Robotica (1997) volume 15, pp 3–10. © 1997 Cambridge University Press

# Obstacle avoidance inverse kinematics solution of redundant robots by neural networks

*Ziqiang Mao & †T.C. Hsia

**SUMMARY**

This paper investigates the neural network approach to solve the inverse kinematics problem of redundant robot manipulators in an environment with obstacles. The solution technique proposed requires only the knowledge of the robot forward kinematics functions and the neural network is trained in the inverse modeling manner. Training algorithms for both the obstacle free case and the obstacle avoidance case are developed. For the obstacle free case, sample points can be selected in the work space as training patterns for the neural network. For the obstacle avoidance case, the training algorithm is augmented with a distance penalty function. A ball-covering object modeling technique is employed to calculate the distances between the robot links and the objects in the work space. It is shown that this technique is very computationally efficient. Extensive simulation results are presented to illustrate the success of the proposed solution schemes. Experimental results performed on a PUMA 560 robot manipulator is also presented.

KEYWORDS: Neural networks; Obstacle avoidance; Redundant robots; Inverse kinematics.

## 1. INTRODUCTION

Solving inverse kinematics of a manipulator is an important problem in robotics. The main difficulty in solving such problems is that they are highly nonlinear and there exist multiple solutions. Each solution provides a different manipulator posture. For the majority of the industrial robots, which are kinematically non-redundant, closed form solutions do exist and the number of solutions are known to be finite. However, the number of inverse kinematic solutions for a redundant manipulator is infinite, and closed form solutions are impossible to find in general.

Kinematically redundant robot manipulators have attracted much attentions in robotics research because of their ability to increase dexterity and avoid obstacles. Solving the inverse kinematic problem of redundant robot manipulators is generally complicated by the fact that it has an infinite number of joint space solutions to a given end-effector Cartesian space position. To obtain an obstacle avoidance solution, it also involves object modeling and distance computation. Clearly closed form solution is neither possible nor necessarily computationally desirable. Thus numerical solution methods are commonly used.[1–6] In these methods, the inverse kinematic solution redundancy is resolved by using the dexterity measures and/or Moore-Penrose pseudoinverse of the Jacobian matrix.[7] In addition, obstacle avoidance joint solution can be obtained by optimizing certain objective functions.[8]

Because a multi-layer neural network can form any continuous nonlinear mapping from one domain to another, neural network methods have been studied by researchers to model the forward and inverse kinematics mapping of robot manipulators. It is shown that a multi-layer neural network with sinusoidal activation functions can be successfully trained to model a given forward kinematic mapping.[9] Modeling of inverse kinematics mappings by multi-layer neural network has also been successfully demonstrated.[10–12] It is shown that for a three-degree-of-freedom manipulator, elbow-up and elbow-down inverse kinematic solutions can be trained by choosing different initial weights in the neural network.[10] Other neural network algorithm such as Kohonen's self-organizing mapping algorithm[13] and cooperative/competitive neural networks[14,15] have been used to model the inverse kinematics mappings.

In this paper, we investigate the idea of solving the inverse kinematics problem of redundant robots using a neural network. Specifically, we develop methodologies to train a neural network to learn the inverse kinematics solution from a given forward kinematics of a manipulator. A training algorithm for finding obstacle avoidance inverse kinematics solutions is developed. The solution scheme is presented in which the user can conveniently control the manipulator posture of an inverse kinematic solution by assigning appropriate initial conditions of the joint variables. This feature is particularly important to redundant robot arms where redundancy can be resolved in favor of desirable arm postures satisfying collision free properties.

The result is derived based on a simple ball-covering object modeling technique which allows distance functions to be calculated very efficiently. The properties of the proposed scheme are validated by simulation results and experimentation. Thus a successfully trained neural network using the proposed schemes can provide accurate continuous inverse kinematics solutions on-line in a given robot work space with or without obstacles.

In the following sections, we will first introduce the neural network solution scheme and the formulation of the inverse kinematic problem for the obstacle free case.

* Intel Corporation, 200 Mission College Blvd, Santa Clara, CA 95052 (USA).
† Robotics Research Laboratory, Department of Electrical and Computer Engineering, University of California, Davis, CA, 95616 (USA).

Then the obstacle avoidance inverse kinematic solution is introduced. Finally, we present both the simulation and experimental results to demonstrate the performance of the proposed algorithms.

## 2. OBSTACLE FREE NEURAL NETWORK SOLUTION SCHEME

The inverse kinematics problem we are considering can be stated as follows: Given the end-effector location (position and orientation) of a robot manipulator whose forward kinematics are known, find a set of joint variable solution, among all possible solutions, which positions the end-effector at the specified location while the corresponding robot posture possesses certain desirable properties. A neural network solution scheme for the obstacle free case is proposed as shown in Figure 1. In this scheme, $\mathbf{x}_d$ and $\mathbf{x}$ are the $m \times 1$ desired and actual end-effector location vectors respectively, $\mathbf{q}$ is the $n \times 1$ robot joint variable vector, $\mathbf{q}_0$ is the prespecified initial value of $\mathbf{q}$, $\mathbf{q}'$ is the neural network output joint variable vector, and $\mathbf{G}(\mathbf{q})$ is the known forward kinematics. Hence, the $m \times n$ Jacobian matrix $\mathbf{J}(\mathbf{Q}) = \dfrac{\partial \mathbf{G}(\mathbf{q})}{\partial \mathbf{q}}$ is also known. A manipulator is kinematically non-redundant when $m = n$, and it is redundant when $m < n$. The neural network weights are trained using the location error $\mathbf{e} = \mathbf{x}_d - \mathbf{x}$. The neural network architecture and training algorithm will be discussed in the next section.

