Strojniški vestnik - Journal of Mechanical Engineering 54(2008)9, 619-628
UDC 681.5:007.52
Paper received: 05.12.2007
Paper accepted: 07.07.2008

Uncalibrated Visual Servo Control with Neural Network

Rok Klobučar[1,*] - Jure Čas[1] - Riko Šafarič[1] - Miran Brezočnik[2]
[1] University of Maribor, Faculty of Electrical Engineering and Computer Science, Slovenia
[2] University of Maribor, Faculty of Mechanical Engineering, Slovenia

Research into robotics visual servo systems is an important content in the robotics field. This paper describes a control approach for a robotics manipulator. In this paper, a multilayer feedforward network is applied to a robot visual servo control problem. The model uses new neural network architecture and a new algorithm for modifying neural connection strength. No a-prior knowledge is required of robot kinematics and camera calibration. The network is trained using an end-effector position. After training, performance is measured by having the network generate joint-angles for arbitrary end effector trajectories. A 2-degrees-of-freedom (DOF) parallel manipulator was used for the study. It was discovered that neural networks provide a simple and effective way of controlling robotic tasks. This paper explores the application of a neural network for approximating nonlinear transformation relating to the robot's tip-position, from the image coordinates to its joint coordinates. Real experimental examples are given to illustrate the significance of this method. Experimental results are compared with a similar method called the Broyden method, for uncalibrated visual servo-control.

Keywords: robots, neural networks, visual servoing, parallel manipulator

0 INTRODUCTION

An animal's brain develops accurate sensory-motor coordination in the face of many unforeseen changes in body dimensions, strength of the muscles, and placement of the sensory organs. This is accomplished, for the most part, without a teacher [1]. Can this skill be implemented in to a robot's control system? This paper presents some new architecture regarding artificial neural network for visual servo control.

Visual servo algorithms have been extensively developed in the robotics field over the last ten years [2]. Vision is a useful robotics sensor since it mimics an animal's sense of vision [3]. Visual control of robots allows for non-contact measurements of the environment, as opposed to the traditional encoder and end limit switches [4]. Owing to the reduction in hardware costs and the increase in computing power, the focus of vision research has turned to introducing visual data into the control-loop of a robot. Using visual data within the control-loop is termed as 'visual servoing' [5]. Visual servoing is the fusion of many areas such as image processing, kinematics, dynamics, motion control, and real-time computing. The task during visual servoing is to control a robot when moving within its environment using vision.

The robot controller is required to solve the inverse kinematics problem in order to move the robot tip to a desired point or along a desired path. This problem involves the computation of a sequence of links angles that will position the robot tip at the desired location.

The computational complexity involved when numerically solving of the inverse kinematics problem, and the capability of neural networks to approximate arbitrary functions, has attracted many researchers into applying neural networks to this problem [6]. Most of these approaches use known solutions for forward kinematics or inverse kinematics problems in order to generate input-output patterns for the neural network training process.

A lot of authors have presented uncalibrated visual servoing and several groups have shown how image Jacobian itself can be estimated online from the measurements of robot and image motions. Hosoda and Asada [7] present an uncalibrated visual servoing for static targets using a fixed camera. Jägersand [8] takes the approach of a nonlinear least-squares optimization method using a trust-region method and Broyden estimation. Pieprmeier [9] develops a dynamic Broyden Jacobian estimation for a moving target, where a steady camera is used within the workspace.

*Corr. Author's Address: University of Maribor, Faculty of Electrical Engineering and Computer Science, Smetanova ulica 17, SI-2000 Maribor, Slovenia, rok.klobucar@uni-mb.si

Our task is to build a system for robot control that is completely independent of robot kinematics, camera calibration, and robust to external modifications. This paper proposes a visual servo-control with artificial neural network serving as a robot kinematics approximator. Our proposed method is compared with the Broyden algorithm for visual servoing, as developed by Jagersand [10].

The key attribute of neural networks is their ability to serve as a general nonlinear model. Thousands of data points are fed through a black box until the output of the network converges into the true system. It has been shown [11] that any function of practicable interest can be closely approximated arbitrarily with a neural network having enough neurons, at least one hidden layer, and an appropriate set of weights. The computation high speed and general modelling capability of neural networks are very attractive properties for nonlinear compensation problems, as indeed robot control problems are.

This paper proposes a method for IBVS (Image Based Visual Servoing) based on a neural network for solving nonlinear (dynamic and kinematic) systems, which takes control over the robot's joints in order to position the end effector into the static point or to track the moving target along an unknown trajectory. This paper is organized as follows. Section 2 discusses major classes of visual systems. Section 3 describes a manipulator control scheme. Section 4 describes Broyden's estimation and the definition of the feature Jacobian. Section 5 describes a visual servo control algorithm with artificial neural network. The experimental system and experimental results are shown in section 6. Finally section 7 summarizes the paper.

