Comparison of Inverse Kinematics Solutions Using Neural Network for 6R Robot Manipulator with Offset

Z. Bingul, H.M. Ertunc and C. Oysu
Kocaeli University, Mechatronics Engineering, Kocaeli, Turkey
e-mails: zaferb@kou.edu.tr

Abstract— **An Artificial Neural Network (ANN) using backpropagation algorithm is applied to solve inverse kinematics problems of industrial robot manipulator. 6R robot manipulator with offset wrist was chosen as industrial robot manipulator because geometric feature of this robot does not allow solving inverse kinematics problems analytically. In other words, there is no closed form solution for this problem. In order to define orientation of robot end-effector, three different representations are used here: homogeneous transformation matrix, Euler angles and equivalent angle axis. These representations were compared to obtain inverse kinematics solutions for 6R robot manipulator with offset wrist. Simulation results show that prediction performance from the approximation accuracy point of view is satisfactory with low effective errors based on 10 degrees data resolution.**

I. Introduction

ANN uses data sets to obtain the models of systems in fields such as robotics, factory automation, and autonomous vehicles. Their ability to learn by example makes artificial neural networks very flexible and powerful. Therefore, neural networks have been intensively used for solving regression and classification problems in many fields. In short, neural networks are nonlinear processes that perform learning and classification. Recently neural networks have been used in many areas that require computational techniques such as pattern recognition, optical character recognition, outcome prediction and problem classification. The current focus in learning research lies on increasingly more sophisticated algorithms for the off-line analysis of finite data sets, without severe constraints on the computational complexity of the algorithms.

In robot inverse kinematics learning, however, special constraints need to be taken into account when approaching a learning task. The complexity in the inverse kinematics problem of industrial robot manipulators arises from their geometry and nonlinear equations (trigonometric equations) occurring between Cartesian space and joint space. Some other difficulties in inverse kinematics problem are : i) kinematic equations are coupled, ii) multiple solutions and singularities may exist. Matematical solutions for inverse kinematics problems may not always correspond to physical solutions and method of its solution depends on the robot configuration. Conventional numerical approaches to the inverse calibration of robots are time-consuming and suffer from numerical problems of ill-conditioning and singularities.

The conversion of the position and orientation of a robot manipulator end-effector from Cartesian space to joint space is called as inverse kinematics problem. This relationship between joint space and Cartesian space is illustrated in Figure 1.

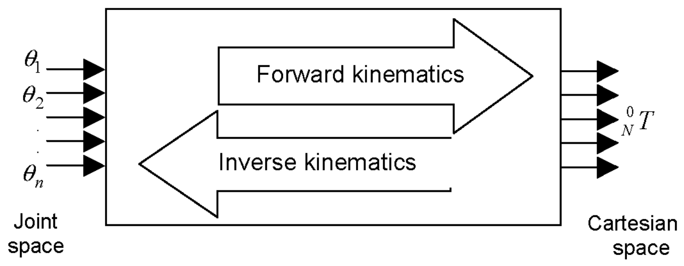

Figure 1. The schematic representation of forward and inverse kinematics.

There are three types of inverse kinematics solution: complete analytical solution (closed form solution), numerical solutions and semi-analitical solutions. In the first type, all of the joint variables are solved analytically according to given configuration data. Closed form solution is preferable because in many applications where the manipulator supports or is to be supported by a sensory system, the results from kinematics computations need to be supplied rapidly in order to have control actions. In the second type of solution, all of the joint variables are obtained iterative computational procedures. There are four disadvantages in these: a) incorrect initial estimations, b) before executing the inverse kinematics algorithms, convergence to the correct solution can not be guarantied, c) multiple solutions are not known, d) there is no solution, if the Jacobian matrix is singular. In the third type, some of the joint variables are determined analytically in terms of two or three joints variables and these joint variables computed numerically. Disadvantage of numerical approaches to inverse kinematics problems is also heavy computational calculation and big computational time.

When coupling of the position and orientation kinematics occurs, there may be no exist efficient closed form solutions. In this case, it is better to use Neural Networks to solve the inverse kinematics problems since trigonometric equations in inverse kinematics problems can not be solved analytically. In other words, it is not possible to formulate a mathematical model that has a clear mapping between Cartesian space and Joint space for inverse kinematics problem. To overcome this problem, ANN uses the samples to obtain the nonlinear model of such systems. Their ability to learn by example makes artificial neural networks very flexible and powerful when the traditional model-based modeling techniques break down. Many researchers have experimented with this approach by applying it to several robot configurations [1-12]. However, this approach has not applied to robot manipulator with offset wrist whose inverse kinematics solution is no exist in efficient closed form.

