Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Inverse Kinematic Control Using Rotational And Joint Space Clustering With Visual Motor Coordination [⋆]

Anjan Kumar Ray [*] **Laxmidhar Behera** [**]

[*] *Department of Electrical Engineering, Indian Institute of Technology, Kanpur, 208016, UP, INDIA (Tel: +91-512-259-7854; e-mail: akray@iitk.ac.in)*
[**] *Department of Electrical Engineering, Indian Institute of Technology, Kanpur, 208016, UP, INDIA (Tel: +91-512-259-7854; e-mail: lbehera@iitk.ac.in and School of Computing and Intelligent System, University of Ulster, U.K., e-mail:l.behera@ulster.ac.uk)*

Abstract: In this paper, the inverse kinematic control of a 6-DOF robot manipulator is achieved using visual motor coordination (VMC). Here the positional data is converted into image plane data of a pair of cameras. The Redundancy resolution is a prime goal for the robot manipulator with higher dimensional joint space than the task-space. In this work, we present five schemes for this redundancy resolution based on hybrid visual motor co-ordination (VMC) for a 6-dof robot manipulator by clustering the rotational space and joint space information with visual feedback from a pair of cameras. The proposed schemes are used with the extended Kohonen's Self Organizing Map (EKSOM) to find out the mapping from 3-dimensional positional task space to the 6-dimensional joint space of the manipulator. The neural network with EKSOM is modified to use the cyclic nature of angular displacement of joints. The visual feedback is obtained through a pair of calibrated cameras. So, each positional data is converted to corresponding camera coordinates and then the modified EKSOM has been trained to obtain the input-output mapping by combining the visual feedback and hybrid system model consisting of forward kinematics of the manipulator. These methods produce smooth joint movements for positional tracking. These schemes are successfully implemented on a model of 6-DOF $PowerCube^{TM}$ robot manipulator from Amtec Robotics.

1. INTRODUCTION

Visual Motor Coordination (VMC) is the process of using visual feedback from a camera system to control a robot manipulator to reach a target point in its workspace. It is similar to the hand-eye coordination of Human being. This involves the process of finding the image coordinates of the robot end-effector as well as the target points and the corresponding joint angles of the manipulator. For this, a priori knowledge of camera model, robot kinematics and dynamics are required. These require the exact knowledge of those models which are quite uncertain in the field of robotics. However, the calibration parameters of the camera model can be estimated through camera calibration technique (Tsai [1987], Horn [2000]). Moreover, finding the exact inverse kinematics model of a robot manipulator is also a complex process (Buss [2004]). In case of redundant robot, where the dimension of joint-space is greater than the robot task-space, the problem spreads out. There will be multiple solution sets of joint angles to reach a particular target point. The problem is to select a feasible solution which also support the physical limits and other constraints of the manipulator. Redundancy resolution is a process to handle this kind of multi solution problem (Patel and Shadpey [2005]). There are several methods available to solve the redundancy resolution problem such as Jacobian transpose, pseudo-inverse, damped-least square (Buss [2004]), configuration control (Seraji [1989]), successive approximation based technique (Goran S. [1999]).

Learning and acquisition paradigm can be used to form a mapping between the task-space and the joint space. In this learning scheme, the manipulator is trained to place its end-effector to a desired location by using visual data. This extracts the mapping existed between the camera output and the desired end-effector position. Various neural network models have been developed which apply biologically inspired control mechanisms for this coordination control task. Kuperstein's (Kuperstein [1987, 1988]) model is an early contribution to the application of topology-conserving maps to visual-motor co-ordination. Ritter, Martinetz and Schulten (T.M. Martinetz and Schulten [1990]) have improved on Kuperstein's model by considering a more general model based on Kohonen's self-organizing scheme. Further modification has been done by Walter and Schulten (Walter and Schulten [1993]) using the visual feedback from the camera for fine tuning of the manipulator's joint space variables.

In this present work, a general framework for kinematic control has been presented using visual feedback. A model of 6-DOF $PowerCube^{TM}$ manipulator is considered to study the

[⋆] This work was supported by Department of Science and Technology (DST), Govt. Of India under the project titled "Intelligent Control Schemes and application to dynamics and visual control of redundant manipulator systems". The project number is DST/EE/20050331.

applicability of the proposed schemes. In the hybrid network approach, we take the forward kinematics model and a neural network based on Extended Kohonen's Self-Organizing Map (EKSOM). The update of neural network is so planned that will always keep the joint-angles within the limits of $[-\pi \ \pi]$. The redundancy of the manipulator has been resolved using the rotational elements of the end-effector frame. We map 9-dimensional rotational space cluster to 6-dimensional joint space during the training. Among the 9-dimensional input space, first 4 elements are from image plane data corresponding to the position in the workspace and rest 5 elements are random rotational elements. We also use the joint space information during training to form a 10-dimensional joint space cluster whose first 4 elements correspond to image plane data and rest 6 elements are joint space elements. The trained network successfully maps the position to the joint space and hence capable of finding the inverse kinematics of the manipulator. The simulation results are shown to validate these training schemes.

