ISSN: 0128-7680
Pertanika J. Sci. & Technol. Supplement 12(2):191 - 200 (2004)

Application of Soft Computing to Serial Manipulator Kinematic Problems

[1]Ahmad Yusairi Bani Hashim, [2]Napsiah Ismail, [2]Megat Hamdan Megat Ahmad, [2]Abdel Magid Hamouda & [2]Mohd Rasid Osman

[1]Faculty of Manufacturing Engineering,
Kolej Universiti Teknikal Kebangsaan Malaysia
Locked Bag 1200, Ayer Keroh 75450, Melaka, Malaysia
[2]Department of Mechanical and Manufacturing Engineering
Faculty of Engineering, Universiti Putra Malaysia
43400 UPM, Serdang, Selangor, Malaysia

ABSTRAK

Kajian ini menggunakan elemen-elemen dalam kaedah *soft computing* untuk menentukan koordinat akhir bagi pengolah robot. Dengan gabungan tiga kaedah dalam cerdik buatan iaitu peraturan-peraturan fuzzy, rangkaian neural, dan algoritma genetik mewujudkan unit *soft computing* yang disulam khas untuk pengolah robot bersiri. Kinematik terus bagi pengolah ini dijadikan sebagai sistem kawalan suapan-terus dan unit *soft computing* pula menggantikan kinematik songsang dalam suapan-semula. Kejituan dalam menentukan koordinat akhir berlaku melalui proses pembelajaran dan kemudian dengan kinematik terus setelah unit *soft computing* mencadangkan masukan dan lelaran. Hasil uji kaji menunjukkan bahawa teknik yang diperkenalkan ini berupaya untuk memberi hasil yang memuaskan.

ABSTRACT

Inspired by the techniques in artificial intelligence, this research applies the constituent of artificial intelligence to perform manipulator positioning tasks. This work combines three methods in artificial intelligence: fuzzy rules, neural networks, and genetic algorithm to form the control block specifically designed to solve kinematic problems of a six degree-of-freedom serial manipulator. The direct kinematic of a manipulator is taken as the feedforward control and the artificial intelligence control block replaces the inverse kinematics when the reverse solving is done. Fine manipulator positioning is achieved from the learning stages executed by the artificial intelligence control block. It is shown experimentally that the technique proposed was capable of producing results with very low errors.

Keywords: Fuzzy logic, neural network, genetic algorithm, direct kinematics, inverse kinematics

INTRODUCTION

Robots are complex systems. Robotics itself is a broad field where a cluster of robots is referred to as industrial robots, or display robots that are typically found in motion pictures and theme parks. Whether it is a display-type robot or an industrial-type robot, the main goal for implementing robots or creating robots is to emulate the ability of bio-mechanical systems performing difficult tasks. This emulation leads to the design of highly complicated mechanical systems that are sufficient to emulate the capability of bio-mechanical systems such as the human arms.

A robot utilises a programming language as a platform that allows programmers to execute manipulations conforming to end users' requirements. There are different types of programming languages, but there is seen a need for intelligent behaviour where a robot is able to make a simple decision to achieve specific work. Mason (1998) inferred that at present, robots are unable to reason about physical processes and robot motions

have to be pre-programmed in complete detail. As a result, robots are extremely limited in the tasks they can perform. They are clumsy, virtually blind, and cannot react to unexpected events.

Robot manipulators are heterogeneous-controlled plants. In a well-structured industrial setting, the robot manipulator is subjected to structured and unstructured uncertainties. The structured uncertainties include inaccurate measurement of length, mass and inertia of the robot manipulator and motor torque constants. The unstructured uncertainties include neglected high-order modes of the manipulator and non-linear friction.

Existing robot arm control systems use a simple joint servomechanism. The servomechanism approach models the varying dynamics of a manipulator inadequately because it neglects the motion and configuration of the whole system of the whole arm mechanism. Changes in the parameter of the controlled system are significant enough to render conventional feedback control strategies ineffective. The result is reduced servo response speed and damping, limiting the precision and speed of the end-effector. This makes it appropriate only for limited-precision tasks. As a result, manipulators controlled this way move at slow speeds with unnecessary vibrations. Any significant performance gain in this area of robot arm control require the consideration of more efficient dynamic models, sophisticated control techniques, and the use of computer architectures.

