An Hyperincursive Method for the Solution of the Inverse Kinematics of Industrial Robots Based on Neural Networks and Genetic Algorithms

Roberto Bussola Rodolfo Faglia Monica Tiboni
Mechanical Engineering Department
Università degli Studi di Brescia
Via Branze 38, 25123 Brescia, ITALY
Phone: +39 030 3715401 Fax: +39 030 3702448
e-mail: bussola@bsing.ing.unibs.it, rfaglia@bsing.ing.unibs.it,
mtiboni@bsing.ing.unibs.it

Abstract
The robotic inverse kinematic problem can be rightly classified as a very felt theme in the field of robotics. Many studies have been carried out in order to find new methods for the solution of the problem as alternatives to the traditional ones. In particular, every method able to improve the calculation speed is more and more appreciated.
In the present paper an innovative method for the numerical inversion of non linear equations sets is shown. The approach is based on some procedures typical of the soft-computing area. In particular, the inverse kinematic problem is solved by a Neural Network optimised by means of a Genetic Algorithm acting inside an Hyperincursive scheme.
After the introduction of the methodology developed, the paper shows some results obtained on a SCARA robot; they appear very good in terms of computational speed, even if the solution precision is not high near the boundaries of the working area.
Keywords: Robotics; Numerical inversion; Neural Network; Genetic Algorithm; Hyperincursion.

1 Introduction

The robotic inverse kinematic problem, that is the inversion of the system of equations that gives the gripper's pose (position and orientation) in the working space as function of the co-ordinates of the gripper in the joint space, appears a very felt problem in the field of robotics. (Paul, 1981; Coiffet, 1981; Vukobratovic, 1989).
Many studies have been carried out in the direction to find methods for the solution of the problem alternatives to the traditional ones, analytical and numerical.
The symbolic definition of the relation which allows to pass from the external co-ordinates to the inner ones (analytical approach), has the advantage to put in evidence the number of solutions of the problem and, usually, it gives origin to very speed calculation algorithms; on the other hand, such a approach is not possible for every kind of geometry of the robot.

On the contrary, the generation of the local inversion by means of a numerical method, when it converges, can be applied to all the manipulators, it gives one solution only and, if high accuracy is requested, it is generally slower than the analytical one.
In the present paper an innovative method for the inversion of non linear equations sets, is shown. (Other interesting new methodologies of the kind can be found in Fukuda and Shimojima, 1997; Alsina and Geholt, 1994; Dreiseitl and Kubik, 1994). The studied approach is based on some procedures typical of the Soft Computing field (Holland, 1992; Floreano, 1996; Back, 1996; Haykin, 1999), Neural Network and Genetic Algorithms in particular, ruled by an hyperincursive feedback cognitive process (Dubois 1996, 1998). The application of the new methodology for the solution of the inverse kinematic problem for industrial robots appears a good alternative to classical methods: i) it gives a global solution; ii) the calculation speed is very high (when the Neural Network weights are established); iii) it can be, in principle, applied to all the manipulators, being not influenced by the robot geometry.

2 The Inverse Kinematic Problem and the SCARA Robot

In the present work we will consider the inverse kinematic problem for usual industrial manipulators, which can be modelled as open chains of rigid bodies.
The kinematic problem can be formalised in the following way: being $\boldsymbol{u}$ the vector of the external co-ordinates of the gripper (displacement and orientation of the gripper in its working space) and $\boldsymbol{q}$ the vector of the inner co-ordinates or joint variables, we have:

DIRECT KINEMATIC PROBLEM INVERSE KINEMATIC PROBLEM

$$\boldsymbol{u} = f(\boldsymbol{q}) \qquad\qquad \boldsymbol{q} = f^{-1}(\boldsymbol{u}) \qquad (1, 2)$$

Equations that constitute set given on the left side are often called 'position equations' and they allow to find $\boldsymbol{u}$ when $\boldsymbol{q}$ is known (direct kinematic problem); their complexity depends on the architecture of the robot, in particular on some dimensions of the links and on the kind of the coupling pairs (revolute or prismatic) that characterise the robot structure.
The relation on the right hand side represents the formalisation of the inverse transformation; solving the inverse kinematic problem means, assigned vector $\boldsymbol{u} = \boldsymbol{u}'$, to determine the corresponding vector $\boldsymbol{q}'$.
Whereas the solution of the direct kinematic problem for serial robots is normally very simple, the solution of the inverse kinematic problem shows some obstacles (greatly documented in literature), intrinsic in the formulation of the problem itself, which are briefly reminded: the non linearity of the position equations, the presence of singular configurations, the difficult choice of angular variables and of the related units of measure and the contemporary presence of angular and linear variables, usually characterised by range of variability significantly different.

