Robotics and Autonomous Systems 58 (2010) 622–633

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Kinematic control of a redundant manipulator using an inverse-forward adaptive scheme with a KSOM based hint generator

Swagat Kumar [a], Laxmidhar Behera [b,*], T.M. McGinnity [b]

[a] Department of Electrical Engineering, Indian Institute of Technology Kanpur, Uttar Pradesh, India
[b] Intelligent Systems Research Centre, University of Ulster, Magee Campus, Londonderry, UK

ARTICLE INFO

Article history:
Received 10 October 2008
Received in revised form
26 November 2009
Accepted 3 December 2009
Available online 13 January 2010

Keywords:
Redundant manipulator
Inverse kinematic solution
Kohonen Self-Organizing Map (KSOM)
Network inversion
Radial Basis Function Network (RBFN)
KSOM-SC architecture
Redundancy resolution

ABSTRACT

This paper proposes an online inverse-forward adaptive scheme with a KSOM based hint generator for solving the inverse kinematic problem of a redundant manipulator. In this approach, a feed-forward network such as a radial basis function (RBF) network is used to learn the forward kinematic map of the redundant manipulator. This network is inverted using an *inverse-forward* adaptive scheme until the network inversion solution guides the manipulator end-effector to reach a given target position with a specified accuracy. The positioning accuracy, attainable by a conventional network inversion scheme, depends on the approximation error present in the forward model. But, an accurate forward map would require a very large size of training data as well as network architecture. The proposed *inverse-forward* adaptive scheme effectively approximates the forward map around the joint angle vector provided by a hint generator. Thus the inverse kinematic solution obtained using the network inversion approach can take the end-effector to the target position within any arbitrary accuracy.

In order to satisfy the joint angle constraints, it is necessary to provide the network inversion algorithm with an initial hint for the joint angle vector. Since a redundant manipulator can reach a given target end-effector position through several joint angle vectors, it is desirable that the hint generator is capable of providing multiple hints. This problem has been addressed by using a Kohonen self organizing map based sub-clustering (KSOM-SC) network architecture. The redundancy resolution process involves selecting a suitable joint angle configuration based on different task related criteria.

The simulations and experiments are carried out on a 7 DOF PowerCube™ manipulator. It is shown that one can obtain a positioning accuracy of 1 mm without violating joint angle constraints even when the forward approximation error is as large as 4 cm. An obstacle avoidance problem has also been solved to demonstrate the redundancy resolution process with the proposed scheme.

1. Introduction

The inverse kinematic problem of a manipulator is a difficult problem in robotics which has attracted a lot of attention not only in the robotics community but also in the soft-computing research community. The main issue in solving these problems lies in the fact that they are highly nonlinear and there exist multiple solutions. In the case of a redundant manipulator, the number of inverse kinematic solutions may become infinite and closed form solutions are impossible to find in general. The methods available for solving these problems may be broadly grouped into two major classes—*classical techniques* based on the differential kinematic relationship between the end-effector velocities and the corresponding joint angle velocities, and *learning based techniques* based on biologically inspired approaches like neural networks, fuzzy logic, machine learning etc.

The classical techniques require previous knowledge of the manipulator geometry represented by the Jacobian matrix. Pseudo-inverse, Jacobian transpose, damped least square methods and their variants [1,2] fall into this category. On the other hand, learning based approaches can be used to learn the kinematic relationship of the manipulators during action–perception cycles without requiring explicit knowledge of their geometry [3].

Among all the learning schemes, neural networks have been extensively used for solving the inverse kinematic problem. This includes Multiple Layer Perceptrons (MLP) [4], CMAC networks [5], Radial Basis Functions (RBFs) [6,7], Locally Weighted Projection Regression (LWPR) [8,9], Reinforcement Learning (RL) [3], Hopfield Networks [10] and Kohonen's Self-Organizing maps (KSOM) [11,12]. The neural network based methods may be distinguished into three different classes: direct inversion modeling

* Corresponding author.
E-mail addresses: swagatk@iitk.ac.in (S. Kumar), l.behera@ulster.ac.uk (L. Behera), tm.mcginnity@ulster.ac.uk (T.M. McGinnity).

0921-8890/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2009.12.002

[4,11,13–15], distal learning [16] and differential kinematics models [17–19]. We focus on a distal learning approach where the inverse kinematic solution is obtained from a network which is trained a priori to learn the forward kinematic relationship of the manipulator. Very few papers are available in the literature that solve the inverse kinematic problem using the distal approach. Readers may refer to two such papers [3,20] that use network inversion to solve the inverse kinematic problem. Since the inverse kinematic problem is ill-posed, authors in [3] train another network to match a constraint measure and invert it to exploit the self-motion capability of the manipulator. The redundancy is used to avoid joint angle limits using this self-motion network. On the other hand, authors in [20] invert several modular neural networks trained over input–output partitioned space to generate multiple solutions for a given end-effector position and then one particular solution is selected by using a task related criterion. In general, it can be said that the limitations of network-inversion schemes render them unattractive for solving the inverse kinematic problems of redundant manipulators.

In this paper we re-address the various limitations of a network inversion approach and provide alternate and simpler solutions to overcome these limitations. A radial basis function network is used to approximate the forward kinematic map between the manipulator joint angle vector and its end-effector position. The trained network is used to update the joint angle vector through network inversion so as to reduce the error between the current end-effector position and the desired target point in the workspace. This approach has following three limitations:

(i) The inverse kinematic solution obtained by the network inversion scheme depends on the approximation error present in the forward map. Hence one needs a large size of training data as well as a large network to reduce the forward approximation error.
(ii) In order to satisfy the joint angle constraints of the manipulator, the network inversion algorithm must be initialized with a suitable hint [13,3,21]. Martin et al. [3] use average permissible joint angle value within the physical bounds as the initial hint for inversion. On the other hand, Mao et al. [13] use the center of the range of joint angle travel as the initial hint and Assal et al. [21] use a fuzzy neural network to provide the initial hint from its relative position and relative velocity compared to their respective limits.
(iii) In the case of a redundant manipulator, more than one joint angle vector is associated with each end-effector position. Thus, the hint generator should be capable of generating multiple hints. The network inversion around each hint will provide an appropriate inverse kinematic solution. This attribute is not present among the existing hint generators.

The first problem is solved by using an *inverse-forward* adaptation scheme where the forward network is re-trained using a new input–output data pair whenever the joint angle vector obtained from the last network inversion cycle does not drive the actual manipulator to the desired end-effector position. The new input–output data pair consists of the current joint angle vector obtained from the last network inversion cycle and the end-effector position attained by the manipulator when driven with this joint angle vector. Since it is difficult to learn a global solution during forward training as discussed earlier, re-training the network using input–output data generated around a local hint using an error corrector loop effectively facilitates local search around the current solution vector and hence, reduces the burden on the forward training. It is shown in the simulation section that the proposed adaptation scheme eliminates the error between the network output and the actual end-effector position effectively even when a large approximation error is present in the forward training. This finding is also verified through experiments on an actual 7 DOF manipulator. The application of network inversion in solving the inverse kinematic problem has been limited owing to its inability to attain very high positioning accuracy. The proposed scheme is significant in this aspect as the results produced in the paper are the best available in this category of solutions and hence promise to increase its applicability.

In order to address the last two problems, a KSOM-based sub-clustering (KSOM-SC) network architecture [22] is used as a hint generator. In this architecture, the Cartesian workspace and the joint angle space of the manipulator are discretized using a 3-dimensional Kohonen's Self-Organizing lattice [23] in such a way that each cluster in the workspace is associated with multiple clusters in the joint angle space. The KSOM-SC network architecture was introduced in [22] where it was shown to provide high accuracy in a *pick-and-place* kind of task. The theories related to KSOM-SC network were further developed in [24] where it was applied to solve the redundancy resolution problem while tracking a trajectory. It was shown to preserve all the desirable qualities of inverse kinematic solutions like, the *conservative property*, the *joint angle boundedness* and high positioning accuracy. These theoretical observations were corroborated with actual experiments on a 7 DOF PowerCube robot.

In this paper, the initial joint angle value needed for the inversion algorithm is obtained from the KSOM-SC architecture. Since the KSOM-SC architecture provides multiple joint angle vectors for a given end-effector position, it is possible to select one out of many available configurations by using different criteria. Unlike conventional techniques, changing the redundancy resolution criterion does not require re-training of the network and hence this redundancy resolution scheme is far simpler than those existing in the current literature. Readers are referred to [24] for an in-depth study of this architecture.

Finally, the proposed schemes are implemented on a 7 DOF PowerCube™ manipulator. The details of an obstacle avoidance experiment is presented to demonstrate the redundancy resolution capabilities of the proposed scheme. Note that only a 3-D positioning task is considered without taking into account the orientation of the end-effector.

The paper is organized as follows. The RBF based network inversion scheme is presented in the next section. A review of KSOM-SC architecture is presented in Section 3. The obstacle avoidance scheme using KSOM and RBF inversion is discussed in Section 4. The simulation and experimental results are provided in Section 5 followed by Conclusion in Section 6.

