Robust Telemanipulator Control Using a Partitioned Neural Network Architecture

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

In this paper the control problem of telemanipulators is considered under the condition that they are subject to modeling and other uncertainties of considerable levels. The design is based on the S. Lee and H. S. Lee teleoperator control scheme, which is modified so as to be able to compensate the uncertainties,and is implemented using a partitioned multilayer perceptron neural network. Several subnetworks are used each one identifying a term of the manipulator's dynamic model. A new learning algorithm is proposed which distributes the learning error to each subnetwork and enables online training. Several simulation results are provided, which show the robustness ability by the partitioned neurocontroller, and compare it with the results obtained through sliding mode control.

1. Introduction

Although research on autonomous robots is expected to reduce the need for humans working in hostile or unpredictable environments, our supreme recognition, analysis, decision making and manipulation abilities, are not easy to be matched. Teleoperators will still be valuable in the future for space, underwater, underground, hazardous or medical applications. Current research topics include the incorporation of autonomous functions and designing of schemes robust to time delay in the communication channel between the master and slave robots.

Neural networks (NNs hereafter), possessing a remarkable ability to identify and control strongly nonlinear, multivariable plants, with minimum need for prior knowledge, through an adaptive, compact and fast system, can provide valuable solutions in various levels of a telemanipulator. The methods developed for single manipulators can be applied locally to the master and slave. In addition, they can assist in the co-ordination 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. Several applications of NNs on single manipulators have been reported (e.g. [1], [2], [5], [8]), but no reference was found concerning an application to telerobotics.

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 [3]. 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 [6], is abandoned. S. Lee and H. S. Lee argue that in the context of semiautonomous control, this concept might mislead the operator, since he is not aware of the automatic functions but "feels" their results. Doubts about the necessity of telepresence have been also previously expressed [6]. In this paper, their scheme is enhanced in order to be able to compensate modeling uncertainties. Moreover, a novel heuristic learning algorithm is designed, suitable for use with a partitioned NN, which identifies the inverse robot dynamics. In [9] a sliding mode robust controller is designed for the same purpose.

Modeling errors can be caused by several factors. In the slave side, they could be due to picking up an object of

0-7803-4122-8/97 $10.00©1997 IEEE

unknown mass and shape (e.g. tool or sample) in order to perform an assembling or some other task with it. 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) significant 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. Other causes of errors, appearing on both robots, are 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

The paper is organized as follows. Some basic elements of the architecture of S. Lee and H. S. Lee are outlined in Section II. The neural controller is introduced in Section III, and incorporated in the teleoperator in Section IV. The resulting system is tested through simulations in Section V, and compared with a sliding mode robust controller in Section VI. Section VII contains the conclusions.

2. Original control scheme

Rigid-link manipulators of n degrees of freedom are considered. Their dynamics are described in [7]:

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

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 architecture 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:

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

$$\boldsymbol{M}_{ds}\ddot{\boldsymbol{x}}_s + \boldsymbol{B}_{ds}(\dot{\boldsymbol{x}}_s - \dot{\boldsymbol{x}}_{ds}) + \boldsymbol{K}_{ds}(\boldsymbol{x}_s - \boldsymbol{x}_{ds}) = \boldsymbol{B}_{fs}\dot{\boldsymbol{f}}_{es} + \boldsymbol{K}_{fs}\boldsymbol{f}_{es} \quad (2b)$$

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 $\boldsymbol{M}_{ds}, \boldsymbol{B}_{ds}, \boldsymbol{K}_{ds}, \boldsymbol{B}_{fs}, \boldsymbol{K}_{fs}$, $\boldsymbol{K}_{sc}$ $\boldsymbol{M}_{dm}, \boldsymbol{B}_{dm}$, 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:

$$\boldsymbol{f}_h + \boldsymbol{f}_{ref} = (\boldsymbol{M}_h + \boldsymbol{M}_{dm})\ddot{\boldsymbol{x}}_m + \boldsymbol{B}_{dm}\dot{\boldsymbol{x}}_m \quad (3)$$

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. (3) 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 [3].

In the initial work [3] the role of modeling errors was not considered. In [4] S. Lee and H. S. Lee 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 order to fulfil the above requirements we considered a robust control solution ([9] and Section VI) and an adaptive one (this paper).

