Engineering with Computers (2014) 30:641–649
DOI 10.1007/s00366-013-0313-2

ORIGINAL ARTICLE

A neural-network committee machine approach to the inverse kinematics problem solution of robotic manipulators

Raşit Köker · Tarık Çakar · Yavuz Sari

Received: 12 August 2012 / Accepted: 8 January 2013 / Published online: 1 February 2013

Abstract In robotics, inverse kinematics problem solution is a fundamental problem in robotics. Many traditional inverse kinematics problem solutions, such as the geometric, iterative, and algebraic approaches, are inadequate for redundant robots. Recently, much attention has been focused on a neural-network-based inverse kinematics problem solution in robotics. However, the result obtained from the neural network requires to be improved for some sensitive tasks. In this paper, a neural-network committee machine (NNCM) was designed to solve the inverse kinematics of a 6-DOF redundant robotic manipulator to improve the precision of the solution. Ten neural networks (NN) were designed to obtain a committee machine to solve the inverse kinematics problem using separately prepared data set since a neural network can give better result than other ones. The data sets for the neural-network training were prepared using prepared simulation software including robot kinematics model. The solution of each neural network was evaluated using direct kinematics equation of the robot to select the best one. As a result, the committee machine implementation increased the performance of the learning.

Keywords Neural networks · Committee machines · Inverse kinematics solution · Robotics

R. Köker (✉)
Electronics and Computer Sciences Department,
Technical Education Faculty, Sakarya University,
54187 Sakarya, Turkey
e-mail: rkoker@sakarya.edu.tr

T. Çakar
Industrial Engineering Department, Engineering Faculty,
Sakarya University, 54187 Sakarya, Turkey
e-mail: tcakar@sakarya.edu.tr

Y. Sari
Electronics and Automation Department,
Hendek Vocational High School, Sakarya University,
54300 Hendek, Sakarya, Turkey
e-mail: sari@sakarya.edu.tr

1 Introduction

The study of kinematics of articulated robots are known as one of the most traditional areas in robotics. Robot kinematics can be described as establishing a mechanism to determine the relationship between the joint and Cartesian coordinates. Denavit and Hartenberg [6] have successfully solved the direct kinematics problem. Currently, there are automatic procedures for assignation of the different parameters to every robot's joints and obtaining the end-effector position and orientation referred to a base frame for each point expressed in the joint coordinate frame regarding direct kinematics problem. On the other hand, the inverse kinematics problem deals with determination of a transformation between the external reference frame, which is generally expressed in terms of the true goal coordinates, and the internal reference frame of robot, which is generally expressed in articulation states. Unfortunately, there was no any analytical solution found for inverse kinematics problem in a general way. In the literature, it can be found analytical solutions for the most used and well-known robotic manipulators. However, these analytical solutions are specific to a particular robot type cannot be applied to other types of robots. In case of considering several kinds of new articulated robots proposed for different tasks, the problem will be worst. These kinds of multi-articulated robots make inapplicable some of the classical procedures to compute the inverse

kinematics essentially because of such methods suppose some specific configurations that are not used in these new articulated robots. They have inherently multi-redundant structure. For instance, a simple biped robot needs 12 degrees-of-freedom (DOF) for reaching the most common configurations to obtain realistic postures. The redundant structure of this degree of freedom makes practically unfeasible to develop an analytic solution for the inverse kinematics [26].

The inverse kinematics problem is usually more complex for redundant robots. Traditionally, three models are used to solve the inverse kinematics problem: geometric [9, 24], algebraic [7, 11, 25, 31], and iterative [22] models. Each method has some disadvantages for solving the inverse kinematics problem. For instance, closed-form solutions are not guaranteed for the algebraic methods, and closed-form solutions for the first three joints of the robot must exist geometrically when the geometric method is used. Similarly, the iterative inverse kinematics solution method converges to only one solution, which depends on the starting point. These traditional solution methods may have a prohibitive computational cost because of the high complexity of the geometric structure of the robotic manipulators [18, 19]. This is why researchers have focused on solving the inverse kinematics problem using artificial neural networks.