2. Network inversion based inverse kinematic solution through inverse-forward adaptive scheme

In a conventional network inversion scheme, a feed-forward network is first trained to approximate the forward kinematic map of a manipulator and then this network is inverted to find the joint angle vector that takes the robot end-effector towards the target position. The network inversion algorithm updates the input (joint angle) vector so as to reduce the error between the network output and the desired end-effector position. The accuracy of the solution obtained from network inversion depends on the accuracy of the forward map. Because of the high nonlinearity present in the forward kinematic map, a large number of examples is needed for training the feed-forward network so as to obtain a low approximation error. One instance of forward training is shown in Fig. 1. This figure shows the average positioning error for a RBF network with 700 hidden nodes during the training process. It is seen that the average positioning accuracy obtained after training over 7×10^5 data points is only 4 cm. The training is carried out with a gradient descent algorithm with a learning rate

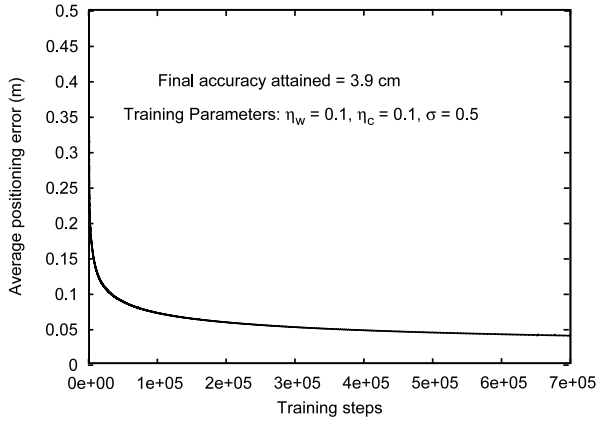


Fig. 1. Forward training of RBFN. Since the forward kinematic equations are highly nonlinear, a large number of data points is needed for good approximation.

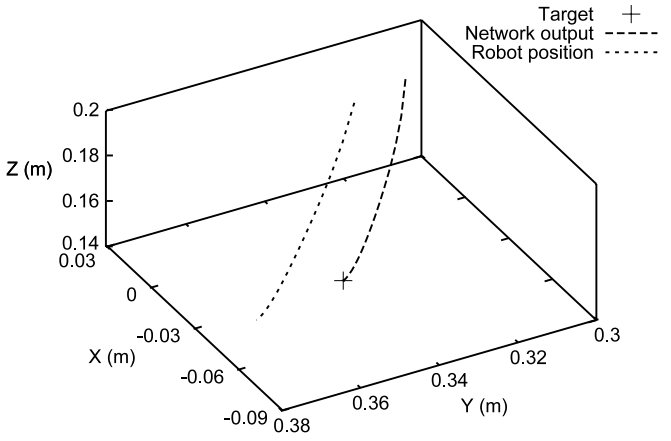


Fig. 2. Reaching a target point with only network inversion. It is seen that while the network output reaches the target point, the actual robot end-effector position does not reach it. It is due the approximation error present in the forward learning. In order to obtain good positioning accuracy, the RBF network must be trained with a huge amount of data points.

of 0.1. The figure shows the best result obtained after trying out several values for the learning rate parameter. Even though some faster algorithms like *Levenberg–Marquardt* may also be used, the amount of data necessary for obtaining good accuracy still remains high. Without any loss of generality, it can be said that it is difficult to obtain very high accuracy in learning the forward kinematic map using feed-forward networks. The necessity of a highly accurate forward model can be understood by studying Fig. 2. In this figure it is seen that even though the network output reaches the target position, the actual robot end-effector does not reach the target point. The positioning error for the actual manipulator is the same as the approximation error present in the forward model. The positioning error obtained is not tolerable in most industrial applications. Apart from dealing with the above mentioned problem, one also needs to initialize the inversion algorithm with a suitable joint-angle value so as to satisfy joint angle constraints. This initial joint angle value determines the pose of the manipulator during robot movement. Since a redundant manipulator can reach a target position with more than one joint angle configuration, it is necessary to resolve redundancy as well during selecting the initial joint angle value. From this discussion, we conclude that the conventional network inversion algorithms are not adequate for dealing with redundant manipulators with high degrees of freedom. In order to address these limitations, a *forward-inverse* adaptive scheme is suggested as explained below.

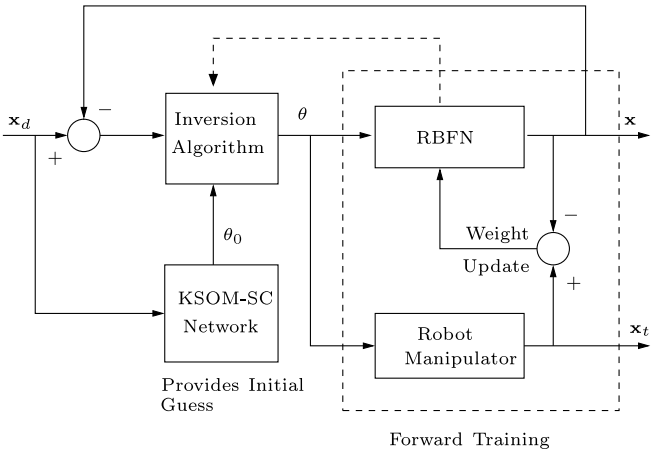


Fig. 3. Scheme for learning inverse kinematics using RBF network. During testing phase, the initial guess of the suitable joint angle vector is provided by a KSOM-SC network architecture. An initially trained forward model is used by the inversion algorithm to compute the next joint angle vector.

The schematic of the proposed method for solving the inverse kinematic problem of a redundant manipulator is shown in Fig. 3. The problem is to find a suitable joint angle vector $\boldsymbol{\theta}$ which can drive the manipulator to reach a desired end-effector position $\mathbf{x}_d$ without violating the joint angle limits. The method consists of following steps:

(i) The forward kinematic map between the manipulator joint angle vector and its end-effector position is approximated using a RBF network. The input to the RBFN is the joint angle vector $\boldsymbol{\theta}$ and the output of the network is the end-effector position $\mathbf{x}$. The actual manipulator end-effector position is represented by $\mathbf{x}_t$. The network is trained using the back-propagation algorithm to minimize the following output error function for a given input vector $\boldsymbol{\theta} = \boldsymbol{\theta}_t$.

$$E_1 = \frac{1}{2}(\mathbf{x} - \mathbf{x}_t)^2. \tag{1}$$

The forward training block is shown in the dashed box in Fig. 3. The input–output pair $(\boldsymbol{\theta}_t, \mathbf{x}_t)$ is generated using the forward kinematic model of the manipulator. It is not necessary to have a very low approximation error during forward training, as it would rather require a large number of training data sets.

(ii) A KSOM-based sub-clustering (KSOM-SC) network is used to provide an initial guess $(\boldsymbol{\theta}_0)$ to the inversion algorithm as shown in Fig. 3. Assal et al. [21] use a fuzzy neural network to provide this initial guess and, Martin et al. [3] start from the current robot configuration. Since this architecture provides multiple joint angle vectors for each end-effector position, one can use a specific criterion to choose one among the available solutions. This initial value determines the pose of the manipulator during its motion. In other words, this architecture facilitates redundancy resolution at the position level without any cumbersome optimization process. An overview of this architecture is provided in Section 3 and the readers are referred to [22,24] for a detailed treatment on the topic.

(iii) For a given target position $\mathbf{x}_d$, update the input vector of the RBF network so as to reduce the error function given by

$$E_2 = \frac{1}{2}(\mathbf{x}_d - \mathbf{x})^2. \tag{2}$$

Using the gradient-descent rule, the update law for the joint angle vector may be written as

$$\boldsymbol{\theta}^{new} = \boldsymbol{\theta}^{old} + \Delta\boldsymbol{\theta} \tag{3}$$

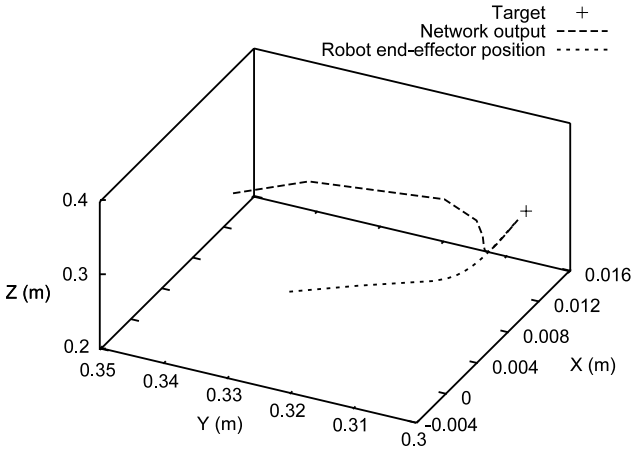


Fig. 4. Reaching a target point using network inversion. In this case, forward training is carried out with a new data pair whenever the positioning error for the actual end-effector is more than the desired value. This scheme reduces the load on forward training.

where

$$\Delta\boldsymbol{\theta} = -\eta\frac{\partial E_2}{\partial\boldsymbol{\theta}} = \eta(\mathbf{x}_d - \mathbf{x})\frac{\partial\mathbf{x}}{\partial\boldsymbol{\theta}} \tag{4}$$

and $\frac{\partial\mathbf{x}}{\partial\boldsymbol{\theta}}$ is the input Jacobian matrix of the RBF network. η is the learning rate for the network inversion.

(iv) The new end-effector position attained by the actual manipulator when driven by the joint angle vector $\boldsymbol{\theta} = \boldsymbol{\theta}_{new}$ is represented by $\mathbf{x}_t$ as shown in Fig. 3. If $\|\mathbf{x} - \mathbf{x}_d\| > \varepsilon$, the RBFN is trained again with the new input–output pair $(\boldsymbol{\theta}, \mathbf{x}_t)$. $\varepsilon > 0$ is an user defined constant which defines the final positioning accuracy to be attained by the manipulator.