In the section 2 we present the forward kinematics model of a 6-DOF $PowerCube^{TM}$ manipulator along with visual motor coordination setup. In the section 3, a modified EKSOM is presented which is the main building block of neural network training. The rotational space clustering and joint space clustering schemes are presented in section 3.2 and 3.3. The simulation results are presented in the section 4 followed by the section 5 which depicts the overall conclusion of these processes.

2. MODELING OF SYSTEMS

2.1 Problem of inverse kinematics

The forward and inverse kinematics of a manipulator are given by mappings

$$S = f(\boldsymbol{\theta}) \tag{1}$$

$$\boldsymbol{\theta} = f^{-1}(S) \tag{2}$$

where, f is the mapping, $\boldsymbol{\theta}$ and $S = [X \ Y \ Z]^T$ are joint space and workspace respectively. Finding the inverse kinematics become difficult for manipulators with more than 3-DOF as it becomes redundant with 3-dimensional positional data. However, we can include the orientation data $O = [\theta_x \ \theta_y \ \theta_z]^T$ to solve for the redundancy for a manipulator with upto 6-DOF where the mapping will be from $[S^T \ O^T]^T \rightarrow \boldsymbol{\theta} = [\theta_1 \ \theta_2 \ \theta_3 \ \theta_4 \ \theta_5 \ \theta_6]^T$. Manipulators with more than 6-DOF become redundant where extra task has to be considered to resolve the redundancy. In this present work, we have addressed the issue of finding the inverse kinematics of manipulator whose orientation data are not available and then resolved the redundancy. The visual-motor coordination for a robot manipulator system consists of a pair of cameras and a robot manipulator. The objective of this method is to find out the inverse kinematics and then place the end-effector of the manipulator to a desired position using the camera data.

2.2 Robot model

In this present work, we consider a 6-DOF $PowerCube^{TM}$ robot manipulator whose D-H parameters are shown in Table 1,

where, the subscripted parameters' values are $a_2 = 0.370$ m, $d_1 = 0.390$ m, $d_4 = 0.310$ m, $d_6 = 0.2656$ m.

Table 1. D-H parameters of $PowerCube^{TM}$

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	d_1	θ_1
2	90^o	0	0	θ_2
3	0	a_2	0	θ_3
4	-90^o	0	d_4	θ_4
5	90^o	0	0	θ_5
6	-90^o	0	d_6	θ_6

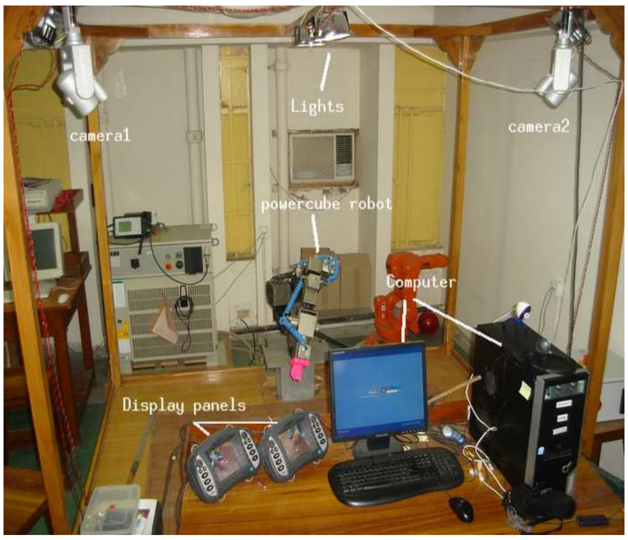

Fig. 1. Experimental setup with stereo-vision cameras and $PowerCube^{TM}$

The position of the end-effector with respect to its base is specified by $S = [X \ Y \ Z]^T$ and the rotational frame is given by the rotational matrix R. The end effector position with respect to end-effector frame $[x_e \ y_e \ z_e]^T$ and rotation is related to the base $[x_b \ y_b \ z_b]^T$ of the manipulator by the following equation,

$$\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = R \begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix} + \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3}$$

where, R is the rotational matrix given by

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{4}$$

The forward kinematics is given by the following (5)

$$\begin{aligned} X =& \ d_4(-c_1c_2s_3 - c_1c_3s_2) + a_2c_1c_2 + d_6(-c_1c_2c_3c_4s_5 \\ &+c_1c_4s_2s_3s_5 + s_1s_4s_5 - c_1c_2c_5s_3 - c_1c_3c_5s_2) \\ Y =& \ d_4(-s_1s_3c_2 - s_1s_2c_3) + a_2s_1c_2 + d_6(-s_1s_5c_2c_3c_4 \\ &+s_1s_2s_3s_5c_4 - c_1s_4s_5 - s_1s_3c_2c_5 - s_1s_2c_3c_5) \\ Z =& \ d_4(-s_2s_3 + c_2c_3) + a_2s_2 + d_1 + d_6(-s_2s_5c_3c_4 - \\ &c_2c_4s_3s_5 - s_2s_3c_5 + c_2c_3c_5) \end{aligned} \tag{5}$$

where, $c_i = cos\theta_i$, $s_i = sin\theta_i$, $c_ic_j = cos\theta_i cos\theta_j$, $s_is_j = sin\theta_i sin\theta_j$, $c_is_j = cos\theta_i sin\theta_j$, $s_ic_j = sin\theta_i cos\theta_j$. $0 < i < 7$, $0 < j < 7$ are integers.

