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™ 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™ robot manipulator is considered to study the
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\textsuperscript{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 which is the main building block of neural network training. In the 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

\[
S = f(\theta) \quad (1)
\]
\[
\theta = f^{-1}(S) \quad (2)
\]

where, \(f\) is the mapping, \(\theta\) and \(S = [X \ Y \ Z]^T\) are joint space and workspace respectively. Finding the inverse kinematics becomes 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 up to 6-DOF where the mapping will be from \([ST \ OT]^T \to \theta = [\theta_1 \ \theta_2 \ \theta_3 \ \theta_4 \ \theta_5 \ \theta_6]^T\). 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\textsuperscript{TM} robot manipulator whose D-H parameters are shown in Table 1.

<table>
<thead>
<tr>
<th>(i)</th>
<th>(\alpha_{i-1})</th>
<th>(a_{i-1})</th>
<th>(d_i)</th>
<th>(\theta_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(\theta_1)</td>
</tr>
<tr>
<td>2</td>
<td>90°</td>
<td>0</td>
<td>0</td>
<td>(\theta_2)</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>(a_2)</td>
<td>0</td>
<td>(\theta_3)</td>
</tr>
<tr>
<td>4</td>
<td>90°</td>
<td>0</td>
<td>(a_4)</td>
<td>(\theta_4)</td>
</tr>
<tr>
<td>5</td>
<td>90°</td>
<td>0</td>
<td>0</td>
<td>(\theta_5)</td>
</tr>
<tr>
<td>6</td>
<td>90°</td>
<td>0</td>
<td>(a_6)</td>
<td>(\theta_6)</td>
</tr>
</tbody>
</table>

Table 1. D-H parameters of PowerCube\textsuperscript{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,

\[
\begin{bmatrix}
  x_b \\
  y_b \\
  z_b
\end{bmatrix} = \begin{bmatrix}
  x_e \\
  y_e \\
  z_e
\end{bmatrix} + \begin{bmatrix}
  X \\
  Y \\
  Z
\end{bmatrix} (3)
\]

where, \(R\) is the rotational matrix given by

\[
R = \begin{bmatrix}
  r_{11} & r_{12} & r_{13} \\
  r_{21} & r_{22} & r_{23} \\
  r_{31} & r_{32} & r_{33}
\end{bmatrix} (4)
\]

The forward kinematics is given by the following (5)

\[
\begin{align*}
X &= d_4(-c_1c_2c_3 - c_1c_2s_3s_2 + a_2c_1c_2 + d_4(-c_1c_2c_3c_4s_5 + c_1c_4s_2s_3c_4s_5 + s_1s_5c_2s_3c_4c_5 - c_1s_2c_5s_3 - c_1c_3c_5s_2)) \\
Y &= d_4(-s_1s_3c_2 - s_1s_2c_3) + a_2s_1c_2 + d_4(-s_1s_5c_2c_3c_4 + s_1s_2s_3s_4c_5 - c_1s_4s_5 - s_1s_2c_5s_4 - s_1s_2c_3c_5)) \\
Z &= d_4(-s_2s_3 + c_2c_3) + a_2s_2 + d_4(-s_2s_3c_4c_5 - c_2c_4s_3s_5 - s_2s_3c_5 + c_2c_3c_5)
\end{align*}
\]

(5)

where, \(c_i = \cos\theta_i, s_i = \sin\theta_i, c_i c_j = \cos\theta_i \cos\theta_j, s_i s_j = \sin\theta_i \sin\theta_j, c_i s_j = \cos\theta_i \sin\theta_j, s_i c_j = \sin\theta_i \cos\theta_j, 0 < i < 7, 0 < j < 7\) are integers.

2.3 Visual motor coordination setup

The experimental setup is shown in Fig. 1. It consists of a PowerCube\textsuperscript{TM} robot manipulator, proper lighting system, computational unit and a pair of C\textsuperscript{a} Zoom\textsuperscript{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