The steps (i) to (iv) are repeated until $\|\mathbf{x}_d - \mathbf{x}\| < \varepsilon$. It is to be noted that the training phase only involves forward training which is carried out off-line. In the testing phase, the network is presented with a desired position $\mathbf{x}_d$ and then the trained network undergoes "inverse-forward" update cycles until the actual manipulator end-effector reaches the desired position. The "inverse-forward" update cycle is carried out during the online operation. Each *inverse-forward* update cycle involves two steps:

(i) An inverse module that updates the input vector so as to reduce the error between the network output $\mathbf{x}$ and the target position $\mathbf{x}_d$.

(ii) A forward update module that modifies the weights of the network so as to reduce the error between its output and the actual end-effector position. This forward update can be considered as a fine tuning process to improve the accuracy of the forward map.

Unlike conventional approaches which aim to learn the global solution through forward training, the proposed *inverse-forward* model aims to learn a local solution around the current robot configuration. The improvement achieved using this scheme is shown in Fig. 4. The initial error between the network output and the actual manipulator end-effector is due to the approximation error present in the forward training. However as the "inverse-forward" update proceeds, the network output as well as the actual end-effector converge towards the target position with the specified accuracy. The joint angle values for this motion are shown in Fig. 5. It is seen that the joint angles satisfy the physical constraints specified by their normalized value ± 1.

In the simulation section, it will be shown that this scheme achieves high positioning accuracy irrespective of the forward approximation error. The consequences of local learning will also be analyzed in detail in this section.

3. Hint generation using KSOM-based sub-clustering (KSOM-SC) network

Kohonen's Self Organizing Maps (KSOM) [23] have been used by several researchers for capturing the topology of the input (Cartesian) and the output (joint angle) spaces of a manipulator system [11,25,26]. In most of these cases, the clusters created in the input and the output spaces are related through a one-to-one map between them. Since a redundant manipulator can reach a target position in the workspace using several joint angle vectors, the authors in [22] proposed a model where each input cluster in the Cartesian space is associated with more than one cluster in the joint angle space. This is called a KSOM based sub-clustering network (KSOM-SC) architecture. Since it is difficult to learn one-to-many mapping using a neural network, this architecture is useful in dealing with redundant inverse kinematic solutions.

The KSOM-SC network architecture is shown in Fig. 6. In this architecture, each lattice neuron γ is associated with a 3-dimensional weight vector $\mathbf{w}_\gamma$ and several 7-dimensional joint angle vectors as shown in Fig. 6. Let us assume that each lattice neuron γ is associated with N_γ numbers of angle vectors given by $\boldsymbol{\theta}^j_\gamma, j = 1, 2 \ldots, N_\gamma$. The number N_γ varies with each γ and is decided on-line based on the actual data distribution. This enables the KSOM-SC network architecture to capture the topology of both input and output spaces properly as shown in Fig. 7. This figure shows the distribution of KSOM nodes in the Cartesian input space and joint angle space after clustering. As one can see, the nodes lie in the same regions as the data generated from the system. Hence, this approach effectively clusters the input and the output spaces without leading to the formation of outliers. An outlier is a node that does not represent a training data set.

As we know the number of inverse kinematic solutions available are not the same for all points in the workspace. This aspect is also captured by the KSOM-SC network. The distribution of the number of joint angle sub-clusters for different points in the manipulator workspace is shown in Fig. 8. It is seen that the number of solutions decreases towards the boundary of the workspace as expected. A detailed treatment on this architecture is provided in [24] where it is shown through simulations as well as experiments that, the joint angle vectors remain bounded for all points in the workspace.

In this paper, the KSOM-SC network architecture is used as a hint generator for the *inverse-forward* scheme as shown in Fig. 3. Using this architecture it is possible to generate multiple hints for a given target position. One can select a suitable initial joint angle vector from the available hints using different task-specific criteria. Let the index of the winner neuron in the task space for a given desired position be μ. As discussed above, this winner neuron is associated with N_μ sub-clusters in the configuration space. Some of the criteria that can be used for selecting the winning angle sub-cluster are given below:

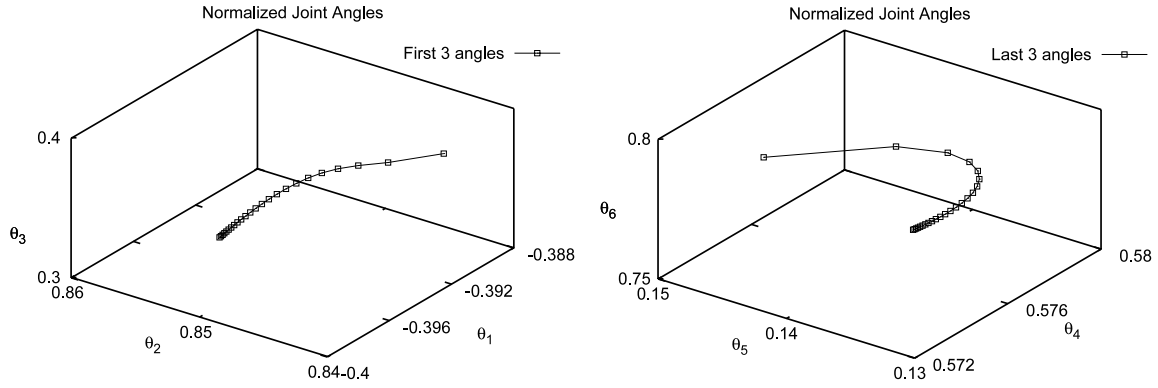
- *Lazy arm movement:* The angle sub-cluster which is closest to the current robot configuration $\boldsymbol{\theta}_c$ is selected as the winner. The winning sub-cluster for this criteria is given by

$$\beta = \arg\min_j \|\boldsymbol{\theta}^j_\mu - \boldsymbol{\theta}_c\|. \tag{5}$$

- *Minimum angle norm:* The angle sub-cluster whose norm is the minimum is selected as the winner. The winning sub-cluster for this criterion is given by

$$\beta = \arg\min_j \|\boldsymbol{\theta}^j_\mu\|. \tag{6}$$

In Section 4, the KSOM-SC architecture is used to avoid an obstacle while reaching a target position using the *inverse-forward* scheme.



(a) First 3 joint angles. (b) Last 3 joint angles.

Fig. 5. The manipulator joint angle trajectory while reaching the target position in the workspace. The joint angle values are normalized between ± 1. It is seen that joint angle values do not exceed their physical limits represented by $+1$ or -1 value.

4. Obstacle avoidance scheme

The collision with an obstacle in the workspace can be avoided by the manipulator if some information about the obstacle can be embedded into the motion planning algorithm. This can be done by using either geometrical models [27,13] for the obstacle and the manipulator or by obtaining information from sensors like vision [28–30] or sonar [3]. Once this information is available, obstacles can be avoided during online operation by using several methods. Among these, artificial potential field based methods [31–34,28] are quite popular where robot motions are under the influence of an attractive potential field pulling the manipulator towards the target and a repulsive potential field pushing the robot away from the obstacles. Some other authors like Zhang et al. [35] avoid obstacles by satisfying certain constraint functions during manipulator motion. Mao et al. [13] use a penalty function approach to avoid obstacles where an augmented cost function including the inequality constraint is minimized along the motion trajectory. Martin et al. [3] use a Reinforcement Learning based collision avoidance scheme that makes use of sonar rings of range sensors to find the distance of obstacles around the robot. On the other hand, Han et al. [26] use multiple KSOMs to learn obstacle-free poses during the training process thereby simplifying the path planning algorithm.

In this section, we demonstrate a simplified obstacle avoidance scheme using the *inverse-forward* scheme explained in Section 2 and the *ball-covering object modeling* method used earlier by Mao et al. [13]. Since the KSOM-SC architecture provides multiple joint angle solutions for a given target point, a suitable initial value of joint angle vector can be obtained by using a criterion that avoids the obstacle. The criterion used for resolving redundancy is based on the ball-covering modeling method. In this method, spheres of different radii are used to cover different shapes of links and objects as shown in Fig. 9. The centers and radii of these spheres are fixed a priori. In this figure, d_i represents the Euclidean distance between the center of the obstacle C_o and the center of each link C_i, $i = 1, 2, 3, 4$. R_i is the radius of the sphere covering the link i and R_o is the radius of the sphere covering the obstacle. While the location of the obstacle is fixed during the robot motion, the location of link centers (C_i, $i = 1, 2, 3, 4$) is a function of the manipulator joint angle vector $\boldsymbol{\theta}$. If we define a scalar function as

$$g(\boldsymbol{\theta}) = \sum_{i=1}^{4} (R_i + R_o - d_i); \quad \{R_i, R_o, d_i\} > 0 \tag{7}$$

then the obstacle avoidance criterion is given by

$$g(\boldsymbol{\theta}) < 0. \tag{8}$$

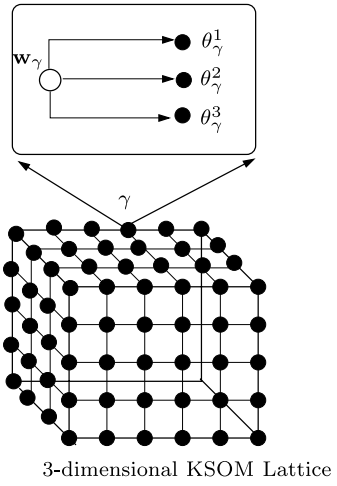


Fig. 6. Sub-clustering in joint-angle space. Each lattice neuron γ is associated with one weight vector $\mathbf{w}_\gamma$ and several joint angle vectors $\boldsymbol{\theta}^\lambda_\gamma$, $\lambda = 1, 2, \ldots, N_\gamma$. Here $N_\gamma = 3$.