To test the adequacy of the new method hereafter described to solve the above stated problems, we considered a simple two degrees of freedom SCARA Robot.
The SCARA robot is a two arms, two revolute joints robot and can be modelled as shown in Fig. 1.
Being ***u*** and ***q*** vectors for such a robot made as follows (3, 4), the system of two equations in two variables (unknown) which describes the direct kinematic is represented by eq. (5).

$$\boldsymbol{u} = \begin{pmatrix} x \\ y \end{pmatrix} \qquad \boldsymbol{q} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{3, 4}$$

$$\begin{cases} x = l_1 \cdot \cos\alpha + l_2 \cdot \cos(\alpha + \beta) \\ y = l_1 \cdot \operatorname{sen}\alpha + l_2 \cdot \operatorname{sen}(\alpha + \beta) \end{cases} \tag{5}$$

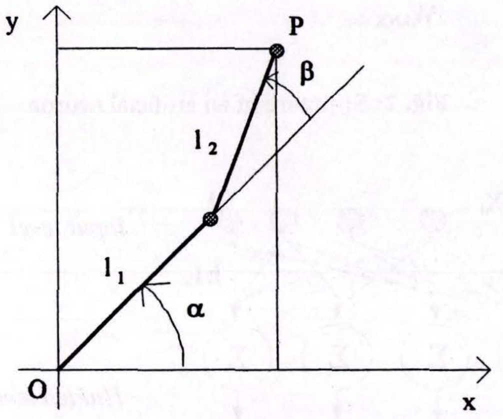

Fig. 1: The scheme of a SCARA robot and the system of position equations.

In the simulating phase we will consider a SCARA robot having both links lengths equal to *0.5 m*.

3 The Neural Network Used

Artificial Neural Networks are information-processing systems that belong to the connessionistic theory, born from the neurological studies on the human brain way of computing.
From these studies emerged that the way of information processing of the human brain is completely different from that of the conventional digital computers; whereas computers are sequential machines, brain is formed by a very large number of elementary units

(neurons) where the stored information is distributed, processed and then, re-formed. As Neural Networks we, thus, mean a parallel distributed processor formed by artificial neurons, where the knowledge is achieved through a learning procedure and where the strengths of the interneuron connections (synaptic weights) are used for storing the knowledge. Each neuron can be modelled as shown in fig. 2.

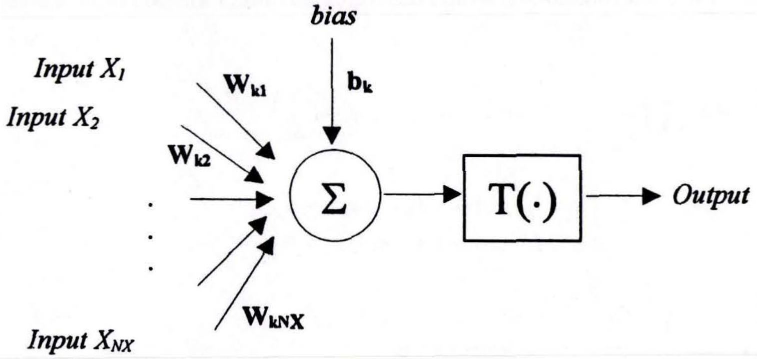

Fig. 2: Structure of an artificial neuron.

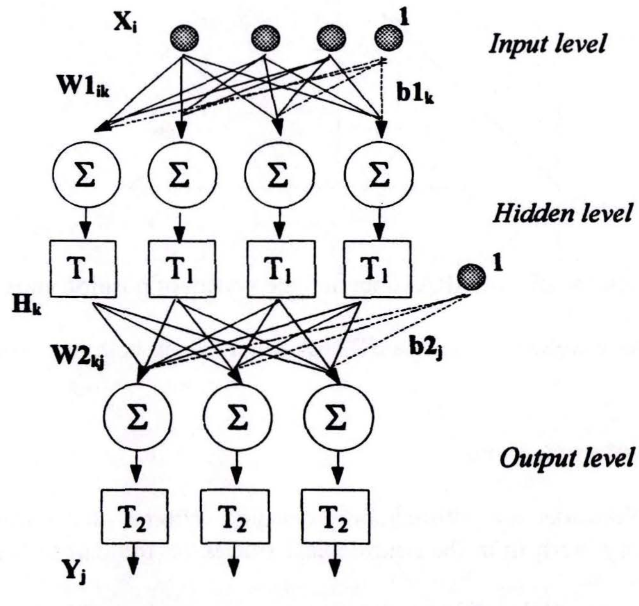

Fig. 3: Structure of a three level Neural Network.

The neuron sums the products between each input signal (X_i) and the related synaptic weight, adding to this partial result also a bias value (b_k) and, applying then to the value so obtained the activation function, which is thought to limit the amplitude of the output of the neuron.
Varying the neurons distribution and the typology of the used connections, we can create many different architectures and many different models of artificial Neural Networks, each one characterised by typical capabilities.
The network architecture that appears as the best one for the studied problem, in authors' opinion, is a feed-forward type, characterised by the fact that each neuron is connected only with neurons of neighbouring layers and connections are all unidirectional, from input to output nodes.
Let us consider, for simplicity, a network made up of three levels (with just one hidden layer): the first one (input level) is formed by *NX* elements, the hidden layer by *NH* neurons and the output level is constituted of *NY* units.
In the net architecture shown in Fig. 3, the bias values for each neuron of the hidden and output layers are considered as the weights for fictitious input units, whose value is assumed equal to one.
As transfer functions, a sigmoid for the neurons of the hidden level ($T1(.)$) and a linear function ($T2(.)$) for the output layer elements have been used, the analytical formalisation of which is:

$$T1(x) = \frac{2}{1+e^{-x}} - 1 \qquad\qquad T2(x) = x \qquad (6, 7)$$

Assigned inputs values (X_i), outputs Y_j can be calculated through the following two steps:

$$H_k = T1\left(\sum_{i=1}^{NX} W1_{ik} X_i + b1_k\right) \qquad\qquad Y_j = T2\left(\sum_{k=1}^{NH} W2_{kj} H_k + b2_j\right). \qquad (8, 9)$$

This procedure is called execution phase.

Such a Neural Network is able to learn an arbitrary function which links input variables (independent variables) to output variables (dependent variables). This capability is achieved in a training phase. To train a Neural Network means to propose to the net a proper set of examples, made of different values for the input variables and of the corresponding values of the dependent variables; during each iteration we modify the synaptic weights of the net in order to minimise the error due to the execution of the proposed examples. To obtain this, several training techniques are known in literature, for example the back-propagation (the best known), the learning with momentum, etc. (Haykin 1999, Floreano 1996).

4 The Genetic Algorithm Used

The optimisation procedure used belongs to a non-conventional class of algorithms called 'evolutionary' due to their characteristic to emulate the natural selection process of an animal species.
In details, the Genetic Algorithm is based on the principles that the individuals of a certain species, during the evolution process, improve their characteristics to fit particular environment conditions.
Mathematically, the algorithm operates on groups (populations) of vector variables (individuals) everyone associated with a quality index (fitness) evaluated on the basis of the specific functional to be optimised (environment). At the end of the process, the optimised solution will be the vector which presents the highest value of the quality index.

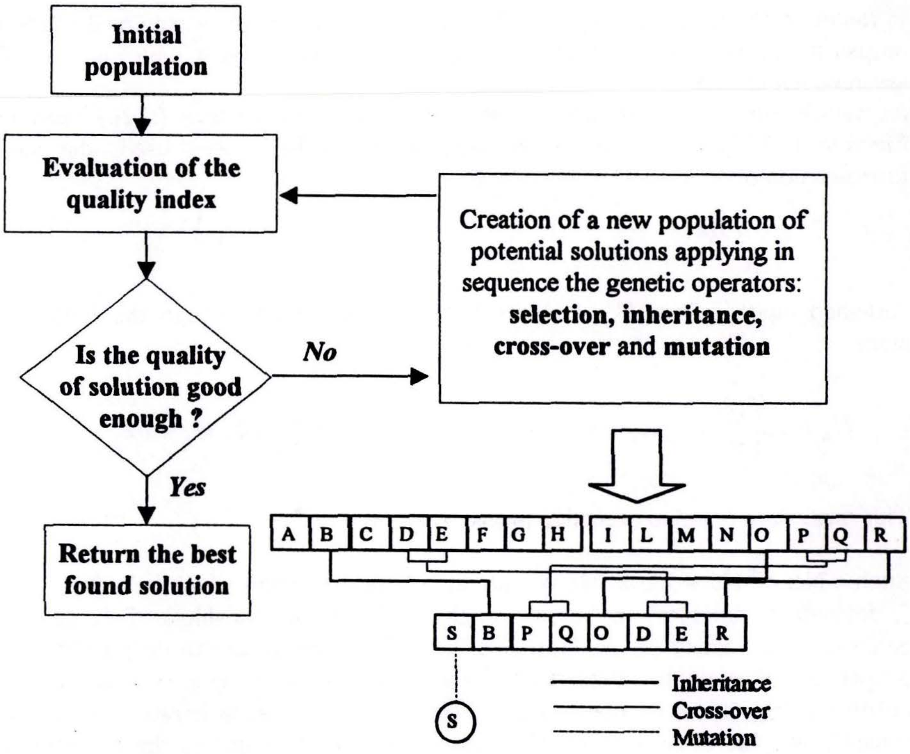
Initial population
Evaluation of the quality index
Creation of a new population of potential solutions applying in sequence the genetic operators: selection, inheritance, cross-over and mutation
Is the quality of solution good enough ?
No
Yes
Return the best found solution
A B C D E F G H I L M N O P Q R
S B P Q O D E R
S
Inheritance
Cross-over
Mutation

Fig. 4 – Flow chart and action scheme of the main operators of the Genetic Algorithm used.

In particular, the algorithm starts from an initial class of random solution esteems and then it builds new groups of vectors manipulating and combining their elements (genes). This is realised sequentially by applying some genetics operators named selection, inheritance, cross-over and mutation (reproductive process). These operators select some vectors from each groups of solutions so to make easier the selection of that ones with highest quality index; then their elements are manipulated and some random components are introduced to realise a new genetic patrimony. In this way, after a high number of iterations, every zone of the functional definition field is analysed.
The convergence of the method is guaranteed by the reintroduction, in each new group of solutions, of the best one found in every previous iteration; on the other hand, every new class of solutions is generally closer and closer to the optimum. (A more detailed description of the particular technique used by the authors can be found in Bussola and Tiboni, 1999).
The elaboration process is stopped when a predefined quality index is achieved, or when the optimisation solution does not improve at all during a pre-fixed number of consecutive iterations.
Fig. 4 shows the flow chart of a classical Genetic Algorithm with the action scheme of the main operators involved in the iterative process.