In our work an approach is developed that proposes a method to obtain a set of joint angles as inputs to the direct kinematics of a six degree-of-freedom (DOF) serial manipulator for a pre-defined tool position. This is done by employing artificial intelligence (AI) techniques named as soft computing (SC). The traditional method of solving the problems is the use of the inverse kinematics (IK) formulation that is by manipulating the matrices of the direct kinematic representation.

SC has been introduced by Zadeh (1993) who claims that it is the new paradigm for an intelligent system that combines existing AI conception and methodologies such as that of fuzzy logic (FL), neural network (NN), and genetic algorithm (GA) to effectively use the computing program to perform intelligent tasks. He believes that the principal constituents of SC are FL, NN and probabilistic reasoning (PR) with the latter subsuming belief networks, GA, chaos theory and parts of learning theory.

FL is a logical system that aims at a formalization of approximate reasoning. In a wider sense, FL is synonymous with fuzzy set theory that is the theory of classes with unsharp boundaries; such a system consists of four important parts. They are the fuzzification interface, the knowledge unit, the decision-making unit, and the output defuzzification interface.

Artificial neural networks (ANN) as the name implies are the networking study, which models the human's neural system. ANN consists of a large number of neurons or simple units referrred to as neurodes (Kulkarni 1994). The fundamental feature of neural network (NN) composes of a large interconnected processing unit. These units resemble the biological neurons found in the human body. A general architecture of NN consists of inputs, hidden layers, and outputs. On the side of hidden layers, its weight is determined by combinations of layers defined.

GA is adaptive and probabilistic search algorithms based on natural selection and natural genetics (Johnson *et al.* 1995). It is a very efficient means for searching for a solution space. The concept behind GA is the genetic information that comes from the genetic codes. The genetic information forms strings that define a particular solution. If 110011 represents a genetic code, then a population of these codes corresponding to a number of individual solutions to the problem is created. There will be possibly some good solutions given by a population of genetic codes.

In the new millennium, computer-aided engineering will focus on its original objective, the integration of engineering functions and in particular, the co-ordinating function of manufacturing engineering. Engineering and its associated technologies are necessity evolving into a production support function, in which internal customers represent the end users in terms of quality (ease of manufacture), cost (robust designs and processes), and delivery (efficient communication)

Theoretical Background

The kinematic models of a six revolute joints serial manipulator is selected for testing our approach. The direct kinematic (DK) as in equation (1) is used to obtain the position of the tool attached to the very last section of the manipulator. DK is the process of calculating the position in space of the end of a linked structure, given the angles of all the joints, and there is only one solution (Koivo 1989). Again, in equation (1), the position of the tool is dictated by its orientation as well as the (x, y, z) coordinate. On the right-hand side of the equation are the transformation matrices. These matrices are combined from linkages within the manipulator system. Knowing parameters belonging to the transfomation matrices may result in the position tool if properly computed. The tools do the work on objects such as hold, grip, push, and touch. In our work the manner of how the tool performs is irrelevant to the expected findings. But if parameters such as the joints' angles are required with known tool position, the traditional inverse kinematics (IK) may be applied. Consider a robotic manipulator with joint variables $(\theta_1,\theta_2,\ldots,\theta_N)$. The relative position of the tool frame {T} with respect to the station frame is shown in expression (2). The DK then is expressed in (3). Equation (4) exhibits IK for the manipulator.

$$T_6 = {}^0A_1\,{}^1A_2\,{}^2A_3\,{}^3A_4\,{}^4A_5\,{}^5A_6 \tag{1}$$

$${}^S_T T_{(\theta_1,\ldots,\theta_N)} = {}^S_B T\, {}^0_1 T_{(a_1,\alpha_0,d_1)(\theta_1)}\, {}^1_2 T_{(a_1,\alpha_1,d_2)(\theta_2)} \cdots {}^{N-1}_N T_{(a_{N-1},\alpha_{N-1},d_N)(\theta_N)}\, {}^W_T T \tag{2}$$

$$(\theta_1,\ldots,\theta_N) \mapsto {}^S_T T(\theta_1,\ldots,\theta_N) \tag{3}$$

$${}^S_T T \mapsto \begin{pmatrix} \theta_1\left({}^S_T T\right) \\ \vdots \\ \theta_N\left({}^S_T T\right) \end{pmatrix} \tag{4}$$

In IK, the objective is to obtain joints' angles with required tool position. For example, if the angles of each joint that are required to achieve a specific position are unknown, tool position and orientation of the joints will have to be defined. Upon successfully completing the computation, the joints' angles will then be obtained. But there is a fact that to solve inverse kinematic problems potentially poses difficulty in getting a unique solution.