There are many papers published about the neural-network-based inverse kinematics solution for robotic manipulators [3, 5, 12–14, 18, 19, 21, 26–28, 30, 33]. Tejomurtula and Kak [33] proposed a solution for the inverse kinematics problem for a three-joint robotic manipulator based on structured neural networks that can be trained quickly to overcome the disadvantages of the back-propagation algorithms, such as training time and accuracy. Karlık and Aydın [18] presented a study for the inverse kinematics solution of a six-joint robot based on identifying the best neural-network configuration. They found that designing six separate neural networks with two hidden layers for each output yields a better solution than designing a single neural network with six outputs. Oyama et al. [30] presented modular neural-network architectures for learning the inverse kinematics model. Their method is based on DeMers' method that involves a number of experts, an expert selector, an expert generator, and a feedback controller, which can accommodate the nonlinearities in the kinematic system. Their method contains certain limitations; for instance, the inverse kinematics computation procedure is highly complex, and the learning speed is low. However, root mean square (RMS) hand position errors of less than 10 mm were achieved. Mayorga and Sanongboon [28] presented a novel neural-network approach for fast inverse kinematics computations and efficient singularity avoidance or prevention for redundant robots based on a set of bounded geometrical concepts used to determine the characteristic matrices. Their algorithm enables the implementation of fast and robust real-time algorithms for safe robot path generation. A neural-network approach using the backpropagation algorithm was presented by Bingul and Ertunc [3] for the inverse kinematics solution of an industrial robotic manipulator without an analytical inverse kinematics solution. The disadvantages of their approach are the large errors in the joint angles and the inability of this approach to provide multiple solutions to the inverse kinematics problem. Hasan et al. [13, 14] presented an inverse kinematics solution for a 6-DOF robotic manipulator. In their first study [13], an adaptive learning strategy using an artificial neural network was proposed to control the motion of a 6-DOF robotic manipulator and to overcome difficulties in solving the inverse kinematics problem such as singularities and uncertainties. In their approach, a network was trained to learn a desired set of joint angle positions from a set of given end-effector positions, and the experimental results showed good mapping over the working area of the robot. After 8×10^6 iterations, the absolute error percentages for joints 1–6 were 3.635, 3.66, 5.31, 1.73, 3.435, and 6.1 %, respectively. The researchers also provided a graphical presentation of these errors by iteration number. They published a second paper [14] based on using artificial neural networks to learn robot system characteristics rather than specifying an explicit robot system model to overcome singularities and uncertainties; this method was implemented for another type of 6-DOF robotic manipulator model. After 3×10^5 iterations, the total error percentages for the test dataset for joints 1–6 were 0.915, 0.135, 0.57, 4.79, 4.81, and 1.11 %, respectively [20].

The researchers are also focusing on designing NNCM to obtain better performance [1, 2, 4, 10, 15–17, 32, 36]. The main purpose of using NNCM is to increase the learning performance using the selection chance of the better result among neural networks in the committee machine.

In this paper, the inverse kinematics solution has been done using NNCM to reduce the end-effector error. The committee machines are designed using more than one neural network working online and in parallel. The output of each neural-network-based inverse kinematics solution has been evaluated using direct kinematics equations of the robotic manipulator to find the best solution among solutions. It was clearly seen that using committee machines reduced the end effector error at the end of neural-network-based inverse kinematics solution. Using neural networks in parallel is reducing the error, however, after using six neural networks in parallel, there is no significant effect on the learning performance of the solution system. This paper is organized as follows. In Sect. 2, robot models are

described and robot kinematic analysis is explained. In Sect. 3, the usage of neural Networks in robotics has been given. In Sect. 3.1, the structure of committee machines and NNCM design have been presented. In Sect. 4, results and discussions have been given.

2 Kinematics analysis of robotic manipulators

Robotic manipulators and kinematic mechanisms are typically constructed by connecting different joints together using rigid links. A robot can be modeled as an open-loop articulated chain with these several rigid links connected in series by either revolving or prismatic joints driven by actuators. Robot kinematics deals with the analytical study of the geometry of motion of a robot with respect to a fixed reference coordinate system as a function of time, without regard to the forces or moments that cause the motion [23]. For this reason, it deals with the analytical description of the robot as a function of time, particularly the relations between the joint-variable space and the position and orientation of the end effector of a robotic manipulator [11, 33]. Figure 1 shows the structure of the robotic manipulator used in this study [34]. In addition, the Denavit–Hartenberg parameters of a Hitachi M6100 6-DOF robot are given in Table 1 [34].

The forward or direct kinematics deal with the motion of the end effector of the robot according to the world coordinate system. The world coordinate frame (X_0, Y_0, Z_0) is located at the immobile base of the arm, as shown in Fig. 1. Each of the manipulator links is modeled. This modeling

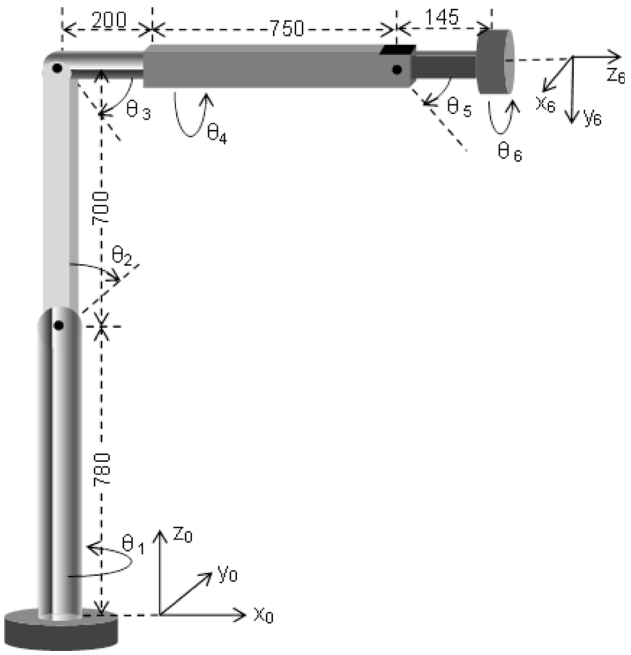

Fig. 1 The kinematic model of the robotic manipulator

describes the homogeneous transformation matrix A, which uses four link parameters [11]. This transformation is known as the Denavit–Hartenberg notation:

$$A = \mathrm{Rot}(z,\theta)\mathrm{Trans}(0,0,d)\mathrm{Trans}(a,0,0)\mathrm{Rot}(x,\alpha) \quad (1)$$

where θ_i is the joint angle from the X_{i-1} axis to the X_i axis about the Z_{i-1} axis, d is the distance from the origin of the $(i-1)$-th coordinate frame to the intersection of the Z_{i-1} axis along the Z_{i-1} axis, a_i is the offset distance from the intersection of the Z_{i-1} axis with the X_i axis to the origin of the i-th frame along the X_i axis, and $\propto_i$ is the offset angle from the Z_{i-1} axis to the Z_i axis about the X_i axis [11]:

$$T_6 = A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6 = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