A particular set of joint points $\mathbf{q}_d$ are the inverse kinematics solutions for $\mathbf{q}$ if $\mathbf{G}(\mathbf{q}_d) = \mathbf{x}_d$. It is well known that $\mathbf{q}_d$ is not unique for a given $\mathbf{x}_d$ in general. For a non-redundant robot, the possible inverse kinematics solutions are isolated in joint variable space. However, for a redundant robot, the set of all possible inverse kinematics solutions contains several disjoint subsets in the joint space. Each subset is topologically connected[16] and consists of infinite number of joint points. The role that the initial value $\mathbf{q}_0$ plays in the proposed inverse kinematics solution is that it defines the center of a subset of $\mathbf{q}$ in which a solution $\mathbf{q}_d$ exists in the neighborhood. Thus by judiciously choosing $\mathbf{q}_0$, we can select the class of inverse kinematic solutions for which the corresponding robot postures are most desirable for the task to be performed.

One traditional way for solving redundancy in inverse kinematic solutions is to require the solution to optimize

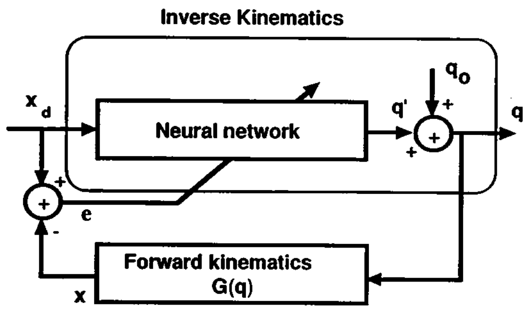

Fig. 1. The Learning Scheme for solving the Inverse Kinematics Problem.

certain dexterity measures.[7] A commonly used measure is the minimum of a joint range availability function defined as

$$h(\mathbf{q}_p) = (\mathbf{q}_c - \mathbf{q})^T(\mathbf{q}_c - \mathbf{q}) \tag{1}$$

where $\mathbf{q}_c$ is the center of the range of joint travel. Therefore, if the desired range of each joint variable is specified, then $\mathbf{q}_c$ can be appropriately identified. Thus we propose to choose the initial value $\mathbf{q}_0$ of $\mathbf{q}$ in the same manner as we choose $\mathbf{q}_c$. For example, we can place $\mathbf{q}_0$ of a non-redundant robot arm inside any one of the finite and distinct subspaces of $\mathbf{q}$ to obtain an inverse kinematic solution which provides either elbow up or elbow down, right arm or left arm configurations as desired. In the case of redundant arms, more flexibility exists in choosing $\mathbf{q}_0$ and additional constraining condition on the arm configuration, such as collision avoidance, can be imposed in obtaining a desired inverse kinematics solution.

## 3. PROBLEM FORMULATION

Before we formulate the inverse kinematics problem in detail, we briefly introduce the multilayer neural network to be used in the proposed solution scheme of Figure 1. A multi-layer neural network is a parallel, distributed information processing system consisting of processing elements interconnected together with unidirectional signal channels. Each processing element, called as neuron, has a single output and performs an activation function of the weighted sum of all its inputs. The activation function is usually chosen as a sigmoid function or a linear function. After the structure of the network is specified, the weights between the neurons are updated step by step so that the input-output mapping of the neural network approximates a desired nonlinear continuous function. The updating rule of the weights to be used is the back-propagation algorithm proposed by Rumelhart et al.[17] The theoretical basis of the neural network approximation to a nonlinear continuous function is the Kolmogorov's theorem.[18] It has been reported[19] that any nonlinear continuous function can be approximated by a two-layer neural network, as shown in Figure 2.

In Figure 2, $x_j$ is the $j$th element of the input vector at the input layer, $q'_k$ is the $k$th element of the output vector at the output layer, $z_i$ is the $i$th element of the output vector at the hidden layer, $w_{oki}$ is the weight on the channel from $z_i$ to the input of $k$th output $q'_k$, $w_{ok0}$ is the bias entering the $k$th neuron at output layer, $w_{hij}$ is the weight on the channel from input $x_j$ to the input of $i$th hidden layer neuron, and $w_{hi0}$ is the bias entering the $i$th neuron at the hidden layer. $z_i$ and $q'_k$ are computed by

$$z_i = f_h(u_i) \quad \text{where} \quad u_i = w_{hi0} + \sum_{j=1}^{m} w_{hij} x_j \tag{2}$$

$$q'_k = v_k \quad \text{where} \quad v_k = w_{ok0} + \sum_{i=1}^{r} w_{oki} z_i \tag{3}$$

where $i = 1, 2, \ldots, r$, $j = 1, 2, \ldots,$ $k = 1, 2, \ldots,$ , $r$ is the