1 FUNDAMENTALS OF VISUAL SERVOING

One of the most basic design issues in any vision-based robotic system is the open or closed loop control. Many industrial systems use open-loop control. Open-loop control may be termed as a "look and move" kind of action, wherein, at each step, the system halts its action, processes the visual information and then executes the next step. Open-loop control may be used in cases where the vision processing system is too slow for real-time control.

Closed-loop control requires the visual data to be used as a feedback signal in the manipulator control and requires vision processing with acceptable speed and delay factors for real-time applications. Closed-loop control also allows visual data to compensate manipulator positioning inaccuracies and sensor noise.

Visual servo control systems typically use one of two camera configurations: end-effector mounted, or fixed in the workspace [12]. The first one is often called an eye-in-hand configuration. In this case, often constant relationship between the pose of the camera and the pose end-effector exists. The second configuration has the camera fixed in the workspace. In this case, the camera image of the target is independent of the robot's motion, unless the camera is connected to the second robot.

Classical approaches to visual servoing are position-based and image-based systems.

During position-based control, features are extracted from the image during iteration of the control loop and evolving of an estimate of the target's pose with respect to the camera. This error signal is computed as the difference between the current and desired poses. The advantage of position-based control is that it is possible to describe tasks in terms of positioning as Cartesian coordinates and the disadvantage is that it is often highly calibration-dependent. Hutchinson, Hager and Corke [3] contend that a key issue in position-based visual servo is the estimating of quantities to parameterize the feedback. Hence, it follows that position-based visual servoing is closely related to the problem of recovering scene geometry from the camera. Figure 1 shows position-based control.

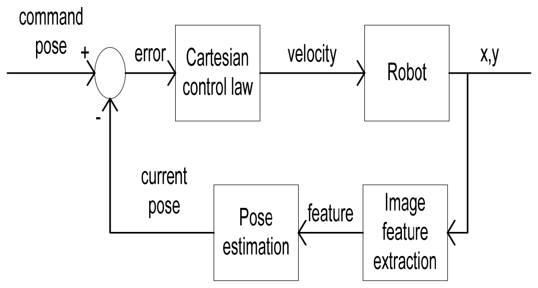

Fig.1. *Position based control*

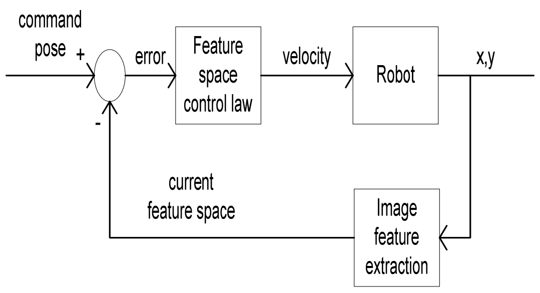

Fig.2. *Image based control*

Image-based control consists of specifying the positioning task directly from the image without an estimation of the target's pose. Figure 2 shows image based control. Feedback is computed from the image plan, and the error is computed as the difference between the desired feature vector and the current feature vector. Elements of the task are therefore specified in the image space rather than the world space. i.e in pixles rather than Cartesian coordinates [5]. The task of the image-based control is to reduce an appropriate error function. The error function is defined within the image parameter space and input to the robot is in the task space. The feature Jacobian captures these relationships, and it is a linear transformation from image feature parameters to changes in the robot's position.

The main advantage of image-based control over position-based control is that no camera calibration is needed.

2 MANIPULATOR CONTROL SCHEME

The objective is to move the robot tip of robotic manipulator towards a target point. Movement of the robot's arm is achieved by generating the control signals through the use of visual information and a neural network.

This method is evaluated by experiment using a 2DOF planar robot manipulator built at the Institute for Robotics, University of Maribor. The camera is fixed within the workspace and can provide positional information of the robot's tip and the target in the robot's workspace. Figure 3 shows the robot and the visual system model.

The robot's forward kinematics is given by the following equation:

$$\vec{x} = L \cdot \begin{bmatrix} \cos(q_1) + \cos(q_2) \\ \sin(q_1) + \sin(q_2) \end{bmatrix}, \tag{1}$$

where q_1, q_2 are the robot's joint angles, and $\vec{x}$ is a vector of the robot's tip coordinates in the Cartesian coordinates. Figure 4 shows the corresponding manipulator and its coordinate system.

