Proceedings of the 12th IEEE
International Symposium on Intelligent Control
16-18 July 1997, Istanbul, Turkey

Telemanipulator Neurocontrol Using Multiple RBF Networks

Spyros G. Tzafestas (*), *Platon A. Prokopiou* (*)
and Costas S. Tzafestas (**)

(*) Intelligent, Robotics and Automation Laboratory,
Department of Electrical and Computer Engineering, National Technical University of Athens,
Zografou, 15773, Athens, Greece.
E-mail: tzafesta@softlab.ece.ntua.gr

(**) Laboratoire de Robotique de Paris, CNRS-UPMC-UVSQ,
10-12 Av. de l' Europe, 78140 Velizy, France.

Abstract

This paper addresses the control problem of master-slave systems which involve severe modeling errors and other high - level uncertainties, using Neural Networks. The solution approach is based on a recent teleoperator control scheme (S. Lee and H. S. Lee), which is suitably enhanced such that to become capable of compensating the uncertainties. The class of radial-basis functions (RBF) neural networks are employed in a multipartitioned neural network architecture, and a special learning scheme is adopted which distributes the learning error to each subnetwork and allows on-line learning. The effectiveness of the present RBF neurocontroller was investigated through extensive simulation and compared to that of MLP (multi-layer perceptron) neurocontroller and a robust sliding-mode controller representative.

1. Introduction

Neural networks (NNs) possess a remarkable ability to identify and control strongly nonlinear multivariable plants, with minimum need for a priori knowledge, through an adaptive, compact and fast system, and therefore can provide valuable solutions in the field of nonlinear control systems. In Robotics they are exploited in a great variety of tasks, including path planning, control, visual image and sensor data recognition. Being among the most complex robotic systems, telemanipulators can take advantage of NNs research in many ways. The methods developed for single manipulators can be applied locally to the master and slave. In addition, they can assist in the coordination between the human operator and the machine (e.g. visual representation, force feedback redefinition, incorporation of human and environmental dynamics in the control loop). Certain autonomous functions can also be assigned to NNs. However, up to date the literature on teleoperators focuses on conventional techniques. Current research topics include the incorporation of autonomous functions and the design of schemes robust to time delay in the communication channel between the master and slave robots. Applications of NNs on this field are rare, and include [1],[11].

In this paper, NNs are exploited at the control level. Our work is based on the teleoperator architecture previously proposed by S. Lee and H. S. Lee [5]. In this, the traditional concept of *telepresence*, according to which the *exact* position and force sensed at the slave side is fed back to the operator, so that he feels as if he is "physically present" at the slave workspace, is abandoned. In our contribution, their scheme is enhanced in order to be able to compensate modeling errors. These can be caused by driving the slave to pick up an object of unknown mass and shape (e.g. tool or sample) in order to perform a task with it, or due to accidental deformation of the last link(s), unidentified nonlinear terms or terms deliberately omitted in the model used in the controller to simplify the design. On the master side modeling errors are not likely to be large, since the master operates in a safe and well known environment. However, in certain advanced applications (e.g. telesurgery) uncertainty may arise by providing the possibility to change the master's end effector's shape and feel, so as to best suit the task and the operator.

In the initial work of [5] the role of modeling errors was not considered. In [6] the same authors proposed an improved control scheme for applications with significant time delay. The new controller was reported to perform well also under 5% - 10% errors in the slave model. However, this is obtained as a side effect, without being an explicit specification of the design from the start. At short time delays the new version coincides with the previous one. The teleoperator design presented in the following: a) ensures proper functioning under disturbances, computational delay and significant modeling errors, b) enhances the original scheme so that it can be used in more complex tasks, c) improves the ability of the system to compensate control errors and the fatigue of the operator, d) is an essential step towards the practical implementation of the original scheme.

In our previous contributions sliding mode robust controllers [9], [10], as a representative of classical control schemes, and Multi Layer Perceptron (MLP) NNs [11] were used for the same purpose. Here results obtained with Radial Basis Function (RBF) NNs are reported. Moreover, a novel heuristic learning algorithm (HERD) is designed, suitable for use with a partitioned NN, which identifies the inverse robot dynamics. This algorithm tries to distribute the total error to each subnet so that the subnets preserve their original role, even though measurable data is very poor in useful quantitative information. The subnets are trained with conventional algorithms.

