Presented at 1992 American Control Conf. (ACC), Chicago, June 24-26, 1992

ROBOTIC VISUAL SERVOING AROUND A STATIC TARGET: AN EXAMPLE OF CONTROLLED ACTIVE VISION

N. P. Papanikolopoulos and P. K. Khosla

Department of Electrical and Computer Engineering
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

This paper addresses the problem of robotic visual servoing (eye-in-hand configuration) around a static rigid target. The objective is to move the image projections of certain feature points of the static rigid target to some desired image positions. The eye-in-hand configuration consists of a CCD camera mounted on the end-effector of the robotic manipulator to provide visual measurements of the motion of the target's features. The vision algorithm is based on a cross-correlation technique, called SSD optical flow. The camera model introduces a number of parameters that must be estimated on-line. An adaptive control algorithm compensates for the servoing errors and the computational delays which are introduced by the time-consuming vision algorithms. Stability issues along with issues concerning the minimum number of required feature points are discussed. Experimental results are presented to verify the validity and the efficacy of the proposed algorithms.

1. Introduction

One of the biggest challenges of robotic visual servoing is the extension of robotic visual control to 3-D tasks. We choose to deal with a subproblem in this area, called robotic servoing around a static target (Fig. 1). This problem can be defined as "move the manipulator (the camera being mounted on the end-effector) such that the image projections of certain feature points of the target reach some desired image positions." Contrary to previous research efforts [1], we assume only partial knowledge of the inverse perspective transformation. In this paper, we address a problem which is an example of the *controlled active vision paradigm* that was introduced in [2]. This paradigm states that a controlled and not accidental motion of the camera can maximize the performance of any active vision algorithm. In order to achieve the objective of robotic visual servoing, computer vision techniques for the detection of motion are combined with appropriate control strategies. The result is the computation of the actuating signal for driving the manipulator. The problem is formulated from the systems theory point of view. An advantage of this approach is that the dynamics of the robotic device can be taken into account without changing the basic structure of the system. In order to circumvent the need to explicitly compute the depth map of the target, adaptive control techniques are proposed. In other words, the adaptive control algorithms compensate for the partially unknown inverse perspective transformation. Experimental results are presented to show the strengths and the weaknesses of the proposed approach.

The organization of this paper is as follows: Section 2 describes the mathematical formulation of the visual servoing problem. Section 3 gives an outline of the vision techniques (optical flow) used for the estimation of the positions of the features' image projections. The control and estimation strategies are discussed in Section 4. The experimental results are presented in Section 5. Finally, in Section 6, the paper is summarized.

2. Modeling of the Visual Servoing Problem

We assume a pinhole camera model with a frame R_s attached to it, and a perspective projection. Consider a static target with a feature located at a point P with coordinates (X_s, Y_s, Z_s) in R_s. The projection of this point on the image plane is the point p with image coordinates (x, y) given by

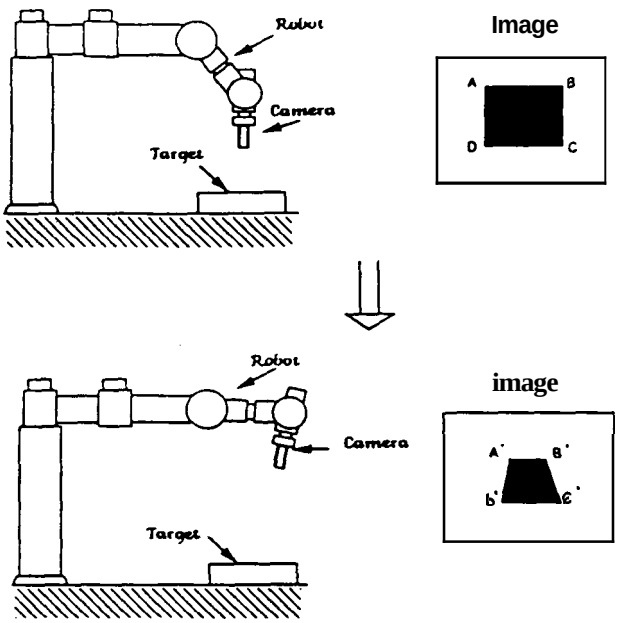


Figure 1: Task of Visual Servoing Around a Static Target.