5 The Hyperincursive Method of Solution of the Inverse Kinematic Problem

In previous works (see Faglia and Tiboni, 1997, for example), the authors inquired into the possibility of teaching to a back propagation Neural Network the inverse kinematic equations, which link the working space co-ordinates with the joint space variables of an industrial robot.

The main problem of this approach is the creation of the examples on which to make the net learning. We proved that if we produce a matrix of *NE* (number of examples) made up by α and β values, randomly distributed into interval $[0, 2\pi]$ and then we calculate the corresponding x and y values with the direct kinematics equations, the Neural Network appears not able to learn the link between the values.
We can understand such a behaviour if we think that in this kind of examples two different mathematical solutions of the problem are present at the same time: the robot, in fact, is able to reach the same point P by two different sets of values α and β and a back propagation Neural Network is not able to understand this. Again, submitting to the network's attention examples with very similar (or equal at least) (x, y) values corresponding to different couples (α, β), we cause confusion in the training phase and compromise the Neural Network learning capability.

On the other hand, a one hidden layer Neural Network showed itself able to learn the mathematical relation between (x, y) and (α, β) under condition that the given examples consider only one solution.

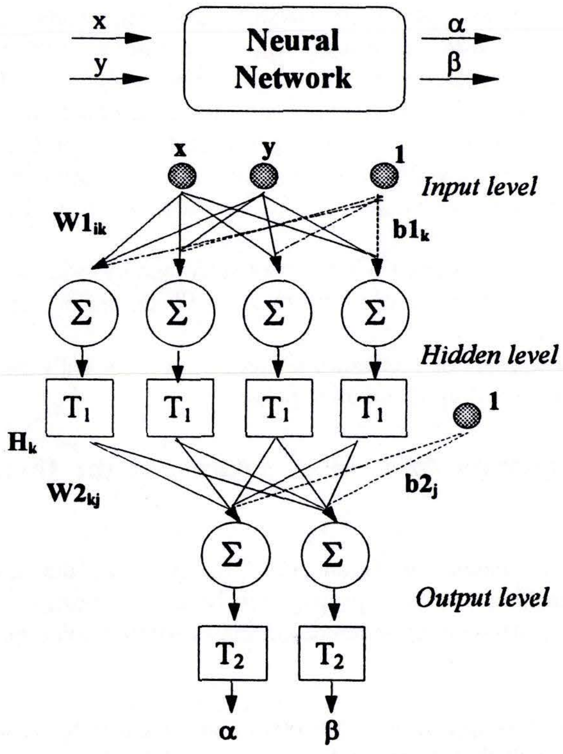

Fig. 5: The architecture of the Neural Network tested.

On this basis, and inspired by the mathematical methodology called Hyperincursion (see Dubois, 1996, 1998), which involves in the evolution of a system not only the past and the present events, but also the future ones, the authors developed an alternative way of solution, which seems able to avoid a-priori the multiple solutions problem.
The basic idea of the Hyperincursion is very well expressed by its author, Prof. Dubois (1998), with the following words: "Classically, feedback processes are based on recursive loops where the future state of a system is computed from the present and past states. With the new concept of incursion, an inclusive recursion, the **future** state of a system is taken into account for computing this future state in a self-referential way. The future state is computed from the mathematical model of system."

In our case, the Net is calibrated by using in series a Direct Kinematics module which utilises the output of the Net itself and simulates the robot future behaviour; the Genetic Algorithm acts on the reference signal *(x, y)* and on the future signal *(x', y')* generated by the robot, controlled by the Net.
Let's suppose to know only the values of the co-ordinates of the point to be reached and to ignore the related values of α' and β': obviously, in this case, the procedure to set the weights and biases of the Neural Network cannot be the back propagation method. Therefore, the training phase must be carried out in another way; a possible solution by an iterative scheme is pictured in Fig. 6.