3. Neural controller

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)) [1],[2],[5]. According to F. Lewis et al [5] 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. [1]) or specific knowledge of the dynamics (e.g. [5]). We searched for a method which would: a) enable on-field adaptation to sudden

parameters' change, b) enable fast output calculation and weight tuning (for this, small NNs are necessary), c) preserve the basic nonlinear functions learned offline. An inappropriate online training method could e.g. allow the gravitational subnet learn some parts of the D matrix, d) enable the learning of new nonlinear terms, induced due to contact with the environment, actuator saturation, or hardware damage and e) encompass minimum a priori information in the subnet structure, so that the net self-optimization and a compact representation are possible. In order to meet requirements (b) and (e) the NN was divided to only three subnets (Fig. 1) corresponding to matrices D, h and 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 [1], [5]). To fulfill point (c) above, a novel, heuristic way to train the subnets was developed.

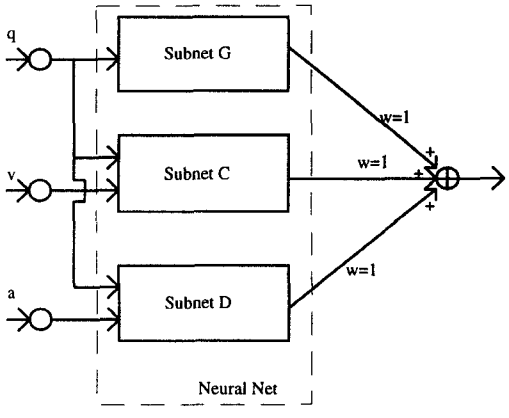

Figure 1. Partition to subnets, q: joint angle, v: velocity, a: acceleration.

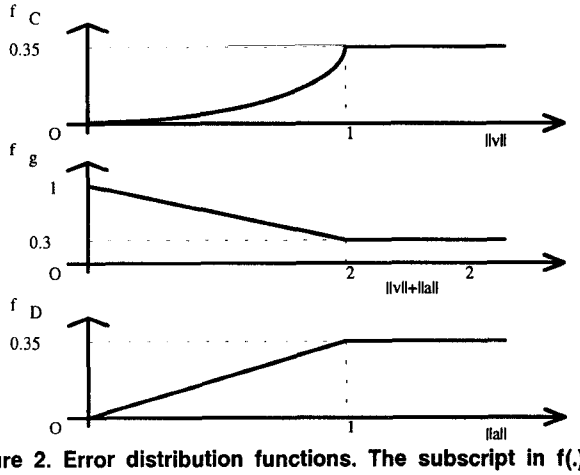

Figure 2. Error distribution functions. The subscript in f(.) stands for the corresponding subnet.

In this work Multilayer Perceptron Networks (MLPs) were used. Each subnet was first trained offline to learn the corresponding term of a nominal model. In this way the required basic functions were formed. The subnets were trained online with a variation of the Back Propagation (BP) algorithm, although any other algorithm can also be used. The main problem with the online training is how to distribute the training error to each subnet. A straightforward way is to include the output neuron weights to the BP process, i.e. first propagate the error through them. However, simulations revealed that this violates requirement (c) above. Much better results were obtained by employing some heuristic functions. Their form was inspired by observing in what way each term of the robot dynamics depends on joint position, velocity and acceleration. They are plotted in Fig.2 and described by:

$$f_D(\|\vec{q}\|) = \begin{cases} 0{,}35\|\vec{q}\| \ldots \text{if..} \|\vec{q}\| \le 1 \\ 0{,}35 \ldots\ldots\ldots \text{if..} \|\vec{q}\| \ge 1 \end{cases}, f_C(\|\vec{q}\|) = \begin{cases} 0{,}35\|\vec{q}\|^2 \ldots \text{if..} \|\vec{q}\| \le 1 \\ 0{,}35 \ldots\ldots\ldots \text{if..} \|\vec{q}\| \ge 1 \end{cases}$$

$$f_g(\|\vec{q}\|+\|\vec{q}\|^2) = \begin{cases} -0{,}35 \cdot (\|\vec{q}\|+\|\vec{q}\|^2) \ldots \text{if..} \|\vec{q}\|+\|\vec{q}\|^2 \le 1 \\ 0{,}3 \ldots\ldots\ldots\ldots\ldots\ldots \text{if..} \|\vec{q}\|+\|\vec{q}\|^2 \ge 1 \end{cases} \quad (4)$$

The error per subnet is obtained by multiplying the total error with these functions. In essence, the various training sections of Guez and Selinsky [1] are here overlapped and take simultaneously place, in a fuzzy logic like way. The constants and the type of norm in eq. (4) can be optimized for a specific robot and task.

