Péter Korondi – Péter Zsíros – Fetah Kolonic *

HUMANOID TYPE HAND MOVED BY SHAPE MEMORY ALLOY

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.

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 sensings 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. The paper [7] 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.

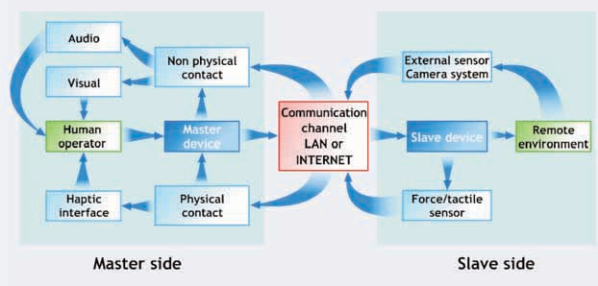


Fig. 1 Information streams of the Telemanipulation

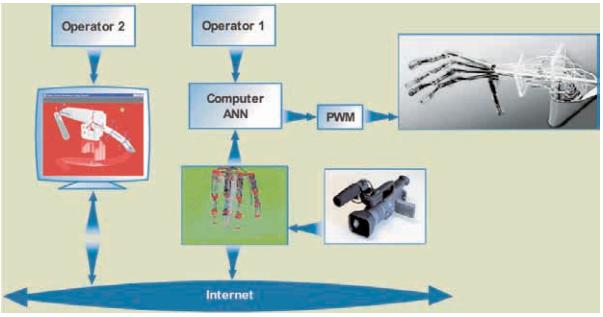


Fig. 2 Data flow of our system

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

* **Péter Korondi[1], Péter Zsíros[1], Fetah Kolonic[2]**
[1]Deptartment of Automation and Applied Informatics Budapest University of Technology and Economics, P.O.Box 91, H-1521, Budapest, Hungary, E-mail: korondi@elektro.get.bme.hu
[2]Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, HR-10000 Zagreb, Croatia, fetah.kolonic@fer.hr

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 – Distal Interphalangeal Predominant (arthritis)) 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. Generalised neural network

A generalised neural network is used to control the hand: The difference between a classical and a generalised neuron is that real numbers are used for the weights of a classical neuron, but the weights are fuzzy functions in the case of generalised neurons. A simple neuron is shown in Fig. 10 and a multi-layer generalised neural network will be discussed in Fig 7.

For each input of a neuron there is a *B* set, which can be fired by an antecedents set *A*. Each antecedent set corresponds 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:

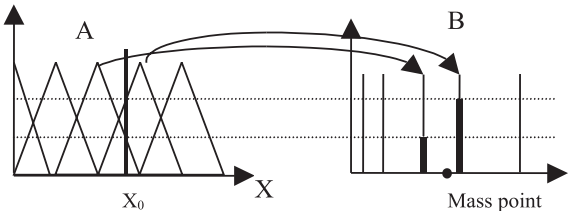


Fig. 4 Calculation of the neurones weight

The value of the mass-point will go into the summation part of the neuron (see in Fig 10). 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: Most functions (which are interesting for engineers) can be approximated by fuzzy functions. 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.

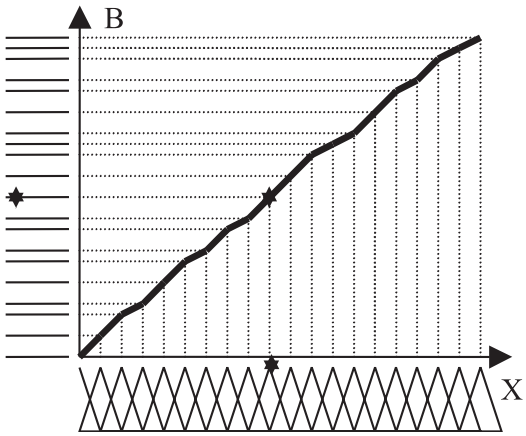


Fig. 5 Approximation of a function by the antecedent and consequent sets

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 network 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 neuron 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.