2.3 Visual motor coordination setup

The experimental setup is shown in Fig. 1. It consists of a $PowerCube^{TM}$ robot manipulator, proper lighting system, computational unit and a pair of $Ca - Zoom^{TM}$ PTZ cameras. The cameras have color image resolution of 320 X 240 pixels. Tsai algorithm (Tsai [1987]) is used to calibrate the cameras. For details of the calibration parameters and their definitions

Table 2. Extrinsic and Intrinsic camera parameters

Parameters	Camera 1	Camera 2
f (mm)	5.436145	4.378799
κ (mm^{-2})	4.590897 X 10^{-3}	2.696366 X 10^{-2}
T_x (mm)	-468.595155	543.932076
T_y (mm)	-245.714726	-356.236535
T_z (mm)	2116.318683	2021.321783
R_x (deg)	-157.688170	-161.992870
R_y (deg)	30.899335	-30.746348
R_z (deg)	130.253939	-127.867017
s_x	0.876342	0.974713
C_x (pixel)	173.287243	117.929013
C_y (pixel)	114.979411	136.855416

please refer to Tsai [1987]. The intrinsic and extrinsic camera calibration parameters are found out using the data as shown in Table 2 where,
Ncx=no. of sensor elements in camera's x direction=320 (sel)
Nfx=no. of pixels in frame grabber's x direction=320 (pixel)
dx=X dimension of camera's sensor element=0.0097 (mm/sel)
dy=Y dimension of camera's sensor element=0.01 (mm/sel)
dpx=Effective X dimension of pixel in frame grabber=0.0097 (mm/pixel)
dpy=Effective Y dimension of pixel in frame grabber=0.01 (mm/pixel).
From the two calibrated cameras we get the image coordinates of the target point $S = [X\ Y\ Z]^T$ as $[u_1\ u_2]$ and $[u_3\ u_4]$ respectively.

3. NEURAL NETWORK TRAINING USING MODIFIED EKSOM

3.1 Learning algorithm

The learning algorithm to find the inverse kinematics is based on extended Kohonen's Self-Organizing Feature Maps which was introduced by Walter and Schulten [1993]. This method was presented for a 3-dof robot manipulator by Behera and Kirubanandan [1999] and Behera and Kumar [2004]. Here, we consider a 3-dimensional neural lattice where each neuron has three fields 'weight vector' w_r, 'Jacobean matrix' A_r and 'output vector' θ_r where subscript r stands for its position in the lattice. From each positional data S we can get a pair of camera coordinates $[u_1\ u_2]$ and $[u_3\ u_4]$ from camera 1 and 2 respectively. These form the visual target as

$$u_{target} = [u_1\ u_2\ u_3\ u_4]^T \tag{6}$$

But for the case of 6-dof manipulator, it is difficult to find out the inverse kinematics with 4-dimensional u_{target} or 3-dimensional positional target S. Here, we have redundant solutions as there may be multiple solutions of $\boldsymbol{\theta}$. So, the additional tasks have to be considered for resolving the redundancy. For each training step, a target position u_{target} is clubbed with a function $g(\boldsymbol{\theta})$ to form an input space of the network as

$$\boldsymbol{ip}_{target} = [u_{target}^T\ g(\boldsymbol{\theta})]^T \tag{7}$$

This $\boldsymbol{ip}_{target}$ is presented at randomly and the corresponding $\boldsymbol{\theta}(ip_{target}) = (\theta_1\ \theta_2\ \theta_3\ \theta_4\ \theta_5\ \theta_6)^T$ are found out using a hybrid model consisting of forward kinematics model, camera model and the defined function $g(\boldsymbol{\theta})$. The output of the network is obtained based on 'winner-takes-all' scheme which produces the joint angles $\boldsymbol{\theta}$ by the Taylor series expansion of $\boldsymbol{\theta}(\boldsymbol{ip}_{target})$, i.e.,

$$\boldsymbol{\theta}(\boldsymbol{ip}_{target}) = \boldsymbol{\theta}_\mu + A_\mu(\boldsymbol{ip}_{target} - w_\mu) \tag{8}$$

where, μ denotes the subscript of the winner neuron based on the Euclidean distance metric in the workspace by (9).

$$||w_\mu - \boldsymbol{ip}_{target}|| = \min_{\forall\rho} ||w_\rho - \boldsymbol{ip}_{target}|| \tag{9}$$

In modified EKSOM, we consider the collective output of the neighbor neurons along with that winning neuron. The neighborhood of winner neuron is chosen by (10).

$$hr_1(r) = exp\left(-\frac{||r-\mu||^2}{2\sigma_1^2}\right) \tag{10}$$

$$\sigma_1 = \sigma_{1initial}\left(\frac{\sigma_{1final}}{\sigma_{1initial}}\right)^{\left(\frac{t_1}{t_{max}}\right)} \tag{11}$$

where, $\sigma_{1initial}$ and σ_{1final} stand for the neighbourhood width parameter σ_1's initial and final values. t_1 is the current number of training step and t_{max} is the maximum number of training step. The manipulator is given a coarse movement $\boldsymbol{\theta}_0^{out}$ which is the network output that moves the end-effector to a position

$$\boldsymbol{ip}_0 = [u_{target}^{0T}\ g(\boldsymbol{\theta}_0^{out})]^T. \tag{12}$$

