Neural Network Based Controller for Visual Servoing of Robotic Hand Eye System

Vikas Panwar and N. Sukavanam

***Abstract*— In this paper a neural network based controller for robot positioning and tracking using direct monocular visual feedback is proposed. The visual information is provided using a camera mounted on the end-effector of the manipulator. A PI kinematic controller is proposed to achieve motion control objective in the image plane. A Feedforward Neural Network (FFNN) is used to compensate for the robot dynamics. The FFNN computes the required torques to drive the robot manipulator to achieve desired tracking. The stability of combined PI kinematic and FFNN computed torque is proved by Lyapunov theory. Simulation results are carried out for 3 DOF articulated robot manipulator to evaluate the controller performance.**

***Index Terms*— Visual Servoing, Non-linear control systems, Feedforward Neural Networks, Perspective Projection.**

I. Introduction

This paper considers the problem of tracking moving objects with a robot end-effector, using visual information acquired with a camera mounted on the end-effector itself. Generally, these vision techniques that provide dynamically closed loop motion control for a robotic manipulator are termed as Visual Servoing [1]. The visual servoing techniques present an attractive solution to motion control of autonomous manipulators evolving in unstructured environments. The robot motion control uses direct visual sensory information to achieve a desired relative position between the robot and a possibly moving object in the robot environment. While accomplishing visual servoing, the camera may either be statically located in the environment (fixed camera configuration) or it may be mounted on the end-effector of the manipulator (known as eye-in-hand configuration). With the former, camera fixed in the environment captures the images of both the robot and target object and the robot is moved in such a way that its end-effector reaches the desired target [2,3]. With the eye-in-hand configuration, the manipulator move in such a way that the projection of a static or moving object will be at a desired location in the image as captured by the camera [4,5,6,7]. The visual servoing systems are also classified as Position Based Visual Servoing (PBVS) and Image Based Visual Servoing (IBVS) [1]. In this paper we develop an IBVS system with eye-in-hand configuration. IBVS depends on the selection of features from the image of the object and uses features directly in the visual sensory output without computing the object position and orientation. Feature based approach was proposed by Weiss et al. in [8]. Feddema et al. [9] proposed a scheme to generate a trajectory in the image feature space. Papanikolopoulos et al. in [10] introduced sum of squared errors (SSD) optical flow and presented many control algorithms e.g. proportional-integral (PI), pole assignment and linear quadratic Gaussian (LQG). Papanikolopoulos and Khosla also considered an adaptive controller based on online estimation of the relative distance of the target and the camera obviating the need of off-line camera calibration [11]. Other adaptive controllers are addressed in [12,13,14]. In most of the above-cited works, the nonlinear robot dynamics is not taken in account in the controller design and very few authors have considered the issue of dynamic control. Nasisi and Carelli [15] considered the linearly parameterized model of robotic manipulator and presented an adaptive controller to compensate for full robot dynamics. Zergeroglu et al. [16] considered the nonlinear tracking controllers with uncertain robot-camera parameters for planar robot manipulators. Bascetta and Rocco considered the dynamic effects of both rigid and flexible motion of manipulators and presented a task space oriented control law with eye-in-hand configuration [17]. Recently, there has been increasing interest in the use of intelligent control techniques e.g. artificial neural networks. Due to their learning capabilities and inherent adaptiveness, ANN finds a good application in control design. Yet the use of neural network technologies with visual feedback in less addressed in literature e.g. [18]. The uses of application of neural networks learning robot dynamics are seen in [19,20,21]. Other works relating to visual servoing are reported in [22,23,24,25,26]. In this paper we use a feedforward neural network to compensate for the full robot dynamics that learns the robot dynamics online and requires no preliminary off line learning. The controller is designed as a combination of a PI kinematic controller and feedforward neural network (FFNN) controller that computes the required torque signals to achieve the tracking. The visual information is provided using the camera mounted on the end-effector and the defined error between the actual image and desired image positions is fed to the PI controller that computes the joint velocity inputs needed to drive errors in the image plane to zero. Then the FFNN controller is designed such that the robot's joint velocities converges to the given velocity inputs. The FFNN controller assumes no knowledge linearly parameterized robot model

Manuscript received March 29, 2006.

Vikas Panwar is with the Department of Mathematics, CDL University Sirsa, Haryana, INDIA. Phone: +91 9354220973; (e-mail: vikasdma@yahoo.co.in).

N. Sukavanam is with the Department of Mathematics, Indian Institute of Technology Roorkee, Roorkee 247667 INDIA. (e-mail: nsukvfma@iitr.ernet.in).

like other adaptive schemes. The stability analysis of combined PI kinematic and FFNN computed torque controller is carried out using Lyapunov theory.

The whole paper is organized as follows. In Section II, the robot and camera models are considered. A review of feedforward neural networks is given in Section III. The NN controller design is presented in Section IV. Numerical simulation results are included in Section V. Section VI gives concluding remarks.

II. Modeling of a Robotic Hand Eye Manipulator

A. Dynamic Model

Based on the Euler-Lagrangian formulation, in the absence of friction, the motion equation of an n-link rigid, non-redundant robotic manipulator can be expressed in joint space as

$$M(q)\ddot{q} + C_m(q,\dot{q})\dot{q} + G(q) = \tau \tag{1}$$

where $q \in R^n$ is the joint displacement vector, $M(q) \in R^{n\times n}$ is the inertia matrix, $C_m(q,\dot{q}) \in R^n$ is the vector characterizing Coriolis and Centrifugal forces, $G(q) \in R^n$ is the gravitational force, $\tau \in R^n$ is the joint space torque. The model (1) has some fundamental properties that can be exploited in the controller design.

Property A.1: $M(q)$ is a symmetric positive definite matrix and bounded above and below i.e. there exits positive constants α_M and β_M such that

$$\alpha_M I_n \le M(q) \le \beta_M I_n$$

Property A.2: The matrix $\dot{M}(q) - 2C_m(q,\dot{q})$ is skew-symmetric.

B. Differential Kinematics

The differential kinematics of a manipulator gives the relationship between joint velocities $\dot{q}$ and the corresponding end-effector linear velocity ${}^W v$ and angular velocity ${}^W\omega$. This mapping is described by a $(6\times n)$ matrix, termed as geometric Jacobian $J_g(q)$, which depends on the manipulator configuration.

$$\begin{bmatrix} {}^W v \\ {}^W\omega \end{bmatrix} = \begin{bmatrix} J_P \\ J_O \end{bmatrix}\dot{q} = J_g(q)\dot{q} \tag{2}$$

where J_P is the $(3\times n)$ matrix relative to the contribution of the joint velocities $\dot{q}$ to the end-effector linear velocity ${}^W v$ and J_O is the $(3\times n)$ matrix relative to the contribution of the joint velocities $\dot{q}$ to the end-effector angular velocity ${}^W\omega$. If the end-effector position and orientation is expressed by regarding a minimal representation in the operational space, it is possible to compute the geometric Jacobian matrix through differentiation of the direct kinematics with respect to joint variables. The resulting Jacobian, termed analytical Jacobian $J_A(q)$ is related to the geometric Jacobian through [28]:

$$J_g(q) = \begin{bmatrix} I & 0 \\ 0 & T(q) \end{bmatrix} J_A(q) \tag{3}$$

where $T(q)$ is a transformation matrix that depends on the minimal parameters of the end-effector orientation.