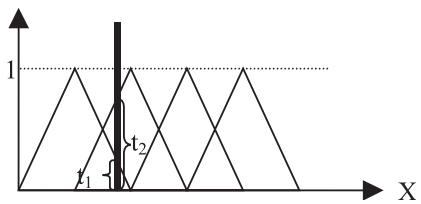


Fig. 6 Ruspini partition

$t_1 + t_2 = 1$ at any X.

4. Training of the generalised neural network

A generalised neural network is shown in Fig. 7. Using the notation introduced in Fig. 7, the outputs of the neural network can be calculated in the following way:

$$Y_j = \sum_i \sum_t A_{i,t}^{(2)}(X_i) \cdot B_{i,j,t}^{(2)} \tag{1}$$

where X_i is the output i-th neuron of the 1-st layer, $A_{i,t}^{(2)}$ is t-th the antecedents of i-th input in the second layer. Each antecedent set corresponds to all neurones. $B_{i,j,t}^{(2)}$ is the consequent set of the j-th neurone fired by $A_{i,t}^{(2)}$

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

$$\frac{\partial Y_j}{\partial X_i} = \frac{\partial \left(\sum_i \sum_t A_{i,t}(X_i) \cdot B_{i,j,t} \right)}{\partial X_i} = \sum_t A_{i,t}^{\prime(1)}(X_i) \cdot B_{i,j,t} \tag{6}$$

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

$$\frac{\partial h^2}{\partial B_{k,i,t}} = \sum_j \left[-2 \cdot h_j \cdot \left(\sum_t A_{i,t}^{\prime(2)}(Xi) \cdot B_{i,j,t} \right) \right] \cdot A_{k,t}^{(1)}(P_k) \tag{8}$$

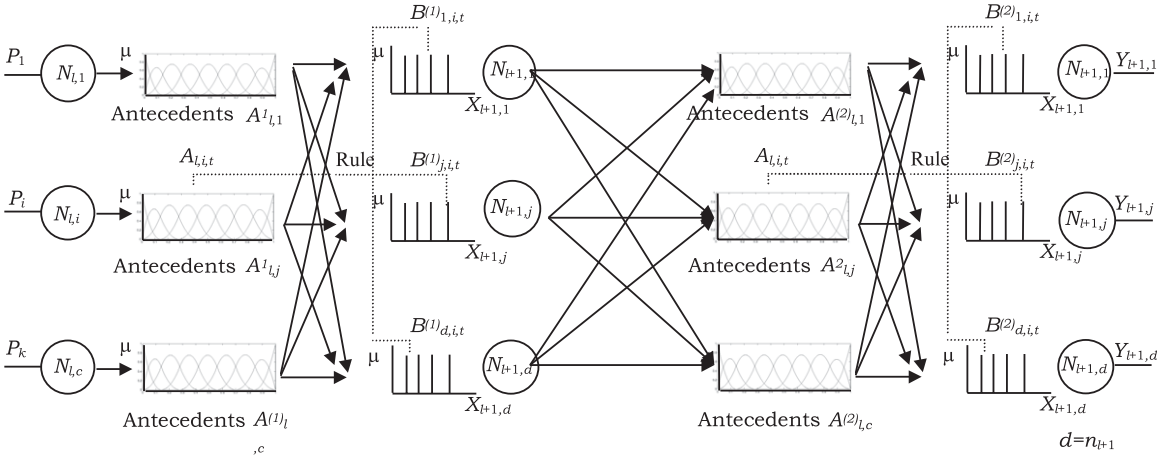


Fig. 7 The generalised neural network

$$Y_i = \sum_k \sum_t A_{k,t}^{(2)}(P_k) \cdot B_{k,i,t}^{(2)} \tag{2}$$

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?

$$\frac{\partial h^2}{\partial B_{k,i,t}} = ? \tag{3}$$

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:

$$\frac{\partial h^2}{\partial B_{k,i,t}} = \sum_j \frac{\partial h^2}{\partial Y_j} \cdot \frac{\partial Y_j}{\partial X_i} \cdot \frac{\partial X_i}{\partial B_{k,i,t}} =$$

$$= \frac{\partial (h_1^2 + h_2^2 + ... + h_n^2)}{\partial Y_j} \cdot \frac{\partial Y_j}{\partial X_i} \cdot \frac{\partial X_i}{\partial B_{k,i,t}} \tag{4}$$