$L = 0.415m$ is the length of the robot's link. The robot has four equal length links. The control system for our robot visual servoing experiment consists of two personal computers. The image processing node and the robot controller is interconnected by the 100 Mbit/s Ethernet. UDP (User Datagram Protocol) is used for exchange data between the PC control nodes. The robot controller is implemented with a DSpace DS1102 motion controller board which executes joint servo-control algorithms at 1ms period. The image processing node acquires the image from the camera, extracts the image features and executes the visual control algorithms. We employed a professional CCD camera. With a full image resolution 640x480, the AVT PIKE F-032B/C offers up to 202 fps and is, thus, particularly suited for fast applications in industrial image processing and product automation. The AVT FireGrab library for grabbing the image and computer vision, and the Gandalf numerical algorithm library were used as written in C language [13].

The relationship between the robot's joint angles and the robot's tip coordinates in the camera image is a non-linear function. The following section presents the known Broyden [8], [10] and [14] method, for uncalibrated visual servo-control.

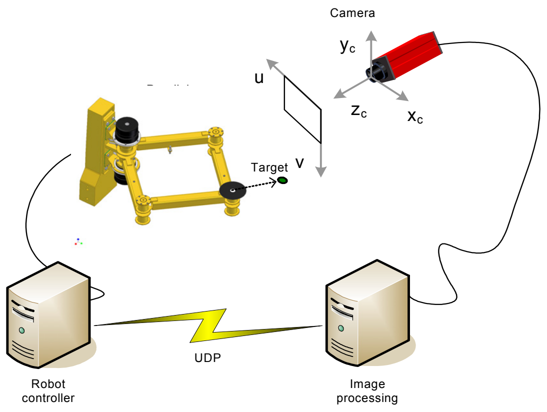

Fig.3. *Experimental system*

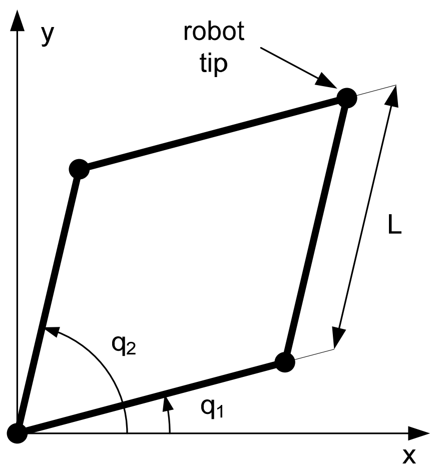

Fig.4. *Planar 2DOF manipulator*

3 THE BROYDEN ALGORITHM

The visual servoing problem has been formulated as a nonlinear least-squares problem [10], [9], and it could be solved using quasi-Newton methods, which consider the linear model, at each iteration:

$$L_k(x;B_k) = F(x_k) + B_k(x - x_k)\,. \tag{2}$$

The model approximates F(x) in the neighbourhood of x_k and computes x_{k+1} as a solution of the linear system $L_k(x;B_k) = 0$. Quasi-Newton methods can be summarized as methods based on the equation:

$$x_{k+1} = x_k - B_k^{-1}F(x_k)\,, \tag{3}$$

followed by computation of B_{k+1}. For the pure Newton method B_k the Jacobian of F is evaluated at x_k, that is a n x m matrix such that entry (i,j) is $\partial F_i / \partial x_j$:

$$B_k = J(x_k) = \nabla F(x_k)^T\,. \tag{4}$$

Broyden [14] proposed a class of quasi-Newton methods based on secant equations, imposing the linear model L_{k+1} to exactly match the nonlinear function at iterates x_k and x_{k+1}, that is

$$L_{k+1}(x_k) = F(x_k), \qquad L_{k+1}(x_{k+1}) = F(x_{k+1}). \tag{5}$$

Subtracting these two equations and defining $y_k = F(x_{k+1}) - F(x_k)$ and $s_k = x_{k+1} - x_k$ we obtain the classical secant equation:

$$B_{k+1}s_k = y_k\,. \tag{6}$$

If the dimension n is strictly greater than 1, there are an infinite number of matrices B_{k+1} satisfying(6). Applying the "least-change secant update", proposed by Broyden, leads to the following updated formula

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)s_k^T}{s_k^T s_k}\,. \tag{7}$$

