders. Optical encoders are mounted on the rotating axis, not at the motor axis directly. This yields a less resolution resulting in small tracking errors.

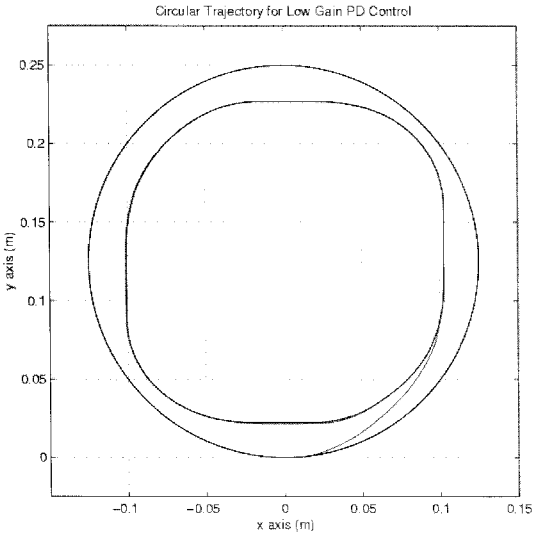


Fig. 4 Circular Trajectory Tracking of a Low PD Controller only

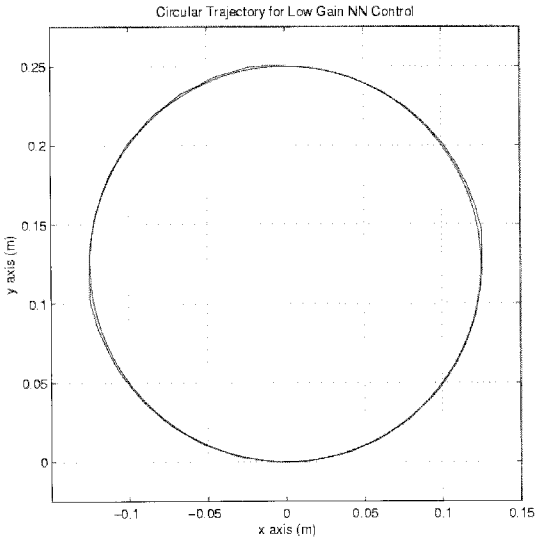


Fig. 5 Circular Trajectory Tracking of a Low PD Controller with NN

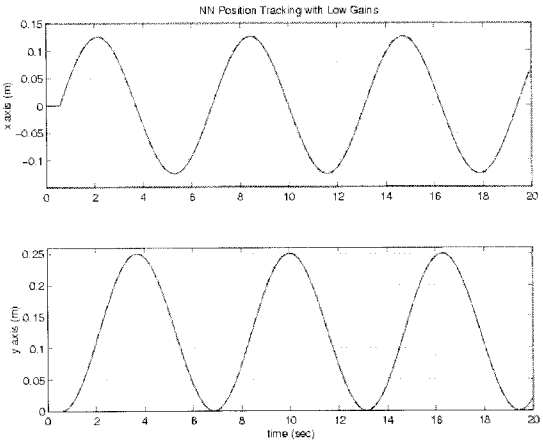


Fig. 6. NN Position Tracking of Each Axis with Low PD Controller Gains

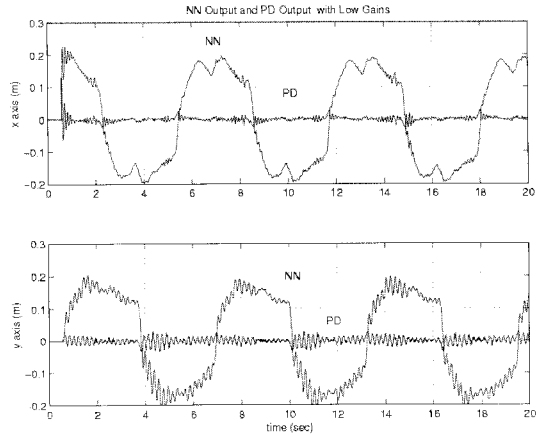


Fig. 7 NN Compensating Outputs for Low PD Gains

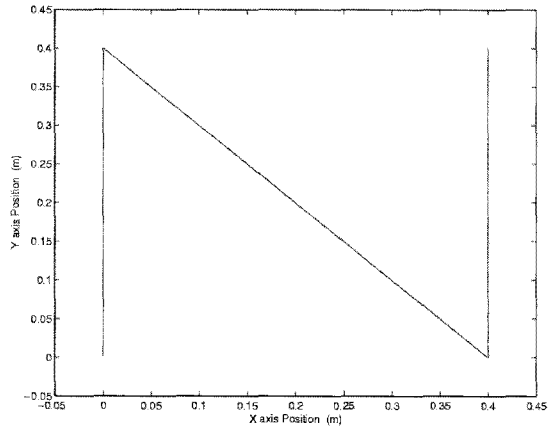


Fig. 8. Position Tracking of a letter N with High NN Controller

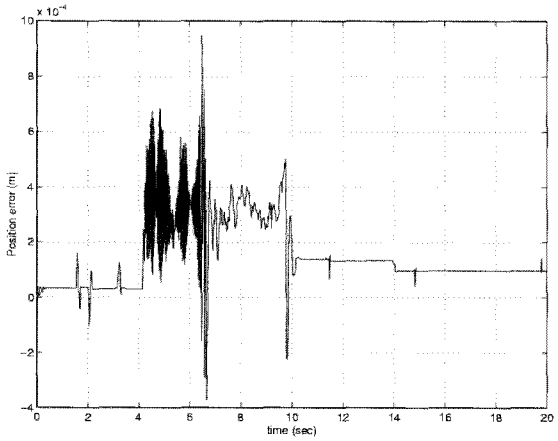


Fig. 9. X axis Position Tracking Errors with High PD Controller Gains

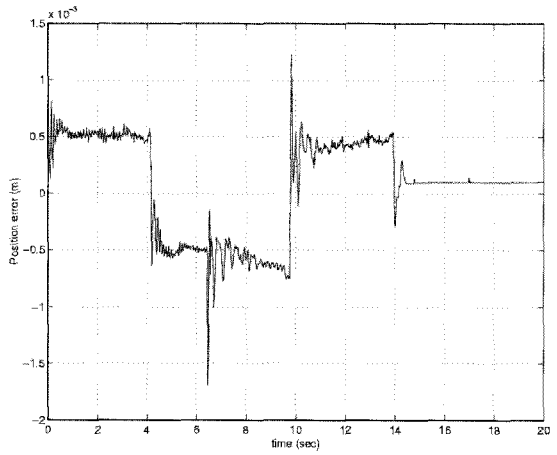


Fig. 10 Y axis Position Tracking Errors with High PD Controller Gains

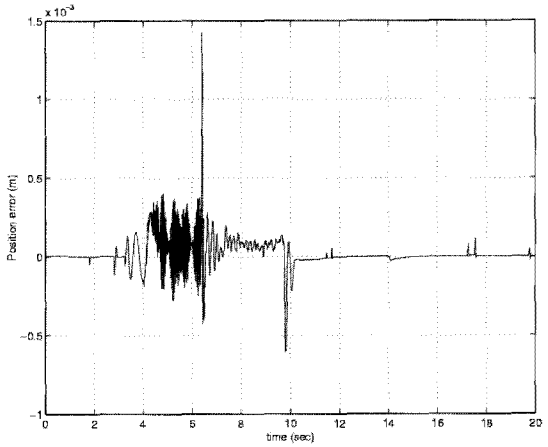


Fig. 11. X axis Position Tracking Errors with an NN Controller