In order to choose a neurocontrol architecture in conjunction to which the proposed partitioned controller would be used, several schemes were extensively tested. The best results were obtained by a variation of the Model Reference Adaptive Control scheme, reported in [2]. 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.

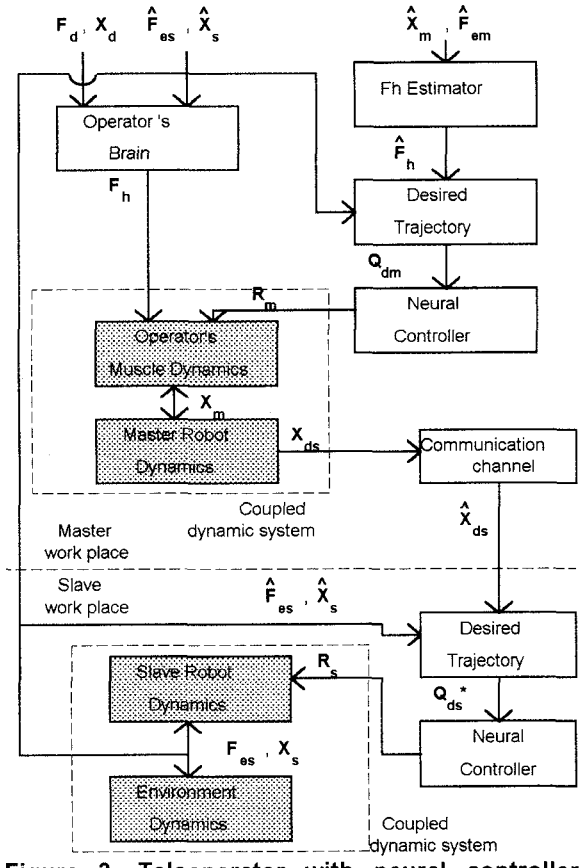

Figure 3. Teleoperator with neural controllers. ^ denotes variables corrupted by noise. Note that f_{es}, x_s are fed back to the master side.

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 (2a) and (3). 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. (2a) 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. S. Lee and H. S. Lee [3] derive a simplified model for the human arm, 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 sliding controllers [9]. In the scheme presented here, the NN achieves satisfactory performance after a short (1 sec), automatically performed training phase, during which their is no interaction with the operator. The robot is commanded to move along a sinusoidal course of small amplitude then stay there for a while, so that the NN learns coarsely the altered dynamics before the main movement begins. In this way the training consists of three phases: a) An initial offline phase, during which the robot is trained on a nominal model and learns the basic nonlinear functions, b) a short, on-field but still offline phase whenever serious modeling errors (are expected to) arise, and finally c) a continuous online phase for perfectly tuning the parameters. Using the terminology of classical identification, the first phase corresponds to the structure identification, the second to a coarse parameter estimation, and the last to the final parameters' tuning.

With the exception of the points discussed above, the design concept of S. Lee and H. S. Lee was preserved without changes (e.g. monitoring forces).

5. Simulations

Simulations were performed with two identical rigid-revolute-link manipulators of 2 DOFs acting as the master and slave arm (refer to Table I). In the following, *all distances are in meters, and forces in Newtons.* 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 (0.8,0.8) 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 $Z_e = \begin{bmatrix} 20 & 0 \\ 0 & 20 \end{bmatrix}$. A point on the border is Xe = (0.8085, 0.8085) and a unity vector vertical to it is (0.7071, 0.7071). All simulations were carried out with sampling period of 1 msec at a PC with