This is especially true for increased DOF manipulator and of this case. As matrices get bigger, inversing the matrix is a complex process. Unless the inversed matrix is square, then a unique solution is attained. Otherwise, multiple solutions or infinite solutions may exist. After this section, we will bring the reader to understand our approach on solving the inverse kinematic problem differently. We employed three methods in artificial intelligence (AI): FL, NN, and GA. They are combined into a

control block (Cb) boundary. While FL, NN, and GP are themselves within their own sub-control block (Cbs) boundaries. Hence, within Cb are three Cbs. *Fig. 1* illustrates the situation.

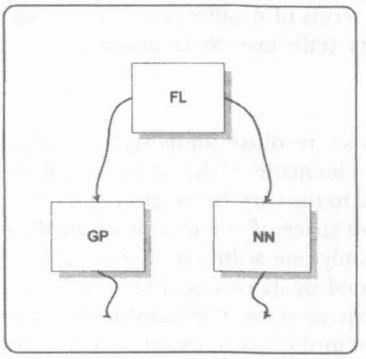

Fig. 1: The main control block with sub-control block within

In order to solve kinematic problem of a serial manipulator, we have created the procedure to complete the task. It begins with a DK equation, where the user defines the required tool position. The required position is $P(X, Y, Z)$ and the obtained position is $p(x, y, z)$. The user enters values for joints angles $(q_1,q_2,\ldots,q_N)$. Upon computation using DK, the tool position is compared to the required position. When the computed value approaches proximity to the required position, the process ends. All attempts for $(\theta_1,\theta_2,\ldots,\theta_N)$ and $p(x, y, z)$ are recorded. Joint angles' data from the record are considered inputs to FL. *Fig. 2* exemplifies the procedure.

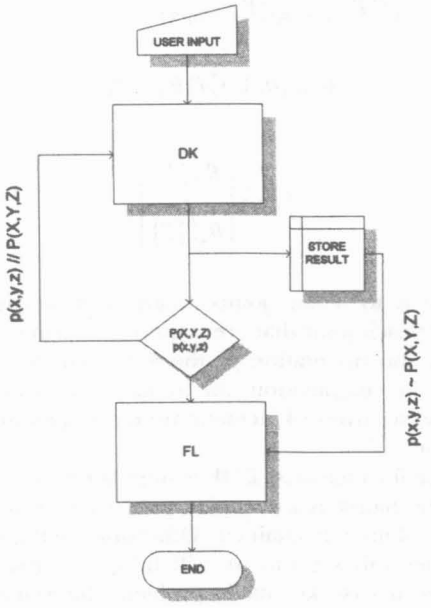

Fig. 2: The procedure for obtaining data

The FL starts with the fuzzification of all input data based on pre-defined fuzzifications. Its linguistic variable (LV) is defined by linguistic descriptions that portray the nature of outputs from the equation (1). The LVs are declared as negative-large (*neglarge*) for (-1), negative-small (*negsmall*) for (-2), positive-small (*possmall*) for (1), and positive-large (*poslarge*) for (2). All declared LVs correspond to equation (5) to (9). This means result from equation [i] is translated by fuzzification using unusual equations (5) to (9).

$$[-2]=\left\{px_i\in\Re\middle|px_i\le PX_{[-2]}\right\}=\left\{py_i\in\Re\middle|py_i\le PY_{[-2]}\right\}=\left\{pz_i\in\Re\middle|pz_i\le PZ_{[-2]}\right\} \quad (5)$$

$$[-1]=\left\{px_i\in\Re\middle|PX_{[-2]}\prec px_i\prec PX_{[0]}\right\}=\left\{py_i\in\Re\middle|PY_{[-2]}\prec py_i\prec PY_{[0]}\right\}=\left\{pz_i\in\Re\middle|PZ_{[-2]}\prec pz_i\prec PZ_{[0]}\right\} \quad (6)$$

$$[0]=\left\{px_i\in\Re\middle|px_i=PX_{[0]}\right\}=\left\{py_i\in\Re\middle|py_i=PY_{[0]}\right\}=\left\{pz_i\in\Re\middle|pz_i=PZ_{[0]}\right\} \quad (7)$$

$$[1]=\left\{px_i\in\Re\middle|PX_{[0]}\prec px_i\prec PX_{[2]}\right\}=\left\{py_i\in\Re\middle|PY_{[0]}\prec py_i\prec PY_{[2]}\right\}=\left\{pz_i\in\Re\middle|PZ_{[0]}\prec pz_i\prec PZ_{[2]}\right\} \quad (8)$$

$$[2]=\left\{px_i\in\Re\middle|px_i\ge PX_{[2]}\right\}=\left\{py_i\in\Re\middle|py_i\ge PY_{[2]}\right\}=\left\{pz_i\in\Re\middle|pz_i\ge PZ_{[2]}\right\} \quad (9)$$