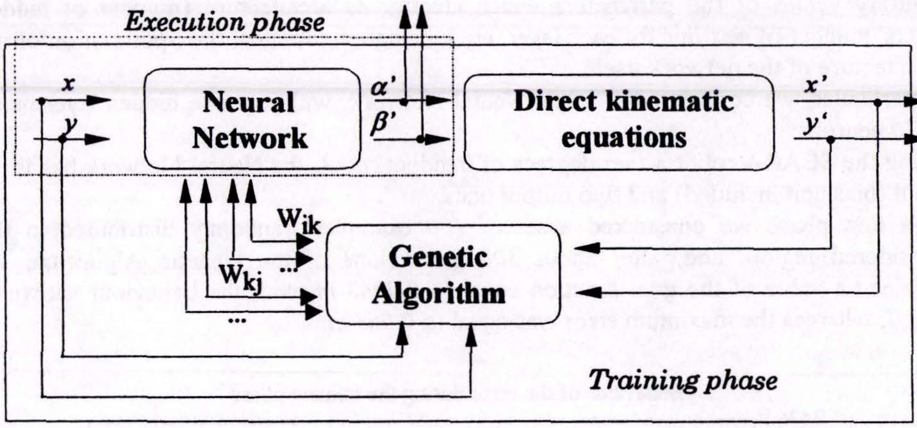

Fig. 6: The scheme of the structure developed to determine the weights and the biases of the Neural Network. Note that the Net is calibrated by the Genetic Algorithm on the basis of a future use in a module solving the Direct Kinematic problem.

It is possible to note that the weights of the net are determined by the action of a Genetic Algorithm working on the values that would be actually reached by the robot (which performs the direct kinematic equations) if the net would be used in its execution phase. In other words, we calibrate the net upon what it will be its **future** use.
During the iterations the Genetic Algorithm changes the net weights in order to minimise function

$$F(x,x',y,y') = \left(\frac{1}{NE} \cdot \sum_{i=1}^{NE} \sqrt{(x'_i - x_i)^2 + (y'_i - y_i)^2} \right) \qquad (10)$$

which represents the distance between the position which would be reached by the system for a specific set of weights of the net and the exact position. At the end of the process the optimal weights set and the net biases will be established.

Once the weights and the biases of the net have been found, when a new couple of x and y value is proposed as input, the execution phase of the Neural Network gives the corresponding α and β values.

6 Results of the Application of the Method to a SCARA Robot

In the application of the described hyperincursive method to a SCARA robot, due to the symmetry of the working area, we restricted our study to the first quadrant.
In the actual study of capability of the method, we considered a Neural Network with arbitrary values of the parameters which identify its architecture (number of hidden layers, number of neurons for each layer, etc.), being a future goal the optimisation of the architecture of the network itself.
In particular, we considered a simple Neural Network, with just one hidden layer made of *10* neurons.
Being the SCARA robot a two degrees of freedom robot, the Neural Network has three input (bias unit included) and two output units.
In a first phase we considered a set of *100* examples, randomly distributed in the considered region, and, after about *300* generations of the Genetic Algorithm, we obtained a value of the goal function equal to *0.0083 m*, with the behaviour shown in Fig. 7, whereas the maximum error was equal to *0.063 m*.

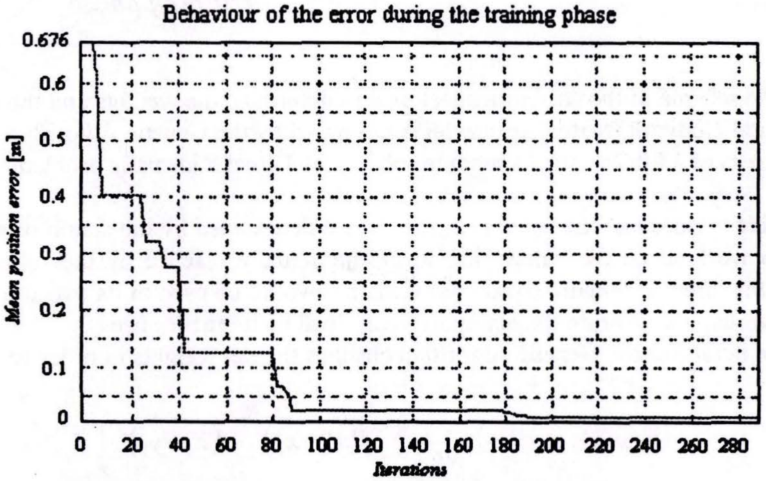

Fig. 7: Behaviour of the mean position error valued on the examples during the training phase

The first good result obtained with the training of the network is the capability of generalisation; tested, in fact, on a set of about *2500* couples (x, y), the mean value for the positioning error obtained was *0.009 m*, a little higher than the one obtained with the training examples, but surely acceptable, thinking that the network was not still optimised.

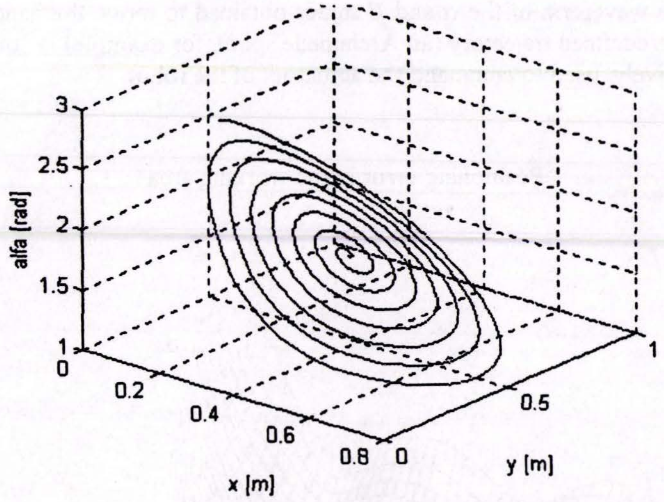

Fig. 8: Behaviour of the co-ordinates of the joints during the execution of a spiral trajectory