C. Image plane Modeling

Let a pinhole camera be mounted at the robot end-effector. The image of a 3D object, captured by the camera consists of a two-dimensional brightness pattern in the image plane that moves in the image plane as the object moves in the 3D space. Let the origin of the camera coordinate frame (end-effector frame) with respect to the robot coordinate frame be ${}^W p_C = {}^W p_C(q) \in R^3$. The orientation of the camera frame with respect to the robot frame is denoted as ${}^W R_C = {}^W R_C(q) \in SO(3)$. The presented control strategy is based on the selection of the feature points of the object's image. It is assumed here that the image features are the projection onto the 2D image plane of 3D points in the scene space and the images are assumed to be noise-free. A perspective projection with a focal length λ is also assumed, as depicted in Fig. 1. An object feature point ${}^C p_O$ with coordinates $\left[{}^C p_x \;\; {}^C p_y \;\; {}^C p_z\right]^T \in R^3$ in the camera frame projects onto a point in the image plane with image coordinates $[u\; v]^T \in R^2$. The position $\xi = [u\; v]^T \in R^2$ of an object feature point in the image will be referred to as an image feature point [27]. In this paper, it is assumed that the object can be characterized by a set of feature points. Some preliminaries concerning feature points with stationary and moving objects are recalled below [15].

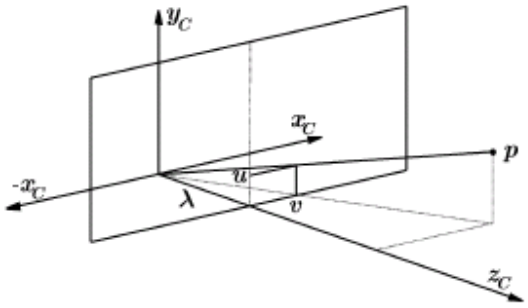


Fig.1 Perspective Projection

C.1. Stationary Object

Let ${}^W p_O \in R^3$ be the position of an object feature point expressed in robotic manipulator coordinate frame. Therefore, the relative position of this object feature located in the robot workspace, with respect to camera coordinate frame is $\left[{}^C p_x \;\; {}^C p_y \;\; {}^C p_z\right]^T$. According to the perspective projection the image feature point depends uniquely on the object feature position ${}^W p_O$ and camera position and orientation, and is

expressed as

$$\xi = \begin{bmatrix} u \\ v \end{bmatrix} = -\alpha \frac{\lambda}{{}^{C}p_z} \begin{bmatrix} {}^{C}p_x \\ {}^{C}p_y \end{bmatrix} \tag{4}$$

where α is the scaling factor in pixels/m due to camera sampling and ${}^{C}p_z < 0$ .The time derivative yields

$$\dot{\xi} = -\frac{\alpha\lambda}{{}^{C}p_z} \begin{bmatrix} 1 & 0 & -\frac{{}^{C}p_x}{{}^{C}p_z} \\ 0 & 1 & -\frac{{}^{C}p_y}{{}^{C}p_z} \end{bmatrix} \begin{bmatrix} {}^{C}\dot{p}_x \\ {}^{C}\dot{p}_y \\ {}^{C}\dot{p}_z \end{bmatrix} \tag{5}$$

On the other hand, the position of the object feature point with respect to the camera frame is given by

$$\begin{bmatrix} {}^{C}p_x \\ {}^{C}p_y \\ {}^{C}p_z \end{bmatrix} = {}^{C}R_W(q)[{}^{W}p_O - {}^{W}p_C(q)] \tag{6}$$

Using general formula for velocity of a moving point in a moving frame with respect to a fixed frame [28] and considering a fixed object point, the time derivative of (6) can be expressed in terms of the camera linear and angular velocities as [29]

$$\begin{bmatrix} {}^{C}\dot{p}_x \\ {}^{C}\dot{p}_y \\ {}^{C}\dot{p}_z \end{bmatrix} = {}^{C}R_W\{-{}^{W}\omega_C \times ({}^{W}p_O - {}^{W}p_C(q)) - {}^{W}v_C\} \tag{7}$$

After performing these operations, we have

$$\begin{bmatrix} {}^{C}\dot{p}_x \\ {}^{C}\dot{p}_y \\ {}^{C}\dot{p}_z \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 & -{}^{C}p_z & {}^{C}p_y \\ 0 & -1 & 0 & {}^{C}p_z & 0 & -{}^{C}p_x \\ 0 & 0 & -1 & -{}^{C}p_y & {}^{C}p_x & 0 \end{bmatrix} \begin{bmatrix} {}^{C}R_W & 0 \\ 0 & {}^{C}R_W \end{bmatrix} \begin{bmatrix} {}^{W}v_C \\ {}^{W}\omega_C \end{bmatrix} \tag{8}$$

where ${}^{W}v_C$ and ${}^{W}\omega_C$ denotes the camera's linear and angular velocities with respect to robot frame respectively. The motion of the image feature point as a function of the camera velocity is obtained by substituting (8) into (5):

$$\dot{\xi} = -\frac{\alpha\lambda}{{}^{C}p_z} \begin{bmatrix} -1 & 0 & \frac{{}^{C}p_x}{{}^{C}p_z} & \frac{{}^{C}p_x\,{}^{C}p_y}{{}^{C}p_z} & -\frac{{}^{C}p_z^2+{}^{C}p_x^2}{{}^{C}p_z} & {}^{C}p_y \\ 0 & -1 & \frac{{}^{C}p_y}{{}^{C}p_z} & \frac{{}^{C}p_z^2+{}^{C}p_y^2}{{}^{C}p_z} & -\frac{{}^{C}p_x\,{}^{C}p_y}{{}^{C}p_z} & -{}^{C}p_x \end{bmatrix} \begin{bmatrix} {}^{C}R_W & 0 \\ 0 & {}^{C}R_W \end{bmatrix} \begin{bmatrix} {}^{W}v_C \\ {}^{W}\omega_C \end{bmatrix} \tag{9}$$

Instead of using coordinates ${}^{C}p_x$ and ${}^{C}p_y$ of the object feature described in camera coordinate frame, which are a priori unknown, it is usual to replace them by coordinates u and v of the projection of such a feature point onto the image frame. Therefore, by using (5)

$$\dot{\xi} = J_{image}(\xi, {}^{C}p_z) \begin{bmatrix} {}^{C}R_W & 0 \\ 0 & {}^{C}R_W \end{bmatrix} \begin{bmatrix} {}^{W}v_C \\ {}^{W}\omega_C \end{bmatrix} \tag{10}$$

where $J_{image}(\xi, {}^{C}p_z)$ is the so-called image Jacobian defined by [29]:

$$J_{image}(\xi, {}^{C}p_z) = \begin{bmatrix} \frac{\alpha\lambda}{{}^{C}p_z} & 0 & \frac{u}{{}^{C}p_z} & -\frac{uv}{\alpha\lambda} & \frac{\alpha^2\lambda^2+u^2}{\alpha\lambda} & v \\ 0 & \frac{\alpha\lambda}{{}^{C}p_z} & \frac{v}{{}^{C}p_z} & -\frac{\alpha^2\lambda^2+v^2}{\alpha\lambda} & \frac{uv}{\alpha\lambda} & -u \end{bmatrix} \tag{11}$$