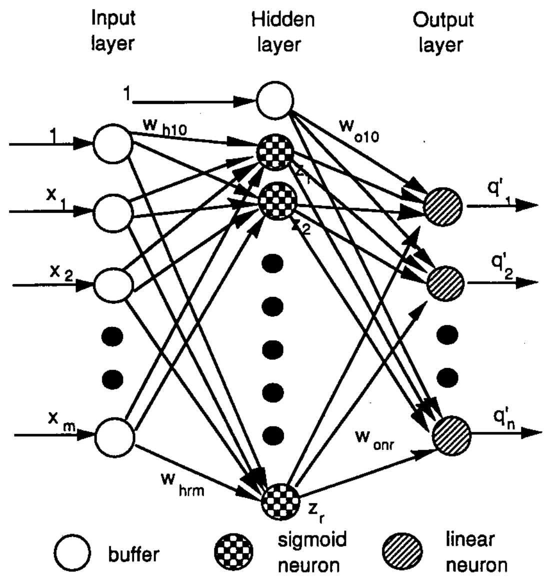

Fig. 2. A Two-Layer neural Network.

number of neurons at hidden layer, and $f_h(u)$ is a sigmoid function at the hidden layer given by $f_h(u) = \frac{1}{1+e^{-u}}$.

The updating rule for adjusting the weights $\mathbf{w}$ at time $k$ is the steepest descent algorithm with momentum term given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \eta\left(-\frac{dE(\mathbf{w})}{d\mathbf{w}}\right) + \alpha(\mathbf{w}(k) - \mathbf{w}(k-1)) \tag{4}$$

where $E(\mathbf{w})$ is the objective function to be minimized, $\alpha$ is the momentum coefficient and $\eta$ is the learning rate.

The task of training the neural network to solve the inverse kinematics problem can be stated as follows. Given $L$ desired end-effector locations $\{\mathbf{x}_{dp} \mid p = 1, 2, \ldots, L\}$ as the input patterns $\mathbf{x}$ to the neural network, adjust the weights $\mathbf{w}$ such that the outputs of the inverse kinematics block in Figure 1 converge to the desired joint variable solution $\{\mathbf{q}_{dp} \mid p = 1, 2, \ldots, L, \mathbf{x}_{dp} = \mathbf{G}(\mathbf{q}_{dp})\}$.

Since we assume that the forward kinematics mapping $\mathbf{x} = \mathbf{G}(\mathbf{q})$ and the $m \times n$ Jacobian matrix $\mathbf{J}(\mathbf{q}) = \frac{\partial \mathbf{G}(\mathbf{q})}{\partial \mathbf{q}}$ are known, we can define a kinematic error as $\mathbf{e}_p = \mathbf{x}_{dp} - \mathbf{x}_p = \mathbf{x}_{dp} - \mathbf{G}(\mathbf{q}_p)$. Minimizing $\mathbf{e}_p$ achieves the goal in finding the desired inverse kinematics mapping. Thus the objective function $E(\mathbf{w})$ in (4) can be defined as the sum of the norm of all squared kinematic errors $\mathbf{e}_p$:

$$E_0 = \frac{1}{2mL}\sum_{p=1}^{L} (\mathbf{x}_{d,p} - \mathbf{x}_p)^T(\mathbf{x}_{d,p} - \mathbf{x}_p). \tag{5}$$

The corresponding gradient $\frac{\partial E_0}{\partial \mathbf{w}}$ is

$$\begin{aligned}\frac{\partial E_0}{\partial \mathbf{w}} &= -\frac{1}{mL}\sum_{p=1}^{L}\left[(\mathbf{x}_{d,p} - \mathbf{x}_p)^T \frac{\partial \mathbf{G}(\mathbf{q}_p)}{\partial \mathbf{q}_p}\frac{\partial \mathbf{q}_p}{\partial \mathbf{w}}\right]^T \\ &= -\frac{1}{mL}\sum_{p=1}^{L}\left[(\mathbf{x}_{d,p} - \mathbf{x}_p)^T \mathbf{J}(\mathbf{q}_p)\frac{\partial \mathbf{q}'_p}{\partial \mathbf{w}}\right]^T. \end{aligned} \tag{6}$$

When the neural network is perfectly trained, $E_0$ is minimized to zero and the neural network solution provides the exact inverse kinematics solution $\mathbf{q}_p$ for each Cartesian position $\mathbf{x}_{dp}$. In practice, we specify an accuracy threshold $\varepsilon > 0$ for the neural network training. The weight updating algorithm (4) can be terminated whenever $E_0$ is less than a threshold $\varepsilon$. The neural network thus obtained provides an approximation to the inverse kinematics solution in the trained domain. Simulation studies, which will be presented later, show that the proposed scheme works very well and converges fast.

## 4. OBSTACLE AVOIDANCE INVERSE KINEMATICS SOLUTION

### *4.1. Object modeling method*

When obstacles are present in the robot work space, inverse kinematics solution must be found in such a way that it avoids the collision between robot links and obstacles. To find obstacle avoidance inverse kinematics solution, we need to appropriately model the obstacles and links and then measure the distance between the obstacles and links. Many researchers[3,8] have dealt with the modeling problem by assuming convexity of links and obstacles and then measuring the distance among convex sets. The advantage of this modeling method is that it can precisely compute the distance between a pair of two objects. The disadvantage is that computing the distance is time consuming.