$$x = \frac{fX_s}{Z_s s_x} \quad \text{and} \quad y = \frac{fY_s}{Z_s s_y} \tag{1}$$

where f is the focal length of the camera and s_x, s_y are the dimensions (mm/pixel) of the camera's pixels. In addition, it is assumed that $Z_s >> f$, and that the camera moves in a static environment with a translational velocity $T = (T_x, T_y, T_z)^T$ and with an angular velocity $R = (R_x, R_y, R_z)^T$ with respect to the camera frame R_s. The optical flow equations are [3]:

$$\dot{x} = u = x\frac{T_z}{Z_s} - \frac{fT_x}{Z_s s_x} + \frac{xys_y}{f}R_x - \left(\frac{f}{s_x} + \frac{x^2 s_x}{f}\right)R_y + \frac{ys_y}{s_x}R_z \tag{2}$$

$$\dot{y} = v = y\frac{T_z}{Z_s} - \frac{fT_y}{Z_s s_y} + \left(\frac{f}{s_y} + \frac{y^2 s_y}{f}\right)R_x - \frac{xys_x}{f}R_y - \frac{xs_x}{s_y}R_z \tag{3}$$

u and v are also known as the optical flow measurements. If we assume $s_x = s_y = f = 1$, equations (2)-(3) become:

$$u = \left[x\frac{T_z}{Z_s} - \frac{T_x}{Z_s}\right] + \left[xyR_x - (1+x^2)R_y + yR_z\right] \tag{4}$$

$$v = \left[y\frac{T_z}{Z_s} - \frac{T_y}{Z_s}\right] + \left[(1+y^2)R_x - xyR_y - xR_z\right] \tag{5}$$

In order to keep the notation simple and without any loss of generality, in the mathematical analysis that follows, we use only the relations described by (4)-(5). Consider now a neighborhood S_w of p in the image plane. Assume that the optical flow of the point p at time kT is $(u(kT), v(kT))$ where T is the time between two consecutive frames. It can be shown that at time kT, the optical flow is:

$$u(kT) = u_c(kT) \tag{6}$$

$$v(kT) = v_c(kT) \tag{7}$$

where $u_c(kT), v_c(kT)$ are the components of the optical flow induced at the time instant kT by the servoing motion of the camera. Without any loss of generality, equations (6) and (7) will be used with k instead of kT. Equations (6) and (7) do not include any computational delays that are associated with the computation and the realization of the servoing motion of the camera. If we include these delays in the model, equations (6) and (7) will be transformed to:

$$u(k) = q^{-d+1} u_c(k) \tag{8}$$

$$v(k) = q^{-d+1} v_c(k) \tag{9}$$

where d is the delay factor ($d \in \{1, 2, \ldots\}$). From the previous analysis, $u_c(k)$ and $v_c(k)$ are given by:

$$u_c(k) = \{ x(k) \frac{T_z(k)}{Z_c(k)} - \frac{T_x(k)}{Z_c(k)} \} + [x(k) y(k) R_x(k) - [1 + x^2(k)] R_y(k) + y(k) R_z(k)] \tag{10}$$

$$v_c(k) = \{ y(k) \frac{T_z(k)}{Z_c(k)} - \frac{T_y(k)}{Z_c(k)} \} + [[1 + y^2(k)] R_x(k) - x(k) y(k) R_y(k) - x(k) R_z(k)] \tag{11}$$