The paper is organized as follows. Some basic elements of the architecture of S. Lee and H. S. Lee are outlined in Section 2. The neural controller and the error distributing algorithm are introduced in Section 3, and incorporated in the teleoperator in Section 4. The resulting system is tested through simulations and compared with a sliding mode robust controller in Section 5. Section 6 contains the conclusions.

2. Original control scheme

Rigid-link manipulators of n degrees of freedom are considered. Their dynamics are described by (see e.g.[8]):

$$\boldsymbol{D}(\boldsymbol{q})\ddot{\boldsymbol{q}}+\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}+\boldsymbol{g}(\boldsymbol{q})=\tau_a-\boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{f}_e \qquad (1)$$

where $\boldsymbol{q}$ is the vector of joint angles, $\boldsymbol{D}(\boldsymbol{q})$ the inertia matrix, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ the vector of Coriolis and centrifugal forces, $\boldsymbol{g}(\boldsymbol{q})$ is related to gravity, τ_a are the driving forces, $\boldsymbol{f}_e$ the forces acting at the end effector from the environment, and $\boldsymbol{J}^T(\boldsymbol{q})$ the Jacobian matrix.

The whole control architecture of S. Lee and H. S. Lee is designed in the Cartecian space. Aiming to provide compliant force control, a previous extension of Impedance Control proposed by the authors is utilized. The desired manipulator impedance is defined through a set of differential equations, termed the *Generalized Impedance* (GI). For the master and slave respectively it is specified as:

$$\boldsymbol{f}_{ref}+\boldsymbol{f}_{em}=\boldsymbol{M}_{dm}\ddot{\boldsymbol{x}}_m+\boldsymbol{B}_{dm}\dot{\boldsymbol{x}}_m \qquad (2)$$

$$\boldsymbol{M}_{ds}\ddot{\boldsymbol{x}}_s+\boldsymbol{B}_{ds}(\dot{\boldsymbol{x}}_s-\dot{\boldsymbol{x}}_{ds})+\boldsymbol{K}_{ds}(\boldsymbol{x}_s-\boldsymbol{x}_{ds})=\boldsymbol{B}_{fs}\dot{\boldsymbol{f}}_{es}+\boldsymbol{K}_{fs}\boldsymbol{f}_{es} \quad (3)$$

In the above and in the following, subscripts m, s, h, and e will denote the master, slave, human and environment respectively. $\boldsymbol{f}_{em}$ is the interaction force between the operator's arm and the master, $\boldsymbol{f}_{ref}$ is the reflected force, $\boldsymbol{x}_{ds}=\boldsymbol{K}_{sc}\boldsymbol{x}_m$, matrices $\boldsymbol{K}_{sc}, \boldsymbol{M}_{ds}, \boldsymbol{B}_{ds}, \boldsymbol{K}_{ds}, \boldsymbol{B}_{fs}, \boldsymbol{K}_{fs}, \boldsymbol{M}_{dm}, \boldsymbol{B}_{dm}$ are parameters of the GI and $\boldsymbol{f}_{es}=\boldsymbol{Z}_e(\boldsymbol{x}_e-\boldsymbol{x}_s)$ represents the contact force at the slave side, where $\boldsymbol{Z}_e$ is the environmental impedance and $\boldsymbol{x}_e$ its location. In order to impose the GI, a control law following the computed torque method was proposed.

Assuming ideal performance, the master and human arm form a dual system described by:

$$\boldsymbol{f}_h+\boldsymbol{f}_{ref}=(\boldsymbol{M}_h+\boldsymbol{M}_{dm})\ddot{\boldsymbol{x}}_m+\boldsymbol{B}_{dm}\dot{\boldsymbol{x}}_m \qquad (4)$$

where $\boldsymbol{f}_h$ is the intentional force of the operator, i.e. the force applied by his brain to his muscle. Note that $\boldsymbol{f}_h$ is the response of his nervous system to the stimuli of vision and force (generated through a screen and the master arm). Eq. (4) reveals that $\boldsymbol{f}_{ref}$ is the reaction force actually felt by the operator. In a conventional scheme it would be equal to (a scaled) $\boldsymbol{f}_{es}$. S. Lee and H. S. Lee define it as a combination of the force and position tracking error and thus actually redefine force feedback.