Equation (2) is used to describe the forward kinematics solution for a robot manipulator. In this equation, n is the normal of the hand. Assuming a parallel-jaw hand, it is orthogonal to the fingers of the robotic arm. s is the sliding vector of the hand pointing in the direction of the finger motion as the gripper opens and closes. a is the approach vector of the hand pointing in the direction normal to the palm of the hand, in other words, normal to the tool-mounting plate of the robotic arm. p is the position vector of the hand pointing from the origin of the base coordinate system to the origin of the hand coordinate system, which is usually located at the center point of the fully closed fingers [14].

A 6-DOF Hitachi M6100 robot model was used in this study. Each link has 1 DOF. Therefore, the manipulator has a 6-DOF Cartesian position of the hand (x, y, z), which is obtained directly from the T_6 matrix (the matrix T_6 describes the position and also the orientation of the manipulator). The orientation of the hand is described according to the RPY (roll–pitch–yaw) rotation [18]. These rotations are the angles around the z, y, and x axes, respectively, as shown in Eq. (3):

$$\mathrm{RPY}(\emptyset_x, \emptyset_y, \emptyset_z) = \mathrm{Rot}(Z_O, \emptyset_z)\mathrm{Rot}(Y_O, \emptyset_y)\mathrm{Rot}(X_O, \emptyset_x) \quad (3)$$

Solving the T_6 matrix, the result is:

$$\emptyset_z = A\tan 2(n_y, n_x) \quad (4)$$

$$\emptyset_y = A\tan 2(-n_z, n_x\cos\emptyset_z + n_y\sin\emptyset_z) \quad (5)$$

$$\emptyset_x = A\tan 2(a_x\sin\emptyset_z - a_y\cos\emptyset_z, o_y\cos\emptyset_z - o_x\sin\emptyset_z) \quad (6)$$

These equations provide information about the position and orientation of the robot with respect to the real-world coordinate framework. The coordinate frames for each joint are used to describe the position and orientation of the

Table 1 Robot D–H parameters

Joint no.	Twist angle (α)	Joint offset, d (mm)	Joint angle, θ	Link length, a (mm)	Ranges (°)
1	0	0	θ_1	0	+150 to −150
2	$-\pi/2$	0	θ_2	780	+45 to −45
3	0	0	θ_3	700	+45 to −30
4	$-\pi/2$	750	θ_4	0	+180 to −180
5	$\pi/2$	200	θ_5	0	+120 to −120
6	$-\pi/2$	145	θ_6	0	+180 to −180

robot. Equation (7) describes the forward kinematics (FK) solution as a function of joint angles:

$$F_{\mathrm{FK}}(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6) = \left(X, Y, Z, \emptyset_x, \emptyset_y, \emptyset_z\right) \tag{7}$$

As can be seen from Eq. (7), the forward kinematics equation can be used to compute the Cartesian coordinates of the robot when the joint angles are known. However, the joint angles need to be computed for any given real-world Cartesian coordinate system in an industrial application. In Eq. (8), the inverse kinematics are shown as a function:

$$F_{\text{inverse kinematics}}\left(X, Y, Z, \Phi_x, \Phi_y, \Phi_z\right) = \left(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\right) \tag{8}$$

Equation (9) describes the inverse kinematics solution for the robotic manipulator [29, 35]. The 12 parameters, which define the position and orientation of the end effector, have been also included in this matrix.

$$\begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6 \tag{9}$$

According to the information given above, the obtained equations have been given below. Simulation software for the kinematics analysis of the robot model was prepared to generate data for the neural network using C++ programming language using the equations and information given above.

$${}^{0}_{1}A = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 780 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

$${}^{1}_{2}A = \begin{bmatrix} s_2 & c_2 & 0 & 700s_2 \\ 0 & 0 & 1 & 0 \\ c_2 & -s_2 & 0 & 700c_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

$${}^{2}_{3}A = \begin{bmatrix} -s_3 & -c_3 & 0 & -200s_3 \\ c_3 & -s_3 & 0 & 200c_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

$${}^{3}_{4}A = \begin{bmatrix} 0 & 0 & 1 & 750 \\ s_4 & c_4 & 0 & 0 \\ -c_4 & s_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

$${}^{4}_{5}A = \begin{bmatrix} 0 & 0 & -1 & 0 \\ s_5 & c_5 & 0 & 145s_5 \\ c_5 & -s_5 & 0 & 145c_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

$${}^{5}_{6}A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ s_6 & c_6 & 0 & 0 \\ -c_6 & s_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