Finally, by using (2) and (3) we can express $\dot{\xi}$ in terms of robot joint velocity $\dot{q}$ as

$$\begin{aligned} \dot{\xi} &= J_{image}(\xi, {}^{C}p_z) \begin{bmatrix} {}^{C}R_W(q) & 0 \\ 0 & {}^{C}R_W(q) \end{bmatrix} J_g(q)\dot{q} \\ &= J_{image}(\xi, {}^{C}p_z) \begin{bmatrix} {}^{C}R_W & 0 \\ 0 & {}^{C}R_W \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & T(q) \end{bmatrix} J_A(q)\dot{q} \end{aligned} \tag{12}$$

C.2. Moving Object

When the object moves in the robot manipulator framework, the derivative of (6) can be expressed as

$$\begin{bmatrix} {}^{C}\dot{p}_x \\ {}^{C}\dot{p}_y \\ {}^{C}\dot{p}_z \end{bmatrix} = {}^{C}R_W\{-{}^{W}\omega_C \times ({}^{W}p_O - {}^{W}p_C(q)) + ({}^{W}\dot{p}_O - {}^{W}v_C)\}$$

$$= \begin{bmatrix} -1 & 0 & 0 & 0 & -{}^{C}p_z & {}^{C}p_y \\ 0 & -1 & 0 & {}^{C}p_z & 0 & -{}^{C}p_x \\ 0 & 0 & -1 & -{}^{C}p_y & {}^{C}p_x & 0 \end{bmatrix} \begin{bmatrix} {}^{C}R_W & 0 \\ 0 & {}^{C}R_W \end{bmatrix} \begin{bmatrix} {}^{W}v_C \\ {}^{W}\omega_C \end{bmatrix} + {}^{C}R_W\,{}^{W}\dot{p}_O \tag{13}$$

where ${}^{W}v_C$ and ${}^{W}\omega_C$ are the linear and angular velocity of the camera with respect to the robot frame. The movement of the feature point into the image plane as a function of the object velocity and camera velocity is expressed by substituting (13) into (5):

$$\dot{\xi} = -\frac{\alpha\lambda}{{}^{C}p_z} \begin{bmatrix} -1 & 0 & \frac{{}^{C}p_x}{{}^{C}p_z} & \frac{{}^{C}p_x\,{}^{C}p_y}{{}^{C}p_z} & -\frac{{}^{C}p_z^2+{}^{C}p_x^2}{{}^{C}p_z} & {}^{C}p_y \\ 0 & -1 & \frac{{}^{C}p_y}{{}^{C}p_z} & \frac{{}^{C}p_z^2+{}^{C}p_y^2}{{}^{C}p_z} & -\frac{{}^{C}p_x\,{}^{C}p_y}{{}^{C}p_z} & -{}^{C}p_x \end{bmatrix} \begin{bmatrix} {}^{C}R_W & 0 \\ 0 & {}^{C}R_W \end{bmatrix} \begin{bmatrix} {}^{W}v_C \\ {}^{W}\omega_C \end{bmatrix} - \frac{\alpha\lambda}{{}^{C}p_z} \begin{bmatrix} 1 & 0 & -\frac{{}^{C}p_x}{{}^{C}p_z} \\ 0 & 1 & -\frac{{}^{C}p_y}{{}^{C}p_z} \end{bmatrix} {}^{C}R_W\,{}^{W}\dot{p}_O \tag{14}$$

Finally, by using (2) and (3) we can express $\dot{\xi}$ in terms of robot joint velocity $\dot{q}$ as

$$\dot{\xi} = J(q,\xi,{}^{C}p_z)\dot{q} + J_O(q,{}^{C}p_O)\,{}^{W}\dot{p}_O \tag{15}$$

where

$$J_O(q,{}^{C}p_O) = -\frac{\alpha\lambda}{{}^{C}p_z}\begin{bmatrix} 1 & 0 & -\frac{{}^{C}p_x}{{}^{C}p_z} \\ 0 & 1 & \frac{{}^{C}p_y}{{}^{C}p_z} \end{bmatrix} {}^{C}R_W \tag{16}$$

III. FEEDFORWARD NEURAL NETWORKS

A two-layer feedforward neural network with n input units, m output units and N units in the hidden layer, is shown in the Fig. 2.

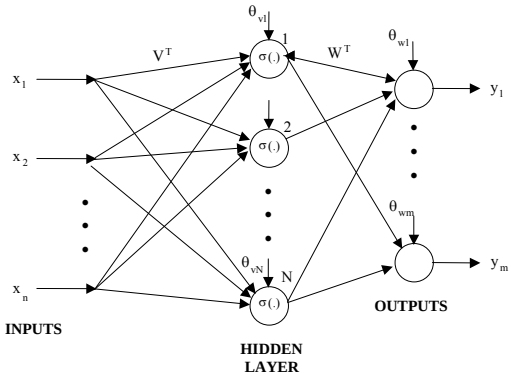


Fig. 2 Feedforward Neural Network

The output vector y is determined in terms of the input vector x by the formula

$$y_i = \sum_{j=1}^{N}\left[w_{ij}\sigma\left(\sum_{k=1}^{n} v_{jk}x_k + \theta_{vj}\right) + \theta_{wi}\right];\ i = 1,\cdots,m \tag{17}$$

where $\sigma(.)$ are the activation functions of the neurons of the hidden-layer. The inputs-to-hidden-layer interconnection weights are denoted by v_{jk} and the hidden-layer-to-outputs interconnection weights by w_{ij}. The bias weights are denoted by θ_{vj}, θ_{wi}. There are many classes of activation functions e.g. sigmoid, hyperbolic tangent and Gaussian. The sigmoid activation function used in our work, is given by

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{18}$$

By collecting all the NN weights v_{jk}, w_{ij} into matrices of weights V^T, W^T, we can write the NN equation in terms of vectors as

$$y = W^T\sigma(V^T x) \tag{19}$$

with the vector of activation functions defined by $\sigma(z) = [\sigma(z_1)\cdots\sigma(z_n)]^T$ for a vector $z \in R^n$. The bias weights are included as the first column of the weight matrices. To accommodate bias weights the vectors x and $\sigma(.)$ need to be augmented by replacing 1 as their first element e.g. $x \equiv [1\ x_1\ x_2\cdots x_n]^T$.