3.1. The Control Scheme

The relationship between the velocity in the robot's joint space $\dot{q}$ and velocity of its end-effector $\dot{x}$ is called a robot Jacobian [15]:

$$\dot{x} = J_R\dot{q} \tag{8}$$

Similarly, the relationship between the velocity of a robot end-effector $\dot{x}$ and the velocity in the image feature space $\dot{f}$, is called image Jacobian:

$$\dot{f} = J_I\dot{x}\,. \tag{9}$$

A feature Jacobian J (10) is a compound of the robot Jacobian J_R and the image Jacobian J_I:

$$J = J_R J_I \tag{10}$$

The relationship between the velocity in a robot joint space and velocity in an image feature space is given by (11):

$$\dot{f} = J\dot{q}\,. \tag{11}$$

The visual algorithm determinates the joint's velocities $\dot{q}$:

$$\dot{q} = J^+ K\mathrm{e}\,, \tag{12}$$

where J^+ is the inverse of the feature Jacobian, K is a gain, and $e = f^d - f$ is the error signal that is obtained by comparing the desired and current image features' parameters. To obtain a vector norm we have to divide (the vector) by its length and thus we get a vector of length 1. The direction of the vector remains unchanged. The same vector is then multiplied with a gain. We norm a vector and multiply it with the gain only in the case where the length of the vector itself surpasses the given gain.

If the equation (12) is multiplied by J we get:

$$J\dot{q} = JJ^{+}Ke\,. \tag{13}$$

If the equation (13) is combined we get:

$$\dot{f} + Kf = Kf^{d}\,. \tag{14}$$

The Feature Jacobian J is obtained by the estimation process. The Broyden algorithm can be used for on-line estimation of the Feature Jacobian. The update equation of its estimate $\hat{J}$ is given by,

$$\hat{J}_{k+1} = \hat{J}_k + \left(\Delta f - J_k \Delta q\right)\Delta q^T\left(\Delta q^T \Delta q\right)^{-1}. \tag{15}$$

This method has proved to be successful. Jägersand [8] demonstrates the robust properties of this type of control and Piepmeier [9] develops a dynamic Broyden Jacobian estimation for moving target tracking.

Next section presents our proposed neural network method for uncalibrated visual servoing. Neural network has the capability to solve non-linear mapping, like direct and inverse kinematics and dynamics of the robot mechanism. Details of the learning and structure of the neural network will be presented in the next section.

4 NEURAL NETWORK STRUCTURE

Let us assume the proposed architecture of the neural network. A four-layered feedforward neural network is used (Figure 5).

It is described by the following equations:

$$\begin{aligned}
&\text{net\_OJ}^1_j = \sum_{a=1}^{i} w^1_{j,a}\cdot \text{in}_a,\\
&\text{OJ}^1_j = f\left(\text{net\_OJ}^1_j\right),\\
&\text{net\_OJ}^2_k = \sum_{a=1}^{j} w^2_{k,a}\cdot \text{OJ}^1_a,\\
&\text{OJ}^2_k = K_1\cdot f\left(\text{net\_OJ}^2_k\right),\\
&\text{net\_OJ}^3_m = \sum_{a=1}^{k} w^3_{m,a}\cdot \text{OJ}^2_a,\\
&\text{OJ}^3_m = f\left(\text{net\_OJ}^3_m\right),\\
&\text{net\_OJ}^4_n = \sum_{a=1}^{m} w^4_{n,a}\cdot \text{OJ}^3_a,\\
&\text{OJ}^4_n = K_2\cdot f\left(\text{net\_OJ}^4_n\right).
\end{aligned} \tag{16}$$

Each layer has a sigmoid function between 0 and 1 for its output. Equation (17) represents this sigmoid function, as shown in Figure 6:

$$f(\text{net}) = \frac{1}{1+e^{-\text{net}}} \tag{17}$$

The network computes all angles between the links in one step, using the desired and actual end-effector location as the network input. The network is trained by the back-propagating of a two errors equations (18). K1 and K2 are the adjustable output scale factors. K1 is set between -π/2 and π/2, this being the output limit. K2 is set between 0 and 640. It presents the range of image resolution. Initial weights have been set randomly between 0.004 and -0.004. The learning rate for the first η1 and the second η2 layer has been set at 0.4 and for the third η3 and the fourth η4 layer has been set at 0.9:

$$\begin{aligned}
\vec{e}_A &= \left(\vec{x}_R - \vec{x}_D\right),\\
\vec{e}_B &= \left(\vec{x}_D - \vec{x}_O\right).
\end{aligned} \tag{18}$$

The first error is the location error which is calculated from the desired and the actual position of the robot's tip. The second error is the location error that is calculated from the actual and the estimated positions of the robot's tip.