$${}^{0}_{6}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16}$$

where

$$\begin{aligned}
r_{11} &= n_x = -c_1s_2c_3c_4c_5s_6 - c_1c_2s_3c_4c_5s_6 - s_1s_4c_5s_6 \\
&+ c_1s_2s_3s_5s_6 - c_1c_2c_3s_5s_6 - c_1s_2c_3s_4c_6 - c_1c_2s_3s_4c_6 \\
&+ s_1c_4c_6 \\
r_{21} &= n_y = -s_1s_2c_3c_4c_5s_6 - s_1c_2s_3c_4c_5s_6 \\
&+ c_1s_4c_5s_6 + s_1s_2s_3s_5s_6 - s_1c_2c_3s_5s_6 - s_1s_2c_3s_4c_6 \\
&- s_1c_2s_3s_4c_6 - c_1c_4c_6 \\
r_{31} &= n_z = -c_2c_3c_4c_5s_6 + s_2s_3c_4c_5s_6 + c_2s_3s_5s_6 \\
&+ s_2c_3s_5s_6 - c_2c_3s_4c_6 + s_2s_3s_4c_6 \\
r_{12} &= s_x = -c_1s_2c_3c_4c_5c_6 - c_1c_2s_3c_4c_5c_6 - s_1s_4c_5c_6 \\
&+ c_1s_2s_3s_5c_6 - c_1c_2c_3s_5c_6 + c_1s_2c_3s_4s_6 + c_1c_2s_3s_4s_6 \\
&- s_1c_4s_6 \\
r_{22} &= s_y = -s_1s_2c_3c_4c_5c_6 - s_1c_2s_3c_4c_5c_6 + c_1s_4c_5c_6 \\
&+ s_1s_2s_3s_5c_6 - s_1c_2c_3s_5c_6 + s_1s_2c_3s_4s_6 + s_1c_2s_3s_4s_6 \\
&+ c_1c_4s_6 \\
r_{32} &= s_z = -c_2c_3c_4c_5c_6 + s_2s_3c_4c_5c_6 + c_2s_3s_5c_6 \\
&+ s_2c_3s_5c_6 + c_2c_3s_4s_6 - s_2s_3s_4s_6 \\
r_{13} &= a_x = -c_1s_2c_3c_4s_5 - c_1c_2s_3c_4s_5 - s_1s_4s_5 \\
&- c_1s_2s_3c_5 + c_1c_2c_3c_5 \\
r_{23} &= a_y = -s_1s_2c_3c_4s_5 - s_1c_2s_3c_4s_5 + c_1s_4s_5 \\
&- s_1s_2s_3c_5 + s_1c_2c_3c_5 \\
r_{33} &= a_z = -c_2c_3c_4s_5 + s_2s_3c_4s_5 - c_2s_3c_5 - s_2c_3c_5 \\
p_x &= -145c_1s_2c_3c_4s_5 - 145c_1c_2s_3c_4s_5 - 145s_1s_4s_5 \\
&- 145c_1s_2s_3c_5 + 145c_1c_2c_3c_5 -
\end{aligned}$$

$$750c_1s_2s_3 + 750c_1c_2c_3 - 200c_1s_2s_3 + 200c_1c_2c_3 + 700c_1s_2$$

$$p_y = 145s_1s_2c_3c_4s_5 - 145s_1c_2s_3c_4s_5 + 145c_1s_4s_5 - 145s_1s_2s_3c_5 + 145s_1c_2c_3c_5 - 750s_1s_2s_3 + 750s_1c_2c_3 - 200s_1s_2s_3 + 200s_1c_2c_3 + 700s_1s_2$$

$$p_z = -145c_2c_3c_4s_5 + 145s_2s_3c_4s_5 - 145c_2s_3c_5 - 145s_2c_3c_5 - 750c_2s_3 - 750s_2c_3 - 200c_2s_3 - 200s_2c_3 + 700c_2 + 780$$

3 Neural networks in robotics

Neural networks have become one of the most popular topics in robotics since it can easily be used in many problems in the robotics. Neural network is a parallel-distributed information processing system. This system is composed of operators interconnected via one-way signal flow channels. NN stores the samples with a distributed coding, thus forming a trainable nonlinear system. It includes hidden layer(s) between the inputs and outputs. The main idea of the NN approach resembles the human brain functioning. Therefore, NN has a quicker response and higher performance than a sequential digital computer. Given the inputs and desired outputs, it is also self-adaptive to the environment so as to respond different inputs rationally. However, it has a complex internal structure, so the NN realized to date are composite systems imitating basic biological functions of neurons [21]. In robotics, neural networks can be used for the inverse kinematics solution, control or trajectory planning of a robotic manipulator. Since the neural network works with a quicker response, it is found successful from the view point of process time comparing to the conventional methods in real time applications.

3.1 Neural-network committee machine design

Using committee machine is an effective method to solve complex problems. A complex problem may be divided into several computationally simple tasks that are assigned to individual experts and these experts compute and produce their own results based on their designed tasks [32]. In the neural-network committee machine approach, n neural network are trained for solving the same problem independently. The neural networks are executed simultaneously for the given input data and their outputs are evaluated and combined to produce the final committee output to obtain better generalization and performance. The output combination module was often performed based on simple functions on the outputs of individual members in the committee machine, such as majority voting for classification and simple/weighted averaging for regression, without involving the input vectors of attributes [1].

