Robotica (2010) volume 28, pp. 795–810. © Cambridge University Press 2009
doi:10.1017/S026357470999049X

Visual motor control of a 7DOF redundant manipulator using redundancy preserving learning network

Swagat Kumar†, Premkumar P.†, Ashish Dutta† and Laxmidhar Behera†,‡,*

† *Department of Electrical Engineering, Indian Institute of Technology Kanpur, Kanpur, India*
‡ *School of Computing and Intelligent systems, University of Ulster, Magee, Northern Ireland, UK*

(Received in Final Form: August 18, 2009. First published online: September 21, 2009)

SUMMARY

This paper deals with the design and implementation of a visual kinematic control scheme for a redundant manipulator. The inverse kinematic map for a redundant manipulator is a one-to-many relation problem; i.e. for each Cartesian position, multiple joint angle vectors are associated. When this inverse kinematic relation is learnt using existing learning schemes, a single inverse kinematic solution is achieved, although the manipulator is redundant. Thus a new redundancy preserving network based on the self-organizing map (SOM) has been proposed to learn the one-to-many relation using sub-clustering in joint angle space. The SOM network resolves redundancy using three criteria, namely lazy arm movement, minimum angle norm and minimum condition number of image Jacobian matrix. The proposed scheme is able to guide the manipulator end-effector towards the desired target within 1-mm positioning accuracy without exceeding physical joint angle limits. A new concept of neighbourhood has been introduced to enable the manipulator to follow any continuous trajectory. The proposed scheme has been implemented on a seven-degree-of-freedom (7DOF) PowerCube robot manipulator successfully with visual position feedback only. The positioning accuracy of the redundant manipulator using the proposed scheme outperforms existing SOM-based algorithms.

KEYWORDS: Visual motor control; Self-organizing map; Sub-clustering; Redundancy resolution; Inverse kinematics.

1. Introduction

Visual motor control (VMC) is the task of guiding a robot manipulator to reach a target position in its workspace using visual feedback. The VMC considered in this paper is a special case of image-based visual-servoing in which a feasible inverse kinematic solution is obtained at position level. Readers can refer to the papers by Hutchinson[10] and Kragic *et al.*[13] for a survey on various visual-servoing techniques. VMC primarily involves two tasks, namely extracting the coordinate information of the robot end-effector and the target from camera images and then finding out necessary joint angle movement necessary for reaching the target. The first task involves camera calibration to find out an approximate model of the camera. This model can be used for mapping a point in the world coordinate to a point in the image coordinate. Tsai's algorithm[25] is one of the most widely used method for this purpose. The second task is more commonly known as solving the inverse kinematic problem. The problem becomes more difficult in case of redundant manipulators, where one has to resolve redundancy as well. Several kinds of kinematic control schemes for redundant manipulators have been suggested both in *Cartesian space*[17,28] and in *image coordinate space*.[7,11] While most of these methods provide the inverse kinematic solution in terms of joint angle velocity vector, this paper deals with the inverse kinematic solution directly in terms of joint angle vector.

Several kinds of learning architectures have been suggested for learning the inverse kinematic mapping of robot manipulators. This includes multiple-layer perceptron and radial basis function network,[21] cerebellar model arithmetic computer (CMAC) networks[18] and self-organizing maps (SOMs).[20] Among these learning architectures, the SOM[12] has been used extensively for VMC of manipulators during last two decades owing to its topology preserving property. A good survey on SOM-based control methods for robot manipulator is provided by Barreto *et al.*[4] Martinetz *et al.* used three-dimensional SOM-based neural architecture for VMC of non-redundant[20] and redundant manipulators.[19] Their method was later extended by Walter *et al.*[26] who introduced neural-gas algorithm along with an error correction scheme based on Widrow–Hoff-type learning rule. They also introduced the neighbourhood concept in SOM-based architecture to improve accuracy. However, the above-mentioned learning schemes employed for VMC of a redundant manipulator are not designed to learn the one-to-many relation.

This paper deals with the VMC of a redundant manipulator with seven degrees of freedom in an *eye-to-hand* configuration, where a pair of cameras are fixed overlooking the manipulator workspace. For this case, the four-dimensional image coordinate space viewed through a pair of cameras is considered as the *input space*, and the seven-dimensional joint angle vector is considered as the *output space*. A three-dimensional SOM-based neural architecture

* Corresponding author. E-mail: lbehera@iitk.ac.in

is selected for learning the inverse kinematic relationship between the input space and the output space. The primary objective is to learn the one-to-many relation, i.e. redundant solutions, so that the redundant manipulator can be used to perform dexterous tasks. SOM-based architecture has been used in the present work because of the following reasons:

1. SOM-based architecture expresses a non-linear relationship as a cluster of linear maps with each linear map being valid over a local region of operation.
2. Although the size of an SOM network may be huge (in terms of memory requirement) depending on the degrees of freedom of the manipulator, the real-time implementation is still efficient because once trained, only few neurons in the neighbourhood of the winning neuron become active, and these few new neurons take part in the decision process.
3. For a redundant manipulator, inverse kinematic solutions are many. Since existing SOM-based networks are not capable of learning this one-to-many relation, it is a worthwhile contribution to design such a scheme.

It is found that in case of redundant manipulators with six or more degrees of freedom, the standard SOM-based schemes such as those proposed by Martinetz and Schulten[20,26] do not properly capture the topology of the input and output spaces simultaneously. This makes the convergence of algorithm sensitive to the initial values of network parameters. Also, the joint angle vector computed using the standard SOM may exceed the physical joint angle limit of the manipulator unless it is explicitly constrained using a hard-limiter.

In their previous work,[15,16] the authors have proposed an SOM-based offline learning scheme that can preserve redundant solutions without an error-corrector loop between the desired target position and the end-effector. In this approach the end-effector reaches the target position in one step, but with a very high value of positioning error of more than 1 cm. Besides, the learning scheme assumes a fixed number of redundant solutions per SOM cluster. The inverse kinematic solution thus learnt is not *conservative*.[24] An inverse kinematic solution is said to be *conservative* if a closed path in end-effector space guarantees a closed path in joint angle space. In this approach, a closed trajectory in the task space does not give rise to a closed trajectory in the configuration space.

In this paper, a scheme that can learn redundant solutions in real time with an uncalibrated camera has been proposed. Simultaneously limitations of our earlier work have been eliminated.[15,16] In a nutshell, the visual control of a redundant manipulator using an error-corrector loop at the position level has been solved comprehensively in this paper. The salient features of the proposed learning architecture and the proposed algorithm are as follows:

1. The learning architecture uses the error-corrector loop between the desired target position and the end-effector.
2. The redundant solutions associated with each SOM cluster in the input space are learnt adaptively by learning sub-clusters in joint space using a real-time adaptive clustering algorithm. Thus multiple linear models are associated with each SOM cluster.
3. Since some clusters are associated with multiple linear models, the conventional neighbourhood concept as used by Martinetz *et al.*[20,26] cannot be used. A modified neighbourhood concept has been proposed in this paper to preserve the conservative property of the inverse kinematic solution.

The proposed scheme has been effectively implemented on a seven-degree-of-freedom (7DOF) robot manipulator. Performance-wise, the proposed scheme has the following advantages:

1. It is possible to attain a positioning error of less than 1 mm. In simulation, it is even possible to reach an error of less than 0.1 mm.
2. It leads to tenfold reduction in the amount of data needed for training the network. Only 50,000 data points are enough for attaining an accuracy of 1 mm. The training can be carried online.
3. The inverse kinematic solution is conservative. In other words, the joint angle trajectories are smooth and continuous for a continuous task-space trajectory.
4. The positioning accuracy attained is almost independent of the initial conditions of the network as compared to the standard SOM-based algorithms.
5. In a multi-step movement, the number of steps needed for obtaining lesser positioning error increases very slowly as compared with the existing SOM-based algorithms.

Since the proposed learning architecture can provide multiple inverse kinematic solutions, three criteria, namely lazy arm movement, minimum joint angle norm and minimum condition number of image Jacobian matrix, are used by the SOM network to resolve redundancy. It is to be noted that the standard SOM-based scheme always provides a unique inverse kinematic solution for any point in the workspace, even when the manipulator is a redundant one. On the other hand, the proposed approach enables the manipulator to reach the same target in more than one possible configuration. Thus dexterity of a redundant manipulator can be efficiently utilized using the proposed learning algorithm. The proposed schemes have been implemented in real time on a 7DOF PowerCube robot manipulator. The experimental and simulation results are found to be in agreement with each other.

The following section describes the system model used for VMC. The standard SOM-based VMC algorithm in the context of redundant manipulators is discussed in Section 3. The proposed algorithm using sub-clustering in joint angle space is explained in Section 4. The simulation and experimental results are provided in Section 5 followed by conclusion in Section 6.

2. System Model

2.1. Schematic of VMC set-up

The schematic of VMC is shown in Fig. 1. It consists of a stereo-camera system, a robot manipulator and a computer.

https://doi.org/10.1017/S026357470999049X

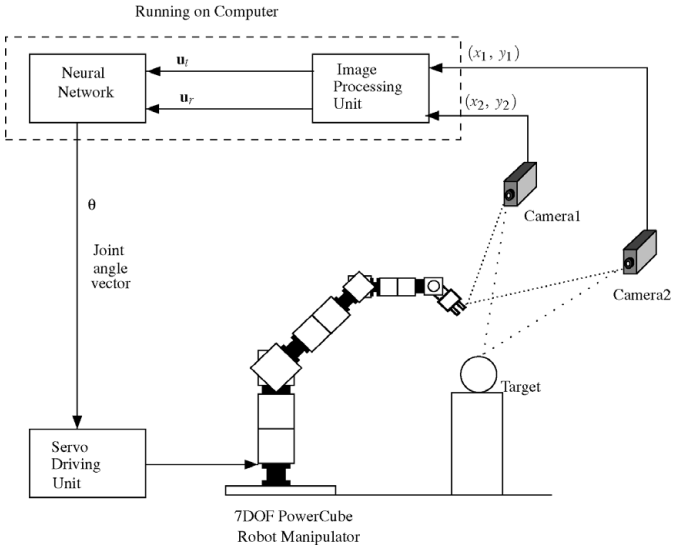


Fig. 1. Schematic of visual motor coordination: $\mathbf{u}_t$ and $\mathbf{u}_r$ are the four-dimensional image coordinate vectors for target point and robot end-effector respectively.

Image processing as well as neural network algorithms are executed on the computer. The final joint angle vector $\boldsymbol{\theta}$ obtained from the algorithm is provided to a servo unit which drives the robot manipulator. The image processing unit is used to extract four-dimensional image coordinate vectors for current robot position ($\mathbf{u}_r$) and target point ($\mathbf{u}_t$) to be reached in the workspace; (x_1, y_1) and (x_2, y_2) are two-dimensional pixel coordinates obtained from each camera.