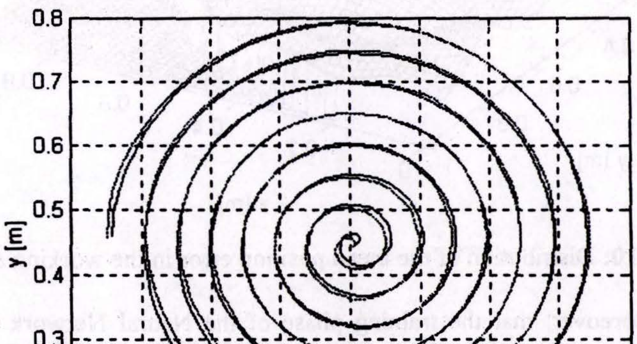

Fig. 9: Accuracy in the reproduction of a spiral trajectory.

In this way, the net behaves as a good path interpolating system.
With the described procedure, in fact, the Neural Network naturally evolves towards one solution only, without falling into the instability observed when both the possible solutions were present inside the given examples. The speed of calculation is very high and, this is very important, the solution found is 'global': how we can observe, in fact, from Fig. 8, the waveform of the α and β angles obtained to move the hand-effector of the robot on a predefined trajectory (an Archimede spiral, for example) is continuous and could be, effectively, used to command the actuators of the robot.

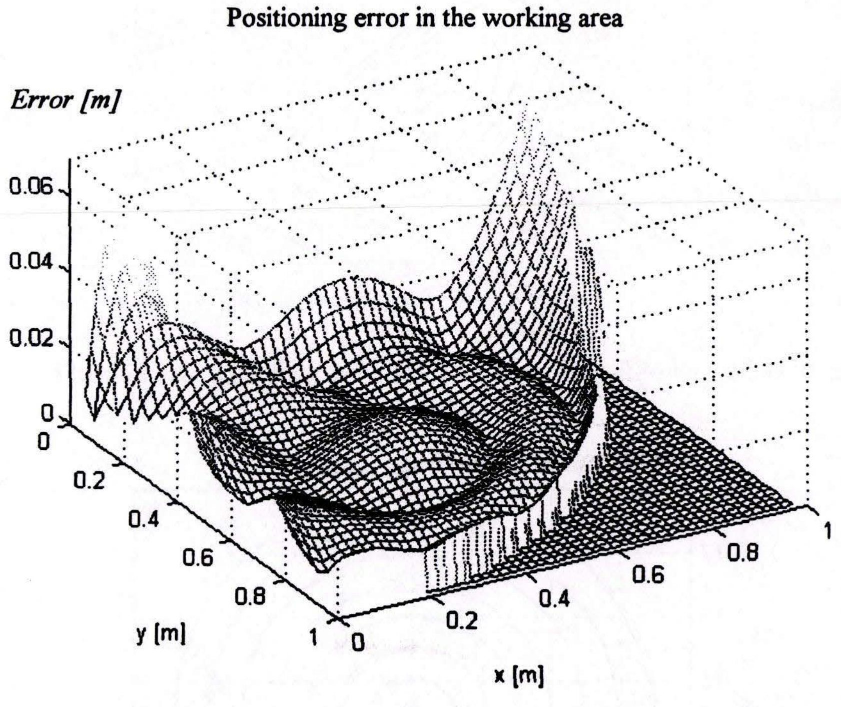

Fig. 10: Distribution of the mean position error in the working area.

We observed, moreover, that the training phase of the Neural Network in the whole quadrant gives not so good performances near its boundaries (sub-region *a*), but interesting results in its inner region (sub-region *b*).
Fig. 10 shows the distribution of the error in the training region of the network: the error assumes low values in the inner region and significantly increases moving towards the boundaries. The mean error of position valued on *5000* points distributed in a region of the first quadrant, reduced of a coastal strip *0.05 m* wide (hereafter named sub-region *a*) has been obtained equal to *0.0063 m*, sensibly smaller than the *0.009 m* of the whole

quadrant, and the maximum error was ***0.0419 m***, instead of ***0.063 m***; if we consider a inner region (sub-region ***b***), neglecting a coastal strip ***0.01 m*** wide the average error decreases up to ***0.0047 m*** and the maximum one to ***0.0195 m***.
Such a behaviour is, probably, due to the bad ability of extrapolation of the network, the examples distribution on the border of the learning region being asymmetrical.

To overlap the problem, the working space has been divided into five sub-regions, so that each sub-region is the internal part of a larger zone, where a proper Neural Network has been trained (see Fig. 11).