Further process goes on with fuzzified data that is translated to another type of coding. The data is converted to genetic codes (Gc) when going through GA. The conversion is made such that an existed LV is considered having a digit 1 while 0 indicated non-existent. For example, if $C_{x,1}$ has LV=(-1), other LVs are non-existent. Therefore, the Gc = {0,1,0,0,0}. Equation (10) exhibits how Gc is represented while equation (11) shows how Gc may be written. The result form conversion is grouped according to respective types. There are type *x*, type *y*, and type *z*. Every type consists of candidates (C). Total up C in *x*, *y*, and *z* becomes the population. All C assumed having both X and Y chomosomes. This X and Y do not resemble the ones we mentioned above. Having both chromosomes mean that they are able to function as male or female according to situations. In other words, unions of each C in *x* to a C in *y* and *z* are possible. In short, the union of any C can be represented by ($Cx\cap Cy$), ($Cx\cap Cz$), and ($Cy\cap Cz$). The symbol $\cap$ does not portray intersection or union as in a theory of sets. It decribes the process of matching C or the marriages (M). All C in the population unioned twice. Only if a couple has the same genetic codes, can they have a child (ch). Mathematical representation (12) explains any union of C can only result in one child or none. It is agreed that any unions that produce a ch reflects the best combinations of ancestors. The ancestors are those who at first begin the population. However any unions that do not produce a ch are considered a reflection of unfit ancestors of theirs and are hence abandoned. The GA used in this work selects data according to definitions made by users.

$$Gc=\{j_i\in bit\middle|j_i=\begin{cases}^{0}if\ LV & non-exist\\ ^{0}if\ LV & exist\end{cases};i=1,2,3,4,5\} \quad (10)$$

$$Gc_k = \{j_1, j_2, j_3, j_4, j_5\}_k \tag{11}$$

$$M = (Cx \cap Cy), (Cx \cap Cz), (Cy \cap Cz) = \begin{cases} 0 \\ or \\ 1 \end{cases} \tag{12}$$

The fuzzified data are also processed by use of NN. The contrast between GA and NN is that NN does not convert any inputs but merely procures and processes. The processing of the data is done by employing the fundamental weighted sum NN method. The NN model consists of absolute inputs, summations of the inputs, the dot product weight function where equal weight is given to all inputs. The hidden layers are determined by the number of possible input combinations. It has been found that the results with the least values obtained from NN are the most excellent combinations which reflect the initial inputs, whereas other values dictate unfavourable combinations and are hence discarded. Our NN searches data according to definitions made by us. Equation (13) is our NN mathematical model where ϖ is the weight.

$$S = \sum_{i=1}^{n} \left(\left| LV_{xi} \right\| LV_{yi} \left\| LV_{zi} \right| \right) (\varpi) \tag{13}$$

Once the GA and NN complete their tasks, the results are then compared. Any results that match the ones of GA and NN are kept for the next tasks. This is the action of data filteration. Comparison of data is made by first returning to the original representation $p(x, y, z)$ and $(\theta_1, \theta_2, \ldots, \theta_N)$. The selected and filtered results $(\theta_1, \theta_2, \ldots, \theta_N)$ are used as inputs to the DK equation. The statement in the algorithm below is the defuzzification for iteration process. In the command <> and e should be placed a proper inequality or equality and tolerance value respectively, which are determined from the results' patterns. If the actual position in *x*-axis is <> (e.g. less than) the desired position in *x*-axis; and the actual position in *y*-axis is <> (e.g. more than) the desired position in *y*-axis; and the actual position in *z*-axis is <> (e.g. less than or equal) the desired position in *z*-axis; then the initial input angle to DK should add or minus the tolerance. The symbol *q* is equivalent to *q* where it is the joint angle. The $p(x, y, z)$ are compared to expected position $P(X, Y, Z)$ while iterations are made until the position approaches the closest proximity to the expected position.

```
REPEAT
If px <> PX AND py <> PY AND pz <> PZ
      THEN q = q ± e;
UNTIL px Æ PX AND py ÆPY AND pz Æ PZ;
```