2.2. *The manipulator model*

A 7DOF PowerCube robot manipulator (Amtec robotics, NC, USA) is used for real-time experiment. The model of the manipulator is derived from its link geometry. The Denavit–Hartenberg (D-H) parameters (shown in Table I) are used to derive forward kinematic equations for the manipulator. The reference frames for all joints needed for computing D-H parameters are shown in Fig. 2. We follow the notation given in the book by Spong and Vidyasagar.[23] The end-effector position is given by following equations:

$$\begin{aligned} x &= c_1(c_2(c_3(c_4c_5d_7s_6 + (c_6d_7 + d_5)s_4) - d_7s_3s_5s_6) \\ &\quad + s_2(-c_5d_7s_4s_6 + c_4(c_6d_7 + d_5) + d_3)) - s_1(s_3(c_4c_5d_7s_6 \\ &\quad + (c_6d_7 + d_5)s_4) + c_3d_7s_5s_6), \end{aligned}$$

Table I. D-H Parameters of PowerCube manipulator.

Link	a_i	d_i	α_i	θ_i
1	0	d_1	-90°	θ_1
2	0	0	90°	θ_2
3	0	d_3	-90°	θ_3
4	0	0	90°	θ_4
5	0	d_5	-90°	θ_5
6	0	0	90°	θ_6
7	0	d_7	0°	θ_7

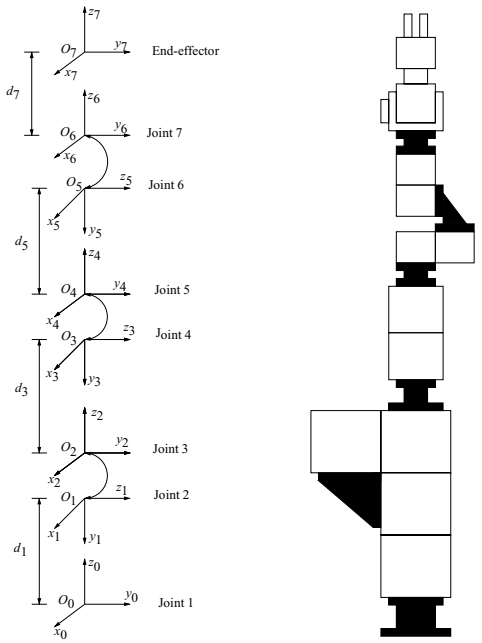


Fig. 2. Frames necessary for computing D-H parameters.

$$\begin{aligned} y &= s_1(c_2(c_3(c_4c_5d_7s_6 + (c_6d_7 + d_5)s_4) - d_7s_3s_5s_6) \\ &\quad + s_2(-c_5d_7s_4s_6 + c_4(c_6d_7 + d_5) + d_3)) + c_1(s_3(c_4c_5d_7s_6 \\ &\quad + (c_6d_7 + d_5)s_4) + c_3d_7s_5s_6), \\ z &= -s_2(c_3(c_4c_5d_7s_6 + (c_6d_7 + d_5)s_4) - d_7s_3s_5s_6) \\ &\quad + c_2(-c_5d_7s_4s_6 + c_4(c_6d_7 + d_5) + d_3) + d_1, \end{aligned} \tag{1}$$

where the various parameters are $d_1 = 0.390$ m, $d_3 = 0.370$ m, $d_5 = 0.310$ m, $d_7 = 0.2656$ m, $c_i = \cos\theta_i$, $s_i = \sin\theta_i$, $i = 1, 2, \ldots, 6$. Note that the end-effector position $[x\ y\ z]^T$ does not depend on the seventh joint angle.

2.3. *The camera model*

The tip of the robot end-effector as well as the target object are viewed through a stereo-camera system. In order to derive the forward kinematic model between the joint angle vector and the end-effector position in image coordinate, one needs to convert the Cartesian coordinate of the end-effector position into the corresponding pixel coordinates in the image plane. This necessitates a camera model. The Camera calibration consists of estimating model parameters for an uncalibrated camera. Tsai's algorithm[25] for non-co-planar camera calibration is used to estimate 11 parameters – extrinsic (R_x, R_y, R_z, T_x, T_y, T_z) and intrinsic (f, κ, C_x, C_y, s_x).

The relationship between the position of a point P in world coordinates (x_w, y_w, z_w) and the point's image in the camera's frame buffer (x_f, y_f) is defined by a sequence of coordinate transformations. The image coordinate of point P is given by

$$\begin{aligned} x_f &= \frac{s_x x_d}{d_x} + C_x, \\ y_f &= \frac{y_d}{d_y} + C_y, \end{aligned} \tag{2}$$

https://doi.org/10.1017/S026357470999049X

where s_x is the scaling factor; d_x and d_y are the x and y dimensions of camera's sensor element respectively; C_x, C_y are pixels coordinates of image centre; and x_d and y_d are true position on image plane considering radial distortions. The detailed derivation of these equations are available in papers by Tsai[25] and Behera *et al*.[5]

2.4. The problem definition

The forward mapping from a seven-dimensional joint angle space to a four-dimensional image coordinate space is obtained using the manipulator forward kinematic model and camera model. This is given by

$$\mathbf{u} = \mathbf{f}(\boldsymbol{\theta}), \tag{3}$$

where $\mathbf{u} = [u_1\ u_2\ u_3\ u_4]^T$ is the end-effector position obtained from a stereo-camera system and $\boldsymbol{\theta} = [\theta_1\ \theta_2\ \ldots\ \theta_7]^T$ is the seven-dimensional joint angle vector. The task is to learn the inverse map given by

$$\boldsymbol{\theta} = \mathbf{f}^{-1}(\mathbf{u}_t) = \mathbf{g}(\mathbf{u}_t). \tag{4}$$

In other words, we want to compute the joint angle vector necessary for reaching a target point $\mathbf{u}_t$ seen through a stereo-camera system. The inverse kinematic problem for a redundant manipulator is ill posed and may have infinite number of solutions. We intend to solve this problem using a real-time learning scheme that can preserve redundant solutions so that standard techniques of redundancy resolution can be applied to pick up one of these solutions for the given task at hand.

It is to be noted that the end-effector position does not depend on the last joint angle. Since we are not considering the orientation of the end-effector, the joint angle of the seventh link has been kept zero. Thus the inverse kinematic relation (4) represents a map from a four-dimensional input space to a six-dimensional output space in this paper. However, the proposed scheme will be able to compute the inverse kinematic solution in seven dimensions provided we are interested in the orientation of the end-effector.

3. Visual Motor Control with SOM Revisited

The inverse kinematic relationship (4) is a one-to-many mapping and hence difficult to learn. In an SOM-based approach, the input space is mapped to a three-dimensional lattice, where each lattice neuron represents a discrete cell. A *linear* map from the input space to the output space is learnt locally for each lattice neuron. The structure of the linear map for the lattice neuron γ is obtained by using first-order Taylor series expansion of Eq. (4) given by

$$\boldsymbol{\theta} - \boldsymbol{\theta}_\gamma = A_\gamma(\mathbf{u} - \mathbf{w}_\gamma), \quad A_\gamma = \left.\frac{\partial \mathbf{g}}{\partial \mathbf{u}}\right|_{(\mathbf{w}_\gamma, \boldsymbol{\theta}_\gamma)}, \tag{5}$$

where $\mathbf{w}_\gamma$ is the centre of the lattice neuron; A_γ is the inverse image Jacobian matrix; and $\boldsymbol{\theta}_\gamma$ is the joint angle vector associated with γ lattice neuron. The discretization process using SOM is illustrated pictorially in Fig. 3. Each lattice neuron γ is represented by a triplet ($\mathbf{w}_\gamma$, $\boldsymbol{\theta}_\gamma$, A_γ), where $\mathbf{w}_\gamma$

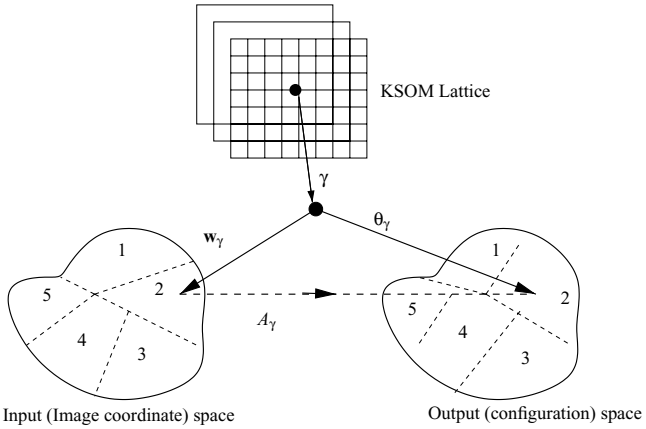


Fig. 3. Discretization of the input and output spaces using SOM: $\mathbf{w}_\gamma$ discretizes the input space and $\boldsymbol{\theta}_\gamma$ discretizes the output space; A_γ represents the linearized map between the input and the output space and KSOM is Kohonen's SOM.

discretizes the four-dimensional input space and $\boldsymbol{\theta}_\gamma$ the six-dimensional output space and the matrix A_γ of dimensions 6×4 represents a linear map from the input space to the output space. Since each weight vector $\mathbf{w}_\gamma$ is associated with only one joint angle vector $\boldsymbol{\theta}_\gamma$, this method gives a unique inverse kinematic solution even for a redundant manipulator.

Given a target position $\mathbf{u}_t$, a winner neuron μ is selected based on its Euclidean distance metric in the image coordinate (input) space. The neuron whose weight vector is closest to the target is declared the winner as shown below:

$$\mu = \arg\min_\gamma \|\mathbf{u}_t - \mathbf{w}_\gamma\|. \tag{6}$$

The arm is given a coarse movement $\boldsymbol{\theta}_0^{\text{out}}$ obtained as the weighted average of individual neuron outputs as shown below:

$$\theta_0^{\text{out}} = s^{-1} \sum_\gamma h_\gamma(\theta_\gamma + A_\gamma(\mathbf{u}_t - \mathbf{w}_\gamma)), \tag{7}$$

where $s = \sum_\gamma h_\gamma$ and $h_\gamma = \mathrm{e}^{-\left(\frac{\|\mu-\gamma\|^2}{2\sigma^2}\right)}$ is the neighbourhood function that is used as a variable weighting coefficient for each individual neuron output. Because of this coarse movement, the end-effector reaches a position $\mathbf{v}_0$. A correcting fine movement $\boldsymbol{\theta}_1^{\text{out}}$ is evaluated as follows:

$$\theta_1^{\text{out}} = \theta_0^{\text{out}} + s^{-1} \sum_\gamma h_\gamma A_\gamma(\mathbf{u}_t - \mathbf{v}_0). \tag{8}$$

This corrective movement results in a final movement of the end-effector to a position $\mathbf{v}_1$. Although one can use several such corrective movements to increase the accuracy of tracking, usually one corrective movement is sufficient to obtain a good tracking accuracy. Note that $\boldsymbol{\theta}_0^{\text{out}}$ and $\boldsymbol{\theta}_1^{\text{out}}$ are joint angle outputs obtained from the network, which are applied directly to the robot manipulator. These are different from their internal representations $\boldsymbol{\theta}_\gamma$, which are the angle vectors associated with lattice neurons $\gamma = 1, 2, \ldots$.