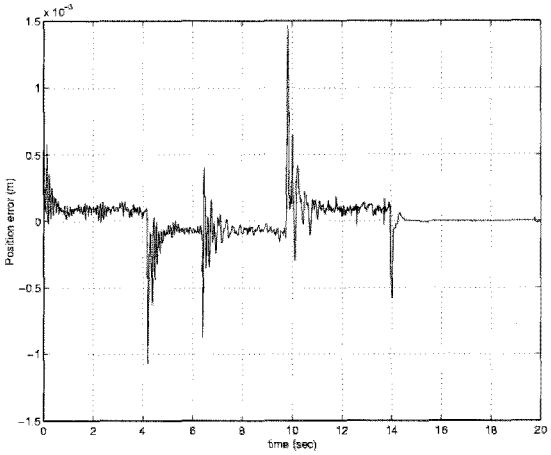


Fig. 12 Y axis Position Tracking Errors with an NN Controller

7. Conclusion

Experimental studies of the neural network control technique for a non-model based x-y table robot control are presented. Due to the decoupled structure of the x-y table robot, decentralized neural networks are used to compensate for uncertainties in system dynamics. Experimental studies on the circular trajectory tracking of the x-y table robot showed that neural network controller works extremely well compared with those of PD controllers only. We have increased P gains just before the saturation to have the better performances of PD controllers. Experimental results also confirm that the RCT as a neural network control algorithm works quite well regardless of values of the controller gains.

These results show us that the extension of the RCT algorithm application to other PD or PID controlled non-model based systems can be possible.