3. Neural controller and error distribution algorithms

An effective way to incorporate NNs in a robot controller, is to use them for the identification of the robot's dynamics. In this context some researchers proposed the partition of the NN to subnets, each one identifying a part of the dynamics (e.g. the terms of eq. (1)) [2],[4],[7]. According to F. Lewis et al [7] this: "1) simplifies the design, 2) gives added controller structure, and 3) makes for faster weight tuning algorithms". However, partition methods proposed up to date either require a lot of subnets (e.g. [2]) or specific knowledge of the dynamics (e.g. [7]). Our method preserves simplicity by dividing the NN to only three subnets (Fig. 1) corresponding to matrices $\boldsymbol{D}, \boldsymbol{C}$ and $\boldsymbol{g}$ of eq. (1). Their outputs were added at a final layer. The only a priori information used was the subnet inputs. In order to achieve on line adaptation, the subnets were chosen to have no fixed part (in contrast to [2], [7]).

In this work RBF Networks are applied. Each subnet was first trained offline with the standard Least Mean Square [3] method, to learn the corresponding term of a nominal model. In this way the required basic functions were formed. The subnets were trained online using a gradient descent procedure outlined in [3]. Neither the position nor the spreads of centers (i.e. the input norm) were trained, since the initial simulations indicated satisfactory performance without such a computation-costly training. Those parameters were selected before learning.

The online training of NNs is a well studied topic. The main problem in our case is how to distribute the training error to each subnet. Note that only the *overall* NN error (i.e. the difference between desired and produced torque) is at hand: the desired *subnet* output is not known from the moment that the dynamics changed. Thus our guess is almost a "blind" one. Some information is however known. We tried to exploit it as much as possible.

Four different ways to distribute the error were examined (Table 1): an equal distribution of 1/3 of the total error to each subnet (*simplistic* solution*),* b) a distribution relative to the contribution of the subnet to the total output of the net, ("*coarse*" solution), c) a distribution exploiting the information about the "measurable part" (Measurable Information (MI) solution), and d) a combination of the two previous methods, named HERD Method (=Heuristic ERror Distribution Method).

These functions were inspired by an analysis of eq. (1): each of the robot torque's element is determined by a complicated matrix ($\boldsymbol{D}, \boldsymbol{C}$ or $\boldsymbol{g}$) related to the structure of the robot, which is multiplied by a fixed measurable vector ($\dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ or 1). The MI method exploits the "*measurable part*" 's information, whereas the coarse method the "*structural part*" 's information.

The coarse method is based on the assumption that the general form of the dynamic model will not change for realistic parametric modifications, i.e. changes in the mass

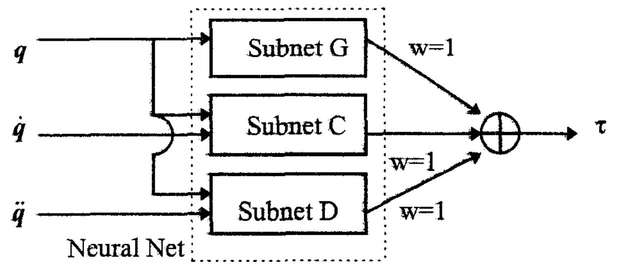

Figure 1. Partition to subnets.

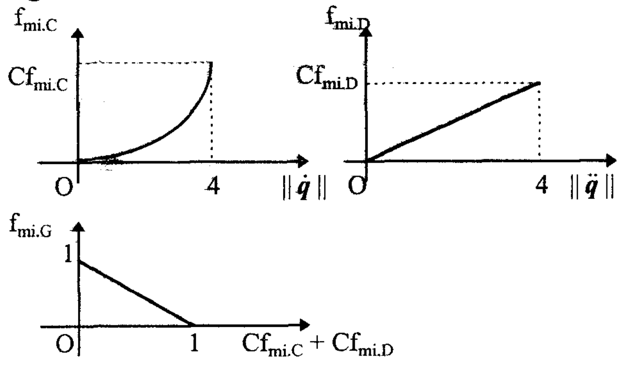

Figure 2. Error distribution functions f_{mi}.