This is followed by some fine movements given by $\boldsymbol{\theta}_{n_1}^{out}$ which brings the end-effector to $\boldsymbol{ip}_{n_1}$. The following equations depict the process of averaged output to calculate $\boldsymbol{\theta}_0^{out}$ and $\boldsymbol{\theta}_{n_1}^{out}$ where n_1 stands for number of fine movements.

$$\boldsymbol{\theta}_0^{out} = s^{-1}\sum_r hr_1(r)\left(\boldsymbol{\theta}_r + A_r\left(\boldsymbol{ip}_{target} - w_r\right)\right) \tag{13}$$

$$\boldsymbol{\theta}_i^{out} = \boldsymbol{\theta}_{i-1}^{out} + s^{-1}\sum_r hr_1(r)A_r\left(\boldsymbol{ip}_{target} - \boldsymbol{ip}_{i-1}\right) \tag{14}$$

where, $s = \sum_r hr_1(r)$, $\boldsymbol{\theta}_i^{out} = [\theta_1\ \theta_2\ \theta_3\ \theta_4\ \theta_5\ \theta_6]_i^{Tout}$ and $i = 1, 2, ...n_1$. It is observed that, with this condition it may happen that $\boldsymbol{\theta}_i^{out}$ becomes unbounded during training. So, to keep the $\boldsymbol{\theta}_i^{out}$ within the range of $\pm 2\pi$ the following corrective measures have been taken.

$$\begin{aligned} &\textit{if}\ \ \theta_j > 0 \\ &\quad \theta_j = \textit{mod}(\theta_j, 2\pi) \\ &\textit{if}\ \ \theta_j < 0 \\ &\quad \theta_j = \textit{mod}(\theta_j, -2\pi) \end{aligned} \tag{15}$$

where, $j = 1, 2, ...6$. θ_j is the element of $\boldsymbol{\theta}_i^{out}$ and the function $mod(a, b)$ is the remainder of the division a by b. This rule will ensure that all element of $\boldsymbol{\theta}_i^{out}$ will stay within the range $\pm 2\pi$ as well as this will enforce a smooth movements of joints. It can be modified further to make the range within $\pm\pi$ as follows

$$\begin{aligned} &\textit{if}\ \ \theta_j > 0 \\ &\theta_j = \begin{cases} \theta_j & ; 0 \le \theta_j \le \pi \\ \theta_j - 2\pi & ; \pi < \theta_j < 2\pi \end{cases} \\ &\textit{if}\ \ \theta_j < 0 \\ &\theta_j = \begin{cases} \theta_j & ; 0 \ge \theta_j \ge -\pi \\ \theta_j + 2\pi & ; -\pi > \theta_j > -2\pi \end{cases} \end{aligned} \tag{16}$$

The neural units are adjusted by the following update rules:

$$w_r \leftarrow w_r + \epsilon hr_2(\boldsymbol{ip}_{target} - w_r) \tag{17}$$

$$\boldsymbol{\theta}_r \leftarrow \boldsymbol{\theta}_r + \epsilon_1 hr_3 \Delta\boldsymbol{\theta}_r \tag{18}$$

$$A_r \leftarrow A_r + \epsilon_1 hr_3 \Delta A_r \tag{19}$$

The update rule for $\boldsymbol{\theta}_r$ is corrected using (15), (16) where, θ_j is the element of $\boldsymbol{\theta}_r$. The ΔA_r is found using a stochastic gradient descent approach to minimize the quadratic cost function

$$E = \frac{1}{2}\left(\Delta\boldsymbol{\theta}_{0n_1}^{out} - A_r\Delta\boldsymbol{ip}\right)^2 \tag{20}$$

The quantities ΔA_r, $\Delta \boldsymbol{ip}$, $\Delta \boldsymbol{\theta}_{0n_1}^{out}$ and $\Delta \boldsymbol{\theta}_r$ are found out using the rules as follows:

$$\Delta A_r = ||\Delta \boldsymbol{ip}||^{-2} \left(\Delta \boldsymbol{\theta}_{0n_1}^{out} - A_r \Delta \boldsymbol{ip}\right) \Delta \boldsymbol{ip}^T \tag{21}$$

$$\Delta \boldsymbol{ip} = \boldsymbol{ip}_{n_1} - \boldsymbol{ip}_0 \tag{22}$$

$$\Delta \boldsymbol{\theta}_{0n_1}^{out} = \boldsymbol{\theta}_{n_1}^{out} - \boldsymbol{\theta}_0^{out} \tag{23}$$

$$\Delta \boldsymbol{\theta}_r = \boldsymbol{\theta}_0^{out} - \boldsymbol{\theta}_r - A_r \left(\boldsymbol{ip}_0 - w_r\right) \tag{24}$$

In this work, we consider the functions hr_2 and hr_3 to be Gaussian as hr_k, $k = 1,\ 2$ as (25).