https://doi.org/10.1017/S026357470999049X

Various network parameters are updated as follows:[5,14,26]

$$\Delta \mathbf{v} = \mathbf{v}_1 - \mathbf{v}_0, \tag{9}$$

$$\Delta \boldsymbol{\theta}^{\text{out}} = \boldsymbol{\theta}_1^{\text{out}} - \boldsymbol{\theta}_0^{\text{out}}, \tag{10}$$

$$\Delta \boldsymbol{\theta}_\gamma = \frac{h_\gamma}{s}\left[\boldsymbol{\theta}_0^{\text{out}} - s^{-1}\sum_\gamma h_\gamma(\boldsymbol{\theta}_\gamma + A_\gamma(\mathbf{v}_0 - \mathbf{w}_\gamma))\right], \tag{11}$$

$$\Delta A_\gamma = \frac{h_\gamma}{s \cdot \|\Delta \mathbf{v}\|^2}\left[\Delta \boldsymbol{\theta}^{\text{out}} - s^{-1}\sum_\gamma h_\gamma A_\gamma \cdot \Delta \mathbf{v}\right]\Delta \mathbf{v}^T, \tag{12}$$

$$\mathbf{w}_\gamma \leftarrow \mathbf{w}_\gamma + \eta_w\, h_\gamma(\mathbf{u}_t - \mathbf{w}_\gamma), \tag{13}$$

$$\boldsymbol{\theta}_\gamma \leftarrow \boldsymbol{\theta}_\gamma + \eta_t\, \Delta \boldsymbol{\theta}_\gamma, \tag{14}$$

$$A_\gamma \leftarrow A_\gamma + \eta_a\, \Delta A_\gamma. \tag{15}$$

The parameters $\eta_w, \eta_t, \eta_a, \sigma$ are varied during the training as follows:

$$\eta = \eta_{\text{init}}\left(\frac{\eta_{\text{fin}}}{\eta_{\text{init}}}\right)^{(t/t_{\max})}. \tag{16}$$

In the above equation, $\eta \in \{\eta_w, \eta_t, \eta_a, \sigma\}$ is a user-defined parameter that varies between the initial value η_{init} and η_{fin} as the iteration step t proceeds from 0 to $t_{\max}$ during the training phase.

The parameter update rules (13)–(15) may be explained as follows: The weight update law (13) is a simple SOM clustering algorithm in which the winner neuron and the neurons in its neighbourhood are moved towards the input vector $\mathbf{u}_t$. The update is aimed at reducing the error term $\|\mathbf{u}_t - \mathbf{w}_\gamma\|$. The parameter A_γ is updated so as to learn the linear mapping (5) which is rewritten as

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}_\gamma + A_\gamma(\mathbf{u}_t - \mathbf{w}_\gamma), \tag{17}$$

where $\boldsymbol{\theta}^*$ is the desired joint angle necessary for reaching the target point $\mathbf{u}_t$. In order to learn this mapping, A_γ is updated so that the following cost function is minimized:

$$E_1 = \frac{1}{2}(\Delta \boldsymbol{\theta}^{\text{out}} - A_\gamma \Delta \mathbf{v})^2. \tag{18}$$

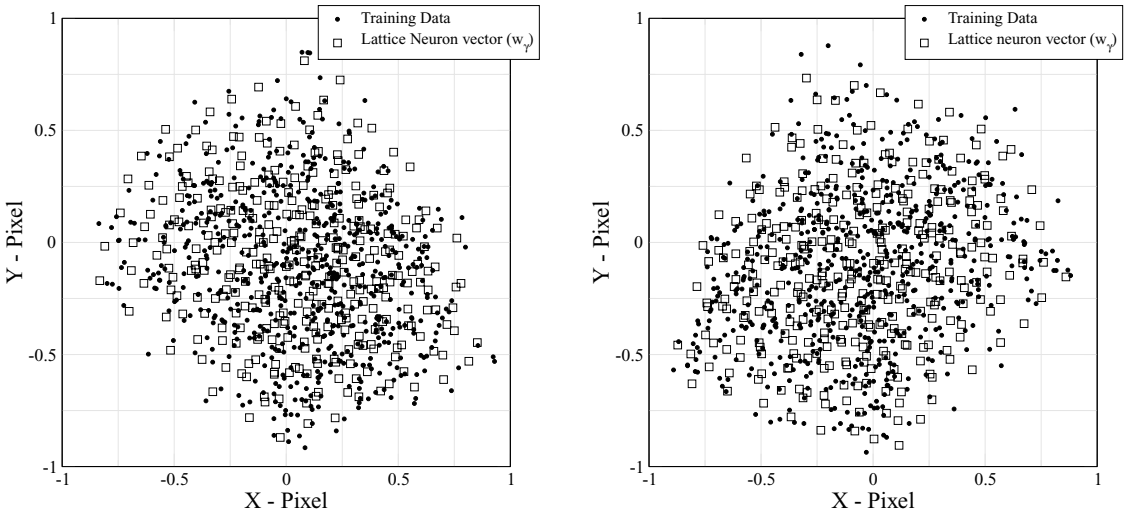
Applying the gradient-descent rule to the above equation, we obtain the Jacobian update law (15). The pre-factor $\|\Delta \mathbf{v}\|^{-2}$ determines the size of adaptation step; $\boldsymbol{\theta}_\gamma$ is the internal representation of the joint angle output $\boldsymbol{\theta}^{\text{out}}$ corresponding to each neuron γ. During the coarse movement, the robot joints are moved by $\boldsymbol{\theta}^{\text{out}}$, and its new position is recorded from the camera as $\mathbf{v}_0$. Now, $\boldsymbol{\theta}_\gamma$ is updated so as to reflect this new joint angle position at the nodal level. In other words, $\boldsymbol{\theta}_\gamma$ is updated so as to minimize the following error function given by

$$E_2 = \frac{1}{2}\left(\boldsymbol{\theta}_0^{\text{out}} - \frac{\sum_\gamma h_\gamma[\boldsymbol{\theta}_\gamma + A_\gamma(\mathbf{v}_0 - \mathbf{w}_\gamma)]}{\sum_\gamma h_\gamma}\right)^2. \tag{19}$$

The update law for $\boldsymbol{\theta}_\gamma$ given by Eqs. (10) and (14) is obtained by applying the gradient-descent rule to the above error function.

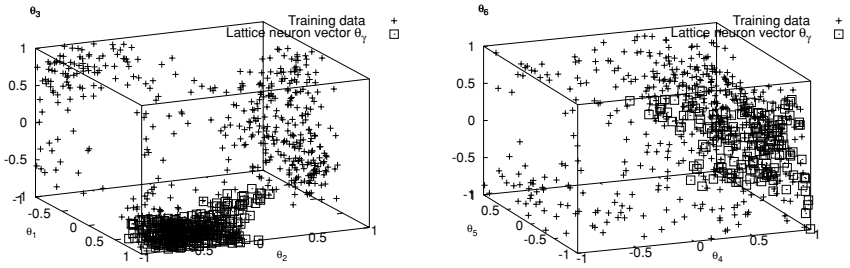
The standard SOM-based VMC scheme has the following limitations which restrict its applicability to redundant manipulators:

1. It is found that for a redundant manipulator with six or more degrees of freedom, although the SOM lattice neurons preserve topology of the input space as shown in Fig. 4, the lattice fails to preserve the topology of the output (joint angle) space as shown in Fig. 5. In Fig. 4, it can bee seen that the weight vectors ($\mathbf{w}_\gamma$) represented by square 'boxes' are spread out uniformly over the input space. On the other hand, in Fig. 5, we find that the clusters in joint angle space ($\boldsymbol{\theta}_\gamma$) represented by square 'boxes' are concentrated at one location. It would be



(a) Camera 1 (b) Camera 2

Fig. 4. Clustering in the image coordinate (input) space. Lattice neurons capture the topology of the input space during training. The dots denote the actual input data generated during training, and the squares represent the cluster centres $\mathbf{w}_\gamma$.



(a) Space formed by first 3 joint angle vectors

(b) Space formed by last 3 joint angle vectors

Fig. 5. The topology of the output space is not captured by the original VMC algorithm during the evolution of parameters (training phase). While the training data shown by the '+' signs are distributed across the entire volume, the cluster centres $\boldsymbol{\theta}_\gamma$ are collected at one location.

shown in the simulation section that because of the fact that the network fails to capture the output topology, the positioning accuracy attained using the standard SOM algorithm is sensitive to the initial conditions.

2. The standard SOM algorithm returns a unique inverse kinematic solution for any target in the manipulator workspace. This might not be desirable in case of redundant manipulators, where one would like to choose a different configuration to satisfy some additional requirement. Even though the training data sets are replete with redundant solutions, there is no provision to *preserve* this redundancy during the evolution of parameters.

To summarize, the standard SOM-based VMC schemes are not efficient to learn the inverse-kinematic relation for redundant manipulators. In the following section, we propose a new architecture in order to overcome the above-mentioned drawbacks of conventional VMC algorithms.

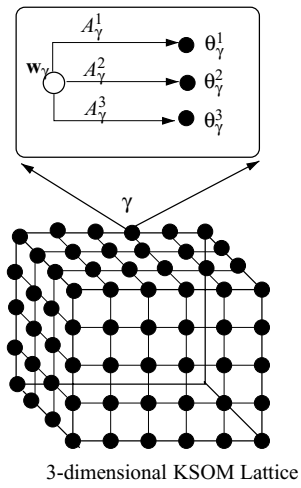


Fig. 6. Sub-clustering in joint angle space. Each lattice neuron γ is associated with one weight vector $\mathbf{w}_\gamma$ and several joint angle vectors $\boldsymbol{\theta}^\lambda_\gamma,\ \lambda = 1, 2, \ldots, N_\gamma$. Here $N_\gamma = 3$.

4. SOM with Sub-Clustering in Joint Angle Space

Any point within the three-dimensional Cartesian workspace of the manipulator may be reached using only three degrees of freedom. The presence of higher degrees of freedom provides dexterity in performing the task at hand. In other words, apart from reaching the point, the extra degrees of freedom may be used to perform some additional tasks like avoiding obstacles, meeting joint angle limits and satisfying other motion constraints. The proper utilization of available degrees of freedom has been an interesting problem for researchers.

In the context of VMC using SOM networks, redundancy resolution has been dealt with by many authors. For instance, Martinetz *et al.*[19] applied SOM algorithm to a 5DOF manipulator and argued that because of neighbourhood function, the redundancy is resolved 'naturally' by using 'lazy arm method'. Han *et al.*[9] and Zha *et al.*[29] used SOM for avoiding obstacles. Zheng *et al.*[30] resolved redundancy by optimizing some task-oriented criteria. Most of these methods have been applied to 3DOF, 4DOF or 5DOF manipulators, and each method resolves redundancy in only one way. The redundancy resolution scheme is learnt during the training phase itself. Hence once the network is trained, it is not possible to change the redundancy resolution criterion while computing the inverse kinematic solution. Since most of the current VMC schemes involve a time-consuming training process, retraining the network for a new redundancy resolution criterion is not desirable.

In case of redundant manipulators, several set of joint angles may lead to the same end-effector position. Thus, the data generated during the training phase consist of redundant data sets. The current VMC algorithms explained in Section 3 do not have provisions to preserve this redundancy in the solution space. In this section, we introduce a concept called *sub-clustering in joint angle space* to preserve this redundancy in a useful manner. The idea is to associate multiple angle vectors with each lattice neuron as shown in Fig. 6. Since the number of redundant solutions available for a target point varies across the workspace of manipulator, the number of sub-clusters to be associated with each neuron is decided online based on the distribution of generated data points.

