APPLICATION OF GENERALISED NEURAL NETWORKS FOR A DEXTROUS HAND

Péter Zsíros[1], Péter Baranyi[2], László Kővári[1], Péter Korondi[1]

[1]Deptartment of Automation and Applied Informatics, zsiros@elektro.get.bme.hu
[2]Department of Telecommunication and Telematics, baranyi@ttt-202.ttt.bme.hu
Budapest University of Technology and Economics,
H-1521, Budapest P. O..Box 91.Hungary

Abstract: The main contribution of this paper is a practical application of generalised neural networks for a dextrous hand moved by Shape Memory Alloys (SMA). Since SMA have highly non-linear characteristics and their parameters depend on the environment (mainly on temperature) so the robot hand is controlled by a generalised neural network, which can learn the actual non-linear characteristics of the robot hand. The experimental setup consists of a 20 degree of freedom hand moved by SMA string used as artificial muscle. A video camera is used to detect the position of joints. The position is then sent to the visual display computer via the Internet, which displays the hand in 3D using OpenGl.

Keywords: telemanipulation, neural networks, object recognition , reduction

1. INTRODUCTION

Telemanipulation is a process where the operator has some task done at the far environment where he/she cannot be present physically. Goertz developed the first modern master-slave system teleoperator at Argonne National Laboratory in 1945. Telemanipulation is divided into two strongly coupled processes. One process is the interaction between the operator and the master device, the other is the interaction between the slave device and the far environment contact. The master device represents the far environment at the operator site, and the slave device represents the operator at the remote site. The information flow between the operator and remote site can be seen in Fig.1, where only three types of information are fed back: visual, audio and sense of touch. Human beings get six types of sensing from the environment surrounding them but only some of these sensing are used during telemanipulation.

Telemanipulation is the extension of human manipulation to a remote location. By providing appropriate feedback telemanipulation can be well utilised in dangerous or otherwise unreachable environment. Human operators perform their tasks mostly by hand, so the master and slave devices must fit the operator's hand. In the field of telemanipulation, many haptic devices have been developed.

The glove type master device allows the most complex and dextrous manipulation. The human hand is the most widely used tool. The final goal is to make a device with which the operator can feel as if the function of his whole hand were expanded. There are several commercial sensor gloves without force feedback on the market. H. Hashimoto (2000) proposed a sensor glove with force feedback to the all 20 joints of operator's hand. If the human hand is covered with these commercial devices, some certain difficult tasks can be done at the remote site but commercial hand type slave devices are not available. This paper proposes a hand type slave device
for precise telemanipulation. Until this point no force feedback has been implemented in this system, thus the actual set up is more simplified than the general one in Fig. 1. The system used is shown in Fig. 2.

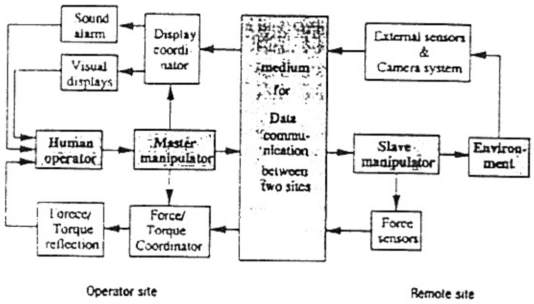

Fig. 1. Information streams of the Telemanipulation

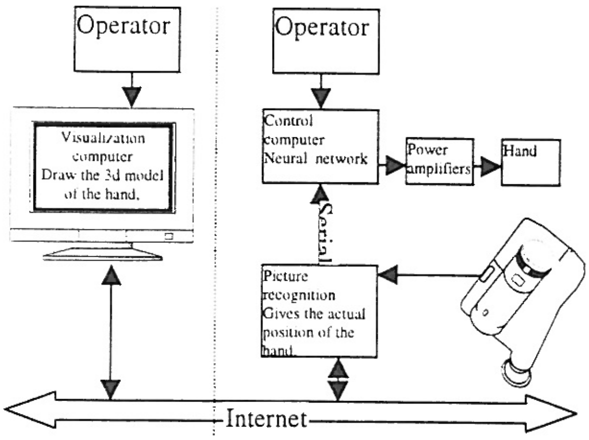

Fig. 2 Data flow of our system

2. HAND

The hand is primarily used for manipulating activities requiring very fine movement incorporating a wide variety of hand and finger postures. Consequently, there is interplay between the wrist joint positions and efficiency of finger actions. The hand region has many stable, yet very mobile segments, with very complex muscle and joint actions.