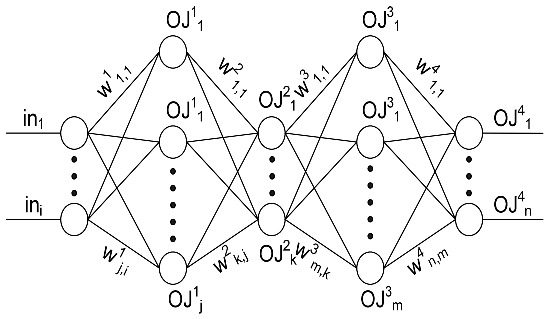

Fig.5. *Neural network*

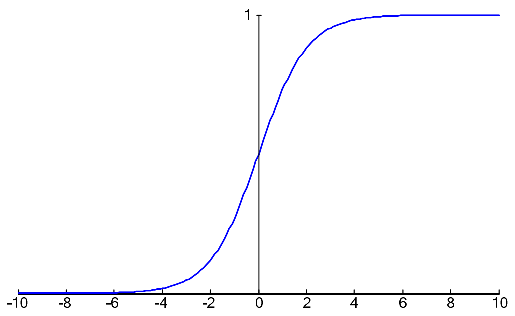

Fig.6. *Sigmoid function*

The back-propagation for error $\vec{e}_A$ and $\vec{e}_B$ is described by:

$$\begin{aligned}
\sigma_m^3 &= \sum_{a=1}^{n} w_{n,m}^4 \cdot e_{B,n}, \\
\sigma_k^2 &= \sum_{a=1}^{m} w_{m,k}^3 \cdot \sigma_m^3, \\
\Delta w_{m,k}^3 &= \eta_3 \cdot \sigma_m^3 \cdot OJ_m^3{}' \cdot OJ_k^2, \\
\Delta w_{n,m}^4 &= \eta_4 \cdot e_{B,n} \cdot OJ_n^4{}' \cdot OJ_m^3,
\end{aligned} \tag{19}$$

$$\begin{aligned}
\sigma_m^3 &= \sum_{a=1}^{n} w_{n,m}^4 \cdot e_{A,n}, \\
\sigma_k^2 &= \sum_{a=1}^{m} w_{m,k}^3 \cdot \sigma_m^3, \\
\sigma_j^1 &= \sum_{a=1}^{k} w_{k,j}^2 \cdot \sigma_k^2, \\
\Delta w_{j,i}^1 &= \eta_1 \cdot \sigma_j^1 \cdot OJ_j^1{}' \cdot in_i, \\
\Delta w_{k,j}^2 &= \eta_2 \cdot \sigma_k^2 \cdot OJ_k^2{}' \cdot OJ_j^1.
\end{aligned} \tag{20}$$

It is important, that this algorithm uses a new neural architecture. The architecture of the neural network is separated into two parts. The first part estimates the inverse kinematic while the second estimates the forward kinematic of the robot and the transformation to the camera image. Both neural networks are executed within the same step. It is important to know that this method also provides the capability of changing the camera's position within the executions of the tasks. Figure 7 shows the depicted architecture.

Unlike the forward kinematic problem, the inverse kinematic problem usually does not have a unique solution. Several joint positions may provide an identical end-effector position. In order to achieve appropriate results, neural networks require correct data preprocessing, architecture selection and network training.

Algorithm starts with random weights. The outputs of IKM NN (inverse kinematic model neural network) are actual joint coordinates, which are the result of random weights. The outputs of FKM NN (forward kinematic model neural network) are the estimated values of the robot tip's Cartesian coordinates.

We used the error BP (back-propagation) supervised learning technique. The first error is the difference between the estimated output values for the FKM NN and the actual values of the robot tip's Cartesian coordinates. The error BP technique helps to set up the weights for FKM NN. Learning for the IKM NN is executed within the same step. The second error is the difference between the actual values of the robot tip's Cartesian coordinates in the camera image and the desired Cartesian coordinates of the robot tip. The error BP technique helps to set up the weights through the FKM NN the IKM NN. Figure 8 represents the learning process while the figure 9 represents the execution process. Vector $\vec{q}$ represents the actual joint-angles. Vector $\vec{x}_{RC}$ represents the desired position in the camera frame. Vector $\vec{x}_{DC}$ represents the actual values of the robot tip in the image frame. Vector $\vec{x}_D$ represents the actual values of the robot tip's Cartesian coordinates.