References

[1] W. T. Miller, R. S. Sutton, and P. J. Werbos, "*Neural networks for Control*", The MIT Press, 1991

[2] K. Narendra and K. Parthasarathy, "Control of nonlinear dynamical systems using neural networks: controllability and stabilization," *IEEE Trans. on Neural Networks*, vol. 4, no. 2, pp. 192-206, 1993

[3] S. Jung and T. C. Hsia, "A new neural network control technique for robot manipulators", *Robotica*, Vol. 13, pp. 477-484, 1995

[4] H. Gomi and M. Kawato, "Learning control for a closed loop system using feedback error learning," Proc. of the *IEEE International Conf. on Decision and Control*, pp. 3289-3294, 1990

[5] S. Omatu and Y. Kishida and M. Yoshioka, "Neuro-control for single-input multi-output systems" *IEEE Conf. on Knowledge Based Intelligent Electronics Systems*, pp. 202-205, 1998.

[6] S. Omatu and T. Fujinaka and M. Yoshioka, "Neuro-pid control for inverted single and double pendulums" *IEEE Conference on Systems, Man, and Cybernetics*, pp. 2685-2690, 2000.

[7] F. L. Lewis, K. Liu, and A. Yesildirek, "Neural net robot controller with guaranteed tracking performance", *IEEE Symposium on Intelligent Control*, pp. 225-231, 1993

[8] F. L. Lewis, S. Jagannathau, and A. Yesildirek, "*Neural network control of robot manipulators and nonlinear systems*," Taylor and Francis, 1999

[9] J. Li and D. Wang, "An NN controller and tracking error bound for robotic manipulators," *IEEE Proc. of Decision and Control*, pp. 872-876, 2000

[10] Y. C. Chang, "Neural network based h tracking control for robotic systems," *IEE Proc. On Control Theory Applications*, vol. 147, no. 3, pp. 303-311, 2000.

[11] S. S. Ge and C. Wang, "Direct adaptive nn control of a class of nonlinear systems," *IEEE Trans. on Neural Networks*, vol. 13, no. 1, pp. 214-221, 2002

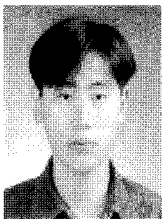
[12] S. Jung and T. C. Hsia, "On reference trajectory modification approach for Cartesian space neural network control of robot manipulators", *Proc. of IEEE Conf. on Robotics and Automations*, pp. 575-580, 1995

[13] S. Jung and T. C. Hsia, "Neural Network Inverse Control Techniques for PD Controlled Robot Manipulators" *Robotica*, pp. 461-455, Vol. 19, No. 3, 2000

[14] T. C. Hsia, "Robustness analysis of pd controller with approximate gravity compensation for robot manipulator control." *Journal of Robotic System*, vol. 11, pp.517-521, 1994

[15] T. H. Lee and S. S. Ge, "Intelligent control of mechatronics systems," *Proc. of IEEE Symposium on Intelligent Control*, pp. 646-660, 2003

[16] S. Jung and T. C. Hsia, "A study of neural network control of robot manipulators", *Robotica*, Vol. 14, pp. 7-15, 1996



Hyun-Taek Cho

Hyun-Taek Cho received his B.S. and M.S. degrees in Mechatronics Engineering from Chungnam National University, in 2000 and 2003, respectively. He is now a researcher in the Sunwoo I. D company at Daejeon. His research interests involve neural network controllers and DSP systems.

Phone :+82-42-821-7232
Fax :+82-42-823-4919
E-mail : hyuntc@empal.com

Sung-su Kim received his B.S. degree in Control and Measurement Engineering from The Kyung-Il University in 2001 and his M.S degree in Mechatronics Engineering from Chungnam National University in 2004. He is presently working at the Agency for Defense Development. His interests include S.O.C design, DSP applications, and robotics.

Phone : +82-42-821-7232
Fax : +82-42-823-4919
E-mail : gubiki780616@hanmail.net

Seul Jung

He received his B.S. degree in Electrical & Computer Engineering from Wayne State University in 1988, and his M.S. and Ph.D. degrees in Electrical & Computer Engineering from the University of California, Davis in 1991 and 1996, respectively. After working at the Advanced Highway Maintenance and Construction Technology Center, he joined the Department of Mechatronics Engineering, Chungnam National University in 1997, where he is presently a professor. His research interests include intelligent systems, hardware implementation of intelligent controllers, and intelligent robotic systems. He is a member of Tau Beta Pi and Eta Kappa Nu.

Phone : +8242-821-6876
Fax : +82-42-823-4919
E-mail : jungs@cnu.ac.kr