TABLE 1: Error Distribution Methods

Method	Er. Distr. Coefficient
Simplistic	1/3
Coarse	$\lvert\tau_i\rvert / \lvert\tau\rvert$ with i=D,C,G
Measurable Info (MI)	$\mathbf{f}_{mi}$
HERD	$(\lvert\tau_i\rvert / \lvert\tau\rvert) * \mathbf{f}_{mi}$ with i=D,C,G
(Error per Subnet) = (EDC) * (Total Error).	

and length of the last link. This is indeed true, as an inspection of the robot dynamic equations can prove and is in detail analyzed in [12]. Thus, based on the old model, we can guess when the subnet output (and hopefully its error also) will be 'big', 'small','increasing' or 'decreasing'.

The form of $\mathbf{f}_{mi}$ was inspired by observing in what way each term of the robot dynamics depends on $\boldsymbol{q}, \dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$. They are plotted in Fig.2 and described by:

$$f_{mi,D}(\|\ddot{q}\|)=Cf_{mi,D}\|\ddot{q}\|, \quad f_{mi,C}(\|\dot{q}\|)=Cf_{mi,C}\|\dot{q}\|^2$$
$$f_{mi,G}\left(\|\ddot{q}\|+\|\dot{q}\|^2\right)=1-Cf_{mi,C}-Cf_{mi,D} \tag{5}$$

The specification of the coefficients $\mathbf{Cf}_{mi}$ is not an easy task. No fixed coefficients, globally optimal for any robot and working parameters, can be found. A procedure to specify them, on the basis of the maximum realistic $\boldsymbol{q}, \dot{\boldsymbol{q}}$ is recommended in [12]. Note that each subnet output is assigned a different $\mathbf{f}_{mi}$, i.e. with different coefficients. The type of the norms in $\mathbf{f}_{mi}$ can also be adjusted according to the task / robot. A reasonable choice is to make them dependent only on the subnet inputs (e.g. for the 2-DOF robot of Section 5 the norm of subnet C's second output should be independent of $\dot{\boldsymbol{q}}_2$).

The HERD method is clearly a concatenation of the "coarse" and "MI" methods. An extra heuristic added was that the subnet C (or D) training error was set equal to the subnet output if $\|\ddot{\boldsymbol{q}}\|<\varepsilon$ (small constant) (or $\|\dot{\boldsymbol{q}}\|<\varepsilon$ (small constant)). Thus a possible inadequate offline learning of the measurable part was healed.

In essence, the various training sections of Guez and Selinsky [2] are here overlapped and take simultaneously place, in a fuzzy logic like way.

Among the problems that could not be solved, two are most important. First, although a guess on the magnitude of the error is made, its sign remains unknown. Second, the *total* error is minimized very soon (less than 0.3% in less than 20 sampling periods). After this period the learning is very slow, since the error to be distributed is small. If in the first few sampling periods the networks confuse their roles, then this confusion will not heal much in the sequence. This was unfortunately the case in all simulations done.

As a consequence of the efforts up to now we conclude that an online-trained multipartitioned network under the presence of severe uncertainties can *successfully act as a controller*, since the total error is kept very small (0.1%-0.5%) but *significantly confuses the role of its subnets* (up to 30% in the worst case, for RBF networks).

In order to choose a neurocontrol architecture in conjunction to which the proposed partitioned controller would be used, several schemes were extensively tested as part of our previous work [11]. The best results were obtained by a variation of the Model Reference Adaptive Control scheme, reported in [4]. Impressive results were also obtained by a variation of Internal Model Control, but the previous scheme was judged to be more appropriate to use within the teleoperator.