II. Robot Inverse kinematics

For a six jointed robot manipulator, the position and orientation of the end-effector with respect to the base is given by

$$ {}^{base}_{end-effector}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where r_{ij}'s represent the rotational elements of transformation matrix (i and j=1, 2 and 3) and p_x, p_y and p_z are the elements of position vector. The product of the link transformations yield forward kinematics of the robot manipulator

$$ {}^0_6T = {}^0_1T(q_1)\,{}^1_2T(q_2)\,{}^2_3T(q_3)\,{}^3_4T(q_4)\,{}^4_5T(q_5)\,{}^5_6T(q_6) \quad (2)$$

where q_i is the joint variable (revolute joint or prismatic joint) for joint i. This forward transformation matrix is equated with end-effector transformation matrix. To find the inverse kinematics solution, it should be solved for q_i as function of the known elements of ${}^{base}_{end-effector}T$. This approach is called homogenous transformation matrix method. In this approach, representation of the end-effector orientation is made with 3x3 rotation matrix (9 elements). Other representation is Euler angle approach in which 3 angles (3 elements) are enough to define the end-effector orientation. The rotation in this approach is described by

$$ R(\alpha,\beta,\gamma) = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix} \quad (3)$$

Third representation is equivalent angle axis approach (4 elements) in which one angles and one vector are enough to define the end-effector orientation. The rotation in this approach is described by

$$ R(\hat{K},\theta) = \begin{bmatrix} k_x k_x v\theta + c\theta & k_x k_y v\theta - k_z s\theta & k_x k_z v\theta + k_y s\theta \\ k_x k_y v\theta + k_z s\theta & k_y k_y v\theta + c\theta & k_y k_z v\theta - k_x s\theta \\ k_x k_z v\theta - k_y s\theta & k_y k_z v\theta + k_x s\theta & k_z k_z v\theta + c\theta \end{bmatrix} \quad (4)$$

where $\hat{K} = k_x\hat{i} + k_y\hat{j} + k_z\hat{k}$ and $v\theta = (1 - c\theta)$. Denavit-Hartenberg (D-H) kinematic parameters of the robot manipulator used in this study are listed in Table 1 and kinematic structure of this robot is shown in Figure 2.

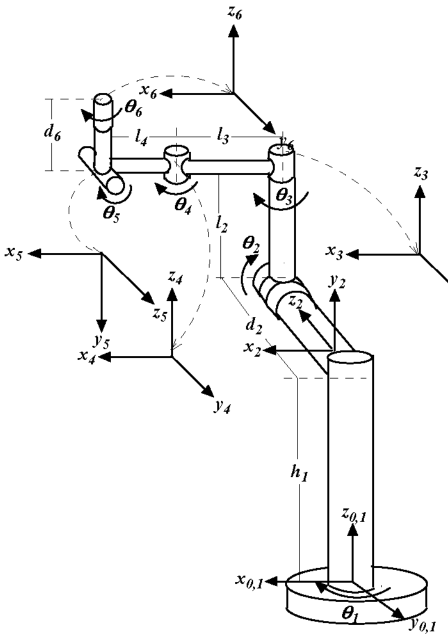

Figure 2. Coordinate frame attached to the rigid body of the robot manipulator.

TABLE I

D-H KINEMATIC PARAMETERS FOR ROBOT MANIPULATOR

i	θ_i	α_{i-1}	a_{i-1}	d_i
1	θ_1	0	0	h_1
2	θ_2	90	0	d_2
3	θ_3	-90	0	l_2
4	θ_4	0	l_3	0
5	θ_5	-90	l_4	0
6	θ_6	90	0	d_6

III. Artificial Neural Networks

An ANN tries to mirror the brain functions in a computerized way by resorting to the learning mechanism as the basis of human behaviour. Utilizing the samples from the experiments, ANNs can be applied to the problems with no algorithmic solutions or with too complex algorithmic solutions to be found. Their ability of learning by examples makes the ANNs more flexible and powerful than the parametric approaches [13].

An ANN consists of massively interconnected processing nodes known as neurons. Each neuron accepts a weighted set of inputs and responds with an output. Such a neuron first forms the sum of the weighted inputs given by:

$$ n = \left(\sum_{i=1}^{P} w_i x_i \right) + b \quad (5)$$

where P and w_i are the number of elements and the interconnection weight of the input vector x_i, respectively, and b is the bias for the neuron [14]. Note that the knowledge is stored as a set of connection weights and biases. The sum of the weighted inputs with a bias is processed through an activation function, represented by *f*, and the output that it computes is:

$$f(n) = f\left[\left(\sum_{i=1}^{P} w_i x_i\right) + b\right] \tag{6}$$

Basically, the neuron model represents the biological neuron that fires when its inputs are significantly excited, i.e., *n* is big enough. There are many ways to define the activation function such as threshold function, sigmoid function, and hyperbolic tangent function.