$$hr_k = exp\left(-\frac{||r-\mu||^2}{2\sigma_k^2}\right) \tag{25}$$

Also the learning rate parameters ϵ, ϵ_1 and the neighborhood width functions σ_2, σ_3 change during the training process according to a general rule as follows:

$$\eta = \eta_{initial}\left(\frac{\eta_{final}}{\eta_{initial}}\right)^{\left(\frac{t_1}{t_{max}}\right)} \tag{26}$$

where $\eta \in \epsilon, \epsilon_1, \sigma_2, \sigma_3$. In this present work, the initial and final values of the parameters $\epsilon, \epsilon_1, \sigma_2, \sigma_3$ are taken as {1.0, 0.05}, {0.9, 0.9}, {2.5, 0.01} and {2.5, 0.01} respectively.

3.2 *Rotational space clustering*

The facility of the hybrid model is that we can directly calculate the elements of rotational matrix R given a set of joint angles. So, during the process of training, the position can be achieved by the two cameras whereas the rotational elements can be calculated using the system model with random joint angles using (27). Given a system model, these information are readily available and is used to form the training cluster and a mapping is achieved to relate directly the positional data with the joint angles of the manipulator without accounting the orientation. After each movement, joint angles are known from the network output and these joint angles are then used to calculate the next phase rotational elements to form the cluster again with updated values.

$$\begin{aligned}
r_{11} &= c_1c_2c_3c_4c_5c_6 - c_1c_4c_5c_6s_2s_3 - s_1s_4c_5c_6 - c_1c_2c_6s_3s_5 \\
&\quad -c_1c_3c_6s_2s_5 - c_1c_2c_3s_4s_6 + c_1s_2s_3s_4s_6 - s_1s_6c_4 \\
r_{21} &= s_1c_2c_3c_4c_5c_6 - s_1s_2s_3c_4c_5c_6 + c_1c_5c_6s_4 \\
&\quad -s_1s_3s_5c_2c_6 - s_1s_2s_5c_3c_6 - s_1s_4s_6c_2c_3 \\
&\quad +s_1s_2s_3s_4s_6 + c_1c_4s_6 \\
r_{31} &= s_2c_3c_4c_5c_6 + c_2c_4c_5c_6s_3 - s_2s_3s_5c_6 + c_2c_3c_6s_5 \\
&\quad -s_2s_4s_6c_3 - c_2s_3s_4s_6 \\
r_{32} &= -s_2s_6c_3c_4c_5 - c_2c_4c_5s_3s_6 + s_2s_3s_5s_6 - c_2c_3s_5s_6 \\
&\quad -s_2s_4c_3c_6 - c_2c_6s_3s_4 \\
r_{33} &= -s_2s_5c_3c_4 - c_2c_4s_3s_5 - s_2s_3c_5 + c_2c_3c_5
\end{aligned} \tag{27}$$

In this way a fusion of camera data and the system model can be achieved which is an important contribution to the proper training. Otherwise, the choice of random rotational elements associated with a specific position may not be feasible for a physical system. The constraint function $g(\boldsymbol{\theta})$ is formulated as (28)

$$g(\boldsymbol{\theta}) = [r_{11}\ r_{21}\ r_{31}\ r_{32}\ r_{33}] \tag{28}$$

The output space for the network is the 6-dimensional joint space represented by (29).

$$\boldsymbol{\theta} = [\theta_1\ \theta_2\ \theta_3\ \theta_4\ \theta_5\ \theta_6]^T \tag{29}$$

The dimensions of the parameters w_r, A_r and θ_r during training are 9×1, 6×9 and 6×1. This clustered network is trained with (6) through (26).

Case I During the testing phase, the mapping $S \rightarrow u_{target}^T$ is found out using calibrated camera parameters. The initial rotational space $g(\boldsymbol{\theta_0})$ is clubbed together to form initial target $\boldsymbol{ip}_{target} = [u_{target}^T\ g(\boldsymbol{\theta_0})]^T$. From network, we get the actual joint space θ and from (27) we get $g(\boldsymbol{\theta})$ which is clubbed together with successive u_{target}^T to form the input space $\boldsymbol{ip}_{target} = [u_{target}^T\ g(\boldsymbol{\theta})]^T$. The dimensions of w_r, A_r and θ_r during testing phase are 9×1, 6×9 and 6×1.

Case II Here the training process is same as $Case\ I$. But, during the testing phase, we choose 4-dimensional input space $\boldsymbol{ip}_{target} = u_{target}$ and the winner neuron μ is selected by (30)

$$||w_\mu - u_{target}|| = \min_{\forall \rho} ||w_{u\rho} - u_{target}|| \tag{30}$$

The dimensions of w_r, A_r and θ_r are 4×1, 6×4 and 6×1.

3.3 *Joint space clustering*

In joint space clustering, we use available joint information to train the network. Here, we club together the u_{target}^T with the output space $\boldsymbol{\theta}$. So, the actual 10-dimensional input space is defined by (31).

$$\boldsymbol{ip}_{target} = [u_{target}^T\ \boldsymbol{\theta}^T]^T \tag{31}$$

We can utilize this joint space information in the following ways.