The robotic hand is made similar to a human hand, theoretically it can realise all the twenty freedoms of movement, but this time we use only eight of them. We do not want to control the movement of the most upper knuckle (DIP) separately because that movement can not be well controlled on a human hand either. We pull only the upper knuckle, and across the rubber it will pull up the middle knuckle too. The structure of the hand is shown in Fig. 3.

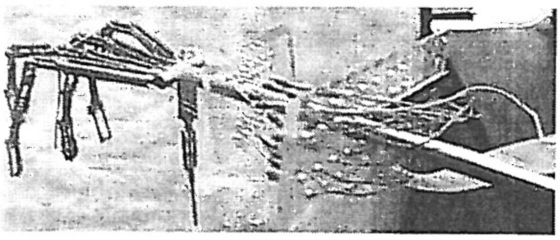

Fig.3. Photo of the artificial hand

3. CONTROL OF THE HAND

A generalised neural network is used for control the hand: The difference between a classical and a generalised neurone is that real numbers are used for the weights of a classical neurone, but the weights are fuzzy functions in the case of generalised neurones. A simple neurone is shown in Fig. 7 and a multi-layer generalised neural network will be discussed in Fig 8.

For each input of a neurone there is a B set, which can be fired by an antecedents set A. Each antecedent set correspond to a consequent set B, which is a singleton set in our case, as shown in Fig. 4. If input X_0 arrives to the input X of the system, that will cut two or more antecedents (depending on the type of antecedent function, triangular sets are assumed here). The B values belonging to the antecedents are weighed and the mass point is calculated, as it can be seen in Fig. 4:

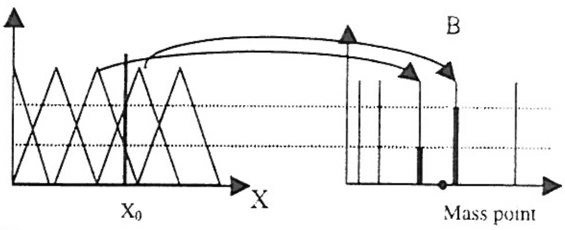

Fig. 4 Calculation of the neurones weight

The value of the mass-point will go into the summation part of the neurone (see in Fig 7). The system can be trained by changing the consequent sets (i.e. B values).

If we draw the antecedents A and the consequent B values to the two axes of a co-ordinate system it will be clearer what is happening: An arbitrary function defined on a compact domain can be approximate by a fuzzy function. The positions of antecedent sets determine the sampling points of the approximated function. The B values are the sampled values. The shapes of the antecedent membership functions determine the characteristics of the interpolation between the sampling points. If we use triangles as antecedent sets then the approximation will be piecewise linear, as shown in Fig. 5.

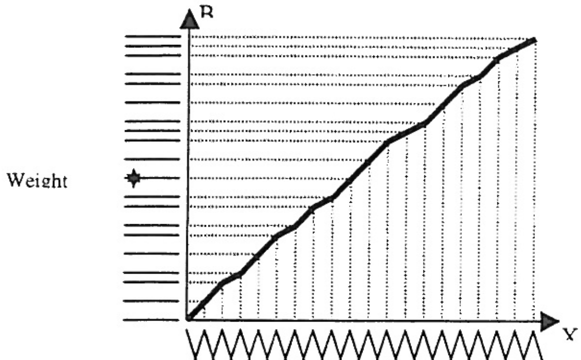

Fig. 5 Approximation of a function by the antecedent and consequent sets

So while in the case of normal neural networks there is the same non-linear function for all the inputs, in case of this generalised neural networks there are different non-linear functions for all inputs. Also by changing the B values one can easily modify these non-linear functions. From the mathematical point of view: the original neurone gives a *non-linear function of* the *linear combination* of its inputs. The generalised neurone gives the *linear combination of* the *non- linear functions* of its inputs. The antecedents of the neurones are triangular functions, because they require the least computation. The antecedents are forming Ruspini-partition which means that the sum of the membership values at any X is 1. This partition is showed in Fig. 6.