4. Incorporation of the neural controller in the teleoperator

The most straightforward way to incorporate the neurocontroller in the teleoperator is to first calculate an ideal trajectory according to the Generalized Impedance (GI), since this determines the desired performance in the original architecture, and then use the NNs to force the robots to track it. The original controller functions in a similar way, but there the GI is explicitly incorporated in the control torque generation formulas, whereas now a two-level approach is followed (Fig. 3): at the higher level, the operator and GI blocks determine the desired trajectories "thinking" in Cartesian coordinates. At the lower level, the controllers try to impose the ideal performance through suitable control inputs.

On the master side the collaboration of the operator and the controller induces a complicated situation. The most important questions arising are: a) how should the desired trajectory be generated? b) how can good controller performance be guaranteed, since it acts in parallel with the *non*-passive human intentional force f_h, c) How can we ensure that the operator will feel f_{ref} as reaction force?

Regarding the point (a), the control scheme of S. Lee and H. S. Lee offers two possibilities: equations (2) and (4). Ideally, the use of either would give the same results, since they both represent the GI. Their advantages are that the second is applied on the dual dynamic system operator's arm + master, whereas the first takes in account only *sensed* variables. In our opinion eq. (4) is the best choice, since it allows to control all the passive parts at the master side and uses f_h as input, rather than its filtered version f_{em}. To provide f_h, an estimator of the human arm dynamics was incorporated in the control scheme. In [5] a simplified model for the human arm is derived, whose estimation is trivial. Even a more detailed one can be identified, e.g. by using an online-adapting NN, to tune to parameters' variation, according to the operator's intention or his fatigue. This way the results of fatigue and control errors of the operator towards his muscle can be compensated. Difficulties (b) and (c) above, are closely related to each other and can both be solved if the controller's performance is satisfactory *right from the start.* This is actually the case with RBF neurocontrollers as well as sliding mode controllers [9],[10]. On the contrary, with MLP NNs, the NN achieved satisfactory performance after a short (1 sec), automatically performed training phase, during which their is no interaction with the operator [11]. With the exception of the points discussed above, the design concept [5] was preserved without changes (e.g. monitoring forces).

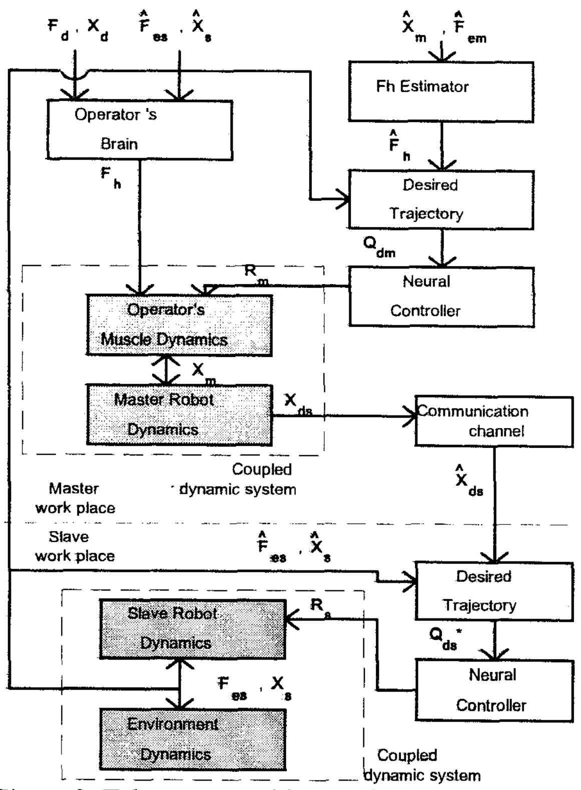

Figure 3. Teleoperator with neural controllers. ^ denotes variables corrupted by noise.

5. Simulations and comparison with sliding mode and MLP - based neural control

Simulations were performed with two identical rigid-revolute-link manipulators of 2 DOFs acting as the master and slave arm (Table 2) [8]. All simulations were carried out with sampling period of 1 msec at a PC with a 90 MHz Intel Pentium processor. Simulations checking the response of the original algorithm of [5] were reported in [11]. It was shown that modeling errors, and even a simple computation delay of a full sampling period, can seriously degrade the performance of the system. These results justified the need for a new controller.