In addition, it is known [3] that:

$$u(k) = \frac{x(k+1) - x(k)}{T} \tag{12}$$

$$v(k) = \frac{y(k+1) - y(k)}{T} \tag{13}$$

If we substitute $u(k)$ and $v(k)$ in (8) and (9) with their equivalent expressions from (12) and (13), then equations (8) and (9) can be written as:

$$x(k+1) = x(k) + T q^{-d+1} u_c(k) \tag{14}$$

$$y(k+1) = y(k) + T q^{-d+1} v_c(k) \tag{15}$$

Further, if we model the inaccuracies of the model (neglected accelerations, inaccurate robot control) as white noise, (14) and (15) become

$$x(k+1) = x(k) + T q^{-d+1} u_c(k) + v_1(k) \tag{16}$$

$$y(k+1) = y(k) + T q^{-d+1} v_c(k) + v_2(k) \tag{17}$$

where $v_1(k), v_2(k)$ are zero-mean, mutually uncorrelated, stationary random variables with variances σ_1^2 and σ_2^2, respectively. The above equations can be written in the state-space form as:

$$\mathbf{x}_F(k+1) = \mathbf{A}_F(k)\mathbf{x}_F(k) + \mathbf{B}_F(k-d+1)\mathbf{u}(k-d+1) + \mathbf{H}_F(k)\mathbf{v}_F(k) \tag{18}$$

where $\mathbf{A}_F(k) = \mathbf{H}_F(k) = \mathbf{I}_2$, $\mathbf{x}_F(k) \in R^2$, $\mathbf{u}(k) \in R^6$, and $\mathbf{v}_F(k) \in R^2$. The matrix $\mathbf{B}_F(k) \in R^{2 \times 6}$ is:

$$\mathbf{B}_F(k) = T \begin{bmatrix} \frac{-1}{Z_c(k)} & 0 & \frac{x(k)}{Z_c(k)} & x(k)y(k) & -(1+x^2(k)) & y(k) \\ 0 & \frac{-1}{Z_c(k)} & \frac{y(k)}{Z_c(k)} & (1+y^2(k)) & -x(k)y(k) & -x(k) \end{bmatrix}$$

The vector $\mathbf{x}_F(k) = (x(k), y(k))^T$ is the state vector, $\mathbf{u}(k) = (T_x(k), T_y(k), T_z(k), R_x(k), R_y(k), R_z(k))^T$ is the control input vector, and $\mathbf{v}_F(k) = (v_1(k), v_2(k))^T$ is the white noise vector. The measurement vector $\mathbf{y}_F(k) = (y_1(k), y_2(k))^T$ for this feature is given by:

$$\mathbf{y}_F(k) = \mathbf{C}_F\mathbf{x}_F(k) + \mathbf{w}_F(k) \tag{19}$$

where $\mathbf{w}_F(k) = (w_1(k), w_2(k))^T$ is a white noise vector ($\mathbf{w}_F(k) \sim N(0, \mathbf{W})$) and $\mathbf{C}_F = \mathbf{I}_2$. The measurement vector is computed using the SSD algorithm which is described in Section 3.

One feature point is not enough for the calculation of the control input vector $\mathbf{u}(k)$ due to the fact that the number of outputs is less than the number of inputs. Thus, we are obliged to consider more points in our model. To make the number of inputs equal to the number of outputs, we should consider three feature points which are not collinear. The reason for the noncollinearity will be investigated in Section 4. Having more than three feature points will result in a larger number of outputs than inputs. Without additional constraints (knowledge of the 3-D model of the target etc.), it will be impossible to control the system so that all the outputs can track arbitrary desired values in the steady-state. In our approach, the robot-camera system is not required to take a certain pose with respect to the static rigid target. The only objective is to move a certain number of features to some desired positions on the image plane. Additional objectives such as a predefined pose require at least four feature points [1]. In our formulation the depth parameter of each one of the feature points is estimated on-line by an adaptive estimator, and therefore, the relative position of the object with respect to the robot-camera system can be computed.

The state-space model for three feature points can be written as:

$$\mathbf{x}(k+1) = \mathbf{A}(k)\mathbf{x}(k) + \mathbf{B}(k-d+1)\mathbf{u}(k-d+1) + \mathbf{H}(k)\mathbf{v}(k) \tag{20}$$

where $\mathbf{A}(k) = \mathbf{H}(k) = \mathbf{I}_6$, $\mathbf{x}(k) \in R^6$, and $\mathbf{v}(k) \in R^6$. The matrix $\mathbf{B}(k) \in R^{6 \times 6}$ is:

$$\mathbf{B}(k) = \begin{bmatrix} \mathbf{B}_F^{(1)}(k) \\ \mathbf{B}_F^{(2)}(k) \\ \mathbf{B}_F^{(3)}(k) \end{bmatrix}$$