B. Function Approximation Property

Let $f(x)$ be a smooth function from R^n to R^m. Let $U_x \subseteq R^n$ then for $x \in U_x$ there exists some number of hidden layer neurons N and weights W and V such that [30]

$$f(x) = W^T\sigma(V^T x) + \varepsilon \tag{20}$$

The value of ε is called the NN functional approximation error. In fact, for any choice of a positive number ε_N, one can find a NN such that $\varepsilon < \varepsilon_N$ in U_x. For a specified value of ε_N the ideal approximating NN weights exist. The, an estimate of $f(x)$ can be given by

$$\hat{f}(x) = \hat{W}^T\sigma(\hat{V}^T x) \tag{21}$$

where $\hat{W}$ and $\hat{V}$ are estimates of the ideal NN weights that are provided by some on-line weight tuning algorithms.

B. Error Backpropagation Algorithm

This is a common weight-tuning algorithm that is based on gradient descent algorithm. If the NN is training off-line to match specified exemplar pairs (x_d, y_d), with x_d the ideal NN input that yields the desired NN output y_d, then the continuous-time version of the backpropagation algorithm for the two-layer NN is given by

$$\begin{aligned} \dot{\hat{W}} &= F\sigma(\hat{V}^T x_d)\mathrm{E}^T \\ \dot{\hat{V}} &= G x_d(\hat{\sigma}'^T\hat{W}\mathrm{E})^T \end{aligned} \tag{22}$$

where F, G are positive definite design learning parameter matrices. The backpropagated error E is selected as the desired NN output minus the actual NN output $\mathrm{E} = y_d - y$. For the scalar sigmoid activation function (18) the hidden-layer output gradient $\hat{\sigma}'$ is

$$\hat{\sigma}' \equiv diag\{\sigma(\hat{V}^T x_d)\}[I - diag\{\sigma(\hat{V}^T x_d)\}] \tag{23}$$

where I denotes the identity matrix, and $diag\{z\}$ means a diagonal matrix whose diagonal elements are the components of the vector z. In the next section design of NN controller is presented.

IV. NN CONTROLLER DESIGN

In this section we consider the design of an image based control algorithm for tracking of an object moving along an unknown curve. It is assumed that the object moves along a smooth trajectory with bounded velocity ${}^{W}\dot{p}_O(t) = {}^{W}v_O(t)$ and acceleration $\mathrm{d}\,{}^{W}\dot{p}_O(t)/\mathrm{dt} = {}^{W}a_O(t)$. It establishes a practical restriction on the object trajectory. Also, there exists a trajectory in the joint space $q_d(t)$ such that the vector of desired fixed features ξ_d is achievable, ensuring that the control problem is solvable. ξ_d is taken to be a constant feature vector in the image plane. The depth ${}^{C}p_z$ i.e. the distance from the camera to the object is available to be used by the controller.

Now the control problem can be formulated. The control problem is to find out a control law so that control error in the image plane $\tilde{\xi}(t) = \xi_d - \xi(t)$ is ultimately bounded in the small ball B_r. An auxiliary velocity control input that achieves tracking for (15) is given by

$$u(t) = J^{+}\left(K_P \tilde{\xi}(t) + K_i \int \tilde{\xi}(t)dt\right) - J^{+} J_O{}^{W} \dot{p}_O \quad (24)$$

where K_P and K_i are design matrices, termed as proportional gain and integral gain matrices and $J^{+} = J^{-1}$ if J is invertible else J^{+} is the pseudo-inverse defined as $J^{+} = J^{T}(JJ^{T})^{-1}$. Assuming perfect velocity tracking and substituting $u(t)$ for $\dot{q}$ in (15) we get

$$\dot{\tilde{\xi}}(t) + K_P \tilde{\xi}(t) + K_i \int \tilde{\xi}(t)dt = 0 \quad (25)$$

Taking the first derivatives of (25) yields

$$\ddot{\tilde{\xi}} + K_P \dot{\tilde{\xi}} + K_i \tilde{\xi} = 0 \quad (26)$$

To analyze the stability of (26), the following Lyapunov function is chosen:

$$V_{\xi} = \frac{1}{2}\dot{\tilde{\xi}}^{T}\dot{\tilde{\xi}} + \frac{1}{2}\tilde{\xi}^{T} K_i \tilde{\xi} \quad (27)$$

From (26) and (27), we have

$$\dot{V}_{\xi} = -\dot{\tilde{\xi}}^{T} K_p \dot{\tilde{\xi}} \quad (28)$$

Therefore, from Lyapunov stability and LaSalle's Theorem, we conclude that $\dot{\tilde{\xi}}(t) \to 0$ as $t \to \infty$. This implies that $\xi(t) \to \xi_d$.

Given the desired velocity $u \in R^{n\times 1}$, define the auxiliary velocity tracking error as $e(t) = u(t) - v(t)$. Differentiating $e(t)$ and using (1), the robot dynamics may be written in terms of the velocity tracking error as

$$M(q)\dot{e}(t) = -C_m(q,\dot{q})e(t) - \tau(t) + f(y) \quad (29)$$

where $f(y) = M(q)\dot{u}(t) + C_m(q,\dot{q})u(t) + G(q)$ is called robot nonlinear function. The vector y is given by $y = [v^T \; u^T \; \dot{u}^T]^T$. The function $f(y)$ contains all the robot parameters such as masses, moments of inertia, gravity, friction etc. This function is partially known or unknown. Therefore, a suitable control input for velocity following is given by

$$\tau(t) = \hat{f} + K_v e(t) \quad (30)$$

with K_v a diagonal positive definite gain matrix, and $\hat{f}(y)$ an estimate of the robot function that is provided by a feedforward neural network (FFNN). Now, with this control the closed-loop error system becomes.

$$M(q)\dot{e} = -(K_v + C_m(q,\dot{q}))e + \tilde{f} \quad (31)$$

where $\tilde{f} = f - \hat{f}$ is the functional estimation error. The functional approximation of $\hat{f}(y)$ with a FFNN may be given as

$$\hat{f}(y) = \hat{W}^{T}\sigma(\hat{V}^{T}y) \quad (32)$$

Using this FFNN functional approximation the velocity error dynamics becomes

$$M(q)\dot{e}(t) = -(K_v + C_m(q,\dot{q}))e(t) + W^{T}\sigma(V^{T}y) + \varepsilon - \hat{W}^{T}\sigma(\hat{V}^{T}y)) \quad (33)$$

In order to proceed further we need the following definitions [20].

Definition 1: The solution of a nonlinear system with state $y(t) \in R^n$ is uniformly ultimately bounded (UUB) if there exists a compact set $U_y \subset R^n$ such that for all $y(t_0) = y_0 \in U_y$, there exists a $\delta > 0$ and a number $T(\delta, y_0)$ such that $\|y(t)\| < \delta$ for all $t \geq t_0 + T$.

Definition 2: We define the norm of a vector $y \in R^n$ as $\|y\| = \sqrt{y_1^2 + y_2^2 + \cdots + y_n^2}$ and the norm of a matrix $A \in R^{m\times n}$ as $\|A\| = \sqrt{\lambda_{\max}[A^T A]}$ where $\lambda_{\max}[\cdot]$ and $\lambda_{\min}[\cdot]$ are the largest and smallest eigenvalue of a matrix. To be specific we denote the p-norm by $\|\cdot\|_p$ and the absolute value as $|\cdot|$.