4.1. Network architecture

A three-dimensional SOM lattice is considered in this case as well. Each lattice neuron γ is associated with a four-dimensional weight vector $\mathbf{w}_\gamma$ and several six-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 $\boldsymbol{\theta}^j_\gamma,\ j = 1, 2, \ldots, N_\gamma$, and an

https://doi.org/10.1017/S026357470999049X

equal number of Jacobian matrices A_γ^j $j = 1, 2, \ldots, N_\gamma$ of dimension 6×4. The number N_γ varies with each γ and is decided online based on the actual data distribution. When lattice neuron γ becomes a winner for a given input vector $\mathbf{u}_t$, this network architecture can actuate N_γ kinematic configurations by which the robot manipulator can reach the same target as per following relation:

$$\theta^j = \theta_\gamma^j + A_\gamma^j(\mathbf{u}_t - w_\gamma); \quad j = 1, 2, \ldots, N_\gamma. \tag{20}$$

For this network architecture, the parameters $\boldsymbol{\theta}_\gamma^j$ and A_γ^j cannot be learnt using standard SOM algorithm. Unlike standard SOM algorithm, we propose an online clustering algorithm to learn θ_γ^j, while the error-correcting gradient learning for A_γ in standard SOM algorithm has been adapted to learn A_γ^j.

4.2. Training algorithm

The training phase consists of following steps:

(1) Set the iteration counter $k = 1$.
(2) *Data generation.* A training data set $(\boldsymbol{\theta}_t, \mathbf{u}_t)$ is generated using robot and camera models during simulation. The robot manipulator commands a movement in joint angle space by generating a random vector θ_t within physical limits, while the input vector $\mathbf{u}_t$ is recorded from camera output.
(3) *Clustering in the input space.* For each four-dimensional target input $\mathbf{u}_t$, a winner neuron μ is selected based on minimum Euclidean distance as shown in Eq. (6). The weight vectors corresponding to the winner neuron μ and the neighbouring neurons are updated as per Eq. (13) for the given target vector $\mathbf{u}_t$.
(4) *Clustering in the output space.* Let's assume that this winner neuron μ is associated with N_μ number of θ vectors given by $\boldsymbol{\theta}_\mu^j$, $j = 1, 2, \ldots, N_\mu$. The incoming target angle vector $\boldsymbol{\theta}_t$ is used to create a new angle vector or update the existing angle vectors as per the following conditions.
   (i) Case I. If $N_\mu = 0$, i.e. if there is no θ vector associated with this neuron, then assign the target joint angle vector $\boldsymbol{\theta}_t$ as its first centre. That is $\boldsymbol{\theta}_\mu^{N_\mu+1} = \boldsymbol{\theta}_\mu^1 = \boldsymbol{\theta}_t$.
   (ii) Case II. If $N_\mu > 0$, the steps given below are followed.
      (a) Find the angle vector $\boldsymbol{\theta}_\mu^j$ which is nearest to the incoming angle vector $\boldsymbol{\theta}_t$. Let's call the winner among these angle vectors $\boldsymbol{\theta}_\mu^\beta$ where

$$\beta = \arg\min_j \left\| \boldsymbol{\theta}_\mu^j - \boldsymbol{\theta}_t \right\|, \quad j = 1, 2, \ldots, N_\mu. \tag{21}$$

      (b) If the minimum distance $d_{\min} = \|\boldsymbol{\theta}_\mu^\beta - \boldsymbol{\theta}_t\| < K$, where K is a user-defined threshold, the angle vectors are updated using a competitive rule given by

$$\boldsymbol{\theta}_\mu^j(k+1) = \boldsymbol{\theta}_\mu^j(k) + \eta h_{\beta j}(\boldsymbol{\theta}_t - \boldsymbol{\theta}_\mu^j(k)), \quad j = 1, 2, \ldots, N_\mu, \tag{22}$$

where $h_{\beta j} = \mathrm{e}^{\frac{-(\beta - j)^2}{2\sigma_t^2}}$ is the neighbourhood function used for sub-clustering. A suitable value of spread of Gaussian function σ_t is selected for this purpose.
      (c) If $d_{\min} > K$, create a new centre and assign the incoming θ_t vector to it and increment the count of angle centres associated with this winner neuron μ from N_μ to $N_\mu + 1$. In other words, $\boldsymbol{\theta}_\mu^{N_\mu+1} = \boldsymbol{\theta}_t$.
(5) *Coarse movement.* Because of sub-clustering, the winner neuron μ is associated with N_μ sub-clusters in joint angle space given by $\boldsymbol{\theta}_\mu^j$, $j = 1, 2, \ldots, N_\mu$. For the given target point $\mathbf{u}_t$, the network has N_μ outputs given by

$$\boldsymbol{\theta}_0^j = \boldsymbol{\theta}_\mu^j + A_\mu^j(\mathbf{u}_t - \mathbf{w}_\mu), \quad j = 1, 2, \ldots, N_\mu. \tag{23}$$

These are called coarse movements. These coarse movements lead to end-effector positions $\mathbf{v}_0^j$, $j = 1, 2, \ldots, N_\mu$, as recorded by the cameras.
(6) *Fine movement.* Based on the current positioning accuracy for each end-effector position $\mathbf{v}_0^j$, a fine movement may also be carried out as

$$\boldsymbol{\theta}_1^j = \boldsymbol{\theta}_0^j + A_\mu^j\left(\mathbf{u}_t - \mathbf{v}_0^j\right). \tag{24}$$

The new end-effector positions are recorded as $\mathbf{v}_1^j$, $j = 1, 2, \ldots, N_\mu$.
(7) The difference $\Delta\mathbf{v}^j = \mathbf{v}_1^j - \mathbf{v}_0^j$ is used to update the corresponding Jacobian matrix A_μ^j so as to minimize the error

$$E_j = \frac{1}{2}\left(\Delta\theta^j - A_\mu^j \Delta\mathbf{v}^j\right)^2. \tag{25}$$

This gives the following update law for Jacobian matrices:

$$A_\mu^j(k+1) = A_\mu^j(k) + \frac{\eta}{\|\Delta\mathbf{v}^j\|^2}\left(\Delta\theta^j - A_\mu^j\Delta\mathbf{v}^j\right)\Delta\mathbf{v}^{jT}. \tag{26}$$

(8) Increment the iteration counter $k = k + 1$ and go to step 1.

Note that during the training phase, the output of the network is not a weighted average of all the neurons as was previously done. It is because of the fact that the number of $\boldsymbol{\theta}$ clusters associated with neurons are not same, and hence the usual neighbourhood concept cannot be used in this case.

4.3. Testing phase

1. For a given target point $\mathbf{u}_t$, the winner neuron μ is computed based on its minimum Euclidean distance from the target in the input space as given by (6). This winner neuron is associated with several, say, N_μ sub-clusters in joint angle space.
2. One can choose among these sub-clusters based on some criterion. The redundancy is resolved using three criteria, namely lazy arm movement, minimum angle norm and minimum condition number of Jacobian matrix. Let the winning joint angle sub-cluster be β. Once the winner indices μ and β are computed, coarse and fine joint angle

https://doi.org/10.1017/S026357470999049X

outputs are given by

$$\boldsymbol{\theta}_0 = \boldsymbol{\theta}_0^{\beta} = \boldsymbol{\theta}_{\mu}^{\beta} + A_{\mu}^{\beta}(\mathbf{u}_t - \mathbf{w}_{\mu}), \tag{27}$$

$$\boldsymbol{\theta}_1 = \boldsymbol{\theta}_1^{\beta} = \boldsymbol{\theta}_0 + A_{\mu}^{\beta}(\mathbf{u}_t - \mathbf{v}_0), \tag{28}$$

where $\mathbf{v}_0$ is the end-effector position recorded after coarse movement. Multiple steps may be taken to improve the positioning accuracy further.

4.4. Redundancy resolution

Sub-clustering gives rise to multiple configurations for every target position. Let the index of winner neuron in the input space be μ, and this winner neuron is associated with N_{μ} sub-clusters. One can select a suitable configuration based on different criteria. The redundancy is resolved using following criteria:

1. *Lazy arm movement.* The angle sub-cluster which is closest to the current robot configuration is selected as the winner. The winning sub-cluster for this criteria is given by

$$\beta = \arg\min_j \left\| \boldsymbol{\theta}_{\mu}^j - \boldsymbol{\theta}_c \right\|, \tag{29}$$

where $\boldsymbol{\theta}_c$ is the current robot configuration.

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

$$\beta = \arg\min_j \left\| \boldsymbol{\theta}_{\mu}^j \right\|. \tag{30}$$

3. *Minimum condition number.* The matrix A_{μ}^j represents a local inverse image Jacobian matrix associated with each joint angle vector $\boldsymbol{\theta}_{\mu}^j$. For VMC, it is desirable to have low condition number for image Jacobian matrices to improve the robustness and numerical stability of the system.[6] Sometimes the condition number of image Jacobian matrix is used as the measure of the *perceptibility* of motion.[8,22] The *perceptibility* is a quantitative measure of the ability of a camera set-up to observe the changes in image feature due to motion of robot end-effector. It is used to evaluate the ease of achieving vision-based control and steering away from singular configurations.[22]

   Since several joint angle configurations are available for a given winner neuron, each associated with an inverse Jacobian matrix, one can choose a particular configuration based on the minimum condition number of these matrices. The winning sub-cluster based on the minimum condition number is given by

$$\beta = \arg\min_j [\mathrm{cond}(A_{\mu}^j)], \tag{31}$$

where $\mathrm{cond}(A_{\mu}^j)$ is the condition number of the matrix A_{μ}^j.

4.5. Tracking a continuous trajectory

Unlike the original VMC algorithm discussed in Section 3, the neighbourhood function is not used during the training phase. This might not be of concern when the task is to reach isolated points in the workspace. However, one would like to have a continuous trajectory in joint angle space for a continuous trajectory in image coordinate space. In other words, the conservative property[24] of inverse kinematic solution is desirable and needs to be preserved.

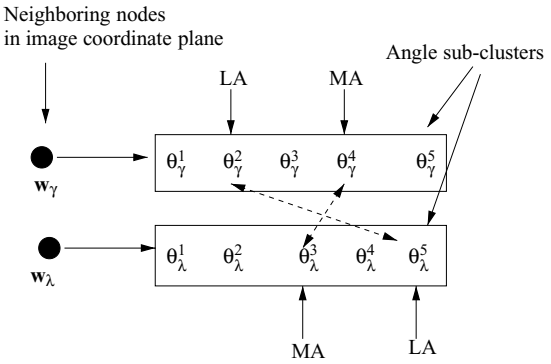


Fig. 7. Defining neighbourhood in a sub-clustered environment: $(\mathbf{w}_\gamma, \boldsymbol{\theta}_\gamma^2)$ and $(\mathbf{w}_\lambda, \boldsymbol{\theta}_\lambda^5)$ are similar configurations for lazy arm criterion when γ and λ are neighbouring neurons in the input space; LA is for the lazy arm criterion and MA for the minimum angle norm criterion.