TABLE I: 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:

$$M_{dm} = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix}, B_{dm} = \begin{bmatrix} 4.0 & 0 \\ 0 & 4.0 \end{bmatrix}$$

$$\mathbf{M}_{ds} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}, \mathbf{B}_{ds} = \begin{bmatrix} 8 & 0 \\ 0 & 8 \end{bmatrix}, K_{ds} = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$$

$$B_{fs} = \begin{bmatrix} 0.75 & 0 \\ 0 & 0.75 \end{bmatrix}, K_{fs} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Communication channel and Grp

$$K_{sc} = \begin{bmatrix} 1.0 & 0 \\ 0 & 1.0 \end{bmatrix}, K_{cs} = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix}, G_{rp} = \begin{bmatrix} 1.0 & 0 \\ 0 & 1.0 \end{bmatrix}$$

Modeling Errors

Unless otherwise stated, 20% and 50% for the master and slave respectively on the mass and moment of inertia of the last link.

a90 MHz Intel Pentium processor.

First some simulations were performed to check the response of the original algorithm. Figs. 4 a,b show that even if only computation delay of a full sampling period is taken into account the slave follows a sinusoidal course instead of the ideal. With modeling errors only at the slave side, the slave moves along a curved course and *stops* at a totally wrong position. With modeling errors at both robots and upon collision with objects, the system becomes unstable. The above results justify the need for a new controller.

Next, simulations were carried out on a *single* robot, identical to the ones of Table I, in order to check the efficiency of the proposed error distributing method. The partitioned NN did not serve as a controller. Data generated from eq. (1) along sinusoidal trajectories with varying frequencies and amplitudes was used to check its ability to learn the inverse dynamics. The NN was initially trained at an erroneous nominal model, with errors as for the slave in Table I. Figs. 4c,d,e demonstrate the superiority of the proposed algorithm in comparison to the BP. In the final set of simulations (Figs. 4f,g,h) the complete teleoperator system was considered. Note that if relative fast movements are commanded after the uncertainty arises (Fig. 4g) the trajectory is not ideal,