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

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

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. Control of the hand

Pulse width modulation is applied to control the movement of the hand. Two different duty rations are used, one to move a certain part of the hand and one to hold it at a certain position (see Fig. 8 and 9). When the operator changes the reference position, the

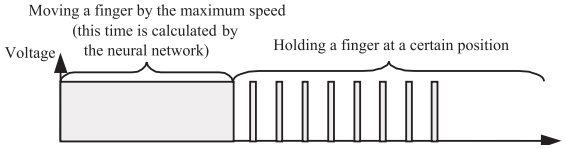


Fig. 8 Moving a finger with maximum speed

neural network calculates how long the voltage should be switched across the wire to move the hand into the new reference point as shown in Figs. 8 and 9. Since the condition (mainly the temperature) of the environment has great influence on the characteristic of SMA wires, the neural network must be retuned continuously.

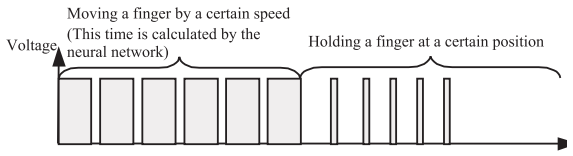


Fig. 9 Moving a finger with less than the maximal speed

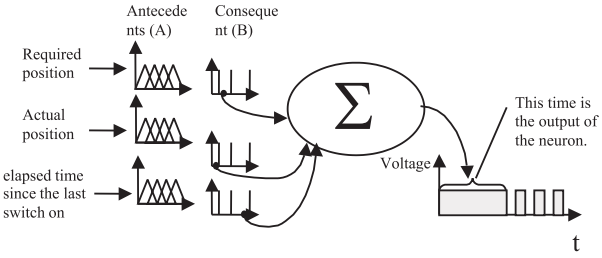
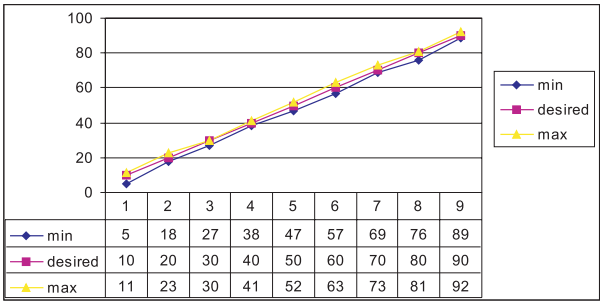


Fig. 10 Neuron model



	1	2	3	4	5	6	7	8	9
min	5	18	27	38	47	57	69	76	89
desired	10	20	30	40	50	60	70	80	90
max	11	23	30	41	52	63	73	81	92

Fig. 11 Experimental results (The desired angle is changing from 10 to 90 degree. The measurement is carried out in 10 times. The minimal and maximal measured angles are shown)

6. 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 added operation but considering the product operation, the computational requirement is characterised as:

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

P_μ indicates the calculation of the membership functions:

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

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

The main objectives of this section are to propose an algorithm that 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 12. The further proofs of this technique discussed by [9,10].

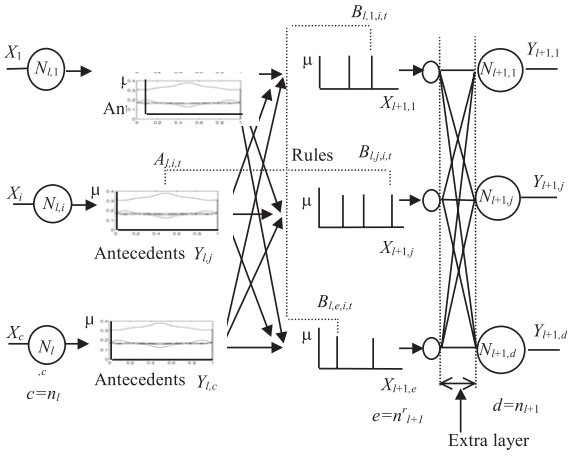


Fig. 12 Reduced neural network

7. 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 color 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 Handycam 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 ("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. The phases of the correction of the picture are shown in Figs. 15–18.