In Fig. 2, IK solution system using neural-network committee machine is given. Training a neural network is the process of determining the best weights for the inputs to each of the units. The goal is to use the training set to produce weights for which the output of the network is as close as possible to the desired output for as many examples as possible in the training set. The training set is a part of the input dataset which is used for neural-network training, i.e., for adjustment of network weights. For training, in this study, three different sets of 5,000 data points, each consisting of the joint angles $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ described by the Cartesian coordinate parameters (X, Y, Z, o_x, o_y, o_z, n_x, n_y, n_z, a_x, a_y, a_z), were first generated separately using Eqs. (1)–(6) in the work volume of the robotic manipulator. This procedure is an attempt to obtain well-structured learning sets to make the learning process successful and easy. These values were recorded in files to form the learning sets for the networks. Each set of 5,000 data points was used in the training of the neural networks. As a validation set, an additional set of 1,000 data points was prepared. The validation set is a part of the dataset which is used to tune the network topology or network parameters other than weights. For example, it is used to define the number of units which are used to detect the moment when neural-network performance starts to deteriorate [8]. To choose the best network (i.e., by changing the number of units in the hidden layer), the validation set is used. As is well known, too much training can cause overfitting, and therefore the validation set may also need to be used to stop the training process early. As for the test dataset, a set of 1,000 data points was prepared and used to test each neural network to determine its level of success on the same dataset. The test dataset is a part of the input

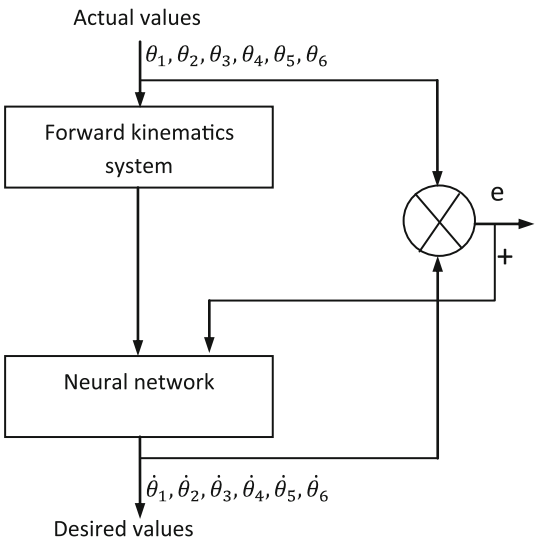

Fig. 2 IK solution system using neural network committee machine

dataset which is used to test how well the neural network will perform on new data. A sample data set produced for the training of neural networks is given in Table 2.

In Fig. 3, the structure of designed NNCM has been given. The designed NNCM consists of ten neural networks; however, using six neural networks in the committee machine has been found optimum for the solution system, the networks after six have been shown using dotted line.

In Fig. 4, the neural-network topology used in the committee machine design has been given. The feed-forward neural-network structure with sigmoid activation function has been used for each network in the committee machine. The gradient descent error learning algorithm has been used for training. The error is computed as the mean squared error (MSE). Graphical representation of MSE values versus the number of Epochs for each neural network has been given in Fig. 5. On the other hand, used parameters and number of epochs during the training of each network have been given in Table 3. In addition, MSE values according to number of epochs have been shown graphically for each network.

In Eq. (10), the euclidian distance equation has been given. This equation has been used to calculate the distance between end effector and the target known as end effector error. The selection of the best result among neural-network results in the committee machine has been done using this equation.

$$dE = [(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2]^{1/2} \quad (10)$$

In Table 4, computed MSE values and error ranges for the NNCM with n number of NN have been given. It is clearly seen in the table, using committee machine increased the success of the IK problem solution. The error has been reduced when we use more than one neural network in parallel. However, using more than six neural network does not make significant change about the decreasing the error. On the other hand, due to the online working feature and quick response ability feature of the NN, there is no significant increase on the process time. This also makes beneficial the usage of NN committee machine in the IK problem solution. As given in Table 4, the end effector error range is between 5.76 and 13.41 mm using a single network; however, the error range has been reduced to between 0.39 and 0.74 mm using six neural networks in the NNCM. On the other hand, using more than six networks does not make significant effect on the solution performance.

It is also evidently seen in Fig. 6 that there is no significant effect on the solution using more than six neural networks. In Fig. 7 logarithmic representation of MSE values versus number of networks in NNCM is given to show the effect of the number of networks in the committee machine more clearly.

Table 2 A sample data set produced for the training of neural networks

Inputs											Outputs						
n_x	n_y	n_z	s_x	s_y	s_z	a_x	a_y	a_z	p_x	p_y	p_z	θ_1	θ_2	θ_313	θ_44	θ_5	θ_6
0	−1	0	0	0	−1	1	0	0	1095	0	1480	0	0	0	0	0	0
0	−1	0	−1	0	0	0	0	−1	950	0	1335	0	0	0	0	90	0
0.83652	0.48296	−0.25882	0.22414	0.12941	0.96593	0.5	−0.86603	0	1295.85119	580.72846	1029.09665	30	45	−30	−90	90	−90
−0.83457	−0.43812	−0.33398	−0.53653	0.50885	0.67320	−0.125	0.74103	−0.65974	563.62881	689.20250	909.33776	45	0	30	60	45	60
−0.1875	0.97428	−0.125	−0.75777	−0.0625	0.64952	0.625	0.21651	0.75	813.11237	1188.598	1383.72475	60	45	−45	30	−60	120
0	0	−1	0	1	0	1	0	0	145	494.97475	324.97475	90	45	45	90	−90	90
−0.41602	0.62584	−0.65974	0.46692	0.76957	0.4356	0.78033	−0.12683	−0.61237	−186.85214	434.6299	1297.42378	120	−30	30	−45	120	−120
0.21875	−0.27063	−0.9375	−0.27063	0.90625	−0.32476	0.9375	0.32476	0.125	1151.54639	−507.87851	929.34278	−30	30	0	120	60	−60
−0.21651	−0.625	−0.75	0.31253	0.68342	−0.65974	0.9249	−0.37724	0.04737	545.47324	−767.19936	1961.86824	−60	0	−30	−120	−45	180
−0.30619.	0.25	−0.91856	0.88388	0.43301	−0.17678	0.35355	−0.86603	−0.35355	51.26524	−1621.81367	1223.70951	−90	45	−45	45	30	30