Use of neighbourhood helps in maintaining a continuous trajectory in the joint angle space by avoiding abrupt changes in the joint motion. This happens because the network output is obtained by taking the weighted average of individual neuron outputs within a neighbourhood around the winner neuron. In order to facilitate further discussion, we divide the available configurations into following two classes:

1. Each joint angle sub-cluster represents a particular robot configuration. Two configurations $\boldsymbol{\theta}_\gamma^i$ and $\boldsymbol{\theta}_\lambda^j$ are said to be *similar* if $\|\boldsymbol{\theta}_\gamma^i - \boldsymbol{\theta}_\lambda^j\| < K$, $i \neq j$, $\gamma \neq \lambda$, and K is an arbitrarily small constant. Otherwise, they would be called *dissimilar* configurations.
2. All angle sub-clusters associated with each neuron γ are *dissimilar*. In other words, $\|\boldsymbol{\theta}_\gamma^i - \boldsymbol{\theta}_\gamma^j\| > K$ for $i \neq j$. Note that K is the threshold that was used for creating a new sub-cluster during training phase. Refer to the discussion under the head 'Clustering in the output space'.

The concept of neighbourhood in a sub-clustered environment employed in this paper is explained in Fig. 7. In this figure, $\mathbf{w}_\gamma$ and $\mathbf{w}_\lambda$ are two neighbouring neurons, represented by their respective weight vectors. The angle sub-clusters associated with them are represented by $\boldsymbol{\theta}_\gamma^i$, $i = 1, 2, \ldots, N_\gamma$, and $\boldsymbol{\theta}_\lambda^j$, $j = 1, 2, \ldots, N_\lambda$, respectively. For simplicity, it is assumed that $N_\gamma = N_\lambda = 5$. Let's assume that the lazy arm criterion selects the angle vector '2' in case of γ neuron and '5' in case of λ neuron (refer to (29)). On the other hand, the minimum angle norm criterion selects angle vector '4' in case of γ and '3' in case of λ neuron (refer to (30)). The dotted lines show that the configurations corresponding to these sub-clusters are *similar*. In other words, the configurations $(\gamma, 2)$ and $(\lambda, 5)$ are *similar* and so are the configurations $(\gamma, 4)$ and $(\lambda, 3)$.

The network output is obtained as a weighted average over all *similar* joint angle vectors. The coarse and fine joint angle movements, after incorporating neighbourhood, are given by

https://doi.org/10.1017/S026357470999049X

following two equations:

$$\boldsymbol{\theta}_0 = \frac{\sum_{\gamma} h_\gamma \left(\boldsymbol{\theta}_\gamma^{\beta(\gamma)} + A_\gamma^{\beta(\gamma)}(\mathbf{u}_t - \mathbf{w}_\gamma)\right)}{\sum_{\gamma} h_\gamma}, \tag{32}$$

$$\boldsymbol{\theta}_1 = \boldsymbol{\theta}_0 + \frac{\sum_{\gamma} h_\gamma A_\gamma^{\beta(\gamma)}(\mathbf{u}_t - \mathbf{v}_0)}{\sum_{\gamma} h_\gamma}, \tag{33}$$

where $h_\gamma = \mathrm{e}^{-\frac{\|\gamma-\mu\|^2}{\sigma^2}}$ is the neighbourhood function defined in the input space; μ is the winner neuron in input (image coordinate) space; $\beta(\gamma)$ is the index of the winning angle sub-cluster associated with the neuron γ. The winning sub-cluster for each neuron γ is obtained using the same criterion which was used for winner neuron μ. In the simulation section, it would be shown that the use of neighbourhood results in a smooth trajectory in the joint angle space.

5. Simulation and Results

5.1. Network architecture and workspace dimensions

A three-dimensional neural lattice with $7 \times 7 \times 7$ neurons is selected for the task. Training data is generated using the forward kinematic model (1) and camera model (3). A Cartesian workspace of dimension of 600 mm × 500 mm × 500 mm is considered for both simulation and experiment. All points within this workspace are visible through both the cameras of the stereo-vision system. Joint angle values are generated randomly within the physical limits of the manipulator, and only those input–output pairs are retained in which the end-effector positions are visible by both the cameras simultaneously. The ranges of the input and output spaces are given in Table II. Since the end-effector positions in the camera plane and joint angles have different range of values, data points are normalized within ±1.

5.2. Training

The network is trained offline using 50,000 data generated using the forward kinematic model (1) and (3). The training can be carried out 'online', which would necessitate generating data by moving the robot continuously. Generating such a large number of data on a real system might not be convenient. Hence we follow the hybrid approach proposed by Behera *et al.*[5], where network is trained offline using approximate models and is then fine-tuned during online operation.

Table II. Ranges of the input and output spaces.

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

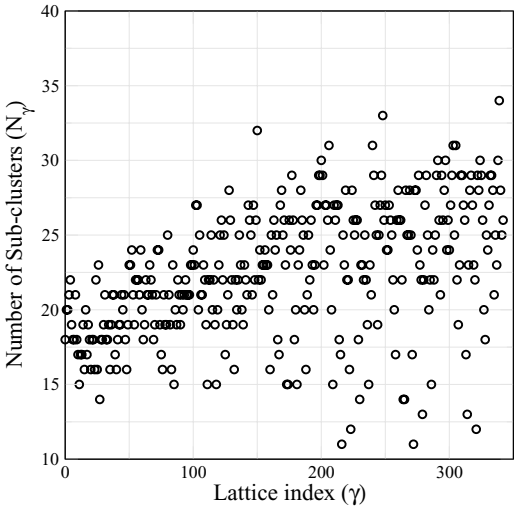


Fig. 8. The number of sub-clusters for lattice neurons. The number of sub-clusters vary from one neuron to another. The number of sub-clusters for a neuron is decided online based on the training data distribution.

A new $\boldsymbol{\theta}$ sub-cluster is formed whenever the distance of incoming $\boldsymbol{\theta}_t$ from the existing nearest sub-cluster exceeds the threshold $K = 1.0$. The distribution of sub-clusters for lattice neurons is shown in Fig. 8. The number of sub-clusters associated with each neuron varies between 10 and 35. The distribution of joint angle sub-clusters in three-dimensional manipulator workspace is shown in Fig. 9. It is seen that the points with less number of redundant solutions lie towards the boundary of the workspace as shown by the '+' symbols (number of solutions <19). The number of points with very large number of redundant solutions is also less as shown by

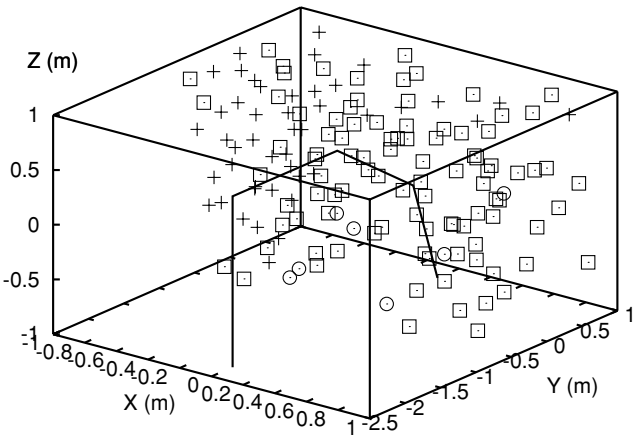
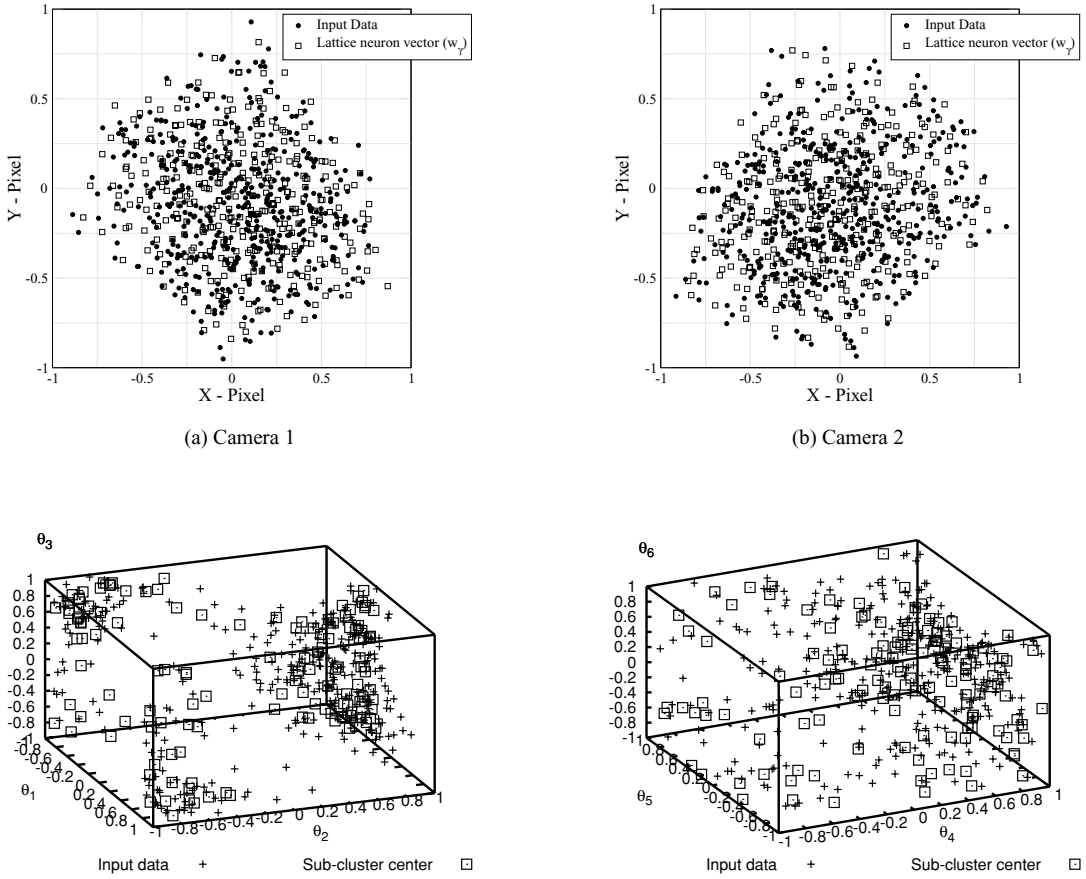


Fig. 9. Distribution of joint angle sub-clusters in the three-dimensional manipulator workspace. The number of solutions available for a given target point varies across the workspace. There are very few points at which the number of available solutions are too high. The number of solutions decreases towards boundary of workspace. In this figure, '+' represents the points at which the number of sub-clusters $N_\gamma < 18$; squares represent the points with N_γ between 19 and 30; and circles represent points with $N_\gamma > 30$. A typical robot configuration is shown by the solid line.

https://doi.org/10.1017/S026357470999049X



(a) Camera 1

(b) Camera 2

(c) First 3 angles

(d) Last 3 angles

Fig. 10. Clustering in the input and output spaces. The SOM lattice learns the topology of the input and output spaces. Since the number of joint angle sub-clusters are decided online based on the actual data distribution, the outliers are automatically avoided. This is because a new sub-cluster is created only when the new training data is far away from current sub-clusters.

the circles (number of solutions > 30). The square symbols represent points with number of solutions between 19 and 30. Hence it is seen that the proposed architecture has been able to learn the distribution of redundant solutions properly.