Among the solutions provided by the KSOM-SC architecture, all those solutions which do not satisfy this constraint are rejected and among the selected configurations the one which is nearest to the current robot configuration is selected as the initial value of joint angle vector for the network inversion scheme.

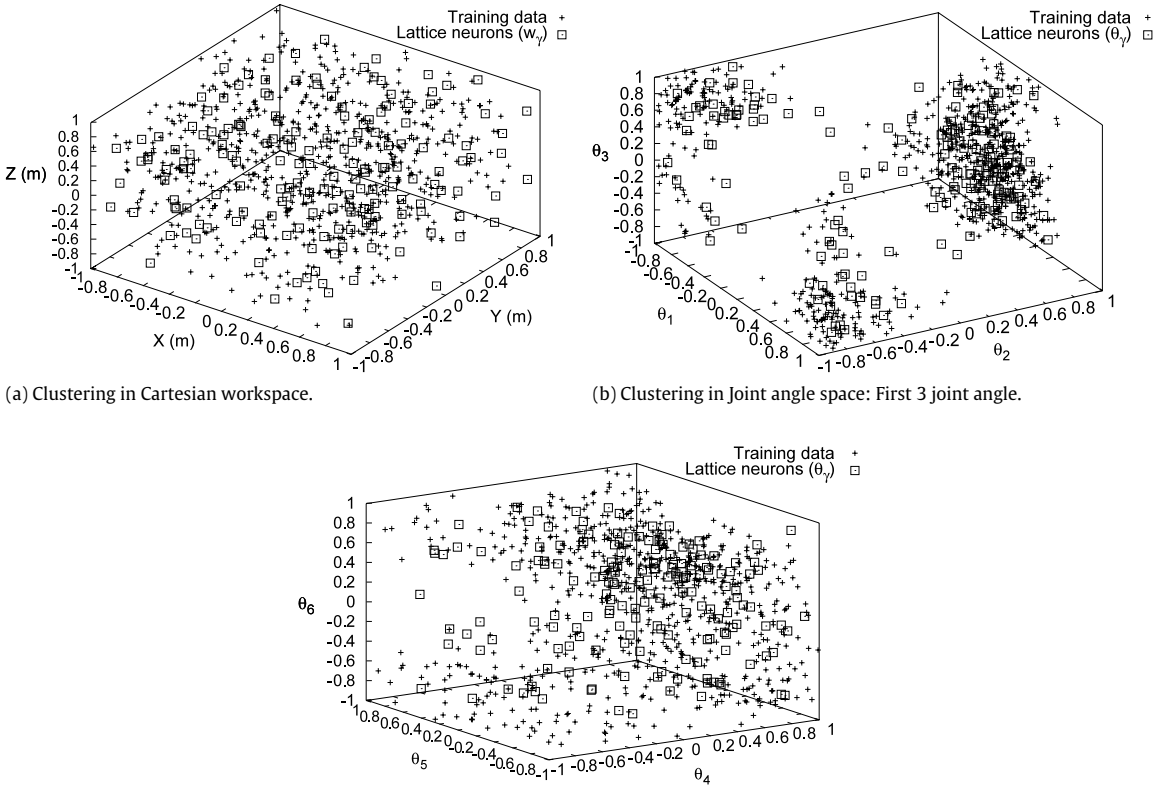
The method proposed in this section is simpler as compared to the currently available methods because once the clustering has been carried out, the obstacle avoidance algorithm involves selecting the suitable joint angle vector using criterion (8). This proposed method does not involve a complicated training process that optimizes certain cost functions.

5. Simulation and experimental results

5.1. Manipulator model

The forward kinematic model of a 7 DOF PowerCube™ [36] manipulator is used for simulation studies. The forward kinematic model of the manipulator is derived using its D-H parameters [37]. The end-effector position is given by following equations:

$$\begin{aligned} x = {} & (-(((c_1c_2c_3 - s_1s_3)c_4 - c_1s_2s_4)c_5 + (-c_1c_2s_3 - s_1c_3)s_5)s_6 \\ & + (-(c_1c_2c_3 - s_1s_3)s_4 - c_1s_2c_4)c_6)d_7 \\ & + (-(c_1c_2c_3 - s_1s_3)s_4 - c_1s_2c_4)d_5 - c_1s_2d_3 \\ y = {} & (-(((s_1c_2c_3 + c_1s_3)c_4 - s_1s_2s_4)c_5 + (-s_1c_2s_3 + c_1c_3)s_5)s_6 \end{aligned}$$



(a) Clustering in Cartesian workspace.

(b) Clustering in Joint angle space: First 3 joint angle.

(c) Clustering in Joint angle space: Last 3 joint angle.

Fig. 7. Discretization of Cartesian workspace and configuration space using KSOM-SC network architecture.

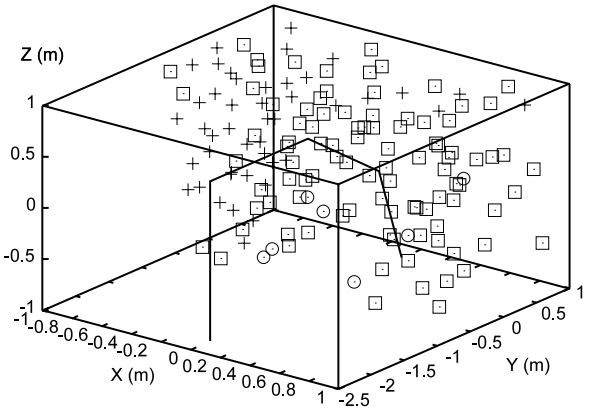


Fig. 8. Distribution of joint angle sub-clusters in the 3-dimensional manipulator workspace. The number of solutions available for a given target point varies across the workspace. There are very few points where the number of available solutions is too high. The number of solutions decreases towards the boundary of the workspace. In this figure, a plus sign ('+') represents a point where the number of sub-clusters $N_\gamma < 18$, squares represent the points with N_γ is between 19 to 30 and circles represent the points with $N_\gamma > 30$. A typical robot configuration is shown with a solid line.

$$
\begin{aligned}
&+(-(s_1c_2c_3+c_1s_3)s_4-s_1s_2c_4)c_6)d_7\\
&+(-(s_1c_2c_3+c_1s_3)s_4-s_1s_2c_4)d_5-s_1s_2d_3\\
z\;=\;&(-((s_2c_3c_4+c_2s_4)c_5-s_2s_3s_5)s_6+(-s_2c_3s_4+c_2c_4)c_6)d_7\\
&+(-s_2c_3s_4+c_2c_4)d_5+c_2d_3+d_1 \qquad (9)
\end{aligned}
$$

where various parameters are: $d_1 = 0.390$ m, $d_3 = 0.370$ m, $d_5 = 0.310$ m, $d_7 = 0.2656$ m, $c_i = \cos\theta_i$, $s_i = \sin\theta_i$, $i = 1, 2, \ldots, 6$. Since the end-effector position $[x\ y\ z]^T$ does not depend on the seventh joint angle, a zero value is assigned to this angle during on-line operation.

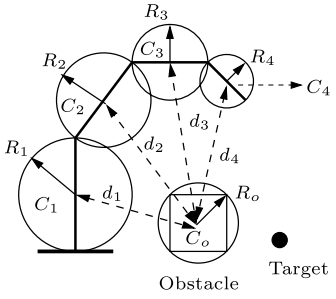


Fig. 9. Obstacle avoidance criterion is obtained by using a ball-covering object modeling scheme. The links and objects are covered with balls of known radii.

5.2. Radial Basis Function network

A 6-700-3 architecture is selected for the RBF network. Training data pairs $(\boldsymbol{\theta}_t, \mathbf{x}_t)$ are obtained by generating random angle vectors $\boldsymbol{\theta}_t$ within their physical limits and obtaining the corresponding end-effector positions $\mathbf{x}_t$ using manipulator forward kinematic equations (8). Only those data points are used for training for which the end-effector position lies within the workspace limits. The physical limits for joint angle vectors and Cartesian end-effector positions are specified in Table 1.

It is seen in Fig. 2 that when a RBF network with 700 hidden neurons is trained over 7×10^5 data pairs, an approximation error of 4 cm is achieved. This accuracy is not good for accurate positioning tasks needed in industrial applications like welding or fitting. This shows the difficulty involved in obtaining high positioning accuracy by forward approximation alone. Since the training processing is a time-consuming process, it would be good to reduce the number of patterns needed for training as much as possible.

Table 1
Ranges of input and output spaces.