The superscript (j) denotes each one of the feature points ($(j) \in \{(1), (2), (3)\}$). The vector $\mathbf{x}(k) = (x^{(1)}(k), y^{(1)}(k), x^{(2)}(k), y^{(2)}(k), x^{(3)}(k), y^{(3)}(k))^T$ is the new state vector, and $\mathbf{v}(k) = (v_1^{(1)}(k), v_2^{(1)}(k), v_1^{(2)}(k), v_2^{(2)}(k), v_1^{(3)}(k), v_2^{(3)}(k))^T$ is the new white noise vector. The new measurement vector $\mathbf{y}(k) = (y_1^{(1)}(k), y_2^{(1)}(k), y_1^{(2)}(k), y_2^{(2)}(k), y_1^{(3)}(k), y_2^{(3)}(k))^T$ for three features is given by:

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{w}(k) \tag{21}$$

where $\mathbf{w}(k) = (w_1^{(1)}(k), w_2^{(1)}(k), w_1^{(2)}(k), w_2^{(2)}(k), w_1^{(3)}(k), w_2^{(3)}(k))^T$ is the new white noise vector ($\mathbf{w}(k) \sim N(0, \mathbf{W})$) and $\mathbf{C} = \mathbf{I}_6$. More feature points can be integrated in our model by augmenting the block matrix $\mathbf{B}(k)$ and the measurement, state, and white noise vectors.

We can combine equations (20)-(21) into a MIMO (Multi-Input Multi-Output) ARX (AutoRegressive with eXternal input) model. This model consists of six MISO (Multi-Input Single-Output) ARX models. In addition, the model's equation is:

$$\mathbf{A}(k)(1 - q^{-1})\mathbf{y}(k) = \mathbf{B}(k-d)\mathbf{u}(k-d) + \mathbf{n}(k) \tag{22}$$

where $\mathbf{n}(k)$ is the white noise vector. The new white noise vector $\mathbf{n}(k)$ corresponds to the measurement noise, modeling errors, and noise introduced by inaccurate robot control. In the next section, we will examine the way we obtain the position of the features' projections on the image plane.

3. Update of the Features' Image Projections

The continuous extraction of the positions of the features' projections on the image plane is based on optical flow techniques (u and v are the optical flow components). For accuracy reasons, we use a modified version of the matching based technique [4] also known as the sum-of-squared differences (SSD) optical flow. For every point $\mathbf{p}_A = (x_A, y_A)$ in image A, we want to find the point $\mathbf{p}_B = (x_A + u, y_A + v)$ to which the point $\mathbf{p}_A$ moves in image B. It is assumed that the intensity values in the neighborhood L of $\mathbf{p}_A$ remain almost constant over time, that the point $\mathbf{p}_B$ is within an area S of $\mathbf{p}_A$, and that velocities are normalized by the time period T to get the displacements. Thus, for the point $\mathbf{p}_A$ the SSD estimator selects the displacement $\mathbf{d} = (u, v)$ that minimizes the SSD measure:

$$\epsilon(\mathbf{p}_A, \mathbf{d}) = \sum_{m,n \in N} [I_A(x_A + m, y_A + n) - I_B(x_A + m + u, y_A + n + v)]^2 \tag{23}$$

where $u, v \in S$, N is an area around the pixel we are interested in, and I_A, I_B are the intensity functions in images A and B respectively. Variations of the previous technique are used in our experiments. In the first variation, image A is the first image ($k = 0$) acquired by the camera while image B is the current image ($k \neq 0$). Thus, for the point $\mathbf{p}_A$ the SSD estimator selects the displacement $\mathbf{d} = (u, v)$ that minimizes the SSD measure:

$$\epsilon(\mathbf{p}_A, \mathbf{d}) = \sum_{m,n \in N} [I_A(x_A + m, y_A + n) - I_B(x_A + m + su + m, y_A + n + sv + n)]^2 \tag{24}$$