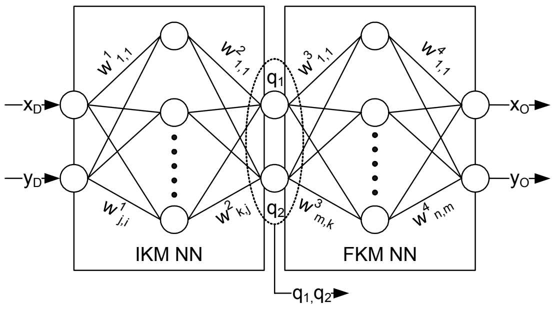

Fig.7. *Neural network sheme*

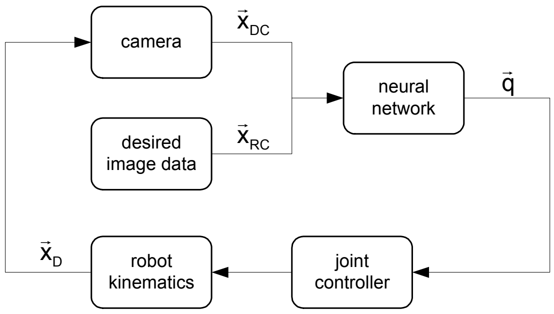

Fig.8. *Block diagram of the learning process*

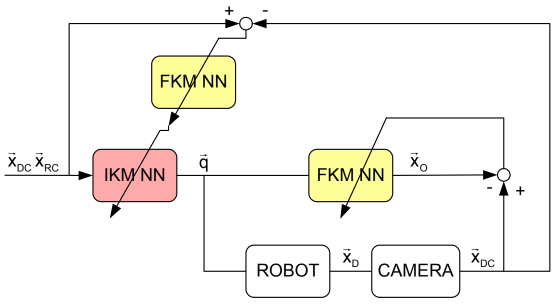

Fig.9. *Block diagram of the execution process*

5 EXPERIMENTAL RESULTS

The work presented in this paper is based on a real experiment using neural network and visual servo-control. The experimental task was compared with the known Broyden method.

The algorithm starts with random weights chosen between the limits 0.004 and -0.004. It could happen that, at the beginning when the initial weights are chosen, the robot's tip does not reach the target, so the algorithm should be started more often. When the robot's tip reached the first target, the first approximation of robot FKM was achieved and the weights for FKM NN had the first good approximation. The FKM NN weights were set better for each of the following targets. More targets in the robot's workspace brought better results, thus better FKM approximation. The algorithm was trained with 100 random targets in the robot's workspace. Then we froze up the weights for FKM NN. Only the IKM NN stayed active, which was then used for controlling the robot's tip and used for minimizing the distance between the target and the robot's tip. Figure 10 shows the trajectory of the robot's tip. When the FKM NN is learned and the weights are set, we could use it for robot tracking and robot positioning, without any camera calibration or known robot-kinematics.

The camera resolution was 640x480, the camera's frame rate was 200 f/s, thus the sampling time of the algorithm execution was 5 ms. The camera was positioned 1 m above the robot, thus the size of pixle was approximately 1.5 mm.

Figure (10) shows the start of the algorithm with random weights. This sample presents appropriately chosen weights. The initial robot tip's position is marked with "0" while the target point is marked with "1". The dotted line presents the robot in the goal position. The curve presents the trajectory of the robot's tip in the learning phase. The initial weights ware set-up randomly thus causing the problem that the robot's tip during the so-called "virgin" learning did not always find the target and the IKM computed by the neural network was badly approximated by the neural network. However, if the first approximation of IKM is achieved the robot's tip finds the target. Better results are given using more targets. It can be compared with teaching a baby to stand on his legs. How many times do children fall before they can stand up and walk without assistance? After crawling for a few weeks they achieve sufficient balance to stand on legs with the support of a wall. It is important to know how many senses a human being has. It all helps to set up the initial weights, which are used in future life.

Figure 11 shows robot tracking. The initial robot tip's position is marked with "0" and the corresponding robot joint-angles have the following values q_1 = -30°, q_2 = 60°. The initial target-point's position is marked with "1", and the consequent positions are marked "2", "3", and "4". The target points' positions were generated in the following order "1"-"2"-"3"-"4"-"1". When the robot's tip reached the target, the target-point was moved to another position in order to provide travelling of the robot's tip through the whole robot work-plane. The position of the target-point determined the corners of a square in the image plane. The linear interpolation generated the reference trajectory within the image. The trajectory tracking speed was set to the following value 0.1 pixle/sample time. This is the difference between reference and actual trajectories.