Ranges of joint angle	Range of Cartesian workspace
$-160^\circ \le \theta_1 \le 160^\circ$	$-0.3\ \text{m} \le x \le 0.3\ \text{m}$
$-95^\circ \le \theta_2 \le 95^\circ$	$0.3\ \text{m} \le y \le 0.8\ \text{m}$
$-160^\circ \le \theta_3 \le 160^\circ$	$0.0\ \text{m} \le z \le 0.5\ \text{m}$
$-50^\circ \le \theta_4 \le 120^\circ$	
$-90^\circ \le \theta_5 \le 90^\circ$	
$-120^\circ \le \theta_6 \le 120^\circ$	
$-360^\circ \le \theta_7 \le 360^\circ$	

5.3. Kohonen Self-Organizing map architecture for sub-clustering

A $7 \times 7 \times 7$ lattice structure is selected for the KSOM network. Nearly 50,000 input–output data pairs $(\boldsymbol{\theta}_t, \mathbf{x}_t)$ are used for learning the clusters in the Cartesian task space and the joint angle space. Each cluster in the Cartesian workspace is associated with multiple clusters in the joint angle space.

The efficacy of the proposed scheme is demonstrated by carrying out the following tasks:

5.4. Reaching isolated points in the workspace

A target position $\mathbf{x}_d$ is selected randomly from the manipulator workspace and is presented to the KSOM network as shown in Fig. 3. The winning lattice neuron is obtained using minimum distance criterion as follows:

$$\mu = \arg\min_{\gamma} \|\mathbf{w}_\gamma - \mathbf{x}_d\| \tag{10}$$

where γ is the lattice index of each neuron. As explained in Section 3, this winning neuron is associated with N_μ joint angle clusters $\boldsymbol{\theta}^j_\mu$, $j = 1, 2, \ldots, N_\mu$. The initial value for the joint angle vector is obtained by using the lazy-arm criterion given by Eq. (5). Now this joint angle vector is updated using the *inverse-forward* scheme as explained in Section 2. The update equation for joint angle vector is given by (3) and (4). The multi-step movement for attaining an accuracy of 1 mm is shown in Fig. 4. Initially, the position obtained from the RBF network does not match with the actual manipulator end-effector position. This error is due to the approximation error present in the forward training. If the network is not trained again during the inversion process, the error persists and the actual manipulator never attains the desired position even when the RBF network output reaches the desired position as shown in Fig. 2. The manipulator joint angle values are shown in Fig. 5. It is seen that the joint angle values do not violate the physical limits during the robot's motion. In order to ascertain our observations, 1000 points are selected randomly from the workspace and the *inverse-forward* learning scheme is used for the angle vector update. The number of steps needed to attain an accuracy of 1 mm is shown in Fig. 10.

The effect of forward training on inverse kinematic solution is shown in Table 2. The forward network is trained with a different number of training examples as shown in the first column. The average positioning error obtained during training is shown in the second column. The third column shows the average number of steps needed to obtain 1 mm accuracy during the testing phase where the network is presented with 1000 random points. Those points for which 1 mm accuracy is not attained within 400 steps are separately counted and are expressed in a percentage of total number of test points in the fourth column. The average positioning error for these non-converging points is shown in the last column. This table shows that the number of steps needed to attain 1 mm accuracy is almost independent of the number of examples used for training the forward network. It is also seen that it is possible to attain 1 mm accuracy for 95% of random points in the manipulator workspace and it increases to 99.7% when the amount of forward training is increased from 10^4 to 7×10^5 examples. Moreover the positioning error of these non-converging points decreases steadily with an increasing amount of forward training. This is expected because, the approximation error in forward training decreases with increasing forward training. Hence it is possible to obtain good positioning accuracy for any point in the manipulator workspace. This shows that the forward training is certainly needed for better performance of the proposed scheme. However it is not necessary to obtain very high positioning accuracy during forward training. In this aspect, the proposed scheme removes the drawback of the conventional network inversion based schemes thereby enhancing its applicability to precision tasks.

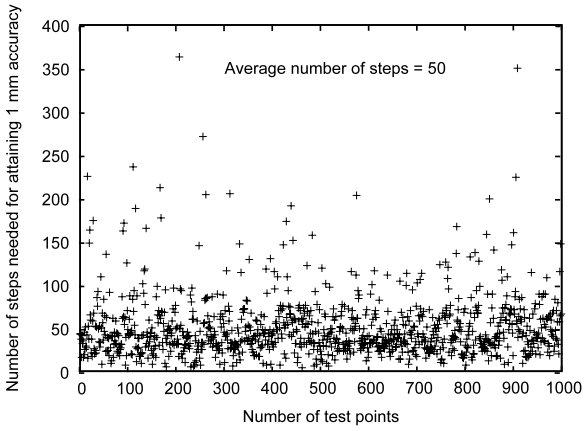


Fig. 10. Multi-step movement: Number of steps needed to attain 1 mm accuracy for 1000 test points using inversion algorithm.

Even though only one criterion called "lazy arm motion" is used to obtain results shown in Table 2, similar observations can be made for other initial values obtained from the KSOM-SC network. A set of valid joint angle configurations obtained from a KSOM-SC network, for a given target end-effector position is shown in Fig. 11. Any one of these configurations could be selected as an initial condition for the inverse-forward scheme.

5.4.1. Discussion

In Table 2, it is seen that the network does not converge for some points in the workspace and it might seem as a drawback of the local learning method. In the following discussion, we would argue that such a limitation does not arise in this case.

If the forward map is trained properly then it is possible to find inverse kinematic solutions for all points within a reasonable amount of time. Due to improper forward training, some points take a greater number of steps to converge than others. If sufficient time (number of steps) is allowed, it is possible to get the desired accuracy for all points within the workspace of the manipulator.

It should be noted that in Table 2, any point which fails to converge within 400 steps is considered to be a non-converging point. If we allow the *inverse-forward* scheme to take more steps then it might converge. For instance, for the last row of the table, we found that the number of non-converging points reduces to zero if the inverse-forward scheme is allowed to take 1000 steps.

The number of non-converging points also depends on the number of nodes used for the KSOM-SC lattice because it provides the initial guess for the inverse kinematic solution. If this initial guess is close to the desired position, a smaller number of steps is necessary for obtaining the solution. If this guess is quite far, the number of steps that is necessary to obtain the solution is large and the scheme may fail to converge within a reasonable amount of time.

Table 2
Effect of forward training on inverse kinematic solution. A: Percentage of target points for which 1 mm positioning accuracy is not achieved by the inverse-forward scheme in 400 steps. B: Average positioning error for the non-converging points listed in column 'A'.

Number of examples used for initial forward training	Avg. positioning error after initial forward training (cm)	Avg. no. of steps to attain 1 mm accuracy using inverse-forward schemes	A	B
1×10^4	13.9	57	5.4	21 cm
2×10^4	11.5	51	2.4	20 cm
5×10^4	9.1	50	1.4	15.9 cm
1×10^5	7.4	50	1.2	4.3 cm
2×10^5	6.1	52	0.7	2.5 cm
5×10^5	4.5	48	0.5	1.3 cm
7×10^5	4.2	51	0.3	2 mm

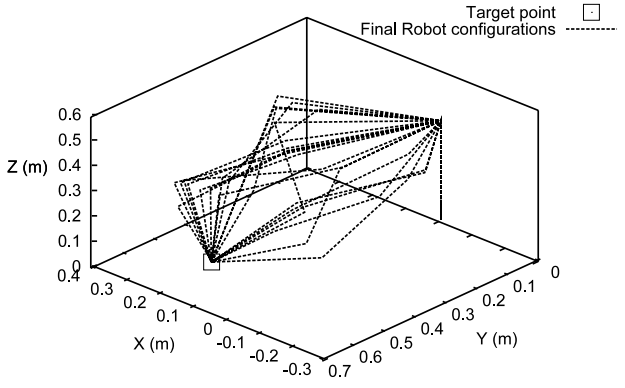


Fig. 11. Inverse kinematic solutions corresponding to all initial conditions obtained from KSOM-SC network. This shows that it is possible to reach the desired end-effector position from all initial conditions with same accuracy. This picture shows the final robot configurations obtained from inverse-forward scheme when initialized with 20 initial conditions provided by the KSOM-SC network. Here the network is trained with 7×10^5 data points and the inverse-forward cycles take nearly 40 steps on an average to attain an accuracy of 1 mm.

5.5. Tracking continuous trajectory

The proposed scheme has also been used to track the following trajectory:

$$x = 0.2 \sin \phi$$
$$y = 0.5 + 0.2 \cos(\phi)$$
$$z = \frac{1}{3}(x + 0.3) \tag{11}$$

where ϕ is varied from 0 to 2π radians with a step size of 0.01. A total of 628 data points is generated over the trajectory as per the above equation in a sequential manner and the *inverse-forward* update scheme is used to compute the required joint angle vectors. The initial configuration for tracking this trajectory is obtained from the KSOM-SC network. This initial configuration is crucial as it determines the pose of the manipulator throughout the robot's movement. The tracking result in Cartesian space is shown in Fig. 13(a). The joint angles are found to be continuous as shown in Fig. 13(b) and the angles lie within their physical limits. The robot's configuration during the motion is shown in Fig. 14. The number of steps needed to obtain 1 mm accuracy at each point is shown in Fig. 15. These results demonstrate that the method proposed in this paper preserves the conservative property [38] of the inverse kinematic solution. In other words, a continuous trajectory in task space gives rise to a continuous trajectory in the joint angle space.

5.5.1. Discussion

In this section, we answer the following question:

Does local optimization that takes place in the inverse-forward scheme lead to unlearning of already learned locations?