Definition 3: Given $A = [a_{ij}], B \in R^{m\times n}$ the Frobenius norm is defined by $\|A\|_F^2 = tr\{A^T A\} = \sum_{i,j} a_{ij}^2$ with $tr\{\cdot\}$ as the trace operator. The associated inner product is $\langle A, B\rangle_F = tr\{A^T B\}$ The Frobenius norm is compatible with 2-norm so that $\|Ay\|_2 \leq \|A\|_F \|y\|_2$.

Definition 4: For notational convenience, define the matrix of all NN weights as $Z \equiv diag\{W, V\}$ and the weight estimation errors as $\tilde{W} = W - \hat{W}$, $\tilde{V} = V - \hat{V}$ *and* $\tilde{Z} = Z - \hat{Z}$. The ideal NN weights bounded so that $\|Z\|_F \leq Z_M$ with known Z_M. Also define the hidden layer output error for a given y as $\tilde{\sigma} = \sigma - \hat{\sigma} = \sigma(V^T y) - \sigma(\hat{V}^T y)$.

Adding and subtracting $W^T\sigma(\hat{V}^T y)$ in (33), we get

$$M(q)\dot{e}(t) = -(K_v + C_m(q,\dot{q}))e(t) + W^{T}\left(\sigma(V^{T}y) - \sigma(\hat{V}^{T}y)\right) + \tilde{W}^{T}\sigma(\hat{V}^{T}y) + \varepsilon \quad (34)$$

The Taylor series expansion of $\sigma(V^T y)$ about given $V^T y$ gives us

$$\sigma(V^{T}y) = \sigma(\hat{V}^{T}y) + \sigma'(\hat{V}^{T}y)\tilde{V}^{T}y + O(\tilde{V}^{T}y)^2 \quad (35)$$

with $\sigma'(\hat{z}) = \left.\frac{d\sigma(z)}{dz}\right|_{z=\hat{z}}$ the Jacobian matrix and $O(z)^2$ denoting terms of second order. Denoting $\hat{\sigma}' = \sigma'(\hat{V}^T y)$ we have $\tilde{\sigma} = \hat{\sigma}'\tilde{V}^T y + O(\tilde{V}^T y)^2$. Replacing for $\tilde{\sigma}$ in (34) we get

$$M(q)\dot{e}(t) = -(K_v + C_m(q,\dot{q}))e(t) + \hat{W}^{T}\hat{\sigma}'\tilde{V}^{T}y + \tilde{W}^{T}\hat{\sigma} + w_1 \quad (36)$$

where the disturbance terms are

$$w_1(t) = \tilde{W}^{T}\hat{\sigma}'\tilde{V}^{T}y + W^{T}O(\tilde{V}^{T}y)^2 + \varepsilon \quad (37)$$

Finally, adding and subtracting $\tilde{W}^T\hat{\sigma}'\hat{V}^T y$ (36) becomes

$$M(q)\dot{e}(t) = -\left(K_v + C_m(q,\dot{q})\right)e(t) + \hat{W}^T\hat{\sigma}'\tilde{V}^T y + \tilde{W}^T\left(\hat{\sigma} - \hat{\sigma}'\hat{V}^T y\right) + w \tag{38}$$

where the disturbance terms are

$$w(t) = \tilde{W}^T\hat{\sigma}'V^T y + W^T O(\tilde{V}^T y)^2 + \varepsilon \tag{39}$$

A. NN Weights Update Law

With positive definite design parameters F, G and $\kappa > 0$, the adaptive NN weight update law is given by

$$\dot{\hat{W}} = F\hat{\sigma}e^T - F\hat{\sigma}'\hat{V}^T y e^T - \kappa F\|e\|\hat{W}$$
$$\dot{\hat{V}} = Gy\left(\hat{\sigma}'\hat{W}e\right)^T - \kappa G\|e\|\hat{V} \tag{40}$$

Then the errors state vector e and $\tilde{W}\ \tilde{V}$ are uniformly ultimately bounded and the error state vector e can be made arbitrary small by adjusting the weights.

Proof: Consider the following Lyapunov function candidate

$$L = \frac{1}{2}e^T M(q)e + \frac{1}{2}tr\left(\tilde{W}^T F^{-1}\tilde{W}\right) + \frac{1}{2}tr\left(\tilde{V}^T G^{-1}\tilde{V}\right) \tag{41}$$

The time derivative $\dot{L}$ of the Lyapunov function becomes

$$\dot{L} = e^T M(q)\dot{e} + \frac{1}{2}e^T\dot{M}(q)e + tr\left(\tilde{W}^T F^{-1}\dot{\tilde{W}}\right) + tr\left(\tilde{V}^T G^{-1}\dot{\tilde{V}}\right) \tag{41}$$

Evaluating (41) along (37) we get

$$\dot{L} = -e^T K_v e - e^T C_m e + \frac{1}{2}e^T\dot{M}(q)e + e^T\hat{W}^T\hat{\sigma}'\tilde{V}^T y + e^T\tilde{W}^T\left(\hat{\sigma} - \hat{\sigma}'\hat{V}^T y\right) + e^T w + tr\left(\tilde{W}^T F^{-1}\dot{\tilde{W}}\right) + tr\left(\tilde{V}^T G^{-1}\dot{\tilde{V}}\right) \tag{42}$$

From property A.2, the matrix $\dot{M}(q) - 2C_m(q,\dot{q})$ is skew-symmetric, hence $-e^T C_m e + \frac{1}{2}e^T\dot{M}(q)e = \frac{1}{2}e^T\left(\dot{M}(q) - 2C_m(q,\dot{q})\right)e = 0$. Also with $\dot{\tilde{W}} = -\dot{\hat{W}}, \dot{\tilde{V}} = -\dot{\hat{V}}$ and adaptive learning rule (40), we have

$$\begin{aligned}\dot{L} &= -e^T K_v e + e^T\hat{W}^T\hat{\sigma}'\tilde{V}^T y + e^T\tilde{W}^T\left(\hat{\sigma} - \hat{\sigma}'\hat{V}^T y\right) + e^T w \\ &\quad + tr\left(-\tilde{W}^T\hat{\sigma}e^T + \tilde{W}^T\hat{\sigma}'\tilde{V}^T y e^T + \kappa\tilde{W}^T\|e\|\hat{W}\right) \\ &\quad + tr\left(-\tilde{V}^T y e^T\hat{W}^T\hat{\sigma}'^T + \kappa\tilde{V}^T\|e\|\hat{V}\right) \\ &= -e^T K_v e + e^T w \\ &\quad + tr\begin{pmatrix}-\tilde{W}^T\hat{\sigma}e^T + \tilde{W}^T\hat{\sigma}'\tilde{V}^T y e^T + \tilde{W}^T\left(\hat{\sigma} - \hat{\sigma}'\hat{V}^T y\right)e^T \\ + \kappa\tilde{W}^T\|e\|\hat{W}\end{pmatrix} \\ &\quad + tr\left(-\tilde{V}^T y e^T\hat{W}^T\hat{\sigma}'^T + \tilde{V}^T y e^T\hat{W}^T\hat{\sigma}'^T + \kappa\tilde{V}^T\|e\|\hat{V}\right)\end{aligned}$$

or $$\dot{L} = -e^T K_v e + e^T w + tr\left(\kappa\tilde{W}^T\|e\|\hat{W}\right) + tr\left(\kappa\tilde{V}^T\|e\|\hat{V}\right)$$