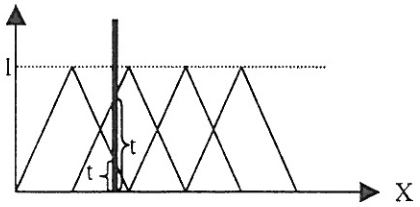

Fig. 6. Ruspini partition

$t_1+t_2 = 1$ at any X.

There are three inputs to the neurone:

- The actual position of the knuckle
- The required position of the knuckle
- The elapsed time from the last switching on

4. TRAINING OF THE GENERALISED NEURAL NETWORK

A generalised neural network is shown in Fig. 8. Using the notation introduced in Fig. 8, the outputs of the neural network can be calculated in the following way:

$$Y_j = \sum_i \sum_t A_{i,t}(X_i)\cdot B_{i,j,t} \qquad (1)$$

where i is the number of the 1-st layer neurons, t is the number of the antecedents.

$$X_i = \sum_k \sum_t A_{k,t}(P_k)\cdot B_{k,i,t} \qquad (2)$$

Where k is the number of inputs, t is the number of antecedents, P_k is the k-th input.

The question is how to modify the $B_{k,i,t}$ -th. weight, if the error is h?

$$\frac{\partial h^2}{\partial B_{k,i,t}} = ?$$

The error can not be derived directly according to $B_{k,i,t}$, the chain rule has to be used as in the Back Propagation Algorithm. The error h can be derived according to the output Y, the output can be derived according to the input X, and finally the input can be derived according to the required B. It will be the following now:

h^2 can be calculated as:

$$\frac{\partial h^2}{\partial B_{k,i,t}} = \sum_j \frac{\partial h^2}{\partial Y_j}\cdot\frac{\partial Y_j}{\partial X_i}\cdot\frac{\partial X_i}{\partial B_{k,i,t}} = \\ = \frac{\partial\left(h_1^2+h_2^2+\ldots+h_n^2\right)}{\partial Y_j}\cdot\frac{\partial Y_j}{\partial X_i}\cdot\frac{\partial X_i}{\partial B_{k,i,t}} \qquad (4)$$

$$\frac{\partial h_j^2}{\partial Y_j} = \frac{\partial\left(d_j - Y_j\right)^2}{\partial Y_j} = -2\cdot\left(d_j - Y_j\right) = -2\cdot h_j \qquad (5)$$

$$\frac{\partial Y_j}{\partial X_i} = \frac{\partial\left(\sum_i\sum_t A_{i,t}(X_i)\cdot B_{i,j,t}\right)}{\partial X_i} = \\ = \sum_t A'^{(1)}_{i,t}(X_i)\cdot B_{i,j,t} \qquad (6)$$

$$\frac{\partial X_i}{\partial B_{k,i,t}} = \frac{\partial\left(\sum_k\sum_t A^{(2)}_{k,t}(P_k)\cdot B_{k,i,t}\right)}{\partial B_{k,i,t}} = A^{(2)}_{k,t}(P_k) \qquad (7)$$

$$\frac{\partial h^2}{\partial B_{k,i,t}} = \sum_j\left[-2\cdot h_j\cdot\left(\sum_t A'^{(2)}_{i,t}(X_i)\cdot B_{i,j,t}\right)\right]\cdot A^{(1)}_{k,t}(P_k) \qquad (8)$$