An ANN can be trained to perform a particular function by adjusting the values of connections, i.e., weighting coefficients, between the processing nodes. In general, ANNs are adjusted/trained to reach from a particular input to a specific target output using a suitable learning method until the network output matches the target. The error between the output of the network and the desired output is minimized by modifying the weights and biases. When the error falls below a determined value or the maximum number of epochs is exceeded, training process is ceased. Then, this trained network can be used for simulating the system outputs for the inputs which have not been introduced before.

The architecture of an ANN is usually divided into three parts: an input layer, a hidden layer(s) and an output layer. The information contained in the input layer is mapped to the output layer through the hidden layer(s). Each unit can send its output to the units only on the higher layer and receive its input from the lower layer. For a given modelling problem, the numbers of nodes in the input and output layers are determined from the physics of the problem, and equal to the numbers of input and output parameters, respectively.

IV. THE STRUCTURE OF BACKPROPAGATION NEURAL NETWORK

The neural network structure was shown in Figure 3. The inputs of the network were selected from three different input data sets, namely first representation, second representation and third representation which have 12 parameters, 6 parameters and 7 parameters, respectively. The network was trained three times separately for each input data set. In the first representation, the 12 inputs to the network are the nine elements of the rotation matrix (r_{ij} *i,j=1:3*) and the position vector (P_x, P_y, P_z). In the second representation, the first three inputs obtained from equation 3 (Euler angles) and the position vector is arranged in vector form. In the final representation, the first four inputs (one vector and one angle) calculated from equation 4 and the position vector is similarly taken. There are six outputs of the network for all three representations. The outputs are three joint angles $_1$, $_2$, $_3$ and three wrist joint angles $_4$, $_5$, and $_6$.

A simulation data set for training and testing of neural network was generated by kinematics equations 2-4. The data set includes $1.5\text{x}10^6$ input-output pairs for every 10° joint angle. A working data set 8000 data points that was composed of input-output vector pairs was selected randomly from the data set. Again, the input vector was arranged at three different representations as explained above. The output vector, or also called as target vector, includes six joint angles. While 70% of the data set was randomly assigned as training set, the remaining 30% was employed for testing. Both the input and output variables were normalized to the [-1,1] range.

As seen from Fig.3, after several trial and errors for determining the architecture of the neural network, a multi-layer backpropagation network was decided to consist of two hidden layers with 12 and 24 neurons and an output layer with 6 neurons. The input layer has the number of neurons that is equal to number of parameters (12, 6 and 7) for each corresponding representation. While the neurons at input and hidden layer have sigmoid activation functions, the output layer has linear neurons so that the network could produce values outside the range -1 to +1. Input vectors and the corresponding target vectors from training set are used to train the network until it can approximate a function between the input and output variables. Training procedure adjusted the weighting coefficients using Levenberg-Marquardt algorithm that is the fastest training algorithm for the networks of moderate sizes. In this procedure, a termination criterion for error goal was set as 0.01; and all the weighting coefficients were initially assigned randomly. Then input vectors from the test data set that were not employed in the training procedure were presented to the trained backpropagation network. The responses of the network, i.e. joint angles, are compared to targets in the test data.

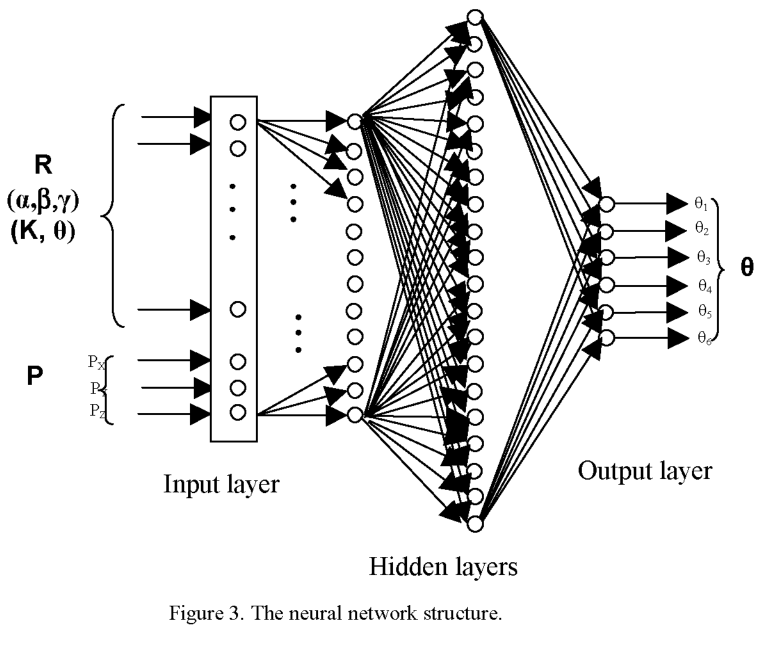

Figure 3. The neural network structure.