Since a robot work space is bounded, the obstacles and links in the work space are also bounded. Mathematical theorem[16] shows that only finite number of balls are required to cover a bounded set. So if robot links and obstacles and links is in the union of all the balls. Of course smaller balls can more accurately describe the set occupied by the links and obstacles, but it would also require more balls and computations. We recommend to use balls with different radii to cover different shapes of links and objects so that more accurate converge is possible without increasing complexity. This object modeling method is called the ball-covering modeling method. Similar method was proposed as the spherical representation in the study of human body modeling.[20]

To illustrate the idea stated above, a planar redundant robot is considered as shown in Figure 3. Suppose that there are $\mathbf{N}_l$ balls covering links and $\mathbf{N}_o$ balls covering

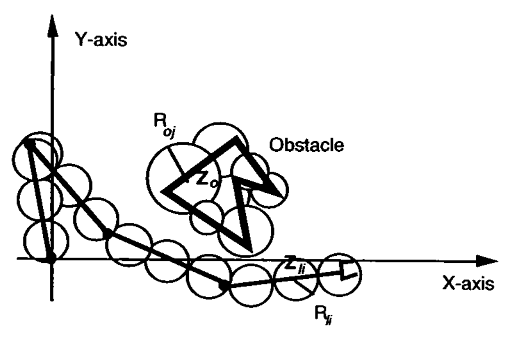

Fig. 3. Object Modeling Method.

obstacles. Let the pairs $(R_{li}, \mathbf{Z}_{li})$ and $(R_{oj}, \mathbf{Z}_{oj})$ be the radii and centers of two balls among all balls covering the links and obstacles, where $i = 1, 2, \dots, N_l$ and $j = 1, 2, \dots, \mathbf{N}_o$. Radii $\mathbf{R}_{li}$ and $\mathbf{R}_{oj}$ are preselected, therefore they are fixed. The center position vector $\mathbf{Z}_{li}$ depends on joint angles, hence $\mathbf{Z}_{li}$ can be written as $\mathbf{Z}_{li}(\mathbf{q})$. $\mathbf{Z}_{oj}$ is fixed if the obstacles are stationary. The sufficient condition for keeping the links and obstacles a minimum distance $\delta$ apart is

$$(R_{li} + R_{oj} + \delta)^2 - \|\mathbf{Z}_{li} - \mathbf{Z}_{oj}\|^2 \leq 0 \tag{7}$$

for all ball pairs one taken from the links and one taken from the obstacles.

Suppose that the obstacles are stationary. For each $p$th output joint variable set $\mathbf{q}_p$, we can define a corresponding $N \times 1$ distance vector $\mathbf{g}(\mathbf{q}_p)$ whose elements are given by

$$g_{ij}(\mathbf{q}_p) = (R_{li} + R_{oj} + \delta)^2 - \|\mathbf{Z}_{li}(\mathbf{q}_p) - \mathbf{Z}_{oj}\|^2 \tag{8}$$

where $i = 1, 2, \dots, N_l$, $j = 1, 2, \dots, \mathbf{N}_o$ and $\mathbf{N} = \mathbf{N}_o * \mathbf{N}_l$. Therefore,

$$\mathbf{g}(\mathbf{q}_p) \leq 0 \text{ for all output index } p = 1, 2, \dots, L, \tag{9}$$

where $\mathbf{g}(\mathbf{q}_p)$ is differentiable with respect to $\mathbf{q}_p$. Satisfaction of this condition for all $\mathbf{q}_p$ guarantees collision free inverse kinematics solutions. This condition leads to an inequality constraint in the neural network solution which is an undesirable complication that must be appropriately dealt with.

### *4.2 Penalty function method*

The inverse kinematics problem at hand becomes one of minimizing the objective function $E_0$ in (5) with the inequality constraint (9). The complexity of the optimization problem with inequality constraints can be reduced if we convert (9) into a penalty function in $E_0$ and then minimize the augmented objective function of the constraint.[21]

Let $P$ be the penalty function

$$P = \frac{1}{L}\sum_{p=1}^{L}\sum_{i=1}^{N_l}\sum_{j=1}^{N_o} g_{ij}^*(\mathbf{q}_p)^2 \tag{10}$$

where

$$g_{ij}^*(\mathbf{q}_p) = \max\{0, g_{ij}(\mathbf{q}_p)\} \tag{11}$$

If all solutions $\mathbf{q}_p$ satisfy condition (9), then $P$ would be equal to zero. $P > 0$ occurs when at least one element $g_{ij}(\mathbf{q}_p)$ of $\mathbf{g}(\mathbf{q}_p)$ is greater than zero which implies that collision occurs. Therefore, the objective is to minimize $P$ and $E_0$ to zero in solving the inverse kinematics problem.

Let us define an augmented objective function as

$$E = E_0 + \mu P \tag{12}$$

where $E_0$ is defined in (5) and $\mu$ is the Lagrange multiplier which is positive. The objective function $E$ is differentiable, so the gradient $\frac{\partial E}{\partial \mathbf{w}}$ can be easily computed. When $\mu$ is sufficiently large, the minimization of $E$ is equivalent to the minimization of $E_0$ with inequality (9). The proper choice of $\mu$ depends on the problem at hand.

## 5. SIMULATIONS

For the purpose of illustration, we examine the inverse kinematics solutions of a four-link planar robot shown in

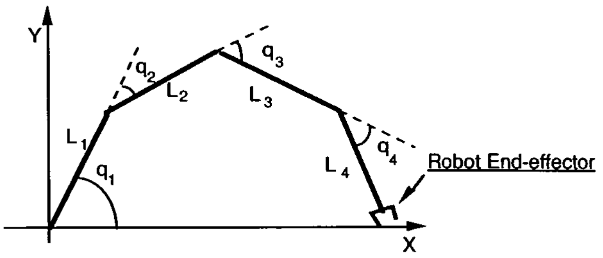

Fig. 4. A Four-link Planar Robot.