A first set of simulations were carried out on a *single* robot, in order to check the efficiency of the proposed error distributing methods. The partitioned NN did not serve as a controller, but simply tried to identify online the modified robot dynamics. The NN was initially trained at an erroneous nominal model, with errors as shown in Table 2 for the slave.

Three of the methods presented in Section 3 were compared (Table 3). Note that the trial trajectory is a hard one. To validate the methods, a special index, named the *Confusion Index* CI_x was used, in an effort to isolate the error due to the error distribution algorithm from the normally appearing error, which is due to the learning algorithm / chosen parameters' values:

TABLE 2: Parameters and Modeling Errors

Operator As in [5], expanded in the 2-DOF case. **Master and slave dynamics** *link 1*: m = 10, l = 1, lc= 0.5 I = 0.8 *link 2*: m = 5, l = 1, lc= 0.5, I= 0.45 SI units (m, kgr, kgr*m^2) Symbols: m: mass, l: length, lc: center of mass position from previous joint. I: moment of inertia **Generalized Impedance**: $\boldsymbol{M}_{dm}=\begin{bmatrix}0.05 & 0\\ 0 & 0.05\end{bmatrix}, \boldsymbol{B}_{dm}=\begin{bmatrix}4 & 0\\ 0 & 4\end{bmatrix}, \boldsymbol{B}_{fs}=\begin{bmatrix}0.75 & 0\\ 0 & 0.75\end{bmatrix}$ $\boldsymbol{M}_{ds}=\begin{bmatrix}0.1 & 0\\ 0 & 0.1\end{bmatrix}, \boldsymbol{B}_{ds}=\begin{bmatrix}8 & 0\\ 0 & 8\end{bmatrix}, \boldsymbol{K}_{ds}=\begin{bmatrix}10 & 0\\ 0 & 10\end{bmatrix} \boldsymbol{K}_{fs}=\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$ **Communication channel and Grp** $\boldsymbol{K}_{sc}=\begin{bmatrix}1.0 & 0\\ 0 & 1.0\end{bmatrix}, \boldsymbol{K}_{cs}=\begin{bmatrix}0.02 & 0\\ 0 & 0.02\end{bmatrix}, \boldsymbol{G}_{rp}=\begin{bmatrix}1.0 & 0\\ 0 & 1.0\end{bmatrix}$
Modeling Errors
20% and 50% for the master and slave respectively on the mass and moment of inertia of the last link.
RBF Networks
Subnet G: 2-49-2 neurons, inputs: **q**, η=0.1 Subnet C: 3-81-2 neurons, inputs: q_2, dq/dt, η=0.01 Subnet D: 3-81-2 neurons, inputs: q_2, d^2q/dt^2, η=0.01

TABLE 3. Error distributing methods (CI)

Subnet	Simplistic	Coarse	HERD
τ_{total}	0.42%	0.45%	0.49%
$\tau_{G,1}$	14.38	7.48	5.01
$\tau_{G,2}$	15.85	11.78	11.14
$\tau_{C,1}$	45.67	16.87	15.83
$\tau_{C,2}$	29.41	19.54	20.17
$\tau_{D,1}$	17.45	19.54	13.56
$\tau_{D,2}$	32.20	24.46	16.98
Task: $q_1 = \pi + 0.8\pi*\cos(2\pi*0.2*kT)$), $q_2 = \pi - 0.8\pi*\cos(2\pi*0.16*(kT))$. A representative part of the joint space is covered. $\dot{q}, \ddot{q}$ up to 4 rad/sec-rad/sec^2 are demanded. ε=0.75 only for subnet C.			

CI =
|(subnet error)-(subnet error by training with real error)| /
/ max(ideal subnet output, subnet error) * 100 %

The coefficients of the HERD method were chosen to allow maximum ($\dot{q}, \ddot{q}$)=(5m/sec, 5m/sec^2), which result in quite high velocities for industrial robots. The final values used were: $\mathbf{Cf}_{mi,G1}$ = 0.27, $\mathbf{Cf}_{mi,G2}$ = 0.22, $\mathbf{Cf}_{mi,C1}$ = 0.55, $\mathbf{Cf}_{mi,C2}$ = 0.6, $\mathbf{Cf}_{mi,D1}$ = $\mathbf{Cf}_{mi,D2}$ = 0.1.