Taking norm of both sides, assuming $w = 0$ (though w can be shown to be bounded) we get

$$\dot{L} \le -\lambda_{\min}(K_v)\|e\|^2 + \kappa\|e\|\left(\left\|\tilde{Z}\right\|_F Z_M - \|Z\|_F^2\right) \tag{43}$$

The following inequality is used in the derivation of (43)

$$tr\left(\tilde{Z}^T\hat{Z}\right) = tr\left(\tilde{Z}^T(Z - \tilde{Z})\right) = \left\langle\tilde{Z}, Z\right\rangle_F - \left\|\tilde{Z}\right\|_F^2$$

Using Cauchy-Schwartz inequality, we have $\left\langle\tilde{Z}, Z\right\rangle_F \le \left\|\tilde{Z}\right\|_F Z_M$, and then the Eq. (4.21) is derived. Completing the square term, we get

$$\dot{L} \le -\|e\|\left[\|e\|\lambda_{\min}(K_v) + \kappa\left(\left\|\tilde{Z}\right\|_F - \frac{1}{2}Z_M\right)^2 - \frac{1}{4}\kappa Z_M^2\right] \tag{44}$$

The expression for $\dot{L}$ given by (43) remains negative as long as the quantity in the bracket is positive i.e. either (45) or (46) hold

$$\|e\| \ge \left(\frac{1}{4}\kappa Z_M^2\right)\Big/\lambda_{\min}(K_v) \equiv C_e \tag{45}$$

$$\left\|\tilde{Z}\right\|_F \ge \sqrt{\frac{1}{4}\kappa Z_M^2} + \frac{1}{2}Z_M \equiv C_{\tilde{Z}} \tag{46}$$

where C_e *and* $C_{\tilde{Z}}$ are the convergence regions. According to Lyapunov theory and LaSalle extension the UUB of e and $\tilde{Z}$ is proved [20].

V. Simulation Results

The simulation has been performed for three-link articulated robotic manipulators (Fig. 3) tracking an object point moving on an elliptical path. The mathematical model of the robotic manipulator is expressed as

$$\begin{bmatrix}m_{11} & 0 & 0\\ 0 & m_{22} & m_{23}\\ 0 & m_{32} & m_{33}\end{bmatrix}\begin{bmatrix}\ddot{q}_1\\ \ddot{q}_2\\ \ddot{q}_3\end{bmatrix} + \begin{bmatrix}c_{11} & 0 & 0\\ c_{21} & c_{22} & c_{23}\\ c_{31} & c_{32} & c_{33}\end{bmatrix}\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\end{bmatrix} + \begin{bmatrix}0\\ g_2\\ g_3\end{bmatrix} = \begin{bmatrix}\tau_1\\ \tau_2\\ \tau_3\end{bmatrix}$$

where

$$m_{11} = \left(\frac{m_2}{3} + m_3\right)a_2^2\cos^2(q_2) + \frac{m_3}{3}a_3^2\cos^2(q_2 + q_3) + m_3 a_2 a_3\cos(q_2)\cos(q_2 + q_3)$$

$$m_{22} = \left(\frac{m_2}{3} + m_3\right)a_2^2 + \frac{m_3 a_3^2}{3} + m_3 a_2 a_3\cos(q_3)$$

$$m_{23} = m_{32} = \frac{m_3 a_3^2}{3} + \frac{m_3}{2}a_2 a_3\cos(q_3)$$

$$m_{33} = \frac{m_3 a_3^2}{3}$$

$$c_{11} = -\begin{pmatrix}\left(\frac{m_2}{3} + m_3\right)a_2^2\sin(2q_2) + \frac{m_3 a_3^2}{3}\sin(2q_2 + 2q_3)\\ + m_3 a_2 a_3\sin(2q_2 + q_3)\end{pmatrix}\dot{q}_2 - \begin{pmatrix}\frac{m_3 a_3^2}{3}\sin(2q_2 + 2q_3)\\ + m_3 a_2 a_3\cos(q_2)\sin(q_2 + q_3)\end{pmatrix}\dot{q}_3$$

$$c_{21} = \begin{pmatrix}\left(\frac{m_2}{6} + \frac{m_3}{2}\right)a_2^2\sin(2q_2) + \frac{m_3 a_3^2}{6}\sin(2q_2 + 2q_3)\\ + m_2 a_2 a_3\sin(2q_2 + q_3)\end{pmatrix}\dot{q}_1$$

$$c_{22} = -m_3 a_2 a_3\sin(q_3)\dot{q}_3$$

$$c_{23} = -\frac{1}{2} m_3 a_2 a_3 \sin(q_3)\dot{q}_3$$

$$c_{31} = \left(\frac{m_3 a_3^2}{6} \sin(2q_2 + 2q_3) + \frac{m_3 a_2 a_3}{2} \cos(q_2)\sin(q_2 + q_3) \right) \dot{q}_1$$

$$c_{32} = \frac{m_3 a_2 a_3}{2} \sin(q_3)\dot{q}_2 - \frac{m_3 a_2 a_3}{4} \sin(q_3)\dot{q}_3$$

$$c_{33} = \frac{m_3 a_2 a_3}{4} \sin(q_3)\dot{q}_3$$

$$g_2 = \left(\frac{m_2}{2} + m_3 \right) a_2 \cos(q_2) g + \frac{m_3 a_3}{2} \cos(q_2 + q_3) g$$

$$g_3 = \frac{m_3 a_3}{2} \cos(q_2 + q_3) g$$

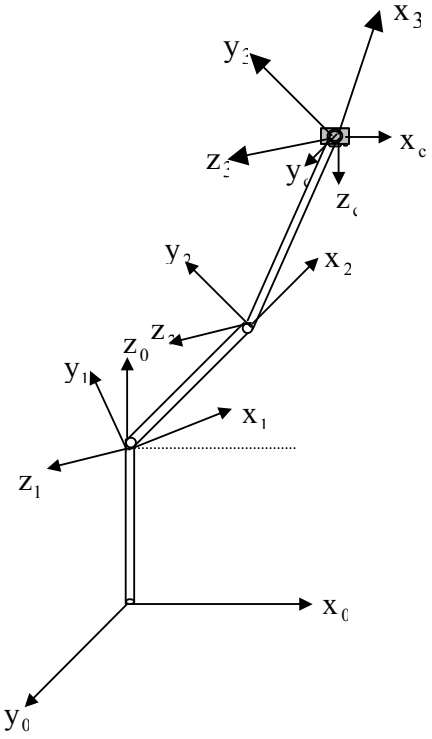


Fig. 3

The parameter values for the model are taken as $m_1 = 5, m_2 = 4, m_3 = 2,\ a_1 = 2.5. a_2 = 2.5, a_3 = 2.5,\ g = 9.8$