where su and sv are the sums of the all the previously measured displacements and defined as:

$$su = \sum_{j=1}^{j=k-1} u(j), \quad sv = \sum_{j=1}^{j=k-1} v(j) \tag{25}$$

This variation of the SSD technique is sensitive to large rotations and changes in the lighting. Another variation of the SSD is the one that updates image A every μ images. This SSD measure is similar to the one previously mentioned (Eq. (24)) except for the fact that su and sv are differently defined. The terms su and sv have the following definition:

$$su = \sum_{j=\mu l+1}^{j=k-1} u(j), \quad sv = \sum_{j=\mu l+1}^{j=k-1} v(j), \text{ and } l=\lfloor k/\mu \rfloor \tag{26}$$

The most efficient variation of the SSD in terms of accuracy and computational complexity proved to be the last one. The continuous computation of the displacement vectors helps us to continuously update the coordinates of the image projections of the feature points.

The next step in our algorithm involves the use of these measurements in the visual servoing process. These measurements should be transformed into cartesian control commands for the robotic system.

4. Control, Estimation, and Stability

The control objective is to move the manipulator in such a way that the projections of the selected features on the image plane move to some desired position (Fig. 1). This section presents the control strategies that realize this motion, the estimation scheme used to estimate the unknown parameters of the model, and the stability analysis of the proposed visual servoing algorithms.

Since the depth information is not directly available, adaptive control techniques can be used for visually servoing around a static object. Adaptive control techniques are used for the recovery of the components of the translational and rotational velocity vectors $\mathbf{T}(k)$ and $\mathbf{R}(k)$ respectively. These adaptive control techniques are based on the estimated and not the actual values of the system's parameters. This approach is often called *certainty equivalence adaptive control* [5]. A large number of algorithms can be generated, depending on the choice of a parameter estimation scheme and the control law. The rest of the section will be devoted to the detailed description of the control and estimation schemes.

4.1. Design of the Controller

The control objective is to move the features' projections on the image plane to some desired positions. The repositioning of the projections is realized by an appropriate motion of the camera. A simple control law can be derived by the minimization of a cost function that includes the control signal:

$$J(k+d) = [\mathbf{y}(k+d) - \mathbf{y}^*(k+d)]^T \mathbf{Q} [\mathbf{y}(k+d) - \mathbf{y}^*(k+d)] + \mathbf{u}^T(k) \mathbf{L} \mathbf{u}(k) \tag{27}$$

The vector $\mathbf{y}^*(k)$ represents the desired positions of the projections of the three features on the image plane. In our experiments, the vector $\mathbf{y}^*(k)$ is known *a priori* and is constant over time. By weighting the control signal, we place some emphasis on the minimization of the control signal in addition to the minimization of the servoing error. The response of the system is slower than having $\mathbf{L}=\mathbf{0}$ but the control input signal is bounded and feasible. This is in agreement with the structural and operational characteristics of the robotic system and the vision algorithm. A robotic system cannot track signals that command large changes in the features' image projections during the sampling interval T. In addition, the optical flow algorithm cannot detect displacements larger than 15 pixels per sampling interval T. The control law which is derived from the minimization of the cost function (27) is:

$$\mathbf{u}(k) = -[\mathbf{B}^T(k) \mathbf{Q} \mathbf{B}(k) + \mathbf{L}]^{-1} \mathbf{B}^T(k) \mathbf{Q} \{ [\mathbf{y}(k) - \mathbf{y}^*(k+d)] + \sum_{m=1}^{m=d-1} \mathbf{B}(k-m) \mathbf{u}(k-m) \} \tag{28}$$