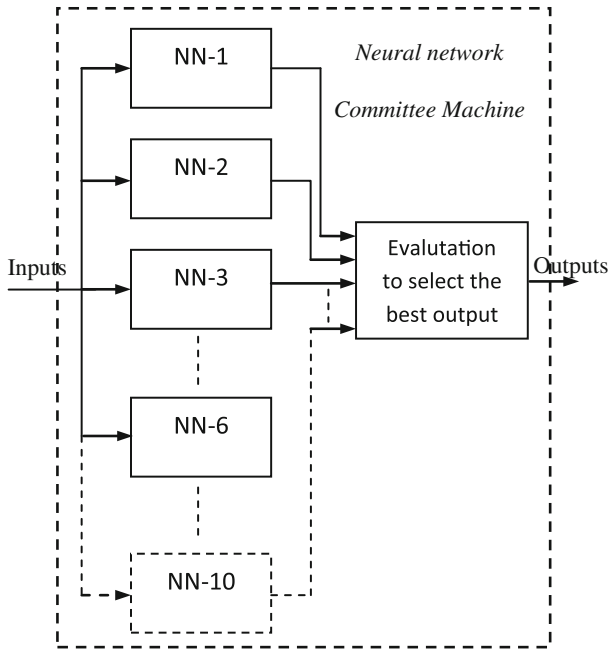

Fig. 3 The structure of designed NNCM

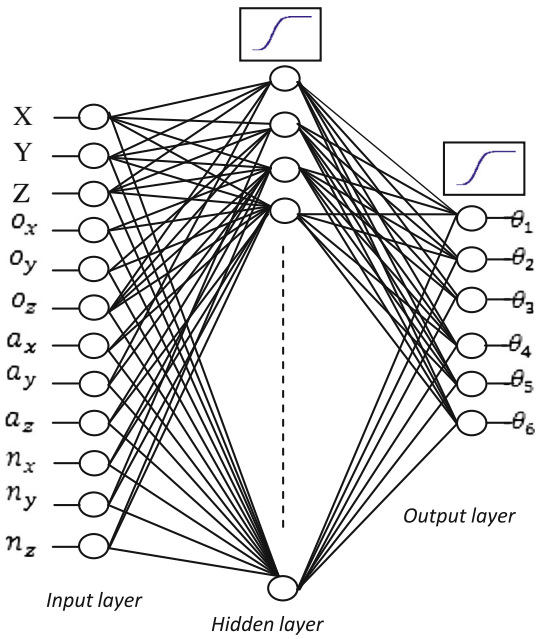

Fig. 4 The neural network topology used in the committee machine design

All algorithms have been coded using C++ programming language; the process times for the execution of the inverse kinematics solution for each committee machines on a six-core Intel Xeon 2.40-GHz computer workstation have presented in Table 5.

Table 3 Training parameters of each neural network in committee machine

Neural network no in the CM	Number of neurons in the hidden layer[a]	Learning rate	Momentum coefficient	Epoch at the learning
NN-1	20	0.30	0.85	1,150
NN-2	24	0.35	0.80	1,388
NN-3	20	0.28	0.82	1,284
NN-4	18	0.40	0.75	1,055
NN-5	25	0.38	0.78	1,182
NN-6	22	0.42	0.80	1,389
NN-7	18	0.36	0.84	1,416
NN-8	21	0.40	0.78	1,548
NN-9	25	0.35	0.88	1,317
NN-10	28	0.45	0.85	1,088

[a] The used range during the training was 5–40

Table 4 Computed MSE values and error ranges for the NNCM with n number of NN

Number of NN in committee machine	MSE value	End effector error range (mm)
A single NN	2.4875	5.76–13.41
CM with 2 NN	1.4461	4.79–8.96
CM with 3 NN	0.9829	3.28–5.81
CM with 4 NN	0.04879	1.27–3.08
CM with 5 NN	0.02819	1.27–2.62
CM with 6 NN	**0.00706**	**0.39–0.74**
CM with 7 NN	0.00639	0.39–0.74
CM with 8 NN	0.00621	0.39–0.74
CM with 9 NN	0.00611	0.39–0.69
CM with 10 NN	0.00601	0.39–0.69

4 Results and discussions

According to the neural-network-based inverse kinematics solution results in the literature [3, 13, 14, 19, 21, 30, 33] and the results obtained in this study, an error-minimization algorithm must be applied to the neural-network-based inverse kinematics solution. Oyama et al. [30] achieved RMS position errors of less than 10 mm. Bingul and Ertunc [3] observed large errors at the end of the learning cycle and explained these errors as a disadvantage of their algorithm. In particular, in one of their most recent papers, Hasan et al. [13, 14] presented errors in the inverse kinematics solution for joints 1–6 of 0.915, 0.135, 0.57, 4.79, 4.81, and 1.11 %, respectively. The graphical plots that they showed for their training set indicated similar joint errors. As it is seen from the results in the literature, since the neural networks are universal approximators with an