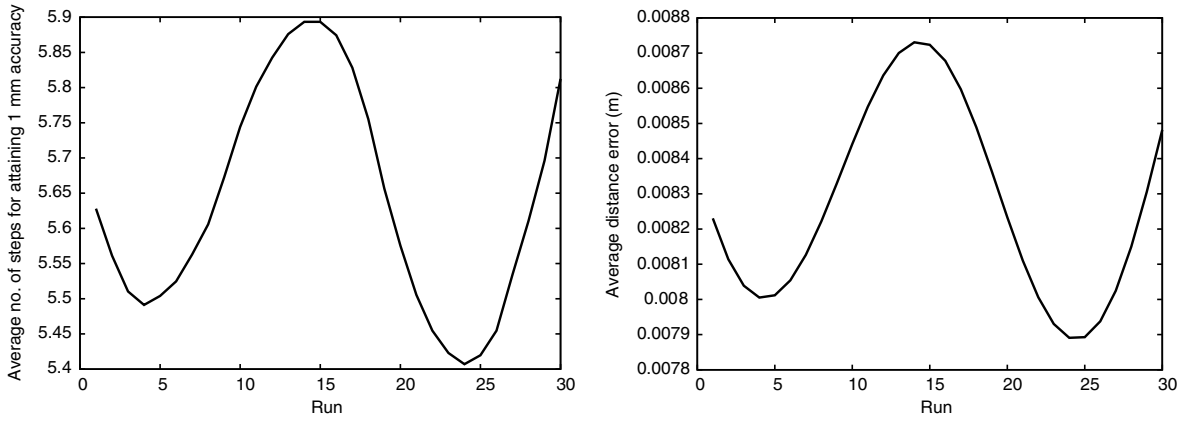
The learning rate for the inversion algorithm (4) is selected in the range of 0.01–0.02 and the learning rate for the back-propagation algorithm used for the forward training is selected to be 0.1. During a local search, the learning and the un-learning of the forward map take place at a very small scale which does not affect the global learning of the forward map. In order to ascertain this fact, the above elliptical trajectory is tracked multiple times using the *inverse-forward* scheme. The outcome is shown in Fig. 12. Fig. 12(a) shows the average number of steps taken for reaching each point on the elliptical trajectory during each run with an accuracy of 1 mm. Each run corresponds to one complete traversal of the elliptical trajectory. It is seen that the average number of steps varies between 5.4 and 5.9 over different runs. In other words, the average number of steps remains almost the same for all runs. *This shows that the local search does not lead to unlearning of the forward map*. To confirm this observation, we performed another simulation where the *inverse-forward* scheme is used to reach each point in only one step and the entire process is repeated over 30 different runs. The result is shown in Fig. 12(b). Once again we see that the average tracking error varies approximately from 8 mm to 9 mm. In other words, the average error remains almost the same over different runs when the trajectory is tracked multiple times.

5.6. Avoiding obstacle

The manipulator base is taken as the origin of the Cartesian reference frame. An obstacle of dimensions 20 cm × 10 cm × 20 cm is placed in the manipulator workspace with its center lying at (0 cm, 35 cm, 10 cm). The manipulator has to reach a target position specified by the coordinates $\mathbf{x}_d = (10\text{ cm}, 50\text{ cm}, 0\text{ cm})$ without colliding with the obstacle. The initial manipulator configuration, the locations of the obstacle and the target are shown in Fig. 16(a). This figure also shows all configurations made available by the KSOM-SC architecture for the given target point. It can be seen that many configurations would collide with the obstacle. All those configurations which do not satisfy the obstacle avoidance criterion given by Eq. (8) are rejected. Among the remaining configurations which avoid the obstacle, the one that is nearest to the current robot's configuration is selected as the initial guess. This ensures that the manipulator will not collide with the obstacle while moving from its current configuration to the configuration provided by the initial guess. Now the *inverse-forward* scheme is used to update the joint angle vector $\boldsymbol{\theta}$ from its initial value $\boldsymbol{\theta}_0$. The complete motion of joint links to reach the target point is shown in Fig. 16.

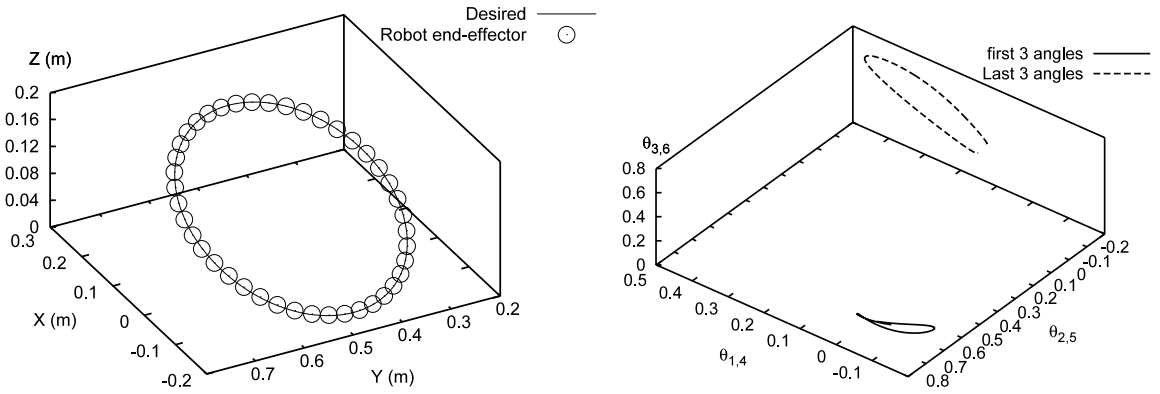
5.6.1. Discussion

Note that choosing the lazy arm criterion to find the initial robot's configuration may not lead to collision-free trajectory in all cases. However, the intention here is to demonstrate



(a) Multi step solution. (b) Single step solution.

Fig. 12. Tracking trajectory multiple times. The learning and unlearning takes place only at a local level.



(a) Tracking in Cartesian space. (b) Joint angle trajectories necessary for tracking.

Fig. 13. Tracking a circular trajectory. The average accuracy attained is 1 mm. Joint angles are found to lie within the physical limits. Angles are normalized and the physical limits are given by ± 1.

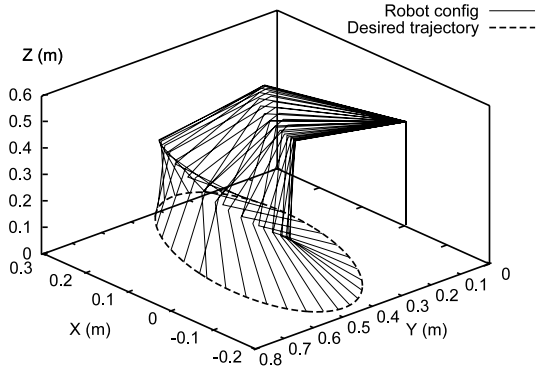


Fig. 14. Robot's configuration during trajectory tracking. Choosing the initial configuration is crucial as it determines the pose of the manipulator throughout the robot's motion.

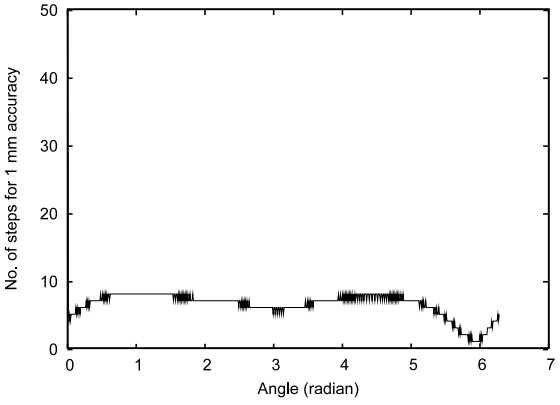
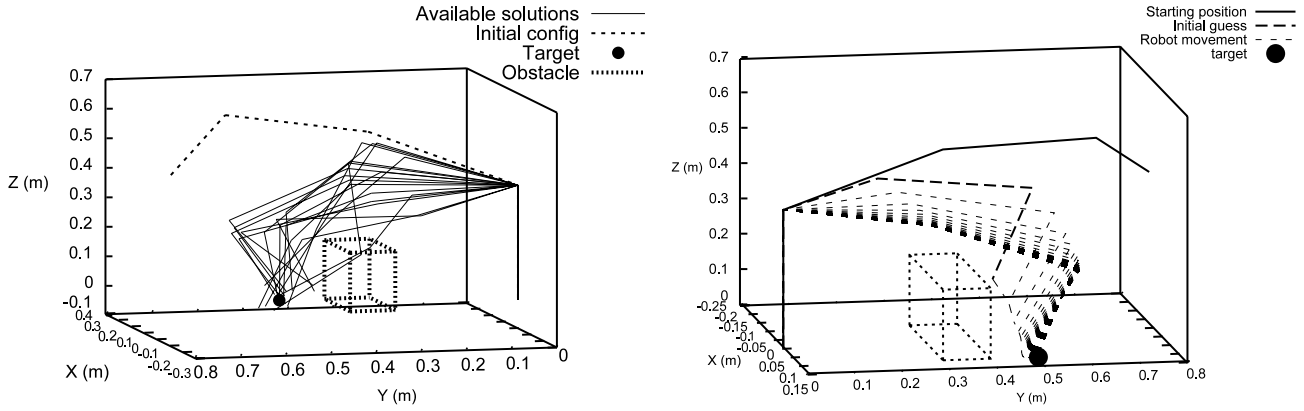


Fig. 15. The number of steps taken by inverse-forward cycle to obtain 1 mm accuracy at each point on the trajectory.

the applicability of our approach to a practical problem. If the input–output space is effectively clustered using a KSOM-SC network, then it would be possible to obtain an initial guess which is fairly close to the desired position. Note that, we are talking about an accuracy of 1 mm to be obtained through our scheme. It is possible to get a positioning accuracy of about 1 cm by using the KSOM-SC architecture as shown in [22,24]. Hence in moving from 1 cm to 1 mm accuracy, it can be safely assumed that one can obtain a collision-free motion in a workspace whose overall dimension is in meters. However, the readers should note that this may not hold in a case when the distance to be traversed is large.