Case III : Here the network parameters w_r, A_r and θ_r have dimensionality 10×1, 6×4 and 6×1. The weight vector w_r is defined as $w_r = [w_u^T \theta_r^T]^T$ where w_u^T corresponds to u_{target}^T. The winner is selected by (9). Except for update of network weights w_r using (17), first 4 elements of ip_{target}, ip_{i-1}, ip_0, ip_{n1} and w_r are used to train the network. In this way, this method reduces the dimensionality of the Jacobian matrix A_r to 6×4 and maps directly the image plane target u_{target} to the output space $\boldsymbol{\theta}$. Equation (18) and (24) are not used in the training process to get the update of θ_r, instead it is assigned with the last 6 elements of updated w_r. A_r is updated with 4-dimensional visual feedback.
During the testing phase, we choose 4-dimensional input space $\boldsymbol{ip}_{target} = u_{target}$. The winner neuron μ is selected by (30). The dimensions of w_r, A_r and θ_r during testing phase are 4×1, 6×4 and 6×1.

Case IV : Here the training network parameters w_r, A_r and θ_r have dimensionality 10×1, 6×10 and 6×1. w_r and θ_r are updated seperately. The constraint function $g(\boldsymbol{\theta})$ is formulated as (32) and this clustered network is trained with (6) through (26).

$$g(\boldsymbol{\theta}) = [\theta_1\ \theta_2\ \theta_3\ \theta_4\ \theta_5\ \theta_6] \tag{32}$$

During the testing phase, 4-dimensional input space u_{target}^T is presented corresponding to S. The mapping $S \rightarrow u_{target}^T$ is found out using calibrated camera parameters. The initial joint space θ_0 is clubbed together to form initial target $\boldsymbol{ip}_{target} = [u_{target}^T\ \theta_0^T]^T$. From trained network, we get the actual joint space θ which is clubbed together with successive u_{target}^T to form the input space $\boldsymbol{ip}_{target} = [u_{target}^T\ \theta^T]^T$. The dimensions of w_r, A_r and θ_r during testing phase are 10×1, 6×10 and 6×1.

Case V : Here the training process is same as $Case\ IV$. But, during the testing phase, we choose 4-dimensional input space $\boldsymbol{ip}_{target} = u_{target}$ and the winner neuron μ is selected by (30).

Table 3. RMS error on the image plane in $pixel$

	u_1	u_2	u_3	u_4
case I	1.1613	0.0691	0.2882	0.5543
case II	0.6370	0.2526	0.5310	0.7602
case III	3.7943	1.6007	2.4603	1.9744
case IV	0.3145	0.0597	0.2932	0.2770
case V	0.7254	0.2492	0.6195	0.5445

The dimensions of w_r, A_r and θ_r are 4×1, 6×4 and 6×1. This scheme does not require any initial joint space information. In the following section 4, the simulation results are presented in support of these schemes.

4. SIMULATION RESULTS

In the simulation, the initial and final values of σ_1 are taken as 1 and 0.1 respectively. The 3-D neural network with $8 \times 8 \times 8$ neurons have been trained with 100000 data points. The data points are generated with the joint-space limits as $-30^0 \leq \theta_1 \leq 30^0$, $60^0 \leq \theta_2 \leq 90^0$, $0^0 \leq \theta_3 \leq 40^0$, $-40^0 \leq \theta_4 \leq 40^0$, $0^0 \leq \theta_5 \leq 40^0$, $-40^0 \leq \theta_6 \leq 40^0$. The Fig. 2 shows the 3-dimensional workspace of the robot. The first 4 elements of weight vector w_r corresponding to the image plane u_{target} of the two cameras and the output vector θ_r are shown in Fig. 3 and Fig. 4 respectively. In the Fig. 3 and Fig. 4, trained w_r and θ_r of $Case\ II$, $Case\ III$ and $Case\ V$ are shown. During the training phase 2 visual corrections are used for fine tuning. The first 3 and the last 3 elements of θ_r in $radian$ are shown in the left and right plots of the Fig.4. The figures depicts that all the joint angles are bounded within the range of $\pm 2\pi$. The trained network has been tested to find the inverse kinematics of the 6-DOF robot. For this, a desired circular trajectory is formed as

$$\begin{aligned} X &= -0.3 + 0.15cos(ja) \\ Y &= 0.05 + 0.15sin(ja) \\ Z &= 0.9 \end{aligned} \tag{33}$$

where, X, Y and Z are in m and $0 \leq ja \leq 2\pi$. $Case\ I$ and $Case\ IV$ need initial joint configurations which are taken as 0^0 for all the 6 joints. The circular trajectory tracking are shown in the Fig. 5 for $Case\ II$, $Case\ III$ and $Case\ V$. The corresponding tracking on the image plane are shown in Fig. 6 and the joint angles of the manipulator are shown in Fig. 7, Fig. 8 and Fig. 9. $Case\ II$ and $Case\ V$ have smooth joint movements and also joints are bounded and $CaseIII$ produces bounded joint movements with slight jerks. The RMS pixel errors on image plane as well as RMS errors in workspace in $Case\ III$ are 3.7943 pixel and 0.014 m maximum. But in all the other cases, the RMS pixel errors are less than 1 pixel and RMS workspace errors are less than 0.0028 m. RMS pixel errors on the image plane and RMS positional errors in workspace are tabulated in Table 3 and Table 4 respectively. From the tables, we see that the proposed methods of training produces the inverse kinematics with very small positional errors and also smooth joint movements are achieved for the desired tracking operation. This is the prime requirement for a real time kinematic control of a redundant robot.