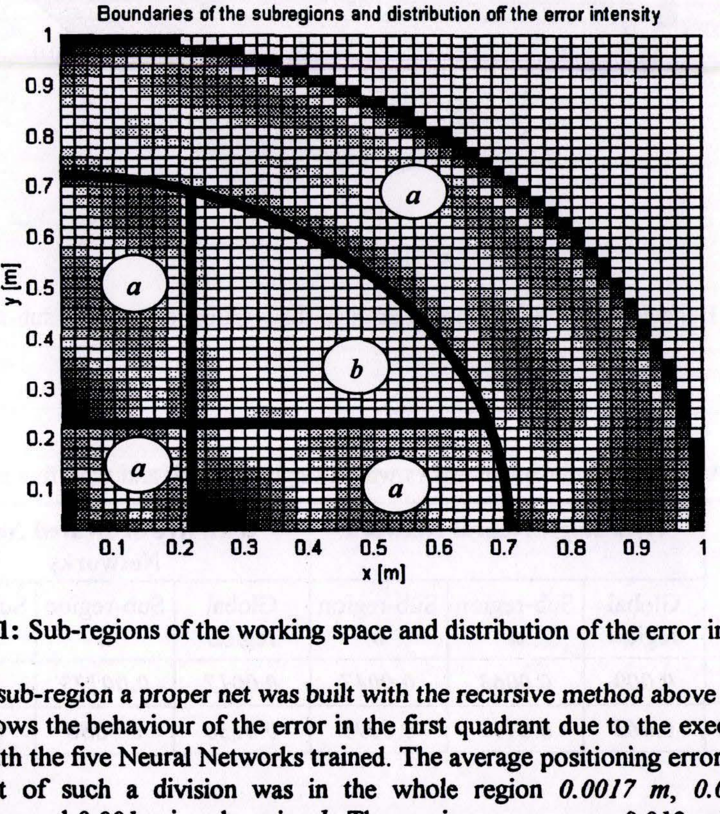

Fig. 11: Sub-regions of the working space and distribution of the error intensity.

For every sub-region a proper net was built with the recursive method above described. Fig. 12 shows the behaviour of the error in the first quadrant due to the execution phase realised with the five Neural Networks trained. The average positioning error distribution consequent of such a division was in the whole region ***0.0017 m***, ***0.00123 m*** in sub-region ***a*** and ***0.001 m*** in sub-region ***b***. The maximum error was ***0.013 m*** in the global region, ***0.0086 m*** in case ***a*** and ***0.0052 m*** in case ***b***. In Table 1 the average and maximum errors just mentioned are compared with the ones obtained with one network only.
The improvement obtained with the partition of the working area can be simply observed by the comparison between Fig. 9 and Fig. 13, which show the reproduction of the same spiral trajectory in the previous case and in the new one. The improvement achieved can be clearly appreciated.

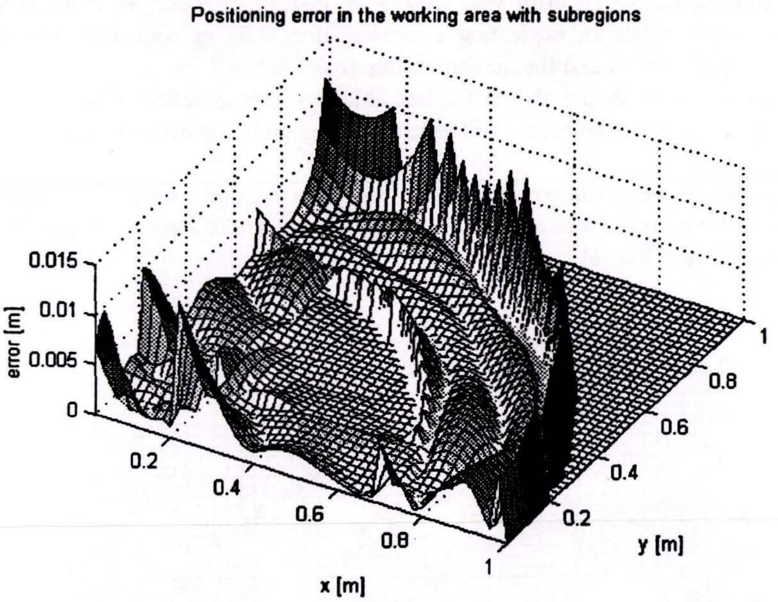

Fig. 12: Behaviour of the positioning error in the working space with sub-regions.

Table 1: Average and maximum errors with a single network and with five networks.

Error value *[m]*	**With single Neural Network**			**With five dedicated Neural Networks**		
	Global region	Sub-region *a*	Sub-region *b*	Global region	Sub-region *a*	Sub-region *b*
Medium	*0.009*	*0.0063*	*0.0047*	*0.0017*	*0.00123*	*0.001*
Maximum	*0.063*	*0.0419*	*0.0195*	*0.0130*	*0.0086*	*0.0052*

The results obtained are not really bad if we consider that no any analytical process has been used at all during the procedure and that no any optimisation of the networks architecture has been developed.