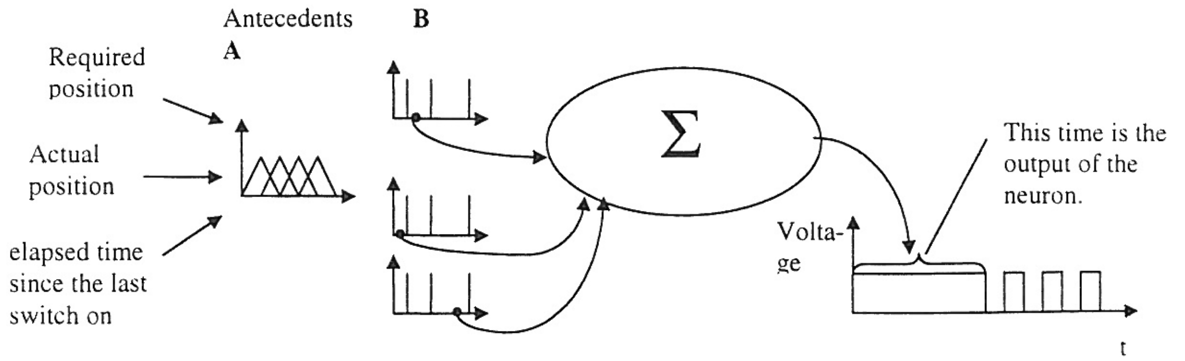

Fig. 7 Working of the control neurone

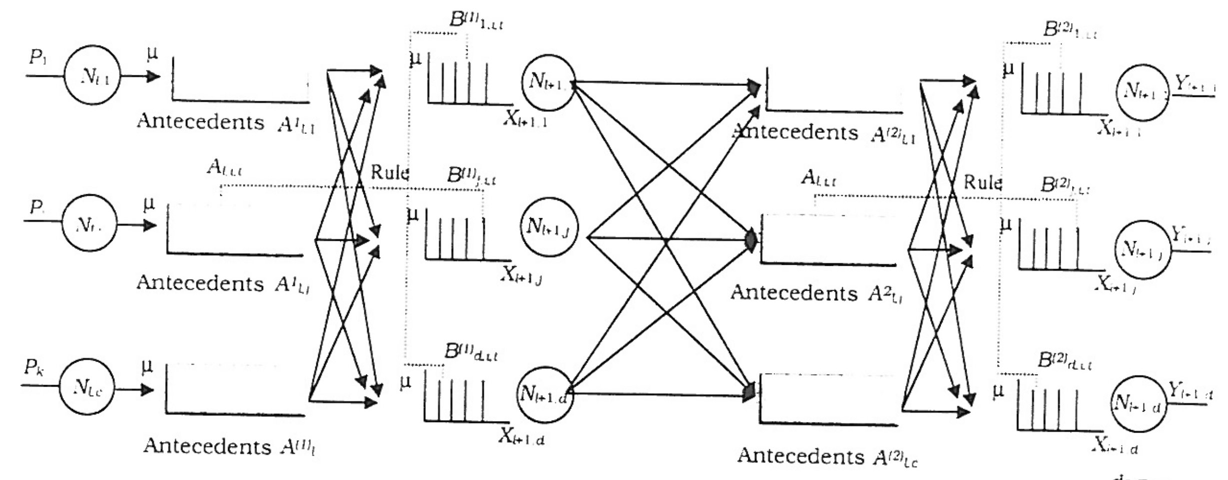

Fig. 8 The generalised neural network

In general if the errors in layer n are $h^{(n)}_j$, then the error of the neurons in layer n-1 can be calculated in the following way:

$$h_i^{(n-1)} = \sum_j h_j^{(n)} \cdot \left(\sum_t A_{i,t}^{(n)}(X_i) \cdot B_{i,j,t}^{(n)} \right) \tag{9}$$

The weight modification is given as:

$$B_{i,j,t}^{(n)new} = B_{i,j,t}^{(n)old} + 2 \cdot p \cdot h_j^{(n)} \cdot A_{i,j}^{(n)}(X_i) \tag{10}$$

5. COMPLEXITY REDUCTION OF THE GENERALIZED ALGORITHM

One of the main problems of applying fuzzy or neural techniques is calculation complexity. Engineers have to face this problem in complex systems or especially in the field of information retrieval, where extremely large information maps of whole libraries or Internet have to be processed at each user's request. These applications apply the generalized type neural networks. Let us have a brief introduction to the results of complexity theorem.