5.7. Details of the actual experiment

This experiment was performed at the ISRC, University of Ulster, Magee Campus in the United Kingdom. The objective of this



(a) Available configurations.

(b) Resolving redundancy to avoid obstacle.

Fig. 16. Avoiding the obstacle while reaching a target point. The first figure shows the available configurations for a given target point obtained from KSOM-SC architecture. In the second figure, the redundancy is resolved by choosing the configuration which is not only closest to the initial configuration but also avoids the obstacle. This is labeled as the 'initial guess'. Then inverse-forward scheme is employed to attain 1 mm accuracy. Different views are provided in figures (a) and (b) to ensure visibility of the obstacle, the target point as well as the robot's end-effector trajectory.

experiment was to reach various points around an obstacle without colliding with it. A small cuboidal box of dimensions 21 cm × 16 cm × 6 cm was taken as the primary obstacle to be avoided. This box was placed over a wooden pedestal of dimensions 70 cm × 50 cm × 80 cm. The primary obstacle was modeled as a sphere of radius 14 cm and the pedestal was modeled as a sphere of radius 65 cm. In order to avoid configurations that would reach the point from the bottom, thereby colliding with the pedestal, the pedestal was also treated as an obstacle. The lazy arm criterion and obstacle avoidance criterion (7) and (8) were used for resolving redundancy while providing the initial hint to the *inverse-forward* algorithm. The forward kinematic model was used to find the current end-effector position. The entire training was carried out in the Cartesian workspace. The modeling of obstacles and final configurations for various points obtained from simulation are shown in Fig. 17. In this figure, the red sphere represents the pedestal and the cyan sphere represents the primary obstacle. The blue squares represent the target points to be reached and the pink lines show the final robot's configurations obtained through inverse-forward scheme. Some of the configurations of the manipulator obtained during the actual experiment are shown in Fig. 18. One has to select the radius of spheres judiciously in order to obtain the right solutions. For instance if the radius is chosen too large, then one may lose all solutions provide by the KSOM-SC hint generator. If the radius is selected too small, then the solution might hit the obstacle. Since accurate models were used for the manipulator as well as the obstacles, the experimental observations were found to concur with the simulation results.

The PowerCube manipulator takes approximately 80 milliseconds to complete one positioning command. Hence, the multi-step inverse-forward scheme is implemented using the forward kinematic model and the final joint angle configuration is provided to the robot manipulator.

6. Conclusion

An online inverse-forward adaptive scheme with a KSOM based hint generator is used to solve the inverse kinematic problem of a redundant manipulator. In this approach, a RBF network is trained to approximate the forward kinematic relationship and this network is inverted to obtain a joint angle vector for the desired target point. It is a well known fact that the

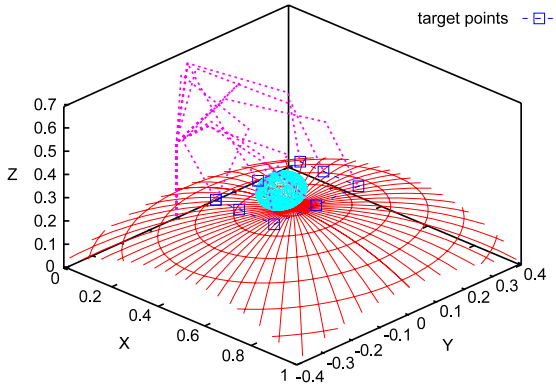


Fig. 17. The base over which obstacle is place is also modeled as an obstacle and the manipulator tries to avoid both of these two spheres while reaching various points around the obstacle.

positioning accuracy obtained by network inversion is limited by the approximation error present in the forward model. Usually a large amount of training data is needed for obtaining a small positioning error during forward training. The difficulty present in forward training can be gauged from the fact that a RBF network with 700 hidden nodes when trained with 7×10^5 data patterns, give rise to a positioning accuracy of only 4 cm. This accuracy is not adequate for most industrial applications. It is shown that by using an *inverse-forward* adaptive scheme, it is possible to obtain high positioning accuracy by network inversion even when the forward model has a large approximation error. This scheme searches for a local solution around the current joint angle configuration that minimizes the output error. The inverse-forward update is carried out online and it is possible to attain 1 mm accuracy for all points in the manipulator workspace.

Another drawback of the network inversion based algorithm is that it provides a feasible joint angle vector (that satisfies physical limits) only when, it is initialized with a suitable joint angle vector. Moreover, since a redundant manipulator can have multiple joint angle vectors for the same end-effector position, one needs to resolve redundancy as well. This initial condition determines the pose of the manipulator during the robot's motion. These two problems are solved by obtaining a initial guess from a KSOM-SC network architecture. This network provides multiple hints for a

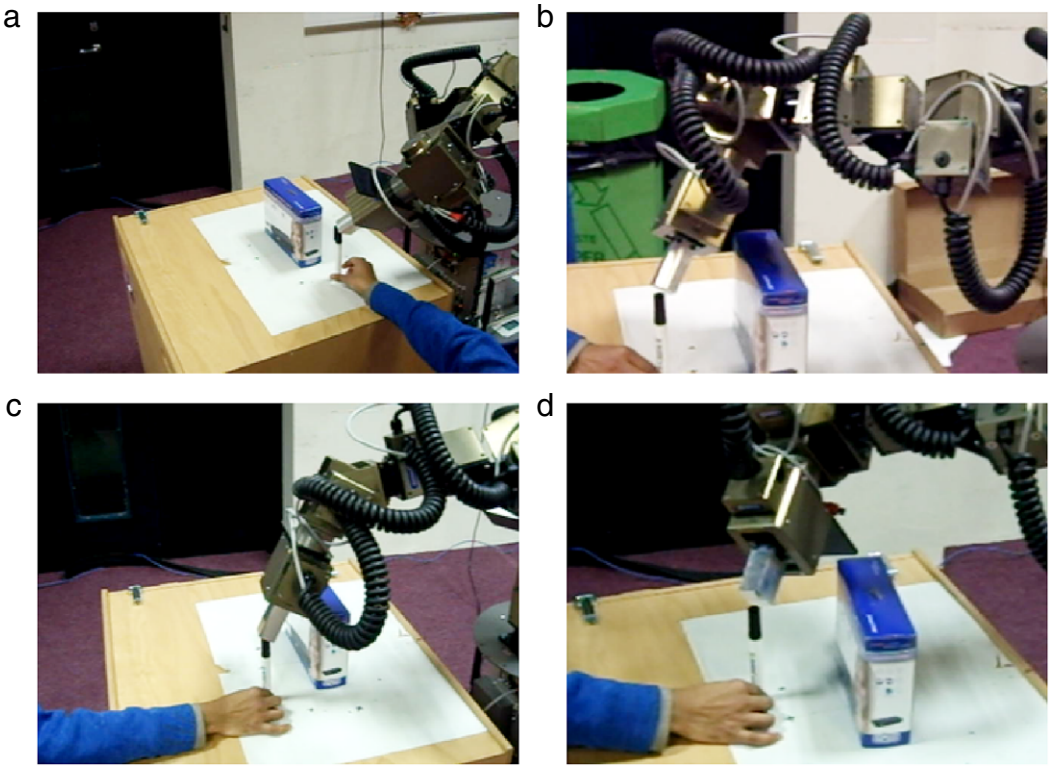


Fig. 18. Obstacle avoidance experiment. The manipulator reaches various points on a closed trajectory without colliding with the obstacle.

given target end-effector position and each hint leads to a valid inverse kinematic solution. Redundancy resolution is carried out by selecting one out of several available configurations by using a task-specific criterion. An obstacle avoidance problem is solved to demonstrate the redundancy resolution process. The theoretical findings made in this paper are corroborated through experiments on an actual 7 DOF PowerCube™ manipulator.

Acknowledgement

This work was supported by Intelligent Systems Research Center (ISRC), University of Ulster, UK.