The discretization of the input and output spaces by the lattice neurons is shown in Fig. 10. It is seen that the topology is captured by the lattice neurons in both the input and the output space. Since the number of sub-clusters for each neuron is decided based on the input data distribution, the outliers are automatically avoided. Outliers are the neurons which do not represent input data. In Figs. 10(c) and 10(d), it is seen that all sub-clusters are surrounded by training data points, and they do not lie in an empty region. These results are in contrast to that of a standard SOM algorithm as shown in figure 5, where clusters are localized and some of them are outliers.

5.3. Testing

The following tasks were performed to demonstrate the efficacy and usefulness of the proposed schemes.

5.3.1. Reaching isolated points in the workspace. The joint angle vectors were computed for 20,000 points located randomly within the manipulator workspace. Only one step is used to compute the necessary joint angles. The performance of the proposed sub-clustering-based scheme is compared with the standard SOM-based scheme.[5,26] The performance comparison is provided in Table III. Since the performance

Table III. Performance comparison for reaching isolated points: LA, lazy arm; MA, minimum angle norm; MC, minimum condition number.

	Average positioning error			
Scheme	Cartesian space (mm)	Image space (pixels)	Angle norm (normalized)	Learning parameters (η_w, η_t, η_a, σ)
Standard SOM	24.93	7.87	1.54	0.1, 0.2, 0.9, 0.1
SC + LA	3.83	1.22	0.83	0.1, 0.5, 0.9, 1.5
SC + MA	3.0	0.95	0.76	0.1, 0.5, 0.9, 1.5
SC + MC	18.67	6.33	1.38	0.1, 0.9, 0.9, 0.2

https://doi.org/10.1017/S026357470999049X

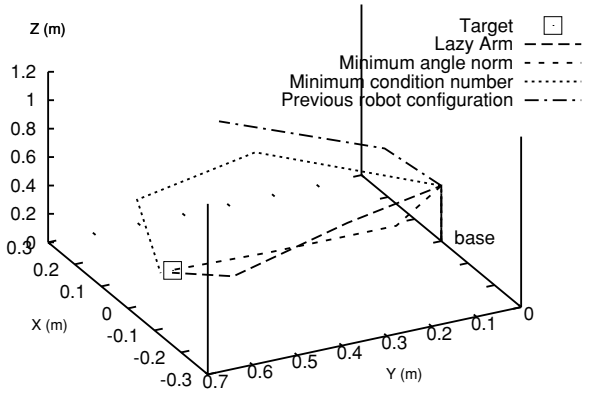


Fig. 11. Redundancy resolution using various criteria. Different criteria give different configurations for the same target point.

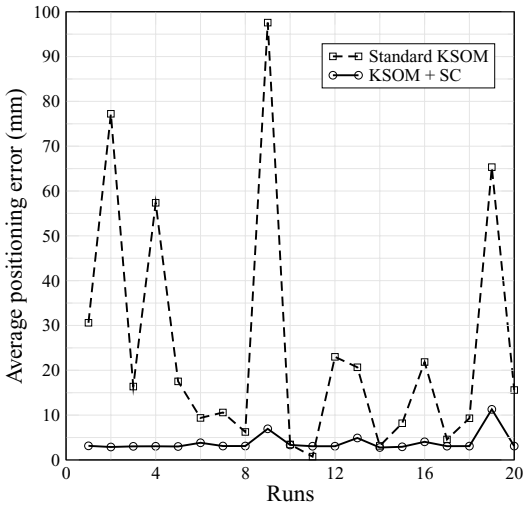


Fig. 12. Dependence of convergence on the initial conditions. The standard SOM algorithm is sensitive to initial conditions and hence varies widely for different runs. The performance of the sub-clustering-based scheme is independent of initial conditions.

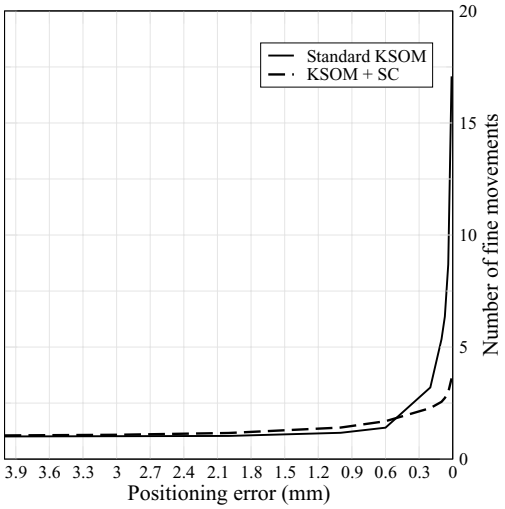


Fig. 13. Improving positioning accuracy using multiple fine movements. The results are obtained by averaging over 1000 test points. The standard SOM algorithm requires very large number of fine movements to attain an accuracy below 1 mm as compared to the sub-clustering-based method.

of standard SOM-based schemes is found to depend on initial values of network parameters, the test results are averaged over 20 different runs. Each run starts with a different random initialization.

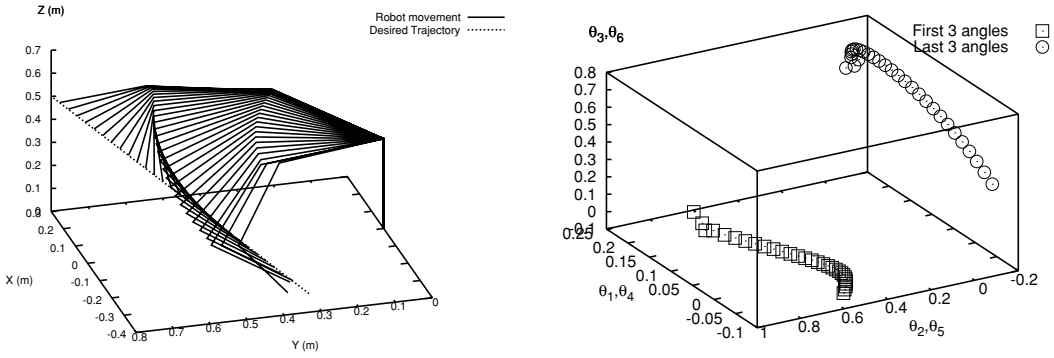
As discussed earlier, the sub-clustering-based scheme preserves the redundancy available in training data and provides a finite number of joint angle vectors for every target position in the manipulator workspace. The redundancy is resolved using three criteria, namely lazy-arm method, minimum angle norm and minimum condition number. Various solutions obtained for a given target point after resolving redundancy are shown in Fig. 11.

From Table III, it is clear that sub-clustering-based methods give better positioning accuracy than the standard SOM-based schemes proposed by Martinetz[20] and Schulten.[26] It is seen in Table III that standard SOM algorithm gives rise to very large joint angle variation as reflected in the magnitude of joint angle norm.

It has been said earlier that the joint angle vectors associated with lattice neurons do not capture the topology of the output space. This makes the convergence of standard SOM-based schemes sensitive to initial values of network parameters as shown in Fig. 12. In this figure, average positioning errors over 20,000 isolated target positions are shown for different runs, where each run starts with a different random initialization and includes a training and a testing phase. It is observed in Fig. 12 that the average positioning error varies widely across different runs and can be as high as 100 mm. On the contrary, the performance of the sub-clustering-based scheme using lazy arm redundancy resolution technique is comparatively less sensitive to initial conditions, and performance remains constant across different runs. The average positioning accuracy over all runs is below 4 mm, implying that the proposed scheme is very accurate in position tracking.

The advantage of an online incremental learning scheme is that one can execute multiple fine movements (refer to (28)) to improve the positioning accuracy as shown in Fig. 13. This figure plots the number of fine movement steps required to achieve a given average positioning error over 1000 test points. Since the standard SOM algorithm is sensitive to initial network parameters, we have taken the initial network parameter corresponding to run 14, where both the standard SOM and proposed algorithms have the same level of performance. Figure 13 shows that the standard SOM algorithm requires 17 fine movement steps to attain the average positioning accuracy of 0.1 mm, while the proposed sub-clustering-based method takes three steps to attain the same accuracy. The result shown in this figure is in sharp contrast to those reported by Angulo *et al.*[3], where the authors have taken more than 1000 steps to attain that level of accuracy even when they have used orientation information to compute the inverse kinematic solution. Note that in Fig. 13, the result for the standard SOM is shown for those initial conditions for which the network performance is comparable with the proposed scheme. Otherwise, the performance of the standard SOM will further deteriorate for other initial conditions.

https://doi.org/10.1017/S026357470999049X



(a) Robot configuration

(b) Joint angle trajectories

Fig. 14. Tracking a straight line using the lazy arm criteria. A continuous trajectory in the task space gives rise to a continuous trajectory in the joint angle space.

5.3.2. Tracking a straight line trajectory. The desired straight line trajectory in Cartesian space is given by

$$y = \frac{5}{6}x + \frac{11}{20},$$
$$z = \frac{5}{6}(x + 0.3), \tag{34}$$

where $-0.3\text{ m} \leq x \leq 0.3\text{ m}$. Six hundred points are generated sequentially on this line, and the joint angles for each point are computed using only one fine movement. Since the learning algorithm has been designed so that it can be used online, the parameters are updated even during the testing phase. This helps in improving tracking accuracy. A typical tracking result obtained using the sub-clustering scheme is shown in Fig. 14(a). In this case, the redundancy is resolved using the lazy arm criterion. The corresponding joint angle trajectory is shown in Fig. 14(b). It is seen that for a continuous trajectory in the Cartesian space, the joint angle movement is continuous, and hence the inverse kinematic solution is conservative in nature.

Performance comparison among various redundancy resolution schemes is provided in Table IV. The solution based on the minimum condition number criterion does not give rise to continuous joint angle movement. While the joint angle trajectory obtained using the minmum angle criterion is unique for a given task-space trajectory, the joint angle trajectory obtained using the lazy arm criterion depends on the current robot configuration. If all the joint angles of the manipulator are reset to zero prior to robot movement, then the solution for lazy arm movement converges to that of the minimum angle norm movement. Different criteria give rise to different trajectories in the configuration space as shown in Fig. 15. This figure shows the solutions obtained using the lazy arm and minimum angle criteria. The purpose of this figure is to show that different criteria lead to different solution trajectories.