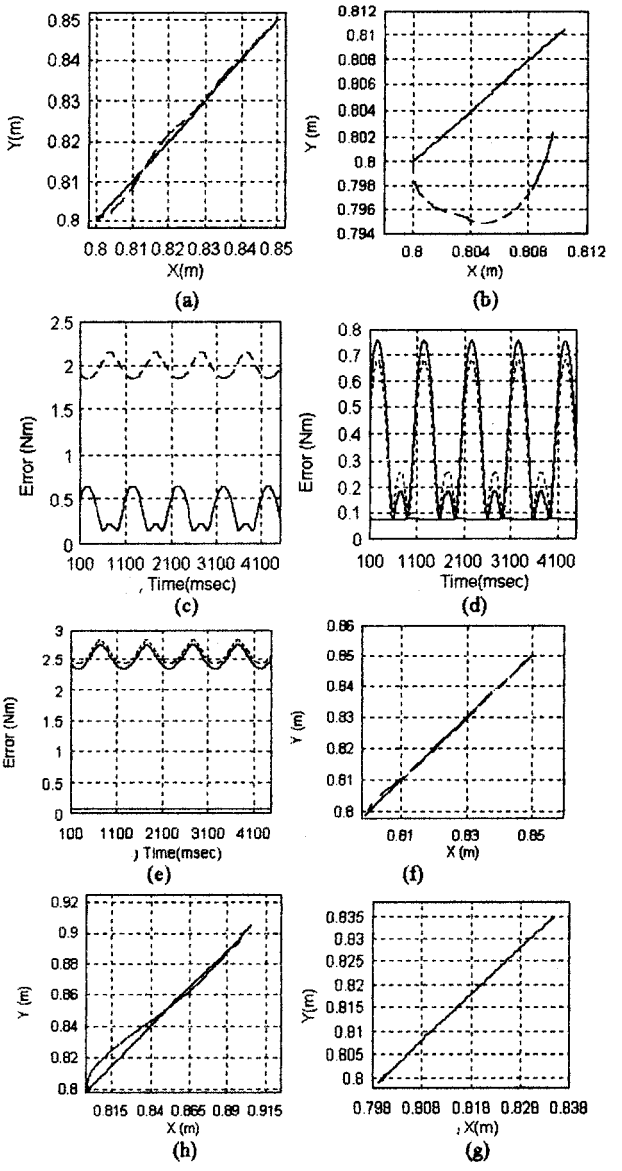

Figure 4. (a)-(b): Slave trajectory, ideal system (solid) (a) system with full period computation delay (dashed). Goal point x_d =(0.85,0.85) (b) system with modeling error at the slave (dashed). Goal point x_d =(0.8105,0.8105), (c)-(e): Simulations *on a single robot*, (c) novel error distribution algorithm (solid) and BP (dashed). Correct output ≈114 Nm. (d) Individual errors for subnets G (solid), h (dashed), D (dotted), compared to each term of eq. (1), with novel algorithm. (e) as above, with BP. (f)-(h): Slave trajectory. (f),(g): free space, (f) ideal system(solid), system with modeling uncertainties (dashed), x_d =(0.85,0.85), (g) system with modeling errors x_d =(0.9105, 0.9105) and back to (0.8,0.8), (h): contact task, x_d =(0.8105, 0.8105), f_d =(5,5). This force will actually be felt if the penetration is 0.1768 m.

since the NN cannot learn so fast. However, when the robot is commanded to go back (effectively along the same trajectory) the tracking error is significantly reduced. This leads us to search for faster learning algorithms (other than BP). The reader should also keep in mind that the initial off line training was not optimal. Fig. 4h shows that also under contact with the environment the system performs well. Note that upon collision, a small deviation from the linear course is observed. To minimize it, the scaling constant K_{cs} was reduced by a factor of 10.

6. Comparison with a sliding mode robust controller

A sliding mode robust controller was also tested as an improvement of the computed torque controller of the control scheme of S. Lee and H. S. Lee [9]. Sliding controllers guarantee trajectory tracking under the presence of modeling uncertainties of known bounds and disturbances [7]. In this section, this alternative control method will be briefly compared with the previously outlined neural one.

The controller was 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 NNs. A problem faced was how to handle force control, since the sliding mode method is primarily aimed at trajectory following. The most straightforward approach was to apply the two step procedure also utilized for the neural controllers.

The results were very good: perfect trajectory following was achieved at all the tasks in free space and upon contact with non-rigid objects. Compared with the NN controller, the robust one could ensure good performance right from the start, and no "collision shock" was observable on the trajectory. Furthermore, in contrast to NNs, good performance is theoretically guaranteed However, the inevitable chattering in the control input was at least one order of magnitude bigger than the one observed with NNs. In addition, the sliding mode technique requires a 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, very suitable for employment within the teleoperator system.

7. Conclusion

In this paper, a previously proposed teleoperator architecture was enhanced to function under modeling errors reaching up to 50% in certain parameters of the robot. Neural network controllers were successfully used. The NN was divided into three subnets and trained by a novel algorithm. The simulation results were proved very encouraging.

References

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

[2] T.H. Lee, W.K. Tan and M.H. Ang, A Neural Network Control System with Parallel Adaptive Enhancements Applicable to Nonlinear Servomechanisms, *IEEE Trans. on Industrial Electronics*, 41(3): 269-276, June 1994.

[3] S. Lee and H.S. Lee, Modeling, Design and Evaluation of Advanced Teleoperator Control Systems with Short Time Delay, *IEEE Trans. on Robotics and Automation*, 9(5): 607-623, October 1993.

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

[5] F.L. Lewis, K. Liu and A. Yesildirek, Neural Net Robot Controller with Guaranteed Tracking Performance, *IEEE Trans. on Neural Networks*, (6)3: 703 -715, May 1995.

[6]T.B. Sheridan, Telerobotics, *Automatica*, 25(4): 487-507, 1989.

[7] S.G. Tzafestas, M. Raibert and C.S. Tzafestas, Robust Sliding Mode Control Applied to a 5-link Biped Robot, *J. Intelligent & Robotic Systems*, 15(1): 67-133, 1996.

[8] S.G. Tzafestas, Neural Networks In Robot Control, in: *Artificial Intelligence in Industrial Decision Making, Control and Automation* (S.G.Tzafestas and H.B. Verbruggen, eds.), Kluwer, Dordrecht/Boston, pp.327-328, 1995.

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