References

[1] L. Sciavicco, B. Siciliano, Modeling and Control of Robot Manipulators, McGraw-Hill, New York, 1996.
[2] T. Yoshikawa, Foundation of Robotics, Analysis and Control, Prentice Hall of India, New Delhi, 2001.
[3] P. Martin, J.R. Millan, Robot arm reaching through neural inversions and reinforcement learning, Robotics and Autonomous Systems 31 (2000) 227–246.
[4] M. Kuperstein, Neural model of adaptive hand–eye coordination for single postures, Science 239 (1988) 1308–1311.
[5] W.T. Miller III, Sensor-based control of robotic manipulators using a general learning algorithm, IEEE Journal of Robotics and Automation RA-3 (2) (1987) 157–165.
[6] G. Sun, B. Scassellati, A fast and efficient model for learning to reach, International Journal of Humanoid Robotics 2 (4) (2005) 391–413.
[7] R.V. Mayorga, P. Sanongboon, A radial basis function network approach for inverse kinematics and singularities prevention of redundant manipulators, in: Proc. of Int. Conf. on Robotics and Automation, ICRA, IEEE, 2002, pp. 1955–1960.
[8] S. Vijayakumar, A. D'Souza, T. Shibata, J. Conradt, S. Schaal, Statistical learning for humanoid robots, Autonomous Robots 12 (2002) 55–69.
[9] A. D'Souza, S. Vijayakumar, S. Schaal, Learning inverse kinematics, in: International Conference on Intelligent Robots and Systems, IROS, IEEE, Maui, Hawai, USA, 2001, pp. 298–303.
[10] G.G. Lendaris, K. Mathia, R. Saeks, Linear hopfield networks and constrained optimization, IEEE Transactions on System, Man and Cybernetics—Part B: Cybernetics 29 (1) (1999) 114–118.
[11] T.M. Martinetz, H.J. Ritter, K.J. Schulten, Three-dimensional neural net for learning visual motor coordination of a robot arm, IEEE Transactions on Neural Networks 1 (1) (1990) 131–136.
[12] J.A. Walter, K.J. Schulten, Implementation of self-organizing neural networks for visual-motor control of an industrial robot, IEEE Transactions on Neural Networks 4 (1) (1993) 86–95.
[13] Z. Mao, T.C. Hsia, Obstacle avoidance inverse kinematics solution of redundant robots by neural networks, Robotica 15 (1997) 3–10.
[14] D. DeMers, K. Kreutz-Delgado, Canonical parameterization of excess motor degrees of freedom with self organizing maps, IEEE Transactions on Neural Networks 7 (1) (1996) 43–55.
[15] J. Peters, S. Schaal, Learning to control in operational space, International Journal of Robotics Research 27 (2) (2008) 197–212.
[16] M.I. Jordan, D.E. Rumelhart, Forward models: Supervised learning with a distal teacher, Cognitive Science 16 (1992) 307–354.
[17] J. Wang, Q. Hu, D. Jiang, A lagrangian network for kinematic control of redundant robot manipulators, IEEE Transactions on Neural Networks 10 (5) (1999) 1123–1132.
[18] W.S. Tang, J. Wang, A recurrent neural network for minimum infinity-norm kinematic control of redundant manipulators with an improved problem formulation and reduced architecture complexity, IEEE Transactions on Systems, Man and Cybernetics—Part B: Cybernetics 31 (1) (2001) 98–105.
[19] J. Peters, D. Nguyen-Tuong, Real-time learning of resolved velocity control on a Mitsubishi PA-10, in: International Conference on Robotics and Automation, ICRA, IEEE, Prasadena, CA, USA, 2008, pp. 2872–2877.
[20] L. Bao-Liang, K. Ito, Regularization of inverse kinematics for redundant manipulator using neural network inversions, in: International Conference on Neural Networks, ICNN, IEEE, Perth, Australia, 1995, pp. 2726–2731.
[21] S.F.M. Assal, K. Watanabe, K. Izumi, Neural network-based kinematic inversion of industrial redundant robots using cooperative fuzzy hint for the joint limits avoidance, IEEE Transactions on Mechatronics 11 (5) (2006) 593–603.
[22] S. Kumar, N. Patel, L. Behera, Visual motor control of a 7 dof robot manipulator using function decomposition and sub-clustering in configuration, Neural Processing Letters 28 (1) (2008) 17–33.
[23] T. Kohonen, Self Organization and Associative Memory, Springer-Verlag, 1984.
[24] S. Kumar, P. Premkumar, A. Dutta, L. Behera, Visual motor control of a 7 DOF robot manipulator using KSOM-based redundancy preserving network, Robotica (2009).

[25] H. Zha, T. Onitsuka, T. Nagata, A self-organization learning algorithm for visuo-motor coordination in unstructured environment, Artificial Life and Robotics 1 (3) (1997) 131–136.
[26] M. Han, N. Okada, E. Kondo, Collision avoidance for a visuo-motor system using multiple self-organizing maps, in: Memoirs of the Faculty of Engineering, Kyushu University, 65 (4), 2005 pp. 129–142.
[27] T.C. Hsia, Z.Y. Guo, New inverse kinematic algorithms for redundant robots, Journal of Robotics Systems 8 (1) (1991) 117–132.
[28] Y. Mezouar, F. Chaumette, Path planning in image space for robust visual servoing, in: Proc. of IEEE Int. Conf. on Robotics and Automation, San Francisco, CA, 2000, pp. 2759–2763.
[29] M. Han, N. Okada, E. Kondo, Coordination of an uncalibrated 3-d visuo-motor system based on multiple self-organizing maps, JSME International Journal Series C 49 (1) (2006) 230–239.
[30] N. Mansard, F. Chaumette, Visual servoing sequencing able to avoid obstacles, in: Proc. of Int. Conf. on Robotics and Automation, IEEE, Barcelona, Spain, 2005, pp. 3143–3148.
[31] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, International Journal of Robotics Research 5 (1) (1986) 90–98.
[32] Y.K. Hwang, N. Ahuja, A potential field approach to path planning, IEEE Transactions on Robotics and Automation 8 (1) (1992) 23–32.
[33] J. Kim, P.K. Khosla, Real-time obstacle avoidance using harmonic potential functions, IEEE Transactions on Robotics and Automation 8 (3) (1992) 338–349.
[34] W. Cho, D. Kwon, A sensor-based obstacle avoidance for a redundant manipulator using a velocity potential field, in: IEEE International Workshop on Robots and Human Communication, Tsukuba, Japan, 1996, pp. 306–310.
[35] Y. Zhang, J. Wang, Obstacle avoidance for kinematically redundant manipulators using a dual neural network, IEEE Transactions on System, Man and Cybernetics, Part B 34 (1) (2004) 752–759.
[36] Powercube manipulators, SCHUNK GmbH & Co. KG, 2009. http://www.schunk.com/.
[37] J.J. Craig, Introduction to Robotics, Pearson Education, Inc., 1989.
[38] G. Tevatia, S. Schaal, Inverse kinematics of humanoid robots, in: Proc. of IEEE Int. Conf. on Robotics and Automation, San Francisco, CA, 2000, pp. 294–299.

Swagat Kumar received his B.E. degree in Electrical Engineering from Orissa School of Mining Engineering Keonjhar, Orissa in 2001. He obtained his M.Tech. and Ph.D. degrees from the Department of Electrical Engineering, IIT Kanpur in the year 2004 and 2009 respectively. Currently, he is a post-doctoral researcher at the Department of Informatics of Kyushu University (Ito Campus), Fukuoka, Japan. His research interest includes, Control Systems, Neural Networks and Robotics.

Laxmidhar Behera received his doctral degree from Indian Institute of Technology Delhi, India in 1996. He is working as Associate professor at Indian Institute Of Technology Kanpur, INDIA. Prior to this, he was a researcher at GMD Germany, an assistant professor at BITS, Pilani India and an honourary faculty at Bhaktivedanta Institute, Mumbai India. Currently he is a reader in cognitive robotics at Intelligent Systems Research Centre, University of Ulster, Magee Campus, UK. He has published more than 120 papers in referred journals and conference proceedings. He is a recipient of All India Council of Technical Education (AICTE) career Award for young teacher in 1997. He is also a senior member in IEEE and a reviewer for several journals. His areas of research include Intelligent control, Evolutionary games, Spatial Stochastic dynamics, Soft computing, visual servoing and robotics.

T.M. McGinnity has been a member of the UU academic staff since 1992 and holds the post of Professor of Intelligent Systems Engineering on the Magee campus. He has a first class honours degree in physics and received his doctorate from the University of Durham. He is a Fellow of the IEE, member of the IEEE, and a Chartered Engineer. He has 26 years experience in teaching and research in electronic and computer engineering, leads the research activities of the Intelligent Systems Engineering Laboratory at the Magee campus of the University, and is Acting Associate Dean of the Faculty of Engineering, with responsibility for research and technology transfer. His current research interests relate to the creation of intelligent computational systems and the area of intelligent systems in general, particularly in relation to hardware and software implementations of neural networks, fuzzy systems, genetic algorithms and bio-inspired intelligent systems. His work is also focused on the topic of implementing intelligence in embedded systems. Following his appointment to UU in 1992, he founded the Intelligent Systems Research Group and has been a key participant in managing its rapid growth to its current position as a major research unit within the Faculty of Engineering. Under his direction, the ISEL laboratory has grown to its current active research membership of twelve academic staff, eight research associates/assistants, fourteen full-time Ph.D. students and three part-time Ph.D. students. TMM has attracted local, national and international funding for his research, with a total research income of over 5 million in the last 5 years. He is the principal grant holder on five current major research projects, SenseMaker (an EU project for which UU is the coordinating partner), InterWave (a north-south project with NMRC Cork), a SRIF2 project, a Department of Enterprise Trade and Investment funded Broadband Flagship project (Walled City to Wireless City) and an Interreg funded EpiCentre project. In addition he is the principal grant holder on an InvestNI Proof of Concept project which commenced in September 2004. An EPSRC project DIESEL (GR/N26753/03), and the EU project QUDOS have recently completed. Previously he was a grant-holder on the Prominent and STEPCAM (EP 23177) EU projects. His national and international collaborations on research projects include the University of Cambridge (QUDOS), University of Heidelberg (SenseMaker), Trinity College Dublin (SenseMaker), Max Planck Institute (QUDOS), CNRS Gif, University of Bordeaux, SINTEF Electronics and Cybernetics, Norway (ProMinent), among many others.