Lemma 1. *The calculation complexity grows proportionally with the number of parallel layers and the neurones.*

Omitting the computational effort of add operation but considering the product operation, the computational requirement is characterised as:

$$P_c = \sum_{i=1}^{n_l} m_{l,i} \cdot n_{l+1} + P_\mu . \tag{11}$$

P_μ indicates the calculation of the membership functions:

$$P_\mu = s \sum_{i=1}^{n_l} m_{l,i} , \tag{12}$$

where s indicates the calculation of one membership value of one antecedent set.

The main objectives of this section are to propose an algorithm which is capable of filtering out common linear combinations and reducing the number of antecedent sets based on the transformation of the weighting functions. One of the main advantages of the proposed method is that the effectiveness of the compression is controlled by the help of a given error threshold.

Theorem 2.: *Equation (4) can always be transformed into the following form:*

$$y_{l+1,j} = \sum_{z=1}^{n^r_{l+1}} a_{l,j,z} \sum_{i=1}^{n_l} \sum_{t=1}^{m^r_{l,i}} \mu_{A^r_{l,i,t}}(y_{l,i}) b^r_{l,z,i,t} \tag{13}$$

where "r" denotes "reduced", further $n^r_{l+1} \le n_{l+1}$ *and* $\forall i : m^r_{l,i} \le m_{l,i}$.

The reduced from is represented as neural network in Fig 9. The further proofs of this technique discussed by P. Barany *et al* (2000 b,c).

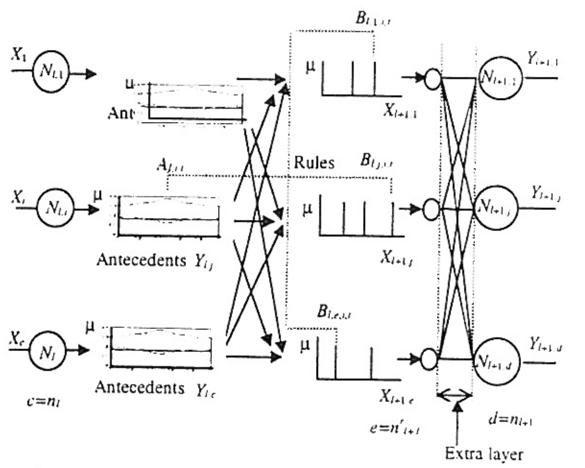

Fig. 9: Reduced neural network

6. FEEDBACK

For finding the joints in the picture of the camera by the computer, the joints were painted red. We chose this color because out of the three primary colours (Red, Green, Blue) red is the most characteristic.

Unfortunately the hand is made of copper, which has a quite high red color content too so the hand had to be painted green. Green has been chosen, because this is another primary colour, and it is used in movie techniques (in spite of the name of the technique, which is blue box, the blue color had been used only until about 10 years ago, when it was changed to green, because that has better characteristics for cameras). Then we recognised that there are a lot of red objects in the laboratory so we made a full green box around the hand. We use a Sony Handicam with a Genius Video Wonder II TV card. The computer is a Pentium-166 with 40 Mbyte RAM and with Windows 98 operating system. We chose this configuration because the programming of the TV card is quite easy under windows, there are drivers for it from the producer, and we simply have to use the routines provided by them. The program is written in Borland Delphi 4. The video-capturing program (eac401) was downloaded from the Internet (from the "Delphi Superpage"), and we modified it to fit our requirements.

We chose the size of the picture to be 160 times 120 points, because this is the smallest standard size. Even in this case the required computation power can be quite high. In this program different camera options can be set, for example brightness, contrast or saturation. By adjusting these options we could get much better pictures. This process can be followed here:

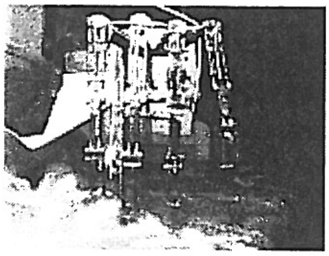

Fig. 10 The normal video picture

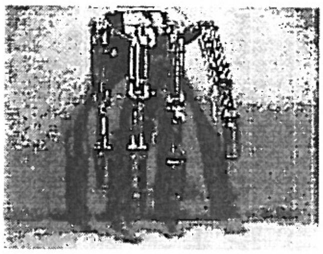

Fig. 11 The picture when the green box is used

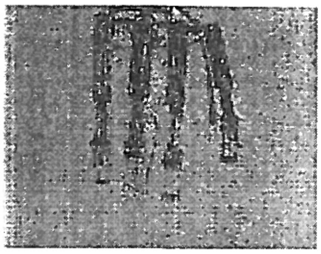

Fig. 12 The picture after decreasing the contrast from 128 to 10

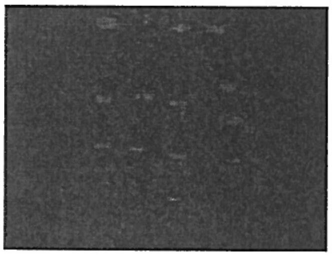

Fig. 13 The picture after decreasing the Brightness from 128 to 16

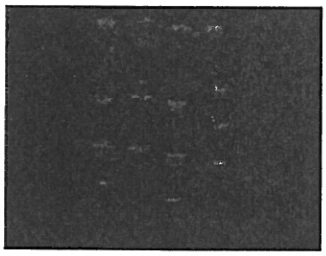

Fig. 14 The picture after increasing the saturation from 128 to 255

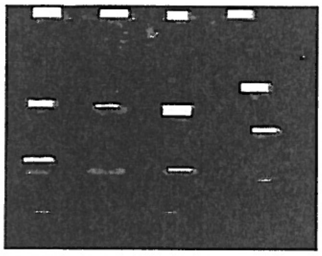

Fig. 15 The picture after the enlargement and recognition of joints

7. COMMUNICATION

Since in telemanipulation there can be a huge distance between the master and the slave operator, an effective way of communication should be chosen. With the huge and possible increasing spread of the Internet the TCP/IP protocol seemed to be the best solution. The communication between the visual interface and the picture recognition computer is carried by packets of 96 bytes, because at this stage 12 joints are controlled, and the position of each joint is described by an 8 byte floating value.

8. VISUAL INTERFACE

The software was developed to give visual feedback of a general hand in telemanipulation. Visual feedback is aimed at giving a quite real representation of the environment, and although at the moment this environment is far from complete, the program enables the user to wander in full three-dimensional space.

The program was designed to run either locally in full simulation mode or over TCP/IP connection, such as Internet or Local Area Network. In simulation mode, the program can be used to represent all the motions of the human hand. To accomplish it, both anatomical and mathematical models were built up, and these models were implemented in OpenGL. The mathematical deduction uses the Denavit-Hartenberg notation, because it can be quite well applied both for hands and in OpenGL. The transformation matrix of one of the fingers is given by (15):

$$\mathbf{T}_1 = \mathbf{A}_{11}^{0}(q_{11})\mathbf{A}_{12}^{1}(q_{12})\mathbf{A}_{13}^{2}(q_{13})\mathbf{A}_{14}^{3}(q_{14})\mathbf{A}_{15}^{4}(q_{15}) \tag{15}$$

where A-s are the homogeneous transformation matrices between two consecutive co-ordinate frames. The position and orientation of one fingertip can be calculated as:

$$\mathbf{P}_1 = \mathbf{T}_1 \quad \mathbf{X}^{15} \tag{16}$$

where X^{15} is the position vector of the fingertip in its local co-ordinate frame.

The position and orientation of all the fingertips with respect to the common base frame (the wrist), is given by (14) as a function of joint angles.

$$P=[f(\theta_{31},\theta_{32},\theta_{33},\theta_{44},\theta_{55})f(\theta_{31},\theta_{32},\theta_{33},\theta_{44},\theta_{55}) \\ f(\theta_{31},\theta_{32},\theta_{33},\theta_{44},\theta_{55})f(\theta_{31},\theta_{32},\theta_{33},\theta_{44},\theta_{55})] \tag{14}$$

Fig. 16 shows all the joints of a finger. To make possible the use of the software on slower personal computers, different models and display modes can be used.