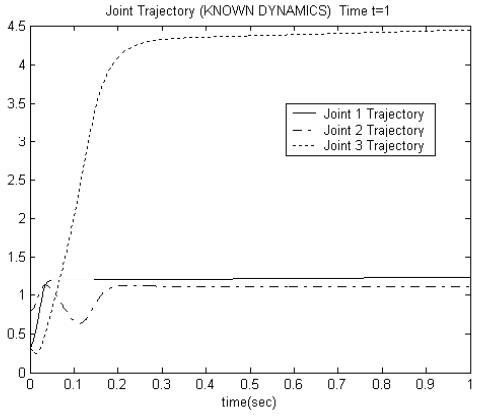
The camera parameters are set to be $\alpha = 1000,\ \lambda = 0.01$. The gain parameters are

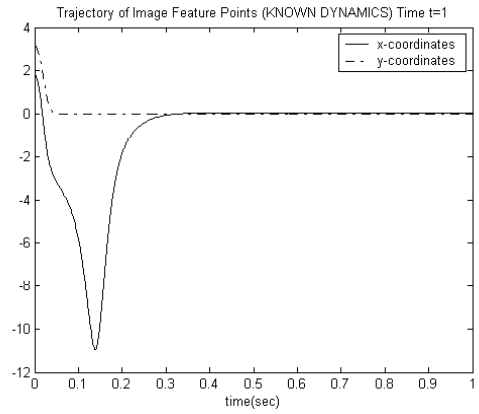
$$K_p = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix},\ K_i = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix},\ K_v = \begin{bmatrix} 1000 & 0 \\ 0 & 1000 \end{bmatrix}.$$

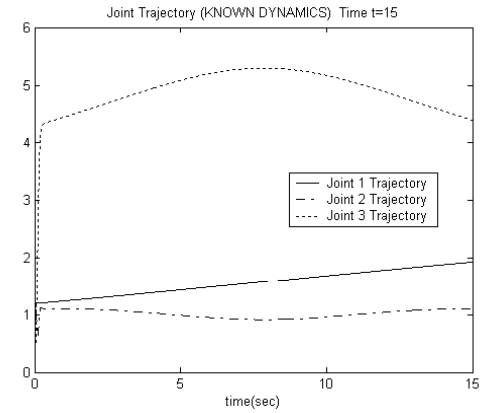
The point object is considered moving on the following elliptical path with $\omega = 0.1$

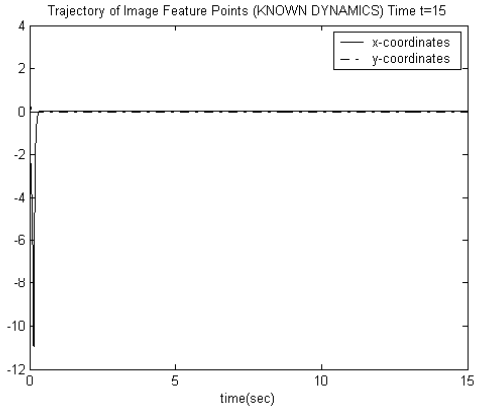
$$^{w}p_O = [\cos(\omega t) \quad 2.5 + 1.5\sin(\omega t) \quad 0]^T.$$

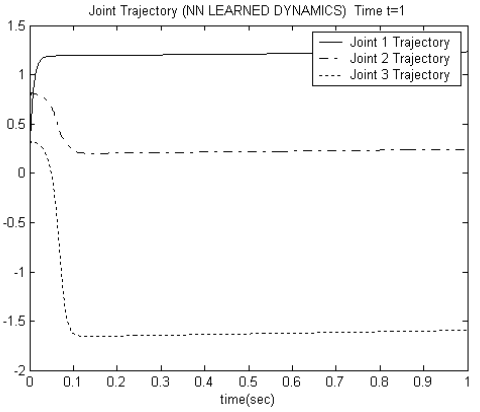
The target point for image feature is taken to be the center of the image plane. The initial values for joint angles and joint velocities are $q_1(0) = \pi/10,\ q_2(0) = \pi/4, q_3(0) = \pi/10$ $\dot{q}_1(0) = 0,\ \dot{q}_2(0) = 0,\ \dot{q}_3(0) = 0$. The architecture of the FFNN is composed of 9 input units & 1 bias unit, 10 hidden sigmoidal units & 1 bias unit and 3 output units. The learning rate in the weight-tuning algorithm is $F = I_{11}, G = I_{10}$ *and* $\kappa = .0001$. The simulation of the whole system is shown for 1 second (transient response) and 15 seconds (steady state response).

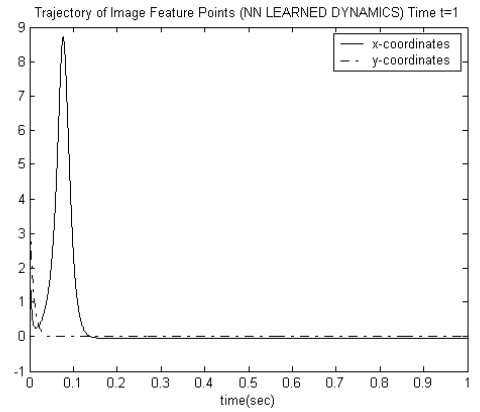


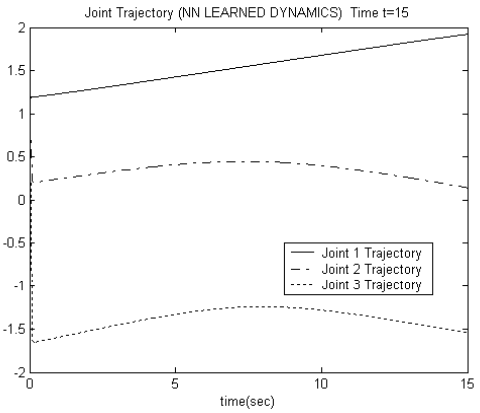


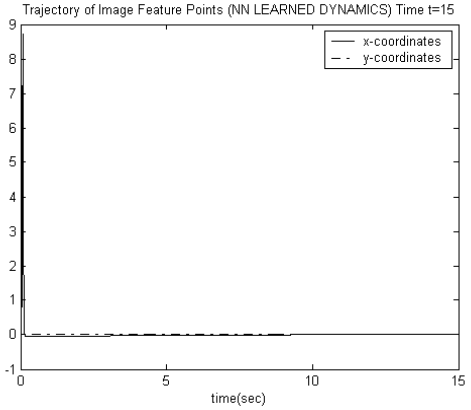












VI. CONCLUSION

In this paper a stable control algorithm is presented for visual tracking of moving objects with for robot manipulators with camera-in-hand configuration using direct visual feedback with unknown robot dynamics. The unknown robot dynamics is fully compensated with a feedforward neural network. The neural network learns the dynamics online and requires no preliminary learning. The NN weights may be simply initialized to zero and errors may be kept arbitrarily small by increasing the gain K_v. The stability of the overall system is proved using Lyapunov function that is generated by weighting matrices. Simulation of 3 DOF manipulator tracking an object point feature, moving on an elliptical path is carried out to illustrate the control methodology. The simulation results show that the feedforward neural network with the on-line updating law can compensate the full robot dynamics efficiently.