Figure 4. The forward kinematics function of the robot is

$$\mathbf{G}(\mathbf{q}) = \begin{bmatrix} L_1c_1 + L_2c_{12} + L_3c_{123} + L_4c_{1234} \\ L_1s_1 + L_2s_{12} + L_3s_{123} + L_4s_{1234} \end{bmatrix} \tag{13}$$

and the corresponding Jacobian matrix is

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} -L_1s_1 - L_2s_{12} - L_3s_{123} - L_4s_{1234} & L_1c_1 + L_2c_{12} + L_3c_{123} + L_4c_{1234} \\ -L_2s_{12} - L_3s_{123} - L_4s_{1234} & L_2c_{12} + L_3c_{123} + L_4c_{1234} \\ -L_3s_{123} - L_4s_{1234} & L_3c_{123} + L_4c_{1234} \\ -L_4s_{1234} & L_4c_{1234} \end{bmatrix}^T \tag{14}$$

where $L_i = 1.0$ is the length of the $i$th link for all $i$, $s_1 = \sin(q_1)$, $s_{12} = \sin(q_1 + q_2)$, $s_{123} = \sin(q_1 + q_2 + q_3)$, $s_{1234} = \sin(q_1 + q_2 + q_3 + q_4)$, and $q_i$ is the $i$th joint angle as shown in Figure 4. The symbol $c$ corresponds to cosine function.

In the simulation study, we use the two-layer neural network as shown in Figure 2 with 40 neurons at hidden layer. The momentum coefficient $\alpha$ is chosen as $\alpha = 0.9$ and the learning rate $\eta$ is chosen as $\eta = 0.5$. The initial values of weights are selected as zero-mean random values with uniform distribution in $[-1, +1]$.

### *5.1 Obstacle free case*

We examine through simulation the neural network training on a specified domain spanned by 25 Cartesian positions $\{(x, y) \mid x = 0.0, 0.6, 1.2, 1.8, 2.4; y = 0.6, 1.2, 1.8, 2.4, 3.0\}$ as shown in Figure 5. If $\mathbf{q}_0$ is specified as

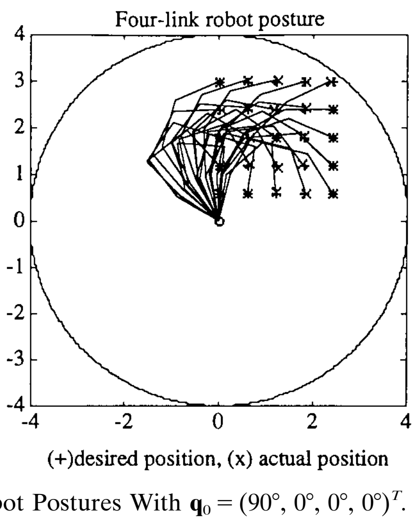

Fig. 5. Robot Postures With $\mathbf{q}_0 = (90°, 0°, 0°, 0°)^T$.

Table I. Training Results of Case 1 and 2

| Case | $E_0$ | Iterations |
|---|---|---|
| 1 | 0.000179 | 1000 |
| 2 | 0.001 | 166 |

$(90°, 0°, 0°, 0°)^T$, which corresponds to an upright robot posture, then the robot postures for the desired solutions are elbow-up.

Table I lists two training results for $\mathbf{q}_0 = (90°, 0°, 0°, 0°)^T$. Case 1 shows the minimized $E_0$ value after 1000 training iterations. The final inverse kinematics solutions are plotted in Figure 5. Case 2 shows that the training is converged at 166 iterations for an error threshold $\varepsilon = 0.001$. Thus the convergence rate is fast for most practical applications.

Shown in Figure 6 is an alternate training result when $\mathbf{q}_0 = (0°, 0°, 0°, 0°)^T$ is used. As expected, robot postures are elbow down. This exercise demonstrates the property that choice of $\mathbf{q}_0$ affects robot postures.

## *5.2 Obstacle avoidance case*

The same four-link planar robot and neural network are used to investigate the obstacle avoidance solution of the inverse kinematics problem. As shown in Figures 7, 8 and 9, the domain is spanned by 10 Cartesian positions $\{(x, y) \mid x = 0.35, 0.7, 1.05, 1.40, 1.75, 2.10, 2.45, 2.80, 3.15, 3.50; y = 0\}$ along a straight line on the $y$ axis. To obtain elbow-up solution we chose $\mathbf{q}_0 = (45°, 0°, 0°, 0°)^T$. Each robot is covered by 4 circles with identical radius of 0.125, and each obstacle is a circle of radius of 0.5. The minimum distance $\delta$ between links and obstacles is chosen as 0.1. The Lagrange multiplier $\mu$ is 3.0. The neural network weights are trained to minimize the objective function $E$ in (12). Training is terminated when $E_0 \leq 0.001$.

Listed in Table II are three simulation results with different environments. Figure 7 shows that the robot is trained for the 10 posiitons on the $x$-axis without obstacles. The training took 125 iterations to reach an accuracy of $E_0 = 0.001$. When one obstacle is introduced as shown in Figure 8, the robot postures are altered to avoid the obstacle. 535 iterations were needed for $E_0$ to converge to the same value of $E_0 = 0.001$. Figure 9 shows the solution when more obstacle is added in the working space in Figure 8. We see that avoidance of two obstacles is successfully accomplished after 591 iterations.

These simulations verify that the proposed learning scheme can solve the inverse kinematics problem with obstacle avoidance.