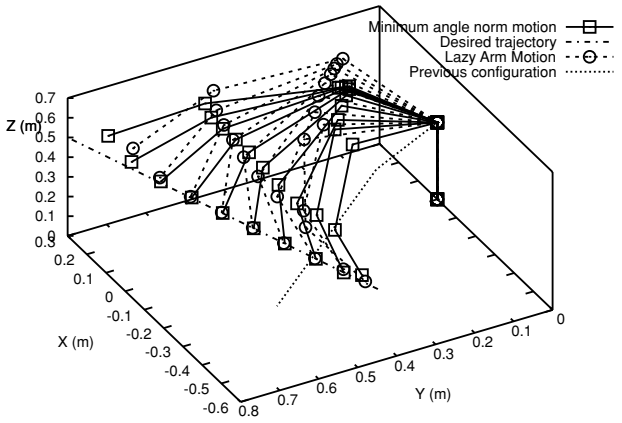


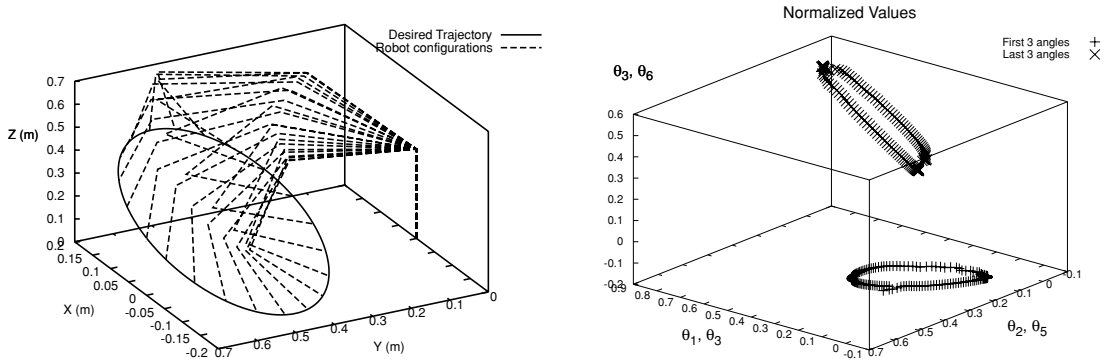
Fig. 15. Redundancy resolution while tracking a straight line trajectory. The minimum angle norm (MA) solution is independent of the initial configuration. The lazy arm (LA) solution depends on the initial condition.

5.3.3. Tracking an elliptical trajectory. The desired trajectory to be traversed is given by

$$x = 0.2 \sin t,$$
$$y = 0.5 + 0.2 \cos t,$$
$$z = \frac{5}{6}(x + 0.3), \tag{35}$$

Table IV. Tracking a straight line trajectory: LA, lazy arm; MA, minimum angle norm; MC, minimum condition number.

	Average positioning error			
Criterion	Cartesian space (mm)	Image space (pixels)	Angle norm (normalized)	Learning parameters (η_w, η_t, η_a, σ)
SC + LA	2.91	0.75	0.89	0.9, 0.9, 0.9, 1.4
SC + MA	2.70	0.63	0.77	0.9, 0.9, 0.9, 1.4
SC + MC	34.78	9.74	1.361	0.9, 0.9, 0.9, 1.4



(a) Robot configuration

(b) Joint angle trajectories

Fig. 16. Tracking an elliptical trajectory using lazy arm movement. The inverse kinematic solution is conservative in the sense that a closed loop trajectory in the task space gives rise to a closed trajectory in the configuration space.

where t varies from 0 to 2π. A total of 628 points are generated sequentially on this trajectory; the joint angles are computed in one step for each point. A typical trajectory obtained using the lazy arm criterion and sub-clustering technique is shown in Fig. 16(a). The corresponding trajectories for joint angles are shown in Fig. 16(b). This reaffirms our previous assertion that the inverse kinematic solution obtained is conservative. The performance comparison for different schemes is provided in Table V. Similar references can be drawn from this table as was done in case of a straight line trajectory.

5.4. Real-time experiment

The actual set-up used for experiment is shown in Fig. 17. The Cartesian workspace visible by both cameras has a dimension of 600 mm × 600 mm × 500 mm. The image frame has a dimension of 320 × 240 pixels. The initial locations of the robot end-effector and the target in the image plane are shown in Fig. 18(a). The regions of interest are extracted using thresholding and filtering operations. The centroid of the region is used by the VMC algorithm to compute necessary joint angles. These joint angles are applied to the robot and it moves to a position as shown in Fig. 18(b). This figure shows the final state of manipulator obtained after robot movement. The error is computed after making corrections for the pixel width of the robot end-effector as well as the target object. The accuracy in detecting tip position is further improved by using LEDs against a dark background. All image processing tasks were carried out using the OpenCV library (Open Source Computer Vision Library, http://www.intel.com/technology/computing/opencv/). The algorithm was implemented using Visual C on a computer

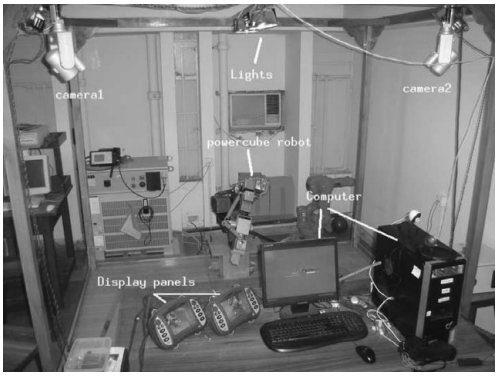


Fig. 17. Experimental set-up for the VMC experiment.

with Intel Pentium 4 1.8-GHz processor. The cameras were calibrated using Reg Wilson's C implementation of Tsai's algorithm .[27]

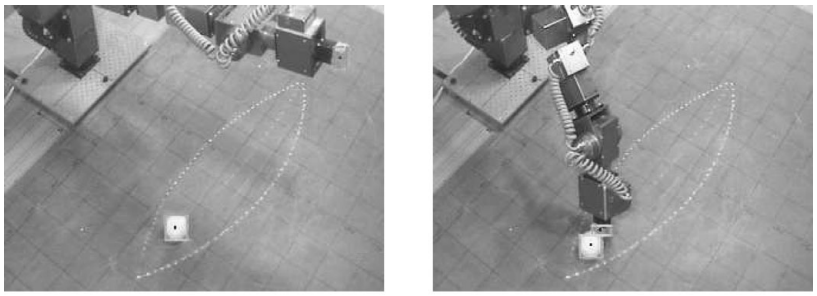
The proposed scheme can be implemented online as was done by Schulten.[20,26] However we used a hybrid approach as suggested by Behera *et al.*[5], where the SOM network is trained offline by generating data from the model rather than from the actual system. This reduces the demand on online data generation.

The trained network was used online to compute joint angle vectors for 20 random locations in the manipulator workspace. Since it was not possible to accurately measure the manipulator tip positions in world coordinates, the distance error was measured directly in pixel coordinates. Only one fine step was used to compute the necessary joint angle vector for each point. The average distance error in image plane is computed to be 12 pixels. This error can be reduced by taking multiple fine steps. It takes 15 steps, on

Table V. Tracking an elliptical trajectory: LA, lazy arm; MA, minimum angle norm; MC, minimum condition number.

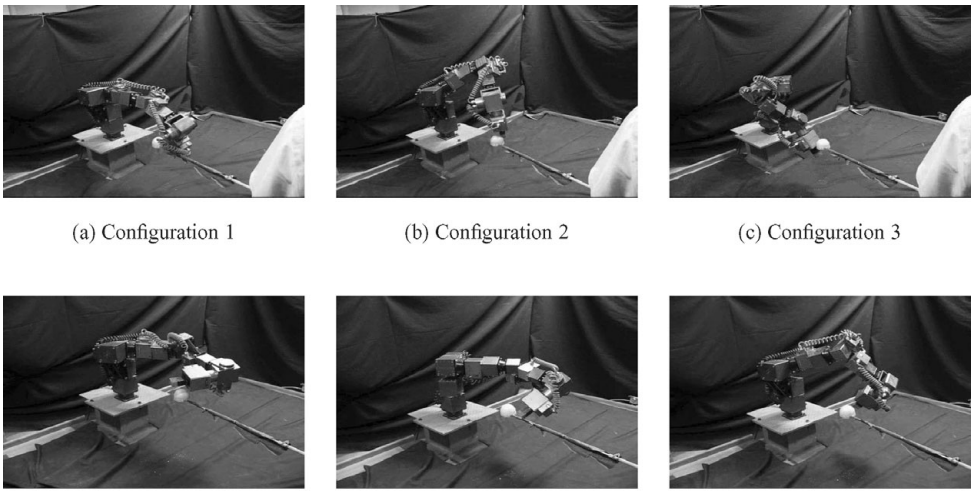
	Average positioning error			
Criterion	Cartesian space (mm)	Image space (pixels)	Angle norm (normalized)	Learning parameters (η_w, η_t, η_a, σ)
SC + LA	1.44	0.42	1.0	0.9, 0.9, 0.9, 2.0
SC + MA	1.28	0.42	0.82	0.9, 0.9, 0.9, 2.0
SC + MC	22.78	6.82	1.55	0.9, 0.9, 0.9, 2.0

https://doi.org/10.1017/S026357470999049X



(a) Initial State (b) Final State

Fig. 18. Extraction of pixel coordinates for the robot end-effector and the target. The initial state is the state before robot movement. The final state refers to state obtained after robot movement.



(a) Configuration 1 (b) Configuration 2 (c) Configuration 3

(d) Configuration 4 (e) Configuration 5 (f) Configuration 6

Fig. 19. Redundant solutions for a given target position: the target is a ball which is reached by the manipulator using various kinematic configurations. Six such kinematic configurations are shown in this figure.

an average, for reaching an accuracy of about 1 pixel for a given target point in the image plane.

The average duration of one feedback cycle is 80 ms. This includes the time needed for image processing, robot movement and communication between the PC and the robot. Hence the program can process the video at a rate of 12 frames s^{-1}.

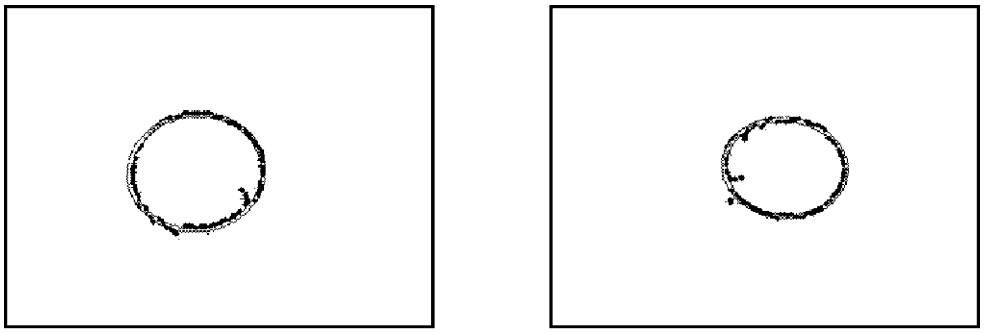
Theoretically, it is possible to attain a positioning accuracy of less than 1 mm in the task space as shown in Section 5.3.1. However in the image plane, the attainable accuracy is about 1 pixel. For a camera of resolution 320×240, an error of 1 pixel in the image plane corresponds to an error of 1 mm to 1 cm in the Cartesian task space. By using higher-resolution cameras, it would be possible to relate each pixel to a smaller range of Cartesian space dimensions. The final positioning accuracy that can be obtained using the scheme does not depend on the size of workspace dimension. It only depends on the resolution of the cameras being used in the experiment. A larger workspace would necessitate greater training for obtaining the same positioning accuracy.

5.4.1. Redundant solutions. Although a redundant manipulator can reach a target point using more than one joint angle configuration, existing learning algorithms can provide only a unique inverse kinematic solution. However, the proposed redundancy preserving network provides multiple solutions simultaneously for any given target position. Figure 19 shows some of these inverse kinematic solutions for a given target position represented by a ball. Readers should be able to appreciate that the ball has been reached by many kinematic configurations. Although the network provides multiple solution, a particular configuration can be selected based on the task requirement.