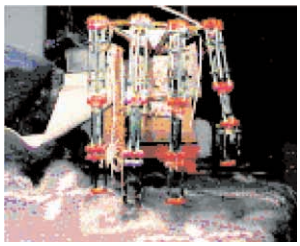


Fig. 13 The normal video picture

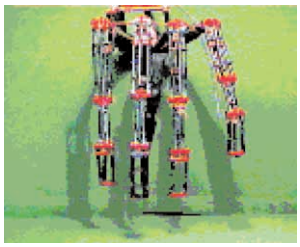


Fig. 14 The picture when the green box is used

Fig. 15 The picture after decreasing the contrast from 128 to 10

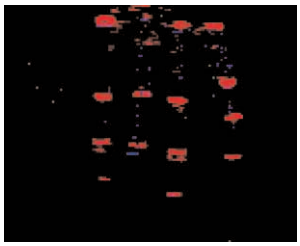


Fig. 16 The picture after decreasing the Brightness from 128 to 16

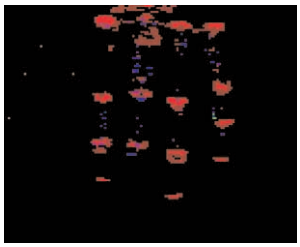


Fig. 17 The picture after increasing the saturation from 128 to 255

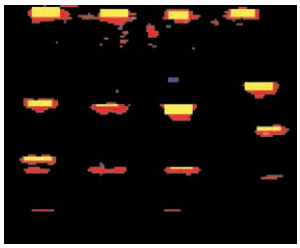


Fig. 18 The picture after the enlargement and recognition of joints

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 model uses several basic assumptions, these are:

- The base coordinate frame is the same for all fingers. This is used for having a completely general case, where the fingertip coordinates are calculated with respect to a common frame, for instance to the wrist. This common frame is considered to be at joint 0. This joint is used in order to have the possibility to extend the model at a later time if needed.
- One finger contains three joints, the first has two degrees of freedom, and the other two have only one, which means planar rotation. The position of the first joint is determined by a variable l_0, which is the distance between the first, common coordinate frame and the first joint.
- Along one finger, the value of d_i is 0, which means that the common normal line on the same line.
- Along one finger the value of a_i is the length of the link.
- The twist angle is taken into account only at the base coordinate frame and at the first joint, since lateral movement of the finger is also considered.

It is important that coordinate frame 0 is the same for all fingers, and is a common reference to the whole hand. The individual θ_i angles and l_0 variables determine the exact position and orientation of the next frame of the next joint in each finger. Between the two frames di is just equal to the length of the link, l_0. Since joint 0 is of the revolute type, q_0 is θ_1 the relationship between the two joints is (14).

$$X = A_1^0 X_1 \tag{14}$$

In (1) A_1^0 is the 4(4-transformation matrix between the two adjacent joints and X^s are position vectors in the respective coordinate frames. In a more detailed form (15, 16)

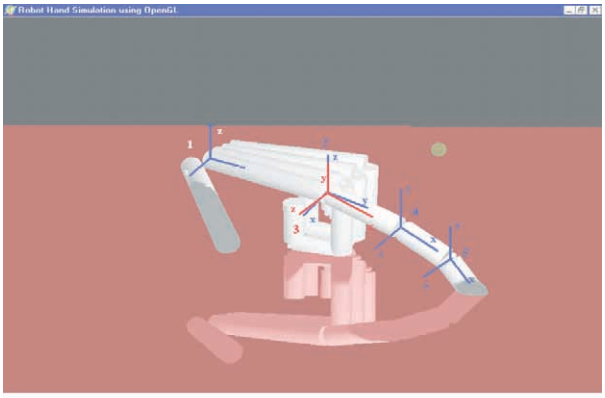


Fig. 19 Computer animation of the robot hand with the coordinate system

$$A_1^0 = \begin{bmatrix} cos\theta_1 & -sin\theta_1\, cos\alpha_1 & sin\theta_1\, sin\alpha_1 & a_0\, cos\theta_1 \\ sin\theta_1 & cos\theta_1\, cos\alpha_1 & -cos\theta_1\, cos\alpha_1 & a_0\, sin\theta_1 \\ 0 & sin\alpha_1 & sin\alpha_0 & d_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (15)$$

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta_1 & -sin\theta_1\, cos\alpha_1 & sin\theta_1\, sin\alpha_1 & a_0\, cos\theta_1 \\ sin\theta_1 & cos\theta_1\, cos\alpha_1 & -cos\theta_1\, cos\alpha_1 & a_0\, sin\theta_1 \\ 0 & sin\alpha_1 & sin\alpha_0 & d_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \quad (16)$$

As it was stated, in this case α is 0°, which means that its cosine is 1 and its sine is 0. For the two joints d is taken to be 0 and a is the length of the link. Substituting these values simplifies the previous equation and yields to (17).