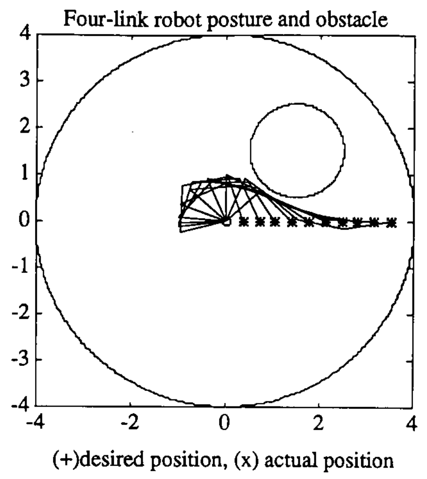

Fig. 8. Robot Postures with One Obstacle.

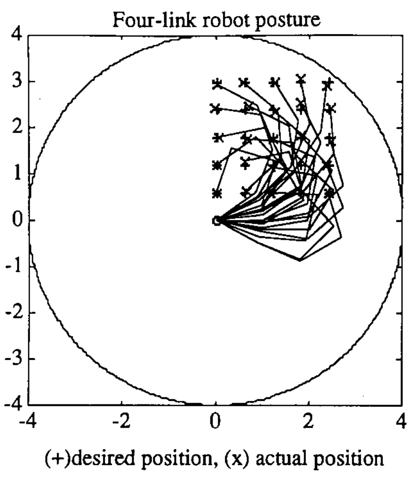

Fig. 6. Robot Postures with $\mathbf{q}_0 = (0, 0°, 0°, 0°)^T$.

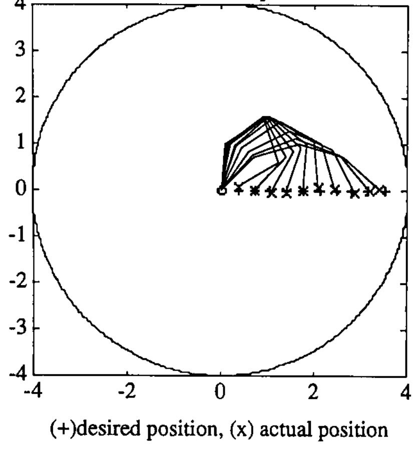

Fig. 7. Robot Postures without Obstacles.

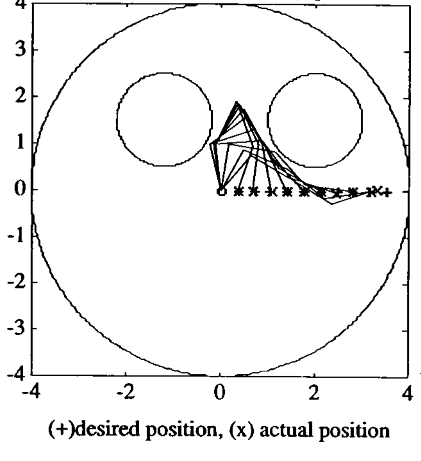

Fig. 9. Robot Postures with Two Obstacles.

Table II. Obstacle Avoidance Results

| Figure | Number of obstacles | $\mathbf{q}_0$ | $E_0$ | iterations |
|---|---|---|---|---|
| 7 | 0 | $(45°, 0°, 0°, 0°)^T$ | 0.001 | 125 |
| 8 | 1 | $(45°, 0°, 0°, 0°)^T$ | 0.001 | 535 |
| 9 | 2 | $(45°, 0°, 0°, 0°)^T$ | 0.001 | 591 |

The values of $\mu$ affects the convergence of training process. For example, when $\mu$ is too large, it takes more iterations to reduce the kinematic error to the preset threshold. The convergence of the kinematic error is faster for a smaller $\mu$, however collision between robot and obstacles may occur even though kinematic error threshold has been met. Thus the appropriate choice of $\mu$ depends on the problem at hand.

### *5.3 Applications to PUMA 560 robot*

The PUMA 560 robot arm has six joints. When both the Cartesian position and orientation are taken into account, the PUMA 560 arm is a non-redundant robot. However, if we only consider the position of the end point of an attached tool, the PUMA 560 arm as shown in Figure 10 will become a redundant robot since the tool tip in the three-dimensional Cartesian space is related to five joint angles (a two degree redundancy). With the joint angle and link parameters defined in Table III, the positional forward kinematics is expressed as follows:

$$X = C_1[d_6(C_{23}C_4S_5 + S_{23}C_5) + S_{23}d_4 + a_3C_{23} + a_2C_2] - S_1[d_6S_4S_5 + d_2) \tag{15}$$

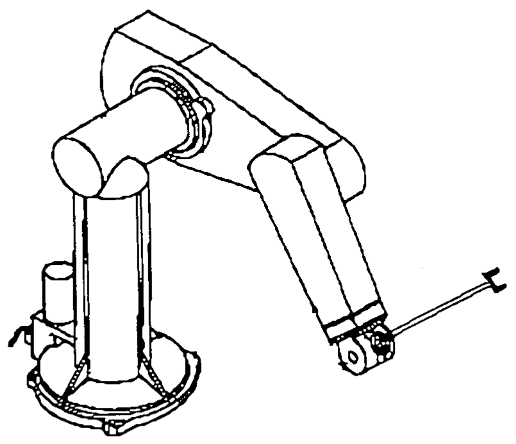

Fig. 10. PUMA 560 Robot with A Long Tool.