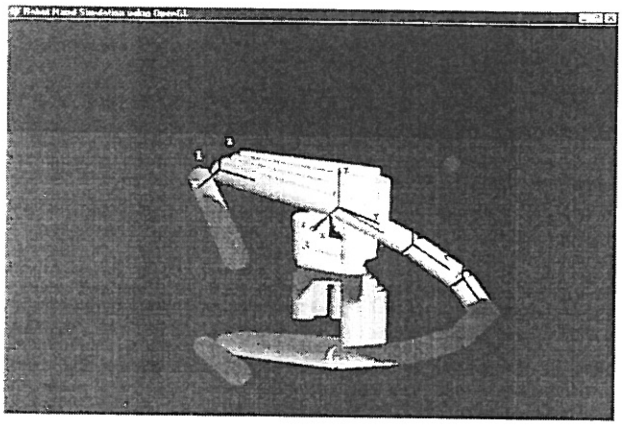

Fig.17 Computer animation of the robot hand

9. CONCLUSION

This paper proposed a dextrous hand as a slave device for telemanipulation. Using shape memory alloy (SMA) as an „artificial muscle" in robot mechanisms efficiently increases the degree of freedom offering much more flexibility than widely adopted hands, that are mostly driven by pneumatic or electric actuators. The SMA wires have much less weight and size than the conventionally used actuators. The construction is much simpler; also its price is comparable. However applying SMA actuators leads to a very difficult control problem, as the feature of SMA is strongly non-linear. This paper demonstrated that generalised neural networks are promoting solutions for control problems caused by SMA actuators.

ACKNOWLEDGEMENTS

The authors wish to thank the Office of Higher Education Support Programs (MKM FKFP 1092/1997), National Science Research Fund (OTKA T029026) and Control Research Group of Hungarian Academy of Science for their financial support

REFERENCES

Rojas R. (1996), Neural Networks, A Systematic Introduction. Springer-Verlag, Berlin.

Hornik,K. Stinchcombe, M. White,H. (1998) "Multi-layer Feedforward Networks are Universal Approxi-mators" Neural Networks 2. pp. 359-366.

Baranyi P, Y. Yam, H. Hashimoto, P. Korondi, P. Michelberger (2000a) "Approximation and Complexity Reduction of the Generalized Neural Network" Submitted to *IEEE Trans. on Fuzzy Systems*.

Baranyi, P. L.T.Kóczy and T.D.Gedeon (1998) "Improved Fuzzy and Neural Network Algorithms for word frequency Prediction in Document Filtering" *Journal of Advanced Computational Intelligence* Vol. 2 No. 3, pp 88-95.

Mizumoto M., (1990)"Fuzzy controls by Product-sum-gravity method" *Advancement of Fuzzy Theory and Systems in China and Japan*, Eds. Liu and Mizumoto, International Academic Publishers, c1.1-c1.4, 1990.

Zam,Y., P.Baranyi and C.T.Yang (2000) „Reduction of Fuzzy Rule Base Via Singular Value Decom-position" *IEEE Trans. on Fuzzy Systems*. Vol.: 7, No. 2, ISSN 1063-6706, pp. 120-131.

H. Hasimoto Korondi, P, P. T. Szemes,(2000) "Human Interfaces for Telemanipulation" Invited plenary paper for EPE-PEMC Conference, Kosice, Slovakia

Sheridan ,T.B.,(1989) "Telerobotics" Automatica, Vol. 25, No. 4, pp 487-507, .

Baranyi, P. K-F. Lei, Y. Yam (2000b)"Complexity Reduction of Singleton Based Neuro-fuzzy Algorithm", IEEE Conference on Systems Man and Cybernetics (IEEE SMC'2000), (submitted)

Lei,K-F, P. Baranyi, Y. Yam (2000c)"Complexity Reduction of Non-singleton Based Neuro-fuzzy Algorithm" IEEE Conference on Systems Man and Cybernetics (IEEE SMC'2000), (submitted)