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Camera 1</th>
<th>Camera 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>f (mm)</td>
<td>5.436145</td>
<td>4.378799</td>
</tr>
<tr>
<td>κ (mm⁻²)</td>
<td>4.590897</td>
<td>2.069636</td>
</tr>
<tr>
<td>T_x (mm)</td>
<td>-468.59135</td>
<td>543.92076</td>
</tr>
<tr>
<td>T_y (mm)</td>
<td>-245.714726</td>
<td>-356.236535</td>
</tr>
<tr>
<td>T_z (mm)</td>
<td>2116.318683</td>
<td>117.929013</td>
</tr>
<tr>
<td>R_x (deg)</td>
<td>-157.688710</td>
<td>-161.992870</td>
</tr>
<tr>
<td>R_y (deg)</td>
<td>30.899335</td>
<td>-30.746348</td>
</tr>
<tr>
<td>R_z (deg)</td>
<td>130.253939</td>
<td>-127.867017</td>
</tr>
<tr>
<td>s_x</td>
<td>0.876342</td>
<td>0.974713</td>
</tr>
<tr>
<td>s_y</td>
<td>4.590897</td>
<td>4.378799</td>
</tr>
<tr>
<td>s_z</td>
<td>543.932076</td>
<td>543.932076</td>
</tr>
<tr>
<td>C_x (pixel)</td>
<td>173.287243</td>
<td>173.287243</td>
</tr>
<tr>
<td>C_y (pixel)</td>
<td>114.979411</td>
<td>114.979411</td>
</tr>
</tbody>
</table>

please refer to Tsai [1987]. The intrinsic and extrinsic camera calibration parameters are found out using the data as shown in Table 2 where,

Nc = number of sensor elements in camera’s x direction = 320 (sel)
Nf = number of pixels in frame grabber’s x direction = 320 (pixel)
dx = X dimension of camera’s x sensor’s element = 0.007 (mm/sel)
dy = Y dimension of camera’s sensor’s element = 0.01 (mm/sel)
dp = Effective X dimension of pixel in frame grabber = 0.0097 (mm/pixel)
dy = 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 manipulator by Behera and Kirubanandan [1999] and Behera and Kumar [2004]. Here, we consider a 3-dimensional neural lattice that each neuron has three fields ‘weight vector’ $w_r$, ‘Jacobian matrix’ $A_r$, and ‘output vector’ $\theta_r$ where subscript $r$ stands for its position in the lattice. From each position 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

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

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 $\theta$. 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 $g(\theta)$ to form an input space of the network as

$$i_p_{target} = [u_{target}^T \ g(\theta)]^T \ (7)$$

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

$$\theta(i_p_{target}) = \theta_\mu + A_\mu(i_p_{target} - w_\mu) \ (8)$$

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

$$||w_\mu - i_p_{target}|| = \min_{\nu_p}||w_\nu - i_p_{target}|| \ (9)$$

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).

$$h_{r1}(r) = \exp\left(-\frac{|r - \mu|^2}{2\sigma^2}\right) \ (10)$$

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

where, $\sigma_{initial}$ and $\sigma_{final}$ stand for the neighbourhood width parameter $\sigma_1$’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 $\theta_{out}^0$ which is the network output that moves the end-effector to a position $i_p_0 = [i_p_{target}^0 g(\theta_{out}^0)]^T$. This is followed by some fine movements given by $\theta_{out}^i$ which brings the end-effector to $i_p_n$. The following equations depict the process of averaged output to calculate $\theta_{out}^0$ and $\theta_{out}^i$ where $n_1$ stands for number of fine movements.

$$\theta_{out}^0 = s^{-1} \sum_r h_{r1}(r) \left(\theta_r + A_r(i_p_{target} - w_r)\right) \ (13)$$

$$\theta_{out}^i = \theta_{out}^{i-1} + s^{-1} \sum_r h_{r1}(r) A_r(i_p_{target} - i_p_{r-1}) \ (14)$$

where, $s = \sum_r h_{r1}(r)$, $\theta_{out}^{i-1} = [\theta_1 \ \theta_2 \ \theta_3 \ \theta_4 \ \theta_5 \ \theta_6]_{out}^{i-1}$ and $i = 1, 2, ... n_1$. It is observed that, with this condition it may happen that $\theta_{out}^i$ becomes unbounded during training. So, to keep the $\theta_{out}^i$ within the range of $\pm 2\pi$ the following corrective measures have been taken.