V. RESULTS AND DISCUSSION

The main drawback of using neural networks to approximate the inverse kinematics of robot arms is the high number of training samples required to attain an acceptable precision. After training 70 % of the input-output vector pairs selected randomly out of the data set having 8000 pairs for each representation separately, the network was tested on the 30 % of the pairs of data set that are not included in the training set. It is expected that trained backpropagation networks give reasonable responses to presented inputs which the networks have never seen. Figure 4 shows 10 points of network output represented by 'x, +, o' for each representation, respectively and corresponding targets represented by a solid line for each angle. As seen from the figure, the neural network predicted the actual joint angle, target, successfully at some angle values. To obtain a better idea for the performance of the neural network prediction, the effective errors or root mean square (RMS) errors of joint angles for each representation was plotted in bar graphic form in Figure 5. The horizontal lines (Figure 5) illustrate the mean of the RMS errors from bottom to top corresponding to first, second and third representations. The mean values are namely 8.78, 10.16 and 10.64. According to the mean values, it can be said that the neural network with inputs from first representation has better performance compared to other ones.

Furthermore, Figure 6 shows how the error values fluctuate around the origin for the same data points given in Figure 4. Note that, the biggest effective error is found for $_4$. In fact, when Figures 4, 5 and 6 are examined, the worst prediction performance and bigger error amplitudes are observed for fourth joint angle.

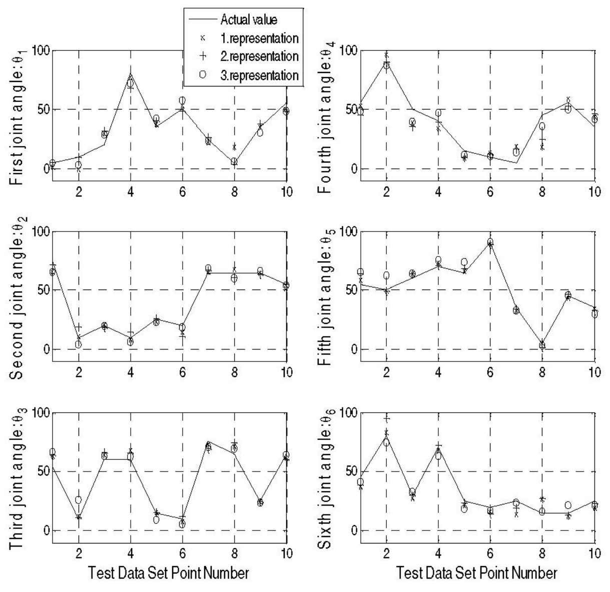

Figure 4. The output of the neural network and target values.

VI. Concilision

Neural Networks using backpropagation algorithm has been applied to inverse kinematics problem for 6R robot manipulator with an offset wrist which does not have closed form solution. Inverse kinematics computation using an artificial neural network that learns the inverse kinematics of a robot manipulator has been employed by many researchers. The inverse kinematics function of the robot manipulator with an offset wrist, much more difficult robot structure, is a multi-valued and discontinuous function. Therefore, it is difficult for a well-known multi-layer neural network to approximate such a function. A coarse mapping can be obtained easily, but an accurate representation of the true mapping is often not feasible or extremely difficult. Because of very wide range robot training data, big neural network structure is needed to obtain finer mapping. Simulation results show that prediction performance from the approximation accuracy point of view is satisfactory with low effective errors based on 10 degrees data resolution.