REFERENCES

[1] Hutchinson S., Hager G.D. and Corke P.I, "A Tutorial on Visual Servo Control," *IEEE Trans. Robot. Automa,* Vol. 12, No. 5, 1996, pp 651-670.

[2] Allen P.K., B. Yoshimi and A. Timcenko, "Real-time Visual Servoing," *Proc. IEEE Inter. Conf. Robot. Autom*, 1991, pp. 851-856.

[3] Koivo A.J. and Hounshangi N., "Real-Time Vision Feedback for Servoing of a Robotic Manipulator with Self-tuning Controller," *IEEE Trans. Sys. Man Cyber*, Vol. 21, No.1, 1991, pp 134-142.

[4] Papanikolopoulos N.P. and Khosla P.K., "Feature-Based Robotic Visual Tracking of 3-D Translational Motion," *Proc. 30th Conf. Decision Contr,* Brighton-England, 1991, pp 1877-1882.

[5] Papanikolopoulos N.P., Nelson B. and Khosla P.K., "Monocular 3-D Visual Tracking of a Moving Target by an Eye-in-Hand Robotic system," *Proc. 31st Conf. Decision Contr*, Tucson-Arizona, 1992, pp 3805-3810.

[6] P. Allen, A. Timcenko, B. Yoshimi and P. Michelman; "Automated Tracking and Grasping of a Moving Object with a Robotic Hand-eye System," *IEEE Trans. Robot. Autom,* Vol. 9, No. 1, 1993, pp 152-165.

[7] K. Hashimoto, T. Ebine and H. Kimura, "Visual Servoing with Hand-Eye Manipulator – Optimal Control Approach," *IEEE Trans. Robot. Autom*, Vol. 12, No. 5, 1996, pp 766-774.

[8] Weiss Lee L., Sanderson A.C. and Neuman C.P., "Dynamic Sensor based Control of Robots with Visual Feedback," *IEEE Journ. Robot. Autom,* Vol. RA-3, No. 5, 1987, pp 404-417.

[9] Feddema John T. and Mitchell Owen R., "Vision-Guided Servoing with Feature-Based Trajectory Generation," *IEEE Trans. Robot. Autom,* Vol. 5, No. 5, 1989, pp 691-700.

[10] Papanikolopoulos N.P., Khosla P.K. and Kanade T., "Visual Tracking of a Moving Target by a Camera Mounted on a Robot: A Combination of

Control and Vision," *IEEE Trans. Robotic. Autom,* Vol. 9, No. 1, 1993, pp 14-35.
[11] Papanikolopoulos N.P. and Khosla P.K., "Adaptive Robotic Visual Tracking: Theory and Experiments," *IEEE Trans. Automatic Contr,* Vol. 38, No. 3, 1993, pp. 429-445.
[12] Asada M., Tanaka T. and Hosoda K., "Adaptive Binocular Visual Servoing for Independently Moving Target Tracking," *Proc. IEEE Inter. Conf. Robotic. Autom*, San Francisco, CA, 2000.
[13] Hsu L., Costa R. and Aquino P., "Stable Adaptive Visual Servoing for Moving Targets," *Proc. 2000 American Contr. Conf*, Vol. 3, June 28–30, 2000, pp. 2008–2012.
[14] Astofli A. Hsu L., Netto M. and Ortega R., "A solution to the adaptive visual servoing problem," *Proc. IEEE Inter. Conf. Robot. Autom,* Vol. 1, May 21–26, 2001, pp. 743–748.
[15] Nasisi O and Carelli R., "Adaptive Servo Visual Robot Control," *Robot. Autonomous Sys*, Vol. 43, 2003, pp 51-78.
[16] Zergeroglu E., Dawson D.M., M. S. de Queiroz and A. Behal, "Vision-Based Nonlinear Tracking Controllers with Uncertain Robot-Camera Parameters," *IEEE/ASME Trans. Mechatron*, Vol. 6, No. 3, 2001, pp. 322-337.
[17] Bascetta L. and Rocco P., "Task Space Visual Servoing of Eye-in-Hand flexible Manipulators," *Proc. IEEE/ASME Intern. Conf. Advanced Intell. Mechatron. AIM*, 2003, pp. 1442-1448.
[18] Hashimoto H., Kubota T., Sato M. and Harashima F., "Visual Control of Robotic Manipulators Based on Neural Networks," *IEEE Trans. Indust. Electron,* Vol. 39, No. 6, 1992, pp. 490-496.
[19] Ishiguro A., Furuhashi T., Okuma S. and Uchikawa Y., "A Neural Network Compensator for Uncertainties of Robot Manipulators," *IEEE Trans. Indust. Electron*, Vol. 39, No. 6, 1992, pp 565-570.
[20] Lewis F.L., Jagannathan S. and Yesildirek A., *Neural Network Control of Robot Manipulators*, Taylor & Francis, 1999.
[21] Lewis F.L., Liu K. and Yesildirek A., "Neural Net Robot Controller with Guaranteed Tracking Performance," *IEEE Trans. Neural Net*, Vol. 6, No. 3, 1995, pp. 703-715.
[22] Cheah C., Lee K., Kawamura S. and Arimoto S., "Asymptotic Stability of Robot Control with Approximate Jacobian Matrix and its Application to Visual Servoing," *Proc. 39th IEEE Conf. Decision Contr*, Vol. 4, December 12–15, 2000, pp. 3939–3944.
[23] Corke P. and S. Hutchison. "Real-Time Vision, Tracking and Control," *Proc. IEEE Inter. Conf. Robot. Autom*, San Francisco, CA, 2000.
[24] Espiau B., Chaumette F. and Rives P., "A New Approach to Visual Servoing in Robotics," *IEEE Trans. Robot. Autom*, Vol. 8, No. 3, 1992, pp 313-326.
[25] Hager G.D., "A modular system for robust positioning using feedback from stereo vision," *IEEE Trans. Robot. Autom*, Vol. 13, 1997, pp 582-670.
[26] Sukavanam N., "Image Based Visual Servo Control in Robotics," *Proc. Conf. Math. Appl. Engg. Indust*, University of Roorkee, Narosa Publishing Home, 1996, pp 263-270.
[27] Feddema J., Lee C. and Mitchell O.R., "Weighted Selection of Image Features for Resolved Rate Visual Feedback Control," *IEEE Trans. Robot. Autom*, Vol. 7, 1991, pp 31-47.
[28] Craig J.J., *Introduction to Robotics Mechanics and Control*, Addison-Wesley, 1986.
[29] Hashimoto K., Kimoto T., Ebine T. and Kimura H., "Manipulator Control with Image – Based Visual Servo," *Proc. IEEE Inter. Conf. Robot. Autom*, 1991, pp. 2267-2272.
[30] Hassoun M.H., *Fundamentals of Artificial Neural Networks*, Prentice-Hall of India, 1998.
[31] Sciavicco L. and Siciliano B., *Modeling and Control of Robot Manipulators*, McGraw-Hill, 1996.