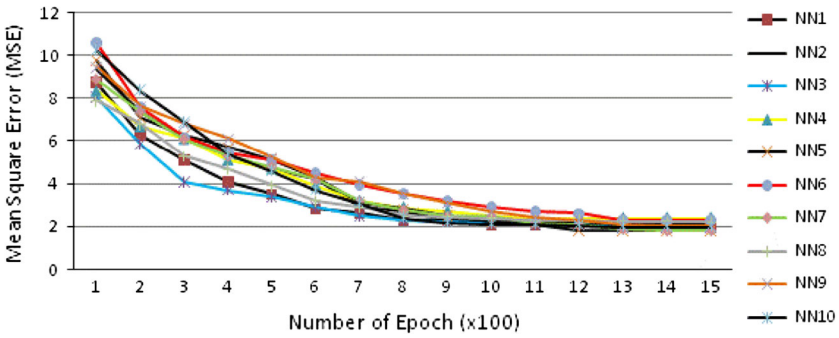

Fig. 5 Graphical representation of MSE values versus the number of Epochs for each neural network

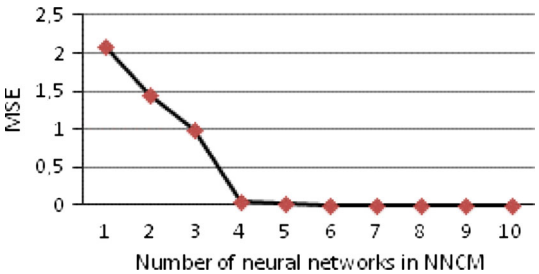

Fig. 6 MSE values versus number of networks in NNCM

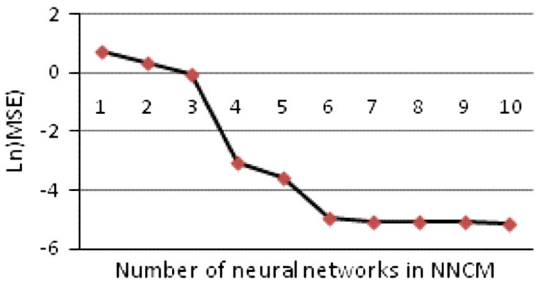

Fig. 7 Logarithmic representation of MSE values versus number of networks in NNCM

arbitrary precision, the obtained solution from a neural network requires improvement.

In this paper, inverse kinematics solution of a 6-DOF robotic manipulator was done using neural-network committee machine. Each neural network was designed for inverse kinematics solution using prepared simulation software. Each neural network was trained using separately prepared training data set. A neural-network output with the least error in the NN committee machine is selected as the solution result that is why the obtained solution will be with less error. It was observed that neural-network committee machine increased the success of the IK solution system from the view point of the error. On the other hand, there was not significant change on the learning performance when more than six neural networks were used. This situation can be seen from the mean square error (MSE) in Table 4 that after a certain number of networks in NNCM, the overall accuracy of the solution system does not improve. The proposed IK solution system is based on using parallel neural networks as a committee machine and evaluating the outputs to find the best prediction result. In the same way it is evidently observed that the performance of the solution has been increased in this paper compared to the use of a unique network.

Using designed NNCM, the end effector error was reduced to less than 1 mm. This solution performance may be satisfactory for many robotic applications. On the other hand, if more sensitive results are needed, any optimization algorithm like genetic algorithms, ant and colonies algorithm or simulated annealing algorithm can be used to reduce this 1 mm error to micrometer levels. But, these algorithms will increase the process time significantly. The online working and quick response feature of the neural networks make useful using NNCM. The proposed solution method can be used where the error around 0.5 mm is satisfactory. On the other hand, this error can be minimized using optimization algorithms and used in the applications, where the micrometer level sensitivity is important, but the process time is not important.

Table 5 The elapsed times for processing the NNCM with n number of NN

Neural network in committee machine	Processing time (μs)
A unique NN	789
CM with 2 NN	805
CM with 3 NN	834
CM with 4 NN	855
CM with 5 NN	871
CM with 6 NN	889
CM with 7 NN	911
CM with 8 NN	929
CM with 9 NN	943
CM with 10 NN	959