$$\text{if } \theta_j > 0 \Rightarrow \theta_j = \text{mod}(\theta_j, 2\pi) \ (15)$$

$$\text{if } \theta_j < 0 \Rightarrow \theta_j = \text{mod}(\theta_j, -2\pi)$$

where, $j = 1, 2, ... 6$. $\theta_j$ is the element of $\theta_{out}^i$ and the function $\text{mod}(a, b)$ is the remainder of the division $a \div b$. This rule will ensure that all element of $\theta_{out}^i$ 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

$$\text{if } \theta_j > 0 \Rightarrow \theta_j = \left\{ \begin{array}{ll} \theta_j & ; 0 \leq \theta_j \leq \pi \\
\theta_j - 2\pi & ; \pi < \theta_j < 2\pi \end{array}\right. \ (16)$$

$$\text{if } \theta_j < 0 \Rightarrow \theta_j = \left\{ \begin{array}{ll} \theta_j & ; 0 \geq \theta_j \geq -\pi \\
\theta_j + 2\pi & ; -\pi < \theta_j < -2\pi \end{array}\right. \ (17)$$

The neural units are adjusted by the following update rules:

$$w_r \leftarrow w_r + \epsilon r h_{r3} \Delta \theta_r \ (17)$$

$$A_r \leftarrow A_r + \epsilon r h_{r3} \Delta A_r \ (18)$$

The update rule for $\theta_r$ is corrected using (15), (16) where, $\theta_j$ is the element of $\theta_r$. The $\Delta A_r$ is found using a stochastic gradient descent approach to minimize the quadratic cost function

$$E = \frac{1}{2} \left(\Delta \theta_{out}^i - A_r \Delta i_p\right)^2 \ (20)$$
The quantities $\Delta A_r$, $\Delta ip$, $\Delta \theta_{out}^0$, and $\Delta \theta_r$ are found out using the rules as follows:

\[
\Delta A_r = ||\Delta ip||^2 - (\Delta \theta_{out}^0 - A_r \Delta ip) \Delta ip^T
\]
\[
\Delta ip = ip_{n1} - ip_0
\]
\[
\Delta \theta_{out}^0 = \theta_{out}^0 - \theta_0
\]
\[
\Delta \theta_r = \theta_{out}^0 - \theta - A_r (ip_0 - w_r)
\]

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

\[
h r_k = \exp\left(-\frac{|r - \mu|^2}{2\sigma_k^2}\right)
\]

Also the learning rate parameters $\epsilon$, $\epsilon_1$, and the neighborhood width functions $\sigma_2$, $\sigma_3$ change during the training process according to a general rule as follows:

\[
\eta = \eta_{\text{initial}} \left(\frac{\eta_{\text{final}}}{\eta_{\text{initial}}}\right)^{\frac{c_1}{c}}
\]

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.

\[
t_{11} = c_1 c_2 c_3 c_4 c_5 c_6 - c_1 c_2 c_3 c_5 c_6 s_2 s_3 - s_1 s_2 c_5 c_6 - c_1 c_2 c_6 s_3 s_5 - c_1 c_2 s_2 s_3 s_4 c_6 + c_1 s_2 s_3 s_4 - s_1 s_2 c_4
\]

\[
t_{21} = c_1 c_2 c_3 c_4 c_5 c_6 - s_1 s_2 s_3 c_5 c_6 + c_1 c_2 c_6 s_3 s_5 - c_1 s_2 s_3 s_4 - c_1 c_2 s_2 s_3 c_6 + c_1 s_2 s_3 s_4 s_6 - s_1 s_2 c_4
\]

\[
t_{31} = s_1 c_2 s_3 c_5 c_6 + c_2 c_5 c_6 s_3 s_5 - s_2 s_3 s_5 c_6 + c_2 c_3 c_6 s_5 - s_2 c_3 s_5 s_6 - c_2 s_3 s_5 s_6 - s_2 c_5 c_6 s_3 s_4 + c_2 c_5 c_6 c_4
\]

\[
t_{32} = -s_2 c_3 s_5 c_6 + c_2 c_3 s_5 c_6 - s_2 s_3 s_5 c_6 + c_2 c_3 c_6 s_5 - s_2 c_3 s_5 s_6 - c_2 s_3 s_5 s_6
\]

\[
t_{33} = -s_2 s_3 c_5 c_4 - c_2 c_3 s_5 s_6 - s_2 s_3 c_5 c_6 - s_2 s_3 c_5 c_6
\]

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 $g(\theta)$ is formulated as (28)

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

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

\[
\theta = [\theta_1 \ \theta_2 \ \theta_3 \ \theta_4 \ \theta_5 \ \theta_6]^T
\]

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

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

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

\[
||w_\mu - u_{\text{target}}|| = \min_{\nu} ||w_\nu - u_{\text{target}}||
\]

The dimensions of $w_r$, $A_r$ and $\theta_r$ are $4 \times 1$, $6 \times 4$ and $6 \times 1$.