5. CONCLUSION

In this work, visual control schemes based on rotational space and joint space clustering are proposed which uses an extended

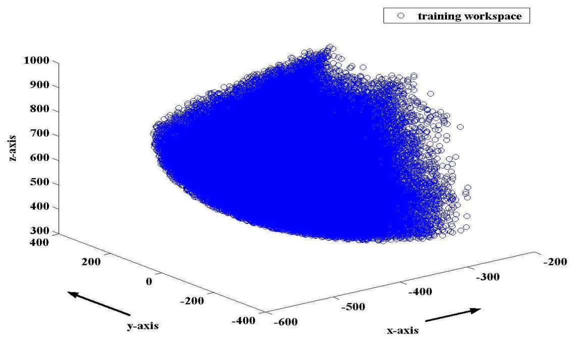

Fig. 2. Training workspace of $PowerCube^{TM}$

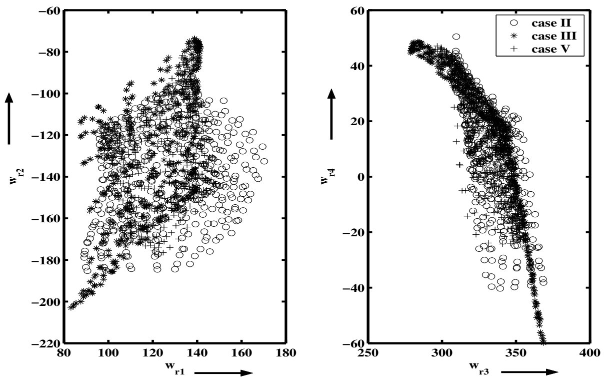

Fig. 3. Trained weight vector w_r corresponding to image plane

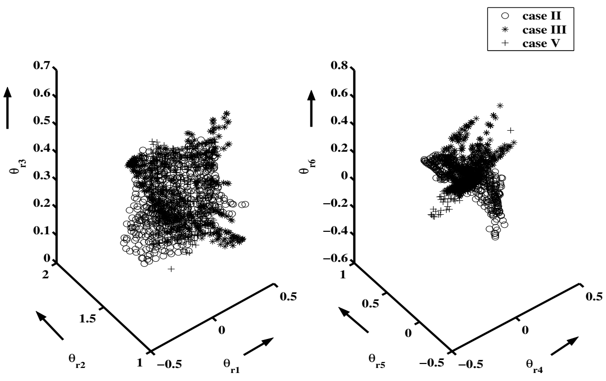

Fig. 4. Trained output vector θ_r in radian

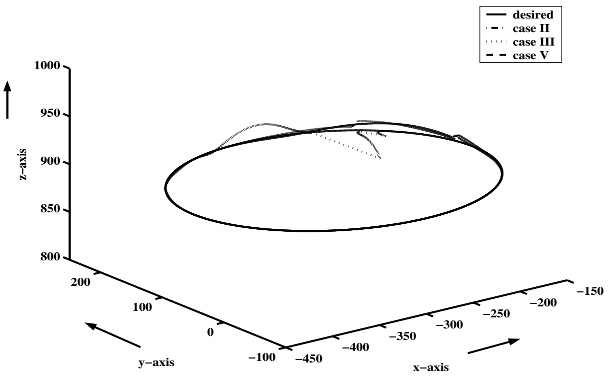

Fig. 5. Tracking of a circular path in the workspace

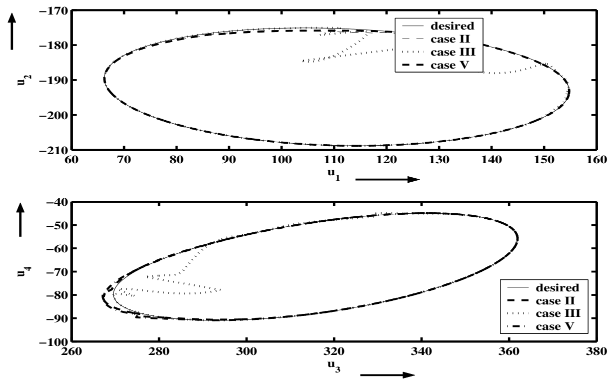

Fig. 6. Circular path tracking on the corresponding image plane

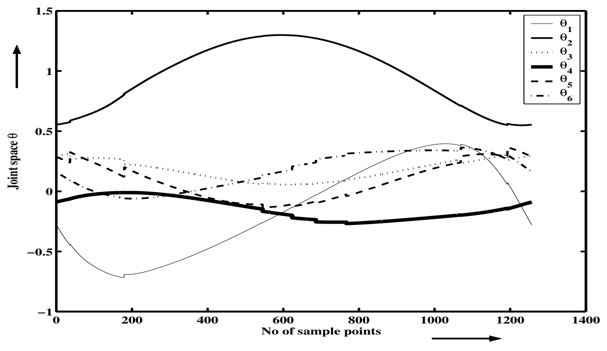

Fig. 7. Joint movements for a circular path tracking in Case II

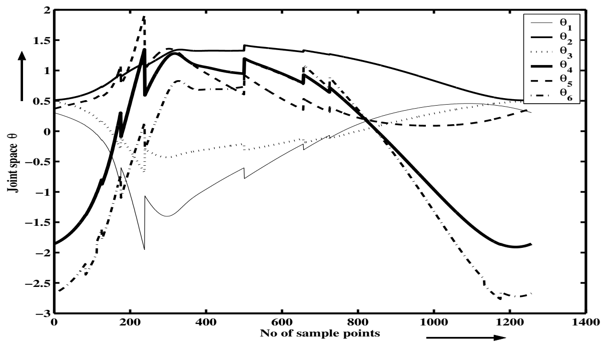

Fig. 8. Joint movements for a circular path tracking in Case III

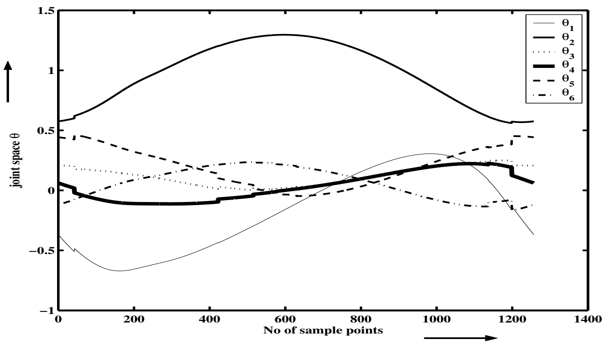

Fig. 9. Joint movements for a circular path tracking in Case V

Table 4. RMS positional error in workspace in m

	X	Y	Z
case I	0.0026	0.0028	0.0005
case II	0.0026	0.0010	0.0016
case III	0.0028	0.0140	0.0038
case IV	0.0012	0.0007	0.0004
case V	0.0018	0.0026	0.0016

Kohonen Self Organizing Map. These methods have been successfully implemented on a 6-DOF $PowerCube^{TM}$ manipulator to achieve its inverse kinematics. The trained networks have been tested for a circular path tracking in the workspace of the manipulator. The simulation results show satisfactory performances. The tracking errors can be further minimized by using more numbers of visual corrections. In this present work, 2 visual corrections are used for training. Here we have achieved smooth bounded joint movements. $Case\ III$ results in larger positional and pixel errors because of the fact that θ_r is not updated by its update rule, instead it is assigned with the last 6 elements of the updated weight vector w_r. The rotational elements are directly related to the orientation of the manipulator. So, $Case\ I$ and $Case\ II$ can be further extended to extract the orientation information. This work assumes a obstacle free environment. In this present work, obstacle avoidance is not considered, which is another way to resolve redundancy.