The superiority of the HERD algorithm over the other methods is evident. Although the error per subnet is significant, it is acceptable since a poor qualitative information is only used. Regardless of this "confusion" error, the overall error is very small and thus the RBF - based scheme *can* be used as a controller.

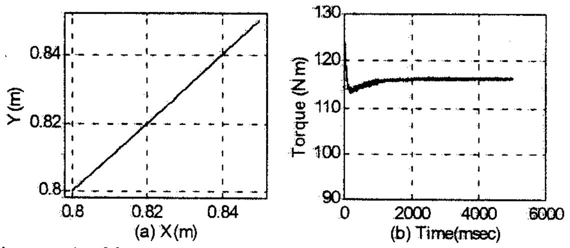

Figure 4. Slave trajectory, free space, (a) ideal system (black), system with modeling uncertainties (gray), x_d=(0.85m,0.85m), (b) control torque of joint 1.

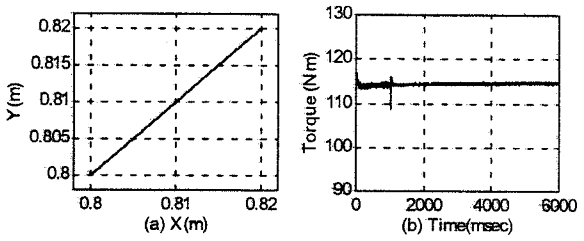

Figure 5. Slave trajectory, contact task, x_d=(0.8105m, 0.8105m), f_d=(5N,5N). This force will actually be felt if the penetration is 0.1768 m. (a) ideal (black), system with modeling uncertainties (gray), (b) control torque of joint 1.

TABLE 4: Comparison of control methods

	MLP	RBF	Sliding mode
Size of network / complexity	small	big	big
offline learning / mathematical computation	big (hours)	small (minu-tes)	big (hours)
online learning: confusion error	bigger	smaller	no learning
online learning: total error	very small	very small	no learning
guaranteed convergence	no	no	yes
robot trajectory	good	excellent	excellent
control chattering in application to teleoperator	small o(0.01) Nm	big o(1)Nm	medium o(0.1)Nm

In the final set of simulations (Figs. 4,5) the complete teleoperator system was considered. In those Figures, when referring to "ideal" response we mean the response computed directly by the GI equations. For the "non-ideal" case the control input was first calculated and then applied to the robot dynamics after a time delay of one sampling period, in order to model the computational delay. At all tasks reported here, the two robots were initially stationary at position (0.8m,0.8m) relative to their base frame. Then the operator tries to move the slave in position x_d. At the contact task he simultaneously estimates to feel a reaction force f_d. The object in the contact tasks is modeled as an elastic, immobile "wall" of infinite dimensions with $Z_e = \begin{bmatrix} 20 & 0 \\ 0 & 20 \end{bmatrix}$ N/m. A point on the border is Xe = (0.8085m, 0.8085m) and a unity vector vertical to it is (0.7071, 0.7071). The trajectory in Figs. 4,5 is almost ideal. However, when the set goal was further away, instability was observed. Since, as proved by previously mentioned trials on a single robot, the overall training error is minimal for a much harder task (Table 3), we conclude that this is due to the original teleoperator control architecture and the parameters chosen (Table 2).

Another NN controller, incorporating MLP NNs, as well as a sliding mode robust controller, as a representative of classical control methods, were previously tested as an improvement of the original computed torque control scheme of S. Lee and H. S. Lee [9],[10],[11]. The controllers were incorporated in the teleoperator system following the concept presented in Fig. 3, where the controller blocks represent now the new type of controllers instead of RBFs. Although a direct quantitative comparison between the methods is not easy, due to the differences in basic concepts and architecture parameters, a qualitative one is attempted in Table 4. A more detailed one is presented in [12]. Among the two NN families tested, RBF controllers performed trajectory better, since they exhibited no initial trajectory overshoot and less "subnetwork confusion". However, the control chattering was, surprisingly, worse than even the sliding controllers, and the MLP had less neurons. Generally, all three methods produced practically ideal trajectories.