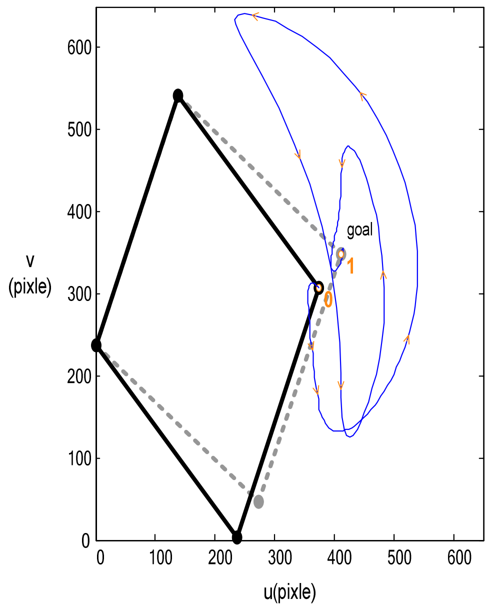

Fig.10. *Virgin learning trajectory*

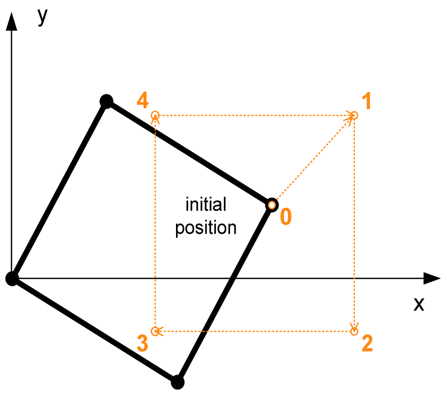

Fig.11. *Target movement*

Figure 12 shows the experimental results for robot-tracking obtained by the Broyden method (a,b) and the proposed method (c,d). Figures 12(b,d) show the experimental results for u and v feature error tracking's for both algorithms.

Both methods yield reliable results. Broyden methods needs initial movements to initialize the first Jacobian approximation. Our proposed method with neural network needs initial learning to set up weights and to estimate the system. Tests for both methods were carried out on the same test-bed under the same conditions.