$$Y = S_1[d_6(C_{23}C_4S_5 + S_{23}C_5) + S_{23}d_4 + a_3C_{23} + a_2C_2] + C_1(d_6S_4S_5 + d_2) \tag{16}$$

$$Z = d_6(C_{23}C_5 - S_{23}C_4S_5) + C_{23}d_4 - a_3S_{23} + a_2S_2 \tag{17}$$

A set of 15 points on a straight line, as shown in Figure 11(a) and listed in Table IV, is used for training. The first run of training is in the environment without obstacles. We use a neural network with 45 neurons at the hidden layer. In order to obtain the elbow-up solution, $\mathbf{q}_0$ is chosen as $(0°, -90°, 90°, 0°, 0)^T$. It took 1765 iterations to reduce the kinematics error $E_0$ to below 0.001. The joint solutions are plotted in Figure 11(b). The robot links of the solution are almost in a plane.

When there is an obstacle in the work space as shown in Figure 12(a), the obstacle avoidance solution took 3826 iterations to reduce the kinematics error to below 0.001. The joint solutions are plotted in Figure 12(b).

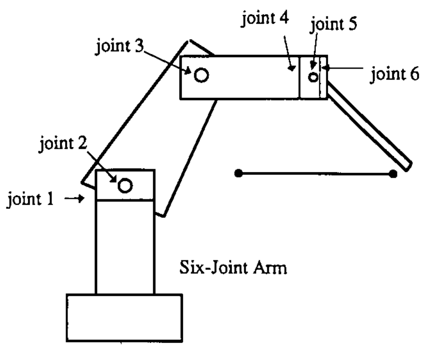

Fig. 11(a). Robot Posture without Obstacle.

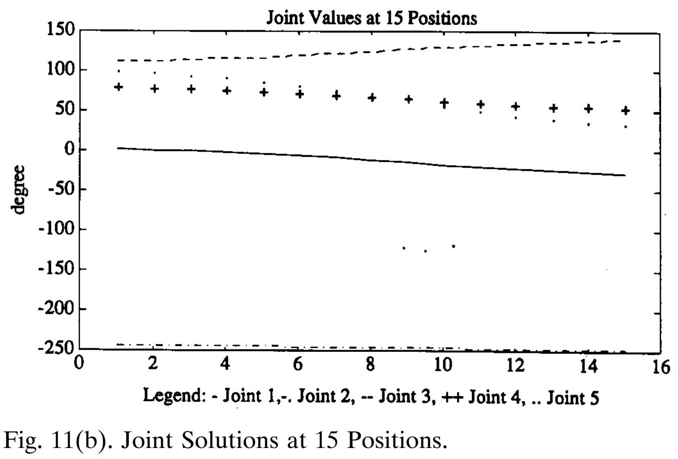

Fig. 11(b). Joint Solutions at 15 Positions.

Table III. PUMA 560 Arm Link Coordinate Parameters

| Joint $i$ | $\theta_i$ | $\alpha_i$ | $a_i$ | $d_i$ | Joint range |
|---|---|---|---|---|---|
| 1 | 90 | −90 | 0 | 0 | −159 to +159 |
| 2 | 0 | 0 | 431.8 mm | 149.09 mm | −180 to −0,0 to 43, +137 to 180 |
| 3 | 90 | 90 | −20.32 mm | 0 | −180 to −128, −51.9 to +180 |
| 4 | 0 | −90 | | 433.07 mm | −110 to +170 |
| 5 | 0 | 90 | | 0 | −100 to +1 |
| tool | | | | 256.25 mm | |

Table IV. 15 Training Points on a Straight Line (in meters)

| *x* | *y* | *z* |
|---|---|---|
| −0.700 | 0.700 | 0 |
| −0.725 | 0.725 | 0 |
| −0.750 | 0.750 | 0 |
| −0.775 | 0.775 | 0 |
| −0.800 | 0.800 | 0 |
| −0.825 | 0.825 | 0 |
| −0.850 | 0.850 | 0 |
| −0.875 | 0.875 | 0 |
| −0.900 | 0.900 | 0 |
| −0.925 | 0.925 | 0 |
| −0.950 | 0.950 | 0 |
| −0.975 | 0.975 | 0 |
| −1.000 | 1.000 | 0 |
| −1.025 | 1.025 | 0 |
| −1.050 | 1.050 | 0 |

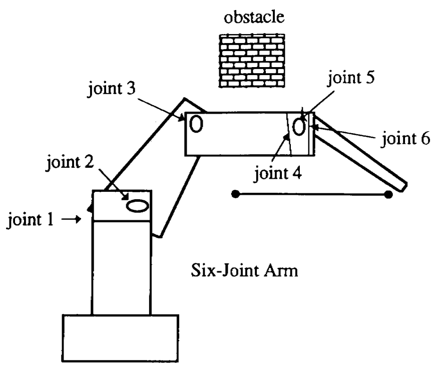

Fig. 12(a). Robot Posture with an Obstacle.