3.3 Joint space clustering

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

\[
\text{ip}_{\text{target}} = [u_{\text{target}}^T \ \theta^T]^T
\]

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

Case III: Here the network parameters $w_r$, $A_r$, and $\theta_r$ have dimensionality $10 \times 1$, $6 \times 4$ and $6 \times 1$. The weight vector $w_r$ is defined as $w_r = \begin{bmatrix} u_{\text{target}}^T & \theta^T \end{bmatrix}^T$ where $u_{\text{target}}^T$ corresponds to $u_{\text{target}}$. The winner is selected by (9). Except for update of network weights $w_r$ using (17), first 4 elements of $ip_{\text{target}}$, $ip_{\text{target}}$, $ip_{\text{target}}$, $ip_{\text{target}}$ and $w_r$ are used to train the network. In this way, this method reduces the dimensionality of the Jacobian matrix $A_r$ to $6 \times 4$ and maps directly the image plane target $u_{\text{target}}$ to the output space $\theta$. Equation (18) and (24) are not used in the training process to get the update of $\theta$, 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 $ip_{\text{target}} = u_{\text{target}}$ and the winner neuron $\mu$ is selected by (30). The dimensions of $w_r$, $A_r$ and $\theta_r$ during testing phase are $4 \times 1$, $6 \times 4$ and $6 \times 1$.

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

\[
g(\theta) = [\theta_1 \ \theta_2 \ \theta_3 \ \theta_4 \ \theta_5 \ \theta_6]
\]

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

Case V: Here the training process is same as Case IV. But, during the testing phase, we choose 4-dimensional input space $ip_{\text{target}} = u_{\text{target}}$ and the winner neuron $\mu$ is selected by (30).
Table 3. RMS error on the image plane in pixel

<table>
<thead>
<tr>
<th>Case</th>
<th>u1</th>
<th>u2</th>
<th>u3</th>
<th>u4</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1.1613</td>
<td>0.0691</td>
<td>0.2882</td>
<td>0.5543</td>
</tr>
<tr>
<td>II</td>
<td>0.6370</td>
<td>0.2526</td>
<td>0.5310</td>
<td>0.7602</td>
</tr>
<tr>
<td>III</td>
<td>3.7943</td>
<td>1.6007</td>
<td>2.4603</td>
<td>1.9744</td>
</tr>
<tr>
<td>IV</td>
<td>0.3145</td>
<td>0.0597</td>
<td>0.2932</td>
<td>0.2770</td>
</tr>
<tr>
<td>V</td>
<td>0.7254</td>
<td>0.2492</td>
<td>0.6195</td>
<td>0.5445</td>
</tr>
</tbody>
</table>

The dimensions of $w_r$, $A_r$ and $\theta_r$ are 4 x 1, 6 x 4 and 6 x 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 $\sigma_1$ 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 $-30^\circ \leq \theta_1 \leq 30^\circ$, $60^\circ \leq \theta_2 \leq 90^\circ$, $0^\circ \leq \theta_3 \leq 40^\circ$, $-40^\circ \leq \theta_4 \leq 40^\circ$, $0^\circ \leq \theta_5 \leq 40^\circ$, $-40^\circ \leq \theta_6 \leq 40^\circ$. 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 $\theta_r$ are shown in Fig. 3 and Fig. 4 respectively. In the Fig. 3 and Fig. 4, trained $w_r$ and $\theta_r$ 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 $\theta_r$ in radian are shown in the left and right plots of the Fig.4. The figures depict 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

$$X = -0.3 + 0.15 \cos(ja)$$

$$Y = 0.05 + 0.15 \sin(ja)$$

$$Z = 0.9$$

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 $0^\circ$ 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 Case III 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...
Kohonen Self Organizing Map. These methods have been successfully implemented on a 6-DOF PowerCube™ 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 $\theta_i$ is not updated by its update rule, instead it is assigned with the last 6 elements of the updated weight vector $w_i$. 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™ robot manipulator.

REFERENCES