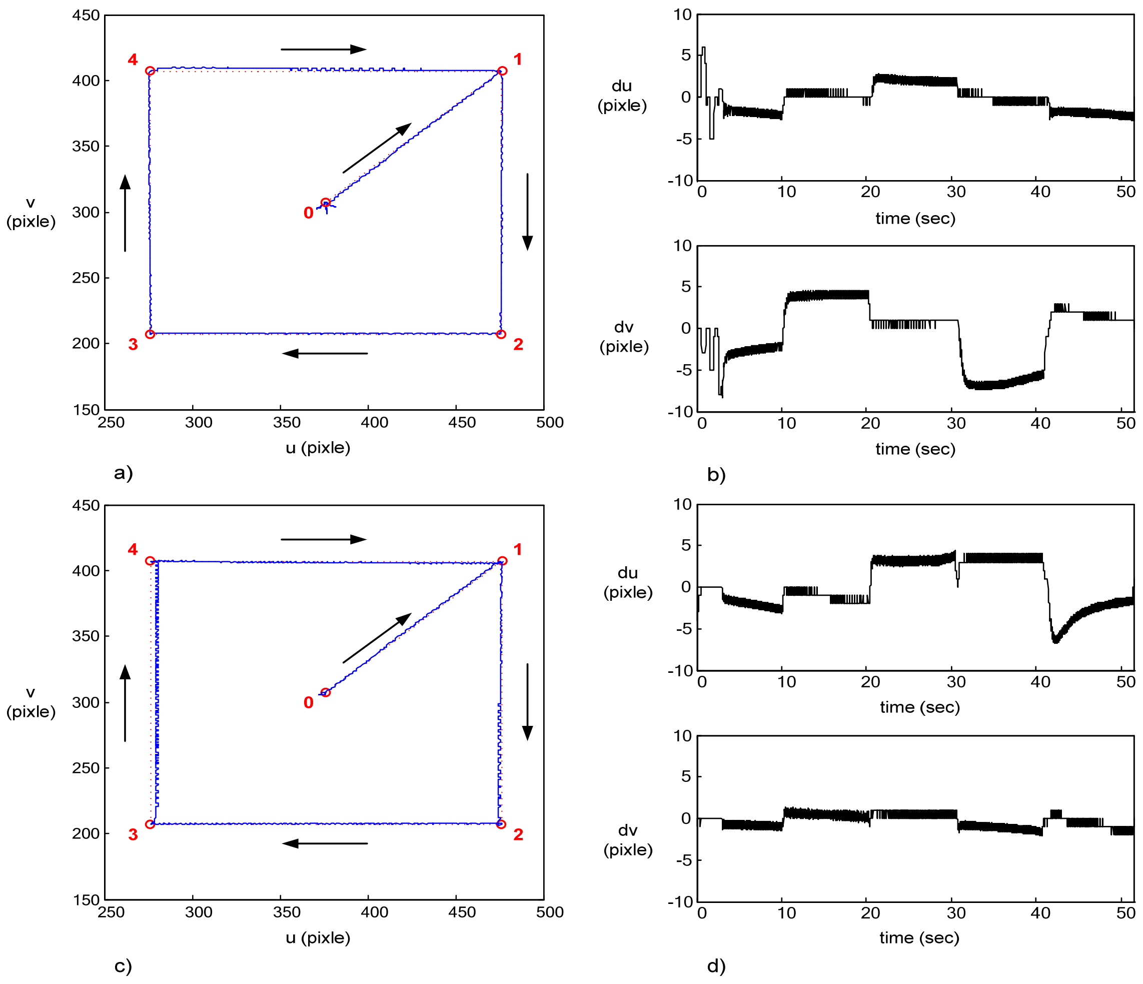

Fig. 12. Experimental results for robot tracking: a) and b) Broyden method, c) and d) proposed method with neural network
image contour : solid line – robot's tip, dotted line – reference, circle – target point
u-v error feature tracking: solid line – difference between reference and actual trajectories

6 CONCLUSION

This paper describes a scheme for the control of robot manipulators based on visual information. The learning control approach discussed is flexible, in that it is easy to apply within a robot control system and can be modified to accommodate system changes. Many learning control approaches involve learning to perform a particular movement. Our research focused on the robot control with two degrees of freedom and one camera. However, we predicted that the neural network algorithm would be able to be used on robots with more degrees of freedom and more cameras.

Achieving successful results for machine vision and robotics in unstructured environments, without using any a-priori camera or kinematic models, has proven hard yet there are many such environments where robots would be useful [8].

A new approach is introduced for the dynamic control of a parallel manipulator. This algorithm uses a completely new neural architecture. The training procedure of the neural network does not require known solutions for the inverse or forward kinematic problems. The neural algorithm can successfully estimate the robot's inverse kinematics and forward kinematics without any prior information, while carrying-out a 2 DOF parallel manipulation task.

7 REFERENCES

[1] Kuperstein, M. Generalized neural model for adaptive sensory-motor control of single postures. In: Proceedings of the IEEE International Conference on Robotics and Automation (1988), pp. 140–143.

[2] Espiau, B., Chaumette, F., Rives, P. A new approach to visual servoing in robotics. IEEE Transactions on Robotics and Automation, 8(3):313-326, 1992.

[3] Hutchinson, S., Hager, G., Corke P. A tutorial on visual servo control, IEEE Transactions on Robotics and Automation, (5), pp. 651–670, 1996.

[4] Soria, A., Garrido, R., Vazquez, R. Improving visual servoing control with high speed cameras, Electrical and Electronics Engineering, 2004.

[5] Lee, P., Dean, T., Yap, A., Walter, D., Kitchen,L., Barnes, N. On- Board Vision Using Visual-Servoing for RoboCup F-180 League Mobile Robots. RoboCup 2003.

[6] Gazit, R. Widrow, B. Backpropagation through links: a new approach to kinematic controlof serial manipulators, Intelligent Control, 1995.

[7] Hosoda, K., Asada, M. Versatile visual servoing withoutknowledge of true Jacobian, in Proc. IEEE/RSJ/GI Int. Conf.Intelligent Robots and Systems, Munich, Germany, Sept. 1994, pp. 186-193.

[8] Jägersand, M. Visual servoing using trust region methods and estimation of the full coupled visual-motor Jacobian, in Proc. IASTED Applications of Robotics and Control, 1996.

[9] Piepmeier J. A., McMurray G. V., Lipkin H. Uncalibrated dynamic visual servoing. IEEE Transactions on Robotics and Automation, 2004, 20(1), p.143-147.

[10] Jagersand, M., Nelson. R.C. On-line estimation of visual-motor models using active vision. In ARPA Image Understanding Workshop, p.677–682, 1996.

[11] Hornik, F. Multilayer feedforward networks are universal approximators, Neural Networks, vol. 2, pp. 359–363, 1989.

[12] Jägersand M., Nelson R. Adaptive Differential Visual Feedback for uncalibrated hand-eye coordination and motor control TR# 579, U. of Rochester 1994.

[13] http://gandalf-library.sourceforge.net/

[14] Broyden, C.G. A class of methods for solving nonlinear simultaneous equations, Mathematics of Computation 19, p. 577-593, 1965.

[15] Qingjie Zhao, Zengqi Sun, Hongbin Deng: Robot Visual Servoing Based on Total Jacobian. ASIAN 2004, p.271-285.