An engineer deciding which algorithm to implement, should also consider the fact that the sliding mode technique requires a mathematical nominal model of the robot dynamics as well as knowledge of the uncertainty bounds. NNs adapt quickly to a wide range of situations and parameters, and the output is calculated easier and faster. Finally, an inverse model of the system is identified. This can prove to be very useful to evaluate the situation and the objects encountered, much better than the "blind" compensation offered by sliding robust controllers. Therefore NNs are, in our opinion, more suitable for employment within the teleoperator system.

6. Conclusion

The work described in this paper constitutes a continuation of the authors' effort to enhance a recent teleoperator control architecture so as to be able to face large modeling errors in the robotic parameters. These errors are unavoidable in practice, and are caused by several reasons. Here the radial basis functions (RBF) neural networks were used in a multipartitioned NN architecture in conjunction with a new heuristic learning algorithm (called HERD algorithm) for identifying the inverse robot dynamics. The results achieved by this algorithm are compared with those obtained by the "simplistic algorithm" and the "coarse algorithm" defined in the text. It was found that the RBF NNs can be effectively used to control the teleopertator system, and its robustness capabilities were compared to those of an MLP neurocontroller and a sliding mode controller. The simulation results of the RBF controller proved very encouraging.

References

[1] Cha, D.H., Cho, H.S. and Kim, S., Design of a Force Reflection Controller for Telerobot Systems using Neural Network and Fuzzy Logic, Journal of Intelligent and Robotic Systems,Vol. 16, 1996, pp. 1-24.

[2] Guez, A. and Selinsky, J., Neurocontroller Design via Supervised and Unsupervised Learning, J. of Intelligent and Robotic Systems, Vol. 2, pp. 307-335, 1989.

[3] Haykin, S., Neural Networks,Macmillan College Publishing.,1994.

[4] Lee, T.H.., Tan, W.K. and Ang, M.H., A Neural Network Control System with Parallel Adaptive Enhancements Applicable to Nonlinear Servomechanisms, IEEE Trans. on Ind. Electronics, Vol. 41, No. 3, June 1994, pp. 269-276.

[5] Lee, S. and Lee, H.S., Modeling, Design and Evaluation of Advanced Teleoperator Control Systems with Short Time Delay, IEEE Trans. on Robotics and Automation, Vol. 9, No. 5, October 1993, pp. 607-623.

[6] Lee, S. and Lee, H.S., Design of Optimal Time Delayed Teleoperator Control Systems, Proc. 1994 IEEE Int. Conf. on Robotics and Autom., San Diego, California, pp. 3252 - 3258.

[7] Lewis, F.L., Liu, K, Yesildirek, A., Neural Net Robot Controller with Guaranteed Tracking Performance, IEEE Trans. on Neural Networks, Vol. 6, No. 3, May 1995, pp. 703 -715.

[8] Tzafestas, S.G. (ed.), Intelligent Robotic Systems (ch. 10), Marcel Dekker, NY, 1991.

[9] Tzafestas, S.G. and Prokopiou, P.A., Sliding-Mode Robust Control of Master-Slave Telemanipulating Systems, 2nd ECPD Int. Conf. on Advanced Robotics, Intelligent Automation and Active Systems, Vienna,1996, pp.599-604.

[10] Tzafestas, S.G. and Prokopiou P.A., Compensation of Teleoperator Uncertainties with a Sliding Mode Controller, Journal of Robotics and Computer Integrated Manufacturing (in Press).

[11] Tzafestas, S. G., Prokopiou, P. A. and Tzafestas, C. S., Robust Telemanipulator Control Using a Partitioned Neural Network Architecture, 1997 IEEE International Conference on Neural Netwroks (ICNN '97), Houston, Texas, June, 1997.

[12] Tzafestas, S. G. and Prokopiou, P. A., The HERD Method, IRAL, DECE, NTUA, Technical Report, 1997.