EXPERIMENTAL PROCESS

The experiment began with inputs as joints' angles to the DK. A target coordinate was assumed to be achieved. This was the expected tool position (Tp). By trials, a set of joint angles was randomly selected as the input to the DK, which resulted in the position of the tool upon computation. Positions obtained through trials were called the experimental positions (Ep). Unless the position of the tool conformed to the Tp, the process was

being repeated until it achieved an acceptable closeness to Tp. Comparisons between the Ep and the Tp were described in position errors. There were five experiments conducted. These experiments were named by romanized letters A, B, C, D, and E. Only experiment A is discussed here. Each experiment, however, has a different number of tests. Experiment A consisted of six tests, experiment B embodied 14 tests, experiment C contained 17 tests, experiment D had two tests, and experiment E carried six tests. Occurrence of different numbers in tests for every experiment was due to the trials performed when defining the inputs to the DK. It took 14 tests in experiment B to get the closeness in tool position as compared to the Tp, but only two tests in experiment D to achieve proximity in tool position as compared to the Tp. The selection and search for suitable data was initiated when a number tests have satisfactorily accomplished proximity to the Tp.

RESULTS

There were 6 tests in experiment *A*. Every test had ten trials. As a sum, there were 60 trials in experiment *A*. The Tp wanted in this experiment was $p(550.00, 550.00, 550.00)$. Testing on selection of the most qualified combinations using NN and GA by applying different position ranges for LVs is shown in Table 1.

TABLE 1
Linguistic variables (LVs) and position ranges

LV	-2	-1	0	1	2
px	x£530	530<x<550	x=550	550<x<570	x≥570
py	y£530	530<y<550	y=550	550<y<570	y≥570
pz	z£530	530<y<550	z=550	550<z<570	z≥570

Note: All co-ordinates are represented with two decimal points.
In the table these co-ordinates are displayed as integers.

Table 2 shows results acquired from NN. Looking at the overall results, the largest values equal 60 and the smallest value equals 38. The least value resulting from NN computation is said to be the finest test/trial, hence it is selected. Thus, the finest trials (as seen shaded) are 5, 6, and 9 of test number 6, and trial 2 of test number 5.

TABLE 2
NN output

Test	Trial										
	1	2	3	4	5	6	7	8	9	10	Ovl
1	6	6	6	6	6	6	6	6	6	6	60
2	6	6	6	6	6	6	6	6	6	6	60
3	6	6	6	6	5	5	6	6	6	6	58
4	5	5	5	5	5	5	5	5	5	6	54
5	6	3	5	5	5	6	5	5	5	4	48
6	4	4	4	4	3	3	4	4	3	5	38

Note: The least value for the Overall (ovl) shows the finest trial

Table 3 represents results obtained from GA. The table portrays genetic codes of population A by samples of unioned couples. An overall (Ovl) result that exhibits Q means the sample is qualified, whereas a DQ means disqualified sample. Therefore, in Table 3 there exist 5 qualified samples, and 1 disqualified sample.

TABLE 3
GA output

S	Couple										
	1	2	3	4	5	6	7	8	9	10	Ovl
1	1	0	1	1	0	1	1	1	1	1	Q
2	1	1	1	1	0	1	1	1	1	1	Q
3	1	1	1	1	0	0	0	0	0	1	Q
4	0	0	0	0	0	0	0	0	0	0	DQ
5	0	1	0	0	0	0	0	0	0	0	Q
6	0	0	0	0	1	1	0	0	1	0	Q

S = Sample, Q = Qualified, DQ = Disqualified

After comparisons among the results from Tables 2 and 3 were made, it was seen that NN and GA proposed common results for trial 2 of test number 5, and trials 5, 6, 9 of test number 6, or couple 2 of S number 5, and couples 5, 6, 9 of S number 6. Therefore, data from these trials were used as the initial conditions for defuzzification rules. Table 4 exhibits the proposed initial condition agreed both by NN and GA.