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta_1 & sin\theta_1 & 0 & l_0\, cos\theta_1 \\ sin\theta_1 & cos\theta_1 & 0 & l_0\, sin\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \quad (17)$$

Now the position and the orientation of the coordinate frame at joint 1 is received, further transformations can be carried out.

Now there is a general formula for calculating the position of the fingertip with reference to the common base coordinate frame. (5) shows this, and it is very similar to (14). Matrix T contains the position and orientation of the tip in the base coordinate system.

$$T = A_1^0(q_1)\, A_2^1(q_2)\, A_3^2(q_3)\, A_4^3(q_4)\, A_5^4(q_5) \quad (18)$$

It is useful to state that joint 2 has two transformation matrices and frames because of the two possible axes of rotation.

$$P_1 = T_1 \quad X^{15} \quad (19)$$

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 (20) as a function of joint angles.

$$P = [f(\theta_{11}, \theta_{12}, \theta_{13}, \theta_{14}, \theta_{15}) f(\theta_{21}, \theta_{22}, \theta_{23}, \theta_{24}, \theta_{25}) \\ f(\theta_{31}, \theta_{32}, \theta_{33}, \theta_{34}, \theta_{35}) f(\theta_{41}, \theta_{42}, \theta_{43}, \theta_{44}, \theta_{45})] \quad (20)$$

The computer-aided animation of the robot hand is shown in Fig. 19.

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 National Science Research Fund (OTKA K62836), Control Research Group and János Bolyai Research Scholarship of Hungarian Academy of Science for their financial support and the support stemming from the Intergovernmental S & T Cooperation Program.

References

[1] ROJAS, R.: *Neural Networks, A Systematic Introduction*, Springer-Verlag, Berlin, 1996
[2] HORNIK, K., STINCHCOMBE, M., WHITE, H.: *Multi-layer Feedforward Networks are Universal Approximators*, Neural Networks 2., 1998, pp. 359–366

[3] BARANYI, P, YAM, Y., HASHIMOTO, H., KORONDI, P., MICHELBERGER, P.: *Approximation and Complexity Reduction of the Generalized Neural Network*, Submitted to IEEE Trans. on Fuzzy Systems, 2000
[4] BARANYI, P., KÓCZY, L. T., GEDEON, T. D.: *Improved Fuzzy and Neural Network Algorithms for word frequency Prediction in Document Filtering*, Journal of Advanced Computational Intelligence Vol. 2 No. 3, 1998, pp. 88–95
[5] MIZUMOTO M.: *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
[6] ZAM, Y., BARANYI, P., YANG, C. T.: *Reduction of Fuzzy Rule Base Via Singular Value Decomposition*, IEEE Trans. on Fuzzy Systems. Vol.: 7, No. 2, 2000, ISSN 1063-6706, pp. 120–131
[7] HASIMOTO, H., KORONDI, P, SZEMES, P. T.: *Human Interfaces for Telemanipulation*, Invited plenary paper for EPE-PEMC Conference, 2000, Kosice, Slovakia
[8] SHERIDAN, T. B.: *Telerobotics,* Automatica, Vol. 25, No. 4, 1989, pp. 487–507
[9] BARANYI, P., LEI, K. F., YAM, Y.: *Complexity Reduction of Singleton Based Neuro-fuzzy Algorithm,* IEEE Conference on Systems Man and Cybernetics (IEEE SMC'2000), submitted
[10] LEI, K. F., BARANYI, P., YAM, Y.: *Complexity Reduction of Non-singleton Based Neuro-fuzzy Algorithm,* IEEE Conference on Systems Man and Cybernetics (IEEE SMC'2000), submitted.