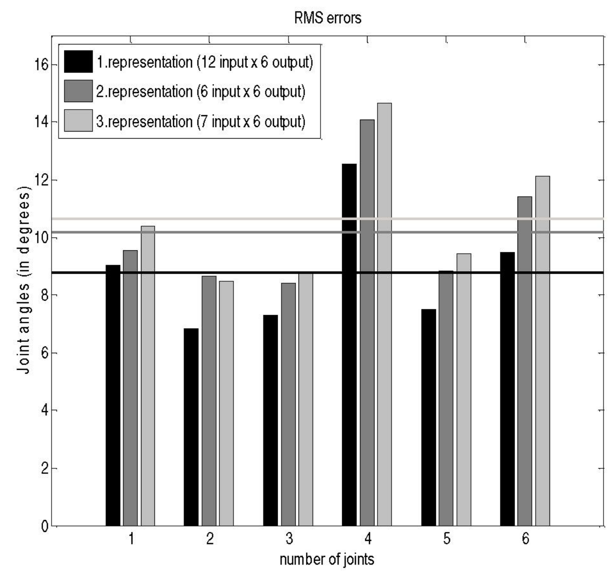

Figure 5. RMS errors of joint angles in degrees for each input representation.

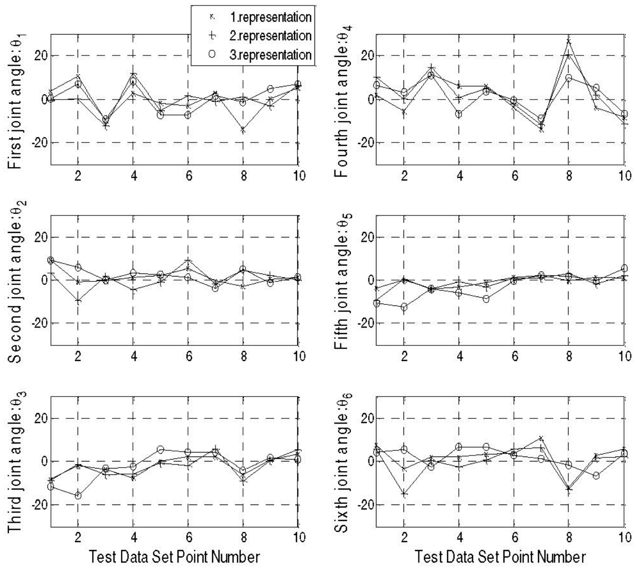

Figure 6. Errors of joint angles in test data set.

References

[1] Guez, A., Ahmad, Z., 1989. A Solution to the Inverse Kinematic in Robotics Using Neural Network Processing. IJCNN89. vol. II, 299–304.

[2] Guez, A., Ahmad, Z., 1989. Accelerated Convergence in the Inverse Kinematics via Multilayer Feedforward Networks. IJCNN89. vol. II, 341–344.

[3] Krose, BJA. and van der Smagt, PP., 1993. An Introduction to Neural Networks. 5th ed. Chap. 7 Robot Control, University of Amsterdam.

[4] Lee, S., Kil, RM., 1990. Robot Kinematic Control based on Bidirectional mapping Neural Network. IJCNN90. vol. III, 327–335.

[5] Lou, YF., Brunn, P., 1999. A Hybrid Artificial Neural Network Inverse Kinematic Solution for Accurate Robot Path Control. Proceedings of the I MECH E Part I Journal of Systems & Control in Engineering, 213: 23-32.

[6] Martin, P., Millan, JDR., 2000. Robot arm reaching through neural inversions and reinforcement learning. Robotics and Autonomous Systems, 31: 227-246.

[7] Nguyen, L., Patel, RV., Khorasani, K., 1990. Neural Network Architectures for the Forward Kinematics Problem in Robotics. IJCNN90. vol. III, 393–399.

[8] Oyama, E., et.al,, 2001. Inverse Kinematics Learning by Modular Architecture Neural Networks with Performance Prediction Networks. Proc. IEEE Int. Conf. on Robotics and Automation, 1006-1012.

[9] Torras C., 2003. Handbook of Brain Theory and Neural Networks, 2nd ed. MIT Press, Cambridge, Massachusetts, 979-983.

[10] Torras, C., 1993. Symbolic planning versus neural control in robots. From Neural Networks to Artifical Intelligence, Research Notes in Neural Computing 4, Springer-Verlag: Berlin Heidelberg New-York, 509-523.

[11] Vijayakumar, S., D'souza, A., Shibata, T., Conradt, J., Schaal, S., 2002. Statistical Learning for Humanoid Robots. Autonomous Robots, 12: 55-69.

[12] Xiaolin, Z., Lewis, J., N-Nagy, FL., 1996. Inverse Robot Calibration Using Artificial Neural Networks. Engineering Applications of Artificial Intelligence, 9: 83-93.

[13] Hagan, M.T., Demuth, H.B., Beale, M., 1995. Neural Network Design, PWS Publishing Company, Boston.

[14] Haykin, S., 1994. Neural Networks: A Compherensive Foundation, Mc Millan, NewJersey .