References

1. Abdel-Aal RE (2005) Improving electric load forecasts using network committees. Electr Power Syst Res 74:83–94
2. Astrom F, Koker R (2011) A parallel neural network approach to prediction of Parkinson's disease. Expert Syst Appl 38:12470–12474
3. Bingul Z, Ertunc HM (2005) Applying neural network to inverse kinematics problem for 6R robot manipulator with offset wrist. In: Proceedings of 7th International Conference on Adaptive and Natural Computing Algorithms, Coimbra, Portugal
4. Chen CH, Lin ZS (2006) A committee machine with empirical formulas for permeability prediction. Comput Geosci 32:485–496
5. Daunicht W (1991) Approximation of the inverse kinematics of an industrial robot by DEFAnet. In: Proceedings of IEEE International Joint Conference on Neural Networks, Singapore, pp 531–538
6. Denavit J, Hartenberg R (1955) A kinematic notation for lower-pair mechanisms based on matrices. ASME J Appl Mech 22:215–221
7. Duffy J (1980) Analysis of mechanisms and robot manipulators. Wiley, New York
8. Dugulena M, Barbuceanu FG, Teirelbar A, Mogan G (2012) Obstacle avoidance of redundant manipulators using neural networks based reinforcement learning. Robotics Comput Integr Manuf 28:132–146
9. Featherstone R (1983) Position and velocity transformation between robot end-effector coordinate and joint angle. Int J Robotics Res 2(2):33–45
10. Fernandes AM, Utkin AB, Lavrov AV, Vilar RM (2004) Development of neural network committee machines for automatic forest fire detection using lidar. Pattern Recogn 37:2039–2047
11. Fu KS, Gonzalez RC, Lee CSG (1987) Robotics: control, sensing, vision, and intelligence. McGraw-Hill, New York
12. Hahala J, Fahner G, Echmiller R (1991) Rapid learning of inverse robot kinematics based on connection assignment and topographical encoding (CATE). In: Proceedings of IEEE International Joint Conference on Neural Networks, Singapore, pp 1536–1541
13. Hasan AT, Hamouda AMS, Ismail N, Al-Assadi HMAA (2006) An adaptive-learning algorithm to solve the inverse kinematics problem of a 6 D.O.F. serial robot manipulator. Adv Eng Softw 37:432–438
14. Hasan AT, Ismail N, Hamouda AMS, Aris I, Marhaban MH, Al-Assadi HMAA (2010) Artificial neural network-based kinematics Jacobean solution for serial manipulator passing through singular configurations. Adv Eng Softw 41:359–367
15. Hirayama V, Fernandez FJR, Salcedo WJ (2006) Committee machine for LPG calorific power classification. Sens Actuators B 116:62–65
16. Jafari SA, Mashohor S, Varnamkhasti MJ (2011) Committee neural networks with fuzzy genetic algorithm. J Petrol Sci Eng 76:217–223
17. Karimpouli S, Fathianpour N, Roohi J (2010) A new approach to improve neural networks' algorithm in permeability prediction of petroleum reservoirs using supervised committee machine neural network (SCMNN). J Petrol Sci Eng 73:227–232
18. Karlık B, Aydın S (2000) An improved approach to the solution of inverse kinematics problems for robot manipulators. Eng Appl Artif Intell 13:159–164
19. Köker R (2005) Reliability-based approach to the inverse kinematics solution of robots using Elman's network. Eng Appl Artif Intell 18:685–693
20. Köker R (2012) A genetic algorithm approach to a neural network-based inverse kinematics solution of robotic manipulators based on error minimization. Information sciences, USA
21. Köker R, Oz C, Cakar T, Ekiz H (2004) A study of neural network based inverse kinematics solution for a three-joint robot. Robotics Auton Syst 49:227–234
22. Korein JU, Balder NI (1982) Techniques for generating the goal-directed motion of articulated structures. IEEE Comput Graphics Appl 2(9):71–81
23. Küçük S, Bingül Z (2004) The inverse kinematics solutions of industrial robot manipulators. In: Proceedings of the IEEE International Conference on Mechatronics, Kuşadası-Turkey, pp 274–279
24. Lee GCS (1982) Robot arm kinematics, dynamics, and control. IEEE Comput 15(12):62–79
25. Manocha D, Canny JF (1994) Efficient inverse kinematics for general 6R manipulators. IEEE Transact Robotic Autom 10(5):648–657
26. Martin JAH, Lope J, Santos M (2009) A method to learn the inverse kinematics of multi-link robots by evolving neuro-controllers. Neurocomputing 72:2806–2814
27. Martinetz T, Ritter H, Schulten K (1990) Three-dimensional neural net for learning visuomotor coordination of a robot arm. IEEE Trans Neural Netw 1(1):131–136
28. Mayorga RV, Sanongboon P (2005) Inverse kinematics and geometrically bounded singularities prevention of redundant manipulators: an artificial neural network approach. Robotics Auton Syst 53:164–176
29. Mi Z, Yang J, Kim JH, Abdel-Malek K (2011) Determining the initial configuration of uninterrupted redundant manipulator trajectories in a manufacturing environment. Robotics Comput Integr Manuf 27:22–32
30. Oyama E, Chong NY, Agah A (2001) Inverse kinematics learning by modular architecture neural networks with performance prediction networks. In: Proceedings of 2001 IEEE International Conference on Robotics and Automation, Seoul, Korea
31. Paul RP, Shimano B, Mayer GE (1981) Kinematics control equations for simple manipulators. IEEE Transact Syst Man Cybernetics SMC-11 6:66–72
32. Takenori K, Dagli C (2009) Hybrid approach to the Japanese candlestick method for financial forecasting. Expert Syst Appl 36:5023–5030
33. Tejomurtula S, Kak S (1999) Inverse kinematics in robotics using neural networks. Inf Sci 116:147–164
34. Vosniakos GC, Kannas Z (2009) Motion coordination for industrial robotic systems with redundant degrees of freedom. Robotics Comput Integr Manuf 25:417–431
35. Zhang D, Lei J (2011) Kinematic analysis of a novel 3-DOF actuation parallel manipulator using artificial intelligence approach. Robotics Comput Integr Manuf 27:157–163
36. Zhao ZQ, Huang DS, Sun BY (2004) Human face recognition based on multi-features using neural networks committee. Pattern Recogn Lett 25:1351–1358