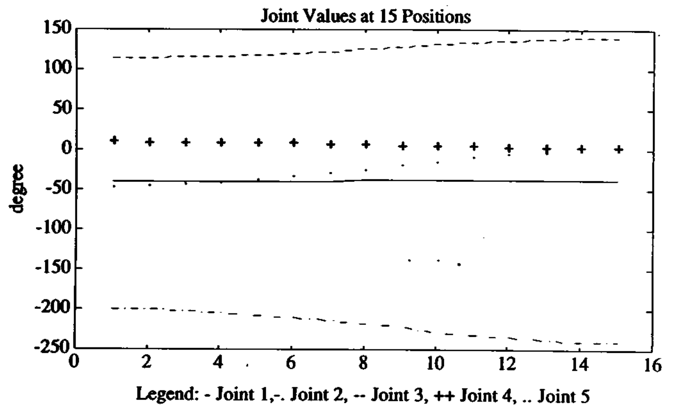

Fig. 12(b). Obstacle Avoidance Joint Solutions at 15 Tool Tip Positions.

The robot links are no longer in a plane, rather they are twisted to avoid the obstacle.

## VI. CONCLUSION

A neural network technique has been introduced in this paper to solve the inverse kinematics problem of redundant robot manipulators with obstacle avoidance capabilities. The solution technique requires only the knowledge of robot forward kinematics including the corresponding Jacobian. Both obstacle free and obstacle avoidance cases were studied. The latter case was effectively solved by employing the ball-covering object modeling technique which is mathematically tractable and computationally efficient. Extensive simulations have been performed, which showed that the proposed neural network schemes are very effective in obtaining accurate robot inverse kinematics solution in a given work space through training in a straightforward manner. The feasibility of the solution technique is also successfully demonstrated on a PUMA 560 arm. Thus the proposed technique is highly useful in practice.

## References

1. W.A. Wolovich and H. Elliot, "A computational technique for inverse kinematics" *Proc. 23rd IEEE Conf. on Decision and Control,* Las Vegas, (1984) pp. 1359–1363.
2. R.J. Vaccaro and S.D. Hill, "A joint-space command generator for Cartesian control of robotic manipulators" *IEEE J. of Robotics and Automation* **RA-4,** 70–76 (1988).
3. T.C. Hsia and Z.Y. Guo, "New inverse kinematic algorithms for redundant robots" *J. of Robotics Systems* **8**(1), 117–132 (1991).
4. Z.Y. Gup and T.C. Hsia, "Joint trajectory generation for redundant robots in an environment with obstacles" *Proc. IEEE Int. Conf. Robotics and Automation* (1990) pp. 157–162.
5. C.A. Klein and C.H. Huang, "Review for pseudoinverse control for use with kinematically redundant manipulators" *IEEE Trans. SMC* **13**(3), 245–250 (Mar./Apr., 1983).
6. L. Sciavicco and B. Siciliano, "A solution algorithm to the inverse kinematic problem for redundant manipulators" *IEEE J. Robotics and Automation* **RA-4,** 403–410 (August, 1988).
7. C.A. Klein and B.E. Blaho, "Dexterity measures for the design and control of kinematically redundant manipulators" *Int. J. of Robotics Research* 72–83 (1987).
8. E.G. Gilbert and D.E. Johnson, "Distance function and their application to robot path planning in the presence of obstacles" *IEEE J. of Robotics and Automation* **RA**-1(1), 21–30 (1985).
9. S. Lee and R.M. Kil, "Robot kinematic control based on bidirectional mapping neural network" *Int. Joint Conf. on Neural Networks* **3,** 327–335 (1990).
10. A. Guez and Z. Ahmad, "Solution to the inverse kinematics problem in robotics by neural networks" *IEEE Int. Conf. on Neural Networks* (1988) **Vol. II,** pp. 617–621.
11. E.S.H. Hou and W. Utama, "An artificial neural network for redundant manipulator inverse kinematics computation" *Proceedings of the SPIE – The International Society for Optical Engineering* (1992) pp. 1607: 668–77.
12. J.A. Francisco, "Multilayer back-propagation network for learning the forward and inverse kinematics" *Proc. Int. Joint Conf. on Neural Networks* (1990) pp. II-319–321.
13. S. Kieffer, V. Morellas and M. Donath, "Neural network learning of the inverse kinematic relationships for a robot arm" *Proc. IEEE Int. Conf. Robotics and Automation,* Sacramento (1991) pp. 2418–2425.
14. T. Iberall, "A ball pack approach to modelling human prehension" *IEEE Conf. on Neural Networks* (1987) **Vol. 4,** pp. 535–544.
15. T. Iberall, "A neural network for planning hand shapes in human prehension" *IEEE Conf. on Decision and Control* (1987) pp. 2288–2293.
16. Serge Lang, "Topological spaces" *Real and Functional Analysis* (Springer-Verlag, New York Inc., 1993) Chapter 2.
17. D.E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning internal representations by error propagation" *Parallel Distributed Processing:Explorations in the Micro-*

*structure of Cognition,* vol. I (ed. D.E. Rumelhart et al.) (1986) pp 318–365.

18. A.N. Kolmogorov, "On the representation of continuous function of many variables by superposition of continuous functions of one variable and addition" *Dolk. Akad. Nauk,* USSR **114,** 953–956 (1957).
19. G. Cybenko, "Approximation by superpositions of a sigmoidal function" *Mathematics of Control, Signals and Systems* **2**(4), 303–314 (1989).
20. N.J. Badler, J. O'Rourke and H. Toltzis, "A spherical representation of a human body for visualizing movement" *Proceeding of the IEEE* **67**(10), 1397–1403 (1979).
21. D.G. Luenberger, "Penalty and barrier methods" *Introduction to Linear and Nonlinear Programming* (Addison-Wesley Publishing Co., Reading, Mass., 1965) pp. 277–304.