5.4.2. Tracking a circular and a straight line trajectory. The real-time experimental results for tracking a circular trajectory and a straight line trajectory are shown in Figs. 20 and 21 respectively. The desired trajectories are specified directly on the image plane using a camera model. In Fig. 20, the desired trajectory is shown as a thin line and the actual positions attained by the manipulator end-effector is shown as a thin dark line. In Fig. 21, the desired and actual positions are shown in light gray and dark gray colours respectively. The end-effector position is detected using an LED against a dark background. The average positioning error in both cases is 8 pixels.

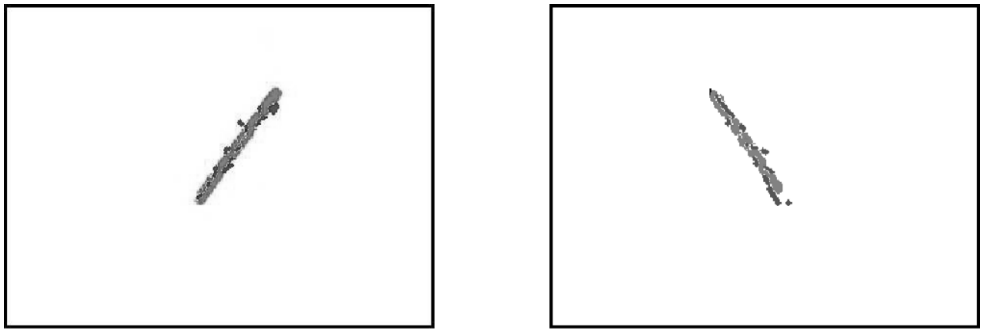
5.4.3. Multi-step movement. In order to improve positioning accuracy of the end-effector, multiple fine movement steps are applied to the manipulator, and the corresponding results are shown in Fig. 22. The first step is a coarse movement as

https://doi.org/10.1017/S026357470999049X



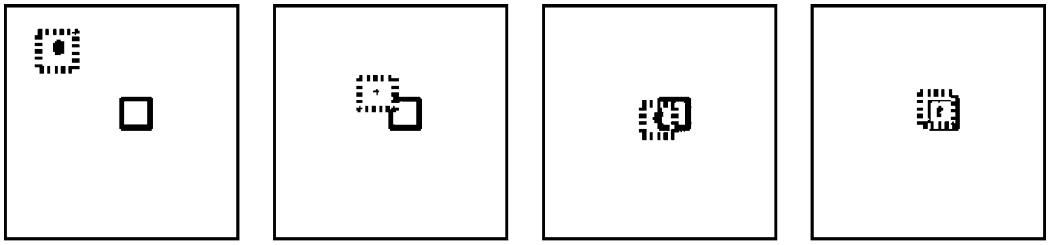
(a) Camera 1 (b) Camera 2

Fig. 20. Tracking a circular trajectory as observed in the image plane. The thin line represents the desired trajectory, and the thick line is the trajectory of the actual end-effector position. The end-effector is detected using an LED in a dark environment. The image has been processed to ensure visibility on paper.



(a) Camera 1 (b) Camera 2

Fig. 21. Tracking a straight line trajectory as observed in the image plane. The desired trajectory is shown in light gray colour and the actual end-effector trajectory is shown in dark gray colour. The end-effector is detected using an LED against a dark background.



(a) Initial State (b) Coarse Movement (c) Fine Movement - Step 5 (d) Fine Movement - Step 15

Fig. 22. Multi-step movement: The rectangular box with solid boundary is the target specified on the image plane that is to be reached. The current end-effector position is shown with a box with dashed boundary. (a) The *initial state*, with the end-effector LED as a black dot. (b) The state after coarse movement. (c) The end-effector position after 5 fine movement steps. (d) The final end-effector position obtained after 15 fine movement steps. The final positioning error is less than 1 pixel. The images have been processed to ensure visibility on paper.

shown in Fig. 22(b). The positioning error is approximately 15 pixels after the coarse movement. This error has been further reduced to 7 pixels and 1 pixel respectively after 5 and 15 fine movements as shown in Figs. 22(c) and 22(d).

6. Conclusion

This paper shows that existing SOM-based VMC algorithms are inefficient for applications in redundant manipulators. The existing learning architectures do not preserve topology of the output space (refer to Fig. 5). Thus such algorithms become sensitive to initial network parameters as shown in Fig. 12. Since existing learning architectures do not preserve redundancy, the redundant manipulator cannot perform dexterous tasks using these VMC algorithms.

Thus an SOM-based redundancy preserving network has been proposed in this paper by using the concept of sub-clustering in joint angle space so that network can yield several kinematic configurations to reach the same target. This is the first learning architecture that has been proposed to preserve redundancy. A real-time algorithm to learn network parameters has been proposed. Since each lattice neuron is associated with multiple solutions in joint angle space, an online adaptive clustering algorithm has been proposed to learn these joint angle vectors. It is shown that this adaptive

https://doi.org/10.1017/S026357470999049X

sub-clustering in the output space leads to the preservation of topology in both the input and the output space by the SOM lattice neurons. It is also shown that the proposed SOM network is insensitive to the initial network parameters unlike the standard SOM network.

Three criteria, namely lazy arm movement, minimum angle norm movement and minimum condition number of Jacobian matrices, are used to resolve redundancy. Apart from providing dexterity, it is shown that the proposed scheme provides the best positioning accuracy as compared to standard SOM-based schemes. Finally, simulation results are validated through experiments on a 7DOF PowerCube robot manipulator.

References

1. V. R. Angulo and C. Torras, "Speeding up the learning of robot kinematics through function decomposition," *IEEE Trans. Neural Networks* **16**(6), 1504–1512 (Nov. 2005).
2. G. A. Barreto, A. F. R. Araujo and H. J. Ritter, "Self-organizing feature maps for modeling and control of robotic manipulators," *J. Intell. Rob. Syst.* **36**, 407–450 (2003).
3. L. Behera and N. Kirubanandan, "A hybrid neural control scheme for visual-motor coordination," *IEEE Control Syst. Mag.* **19**(4), 34–41 (1999).
4. F. Chaumette, "Image moments: A general and useful set of features for visual servoing," *IEEE Trans. Rob.* **20**(4), 713–723 (Aug. 2004).
5. F. Chaumette and E. Marchand, "A redundancy-based iterative approach for avoiding joint limits: Application to visual servoing," *IEEE Trans. Rob. Automat.* **17**(5), 719–730 (Oct. 2001).
6. J. T. Feddema, C. S. George Lee and O. W. Mitchell, "Weighted selection of image features for resolved rate visual feedback control," *IEEE Trans. Rob. Automat.* **7**(1), 31–47 (Feb. 1991).
7. M. Han, N. Okada and E. Kondo, "Coordination of an uncalibrated 3-d visuo-motor system based on multiple self-organizing maps," *JSME Int. J. Ser. C* **49**(1), 230–239 (2006).
8. S. Hutchinson, G. D. Hager and P. I. Corke, "A tutorial on visual servo control," *IEEE Trans. Rob. Automat.* **12**(5), 651–670 (Oct. 1996).
9. P. Jiang, L. C. A. Bamforth, Z. Feng, J. E. F. Baruch and Y. Q. Chen, "Indirect iterative learning control for a discrete visual servo without a camera-robot model," *IEEE Trans. Syst. Man Cybernet. Part B: Cybernet.* **37**(4), 863–876 (Aug. 2007).
10. T. Kohonen, *Self Organization and Associative Memory* (Springer-Verlag, Berlin, Germany, 1984).
11. D. Kragic and H. I. Christensen, Survey on Visual Servoing for Manipulation *Technical Report* (Stockholm, Sweden: Computational Vision and Active Perception Laboratory, KTH, 2002).
12. N. Kumar and L. Behera, "Visual motor coordination using a quantum clustering based neural control scheme," *Neural Process. Lett.* **20**, 11–22 (2004).
13. S. Kumar and L. Behera, "Implementation of a Neural Network Based Visual Motor Control Algorithm for a 7 dof Redundant Manipulator," *International Joint Conference on Neural Networks (IJCNN)*, Hong Kong, China (June 2008) pp. 1344–1351.
14. S. Kumar, N. Patel and L. Behera, "Visual motor control of a 7 dof robot manipulator using function decomposition and sub-clustering in configuration space," *Neural Process. Lett.* **28**(1), 17–33 (Aug. 2008).
15. L. Li, W. A. Gruver, Q. Zhang and Z. Yang, "Kinematic control of redundant robots and the motion optimizability measure," *IEEE Trans. Syst. Man Cybernet. Part B: Cybernet.* **31**(1), 155–160 (Feb. 2001).
16. Y. Li and S. H. Leong, "Kinematics control of redundant manipulators using a CMAC neural network combined with a genetic algorithm," *Robotica* **22**, 611–621 (2004).
17. T. Martinetz, H. Ritter and K. Schulten, "Learning of visuomotor-coordination of a robot arm with redundant degrees of freedom," In *Proceedings of the International Conference on Parallel Processing in Neural Systems and Computers (ICNC)*, (Elsevier, Dusseldorf and Amsterdam 1990) pp. 431–434.
18. T. M. Martinetz, H. J. Ritter and K. J. Schulten, "Three-dimensional neural net for learning visual motor coordination of a robot arm," *IEEE Trans. Neural Networks* **1**(1), 131–136 (Mar. 1990).
19. R. I. V. Mayorgaa and P. Sanongboone, "Inverse kinematics and geometrically bounded singularities prevention of redundant manipulators: An artificial neural network approach," *Rob. Auton. Syst.* **53**, 164–176 (2005).
20. R. Sharma and S. Hutchinson, "Optimizing Hand/Eye Configuration for Visual-Servo Systems," *Proceedings of the International Conference on Robotics and Automation (ICRA)*, Nagoya, Japan (May 1995) pp. 172–177.
21. M. W. Spong and M. Vidyasagar, *Robot Dynamics and Control*, New York, USA (John Wiley, 1989).
22. G. Tevatia and S. Schaal, "Inverse Kinematics of Humanoid Robots." *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA (Apr. 2000) pp. 294–299.
23. R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE J. Rob. Automat.* **RA-3**(4), 323–344 (Aug. 1987).
24. J. A. Walter and K. J. Schulten, "Implementation of self-organizing neural networks for visual-motor control of an industrial robot," *IEEE Trans. Neural Networks* **4**(1), 86–95 (Jan. 1993).
25. R. Wilson, "Tsai Camera Calibration Software," available at http://www.cs.cmu.edu/ rgw/TsaiCode.html.
26. Y. Xia and J. Wang, "A dual neural network for kinematic control of redundant robot manipulators," *IEEE Trans. Syst. Man Cybernet. Part B: Cybernet.* **31**(1), 147–154 (Feb. 2001).
27. H. Zha, T. Onitsuka and T. Nagata, "A self-organization learning algorithm for visuo-motor coordination in unstructured environment," *Artif. Life Rob.* **1**(3), 131–136 (Sep. 1997).
28. X.-Z. Zheng and K. Ito, "Self-organized learning and its implementation of robot movements," *IEEE International Conference on SMC, "Computational Cybernetics and Simulation*," Orlando, FL (1997) pp. 281–286.