ACKNOWLEDGEMENTS

We are hereby acknowledge the work of Mr. Swagat Kumar, a Ph.D. scholar of the Department of Electrical Engineering, Indian Institute of Technology, Kanpur who has helped us to get the camera calibration parameters and D-H parameters of 6-DOF $PowerCube^{TM}$ robot manipulator.

REFERENCES

L. Behera and N. Kirubanandan. A hybrid neural control scheme for visual-motor coordination. *IEEE Control Systems Magazine*, 19:34–41, 1999.

L. Behera and N. Kumar. Visual-motor coordination using a quantum clustering based neural control scheme. *Neural Processing Letters*, 20:11–22, 2004.

Samuel R. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. Tech. rep., University of california, San Diego, April 2004.

Dragan K. Goran S., Milan R. Learning of inverse kinematics behavior of redundant robot. In *Proceedings of the 1999 IEEE international conference on robotics and automation*, pages 3165–3170, Detroit, Michigan, May 1999.

Berthold K. P. Horn. Tsai's camera calibration method revisited. Technical report, 2000.

M. Kuperstein. Neural model of adaptive hand-eye coordination for single postures. *Science*, 239:1308–1311, 1988.

M. Kuperstein. Adaptive visual-motor coordination in multijoint robots using parallel architecture. In *Proc. IEEE Int. Automat. Robotics*, pages 1595–1602, Raleigh, NC, 1987.

R. V. Patel and F. Shadpey. *Control of redundant robot manipulators*. Springer, 2005.

Homayoun Seraji. Configuration control of redundant manipulators: Theory and implementation. *IEEE trans. on robotics and automation*, 5(4):472–490, August 1989.

H.J. Ritter T.M. Martinetz and K. Schulten. Three dimensional neural network for learning visuomotor coordination of a robot arm. *IEEE Trans. NN*, 1(1):131–136, 1990.

Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE journal of robotics and automation*, RA-3(4):323–344, August 1987.

A.J. Walter and K.J. Schulten. Implementation of self-organizing neural networks for visuo-motor coordination of an industrial robot. *IEEE Trans. NN.*, 4(1):86–95, 1993.