TABLE 4
Initial conditions after defuzzification

Joint	Test,trial (angle,degrees)			
	A5,2	A6,5	A6,6	A6,9
1	45.00	45.00	45.00	45.00
2	47.00	46.00	46.00	46.00
3	60.00	61.00	61.50	62.00
4	20.00	20.00	20.00	20.00
5	60.00	62.00	61.50	61.00
6	60.00	60.00	60.00	60.00

The joint angle patterns seen in Table 4 shows unchanged values for joint angles 1, 4, and 6. Joint angles 2 and 5 show a decreasing pattern whereas joint angle 3 displays an increasing pattern. Thus, this pattern was taken to set for defuzzification rules. Table 5 depicts the defuzzification rules for experiment A. The iteration steps for joint angles 2 and 5 were negative 0.00302 and negative 0.00530 respectively. The iteration step for joint angle 3 was positive 0.00621 because of the increasing pattern viewed. Values for iteration step were chosen by estimation. The defuzzification rule obtained is shown below:

$$\text{IF } px \le PX \text{ AND } py \le PY \text{ AND } pz \le PZ \text{ THEN} \begin{cases} q_2 = q_2 - 0.00302 \\ q_3 = q_3 - 0.00621 \\ q_5 = q_5 - 0.00530 \end{cases} \quad (14)$$

By applying the rules into the algorithm above, the final results are shown in Table 5. The results suggest that in order for the manipulator to reach the Tp, P(550.00,550.00,550.00); joint angles of 45.00º, 44.14º, 62.86º, 20.00º, 57.14º, and 60.00º for joint 1 through 6 should become the input values for DK. But the actual position compared to the Ep gave some error (Position error II). Error on px was 0.004%, error on py was 0.006%, and error on pz was 0.002%. We also show the result for the joints' angles using IK which is the inverse of the D-H representation. In order to get the theoretical tool position that is P(550.00,550.00,550.00), the IK proposes the joints' angles of 45.00º, 41.64º, 71.03º, 20.02º, 50.66º, and 56.83º. Errors dictated above are based on tool position only.

TABLE 5
Results

Joint	Angle (Initial)	Angle (Final)	Angle (Inverse Kinematics)	PX 550.00 px_o 535.95	PY 550.00 py_o 535.53	PZ 550.00 pz_o 539.25
1	45.00	45.00	45.00	px_f	py_f	pz_f
2	47.00	41.64	41.64	550.02	549.65	550.13
3	60.00	71.03	71.03		Position error I	
4	20.00	20.00	22.02	2.55%	2.63%	1.95%
5	60.00	50.59	50.66		Position error II	
6	60.00	60.00	56.83	0.004%	0.006%	0.002%

Notes: PX, PY, PZ : Theoretical position.
px_o, py_o, pz_o: Experimental (initial) position.
px_f, py_f, pz_f: Experimental (final) position.
Position error I: Experimental (initial) as compared to theoretical.
Position error II: Experimental (final) as compared to theoretical.

CONCLUSION

We have suggested another approach to solving kinematic problems for a serial manipulator. This approach provides an altenative to solving difficult IK problems in the of six DOF and we believe that it can also be applied to similar problems where DOF is more than six. However, we knew that there were weaknesses to this approach that require refinement. Readers who are into this field of research are invited to enhance this approach. We pinpoint sections where improvement may be carried out. The first is the filtration of data where mutual agreement among methods in AI is employed. In fact, Ng *et al.* (2002) in their work exhibited that genetic programming was able to solve some automatically nontrivial control problems. The second is the statement used in the iteration process where it can be developed as another method in numerical analysis.

REFERENCES

NG, K. L. and R. JOHNSON. 2002. Evolving programs and solution using genetic programming with application to learning and adaptive control. *Journal of Intelligent and Robotic Systems* **19**: 289-307.

WANG, W. and P. BRUNN. 2000. An effective genetic algorithm for job shop scheduling. In *Proceeding Institution of Mechanical Engineers* **214:** 293-300. Part B.

MASON, M. 1998. *Robotics Doctoral Program Guide.* USA: Pennsylvania.

ZADEH, L. A. 1993. *Fuzzy Logic, Neural Networks and Soft Computing.* Berkeley, USA: University of California.

KURKARNI. 1994. *Artificial Neural Networks for Imaging Understanding.* New York: Van Nostrand Reinhold.

JOHNSON, J. and P. PICTON. 1995. *Concepts in Artificial Intelligence.* **2**. Oxford: Butterworth Heinemann.

KOIVO, A. J. 1989. *Fundamentals for Control of Robotic Manipulator.* New York: John Wiley & Sons.