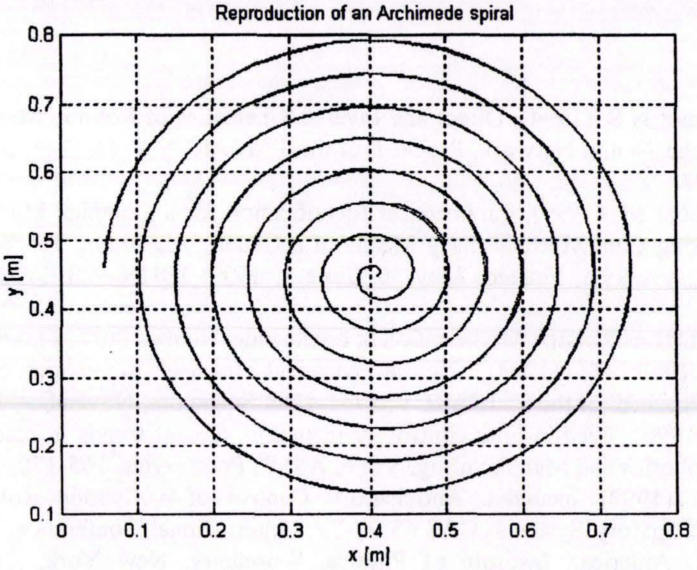

Fig. 13: Accuracy in the reproduction of a spiral trajectory with five networks.

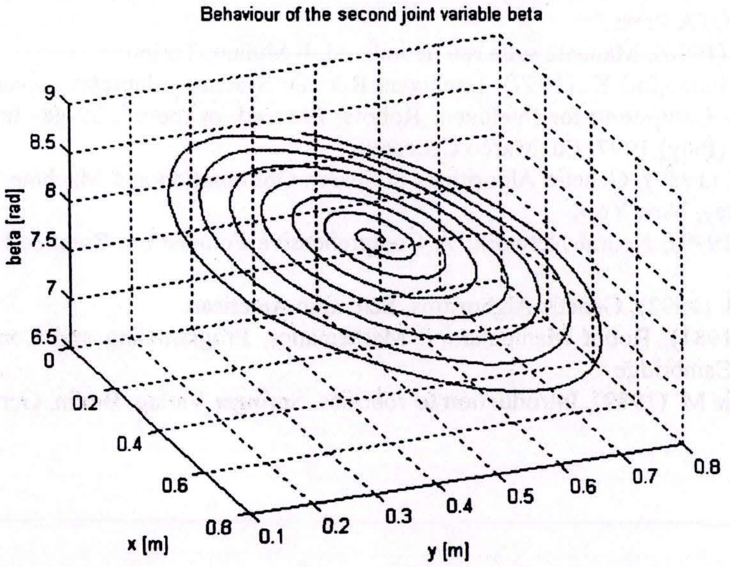

Fig. 14: Behaviour of the β angle during the execution of the spiral trajectory.

References

Alsina P.J., Gehlot N.S. (1994), Direct and Inverse Kinematics of Robotic Manipulators Based on Modular Neural Network, Proceed. of the 3rd ICARCV,1743-1747, Singapore, November, 1994.
Bussola R., Tiboni M. (1999), Parametrical Identification for a Complex Mathematical Model of Indexing Cam Mechanismsny Means of a Genetic Algorithm, Proceedings of EUROGEN99, Jyväskylä, Finland, May 30- June 3, 1999. JOHN WILEY & SONS, LTD.
Coiffet P. (1981), Les Robots. Modelisation et commande. Hermes Publ., France.
Dreiseitl S., Kubik T. (1994), Neural-Processed Inverse Kinematic of Robot Manipulators, Proceed. of the 3rd ICARCV, 1751-1784,Singapore, November, 1994.
Dubois D.M. (1996), Feed-in-time control by incursion. Recent trends in research and applications. Robotics and Manifacturing, Vol.6, ASME Press series. 165-170, 1996.
Dubois D. M. (1998) Incursive Anticipatory Control of a Chaotic Robot Arm, Computing Anticipatory Systems: CASYS'97-First International Conference, edited by D. M. Dubois, American Institute of Physics, Woodbury, New York, Conference Proocedings 437, pp.406-417.
Faglia R., Tiboni M. (1997), Solving the inverse kinematic problem of industrial manipulators by a back propagation Neural Network, Proceed. of the 5th IASTED Int. Conf. on Robotics and Manifacturing, 233-236, Cancun (Mexico), May, 1997. IASTED/IACTA Press.
Floreano D. (1996), Manuale sulle reti neurali, ed. Il Mulino, Torino.
Fukuda T., Shimojima K. (1997), Intelligent Robotic Systems:Adaptation, Learning and Evolutionary Computing for Intelligent Robots, Proceed. of the 6th, 39-48, Int. RAAD '97, Cassino (Italy),1997. Ed. Marco Ceccarelli.
Goldberg D. (1989), Genetic Algorithms in Search Optimisation and Machine Learning, Edison Wesley, New York.
Haykin S. (1999), Neural Network: A Comprehensive Foundation, Prentice-Hall, New Jersey.
Holland J. H. (1992), Genetic Alghoritms, Scientific American.
Paul R.P. (1981), Robot Manipulators: Mathematics, Programming and Control. The MIT Press, Cambridge.
Vukobratovic M. (1989), Introduction to robotics. Springer Verlag, Berlin, Germany.