The design parameters in this control law are the elements of the matrices Q and L. If the matrix $\mathbf{B}(k)$ is full rank then the matrix $[\mathbf{B}^T(k) \mathbf{Q} \mathbf{B}(k) + \mathbf{L}]$ is invertible. On the other hand, the matrix $\mathbf{B}(k)$ is singular when the three feature points art collinear [6]. Therefore, in order for the matrix $B(k)$ to be nonsingular, the three feature points *should not* satisfy the following equalities:

$$\text{if } Z_s^{(2)}(k) \neq Z_s^{(1)}(k)$$
$$\frac{Y_s^{(3)}(k) - Y_s^{(1)}(k)}{Y_s^{(2)}(k) - Y_s^{(1)}(k)} = \frac{X_s^{(3)}(k) - X_s^{(1)}(k)}{X_s^{(2)}(k) - X_s^{(1)}(k)} = \frac{Z_s^{(3)}(k) - Z_s^{(1)}(k)}{Z_s^{(2)}(k) - Z_s^{(1)}(k)} \tag{29}$$

$$\text{or}$$
$$\text{if } Z_s^{(2)}(k) = Z_s^{(1)}(k)$$
$$\frac{Y_s^{(2)}(k) - Y_s^{(1)}(k)}{Y_s^{(3)}(k) - Y_s^{(1)}(k)} = \frac{X_s^{(2)}(k) - X_s^{(1)}(k)}{X_s^{(3)}(k) - X_s^{(1)}(k)} = \frac{Z_s^{(2)}(k) - Z_s^{(1)}(k)}{Z_s^{(2)}(k) - Z_s^{(1)}(k)} \tag{30}$$

In addition, $\mathbf{B}(k)$ becomes singular if $Z_s^{(1)}(k) = Z_s^{(2)}(k) = Z_s^{(3)}(k)$ and at least one of the feature points has a projection on the image plane with coordinates $x^{(j)}(k) = y^{(j)}(k) = 0$ $((j) \in \{(1),(2),(3)\})$. A mathematical proof of the fact that the previous conditions make $\mathbf{B}(k)$ singular can be found in [7]. In addition, [7] illustrates some additional conditions that make $\mathbf{B}(k)$ singular.

By selecting L and Q, one can place more or less emphasis on the control input and the servoing error. There is no standard procedure for the selection of the elements of these matrices. One technique [8] is the optimization approach. Assume that e_{id} $(i = 1, \ldots, 6)$ is the desired maximal servoing error $y_i(k) - y_i^*(k)$ which corresponds to the i-component of the $\mathbf{y}(k)$ vector and T_{xmax}, T_{ymax}, T_{zmax}, R_{xmax}, R_{ymax}, and R_{zmax} are the maximal control amplitudes of T_x, T_y, T_z, R_x, R_y, and R_z respectively. Then, we can choose $\mathbf{Q} = diag\{e_{1d}^{-2}, \ldots, e_{6d}^{-2}\}$ and $\mathbf{L} = diag\{T_{xmax}^{-2}, T_{ymax}^{-2}, \ldots, R_{zmax}^{-2}\}$. In this way, the constraints that the robotic device imposes on the maximal control amplitudes are included in the control law. It is not possible to have infinite maximal control amplitudes. The same is true for the servoing errors. As the positions of the projections of the features on the image plane are measured directly by the SSD algorithm, there is a certain range of values that can be measured. If we want to include the noise of our model and the inaccuracy of the $\mathbf{B}(k)$ matrix in our control law, the control objective (27) will become:

$$J(k+d) = E\{[\mathbf{y}(k+d) - \mathbf{y}^*(k+d)]^T \mathbf{Q} [\mathbf{y}(k+d) - \mathbf{y}^*(k+d)] + \mathbf{u}^T(k) \mathbf{L} \mathbf{u}(k) | F_k\} \tag{31}$$

where the symbol $E\{X\}$ denotes the expected value of the random variable X and F_k is the sigma algebra generated by the past measurements and the past control inputs up to time k. The new control law is:

$$\mathbf{u}(k) = -[\hat{\mathbf{B}}^T(k) \mathbf{Q} \hat{\mathbf{B}}(k) + \mathbf{L}]^{-1} \hat{\mathbf{B}}^T(k) \mathbf{Q} \{ [\mathbf{y}(k) - \mathbf{y}^*(k+d)] + \sum_{m=1}^{m=d-1} \hat{\mathbf{B}}(k-m) \mathbf{u}(k-m) \} \tag{32}$$

where $\hat{\mathbf{B}}(k)$ is the estimated value of the matrix $\mathbf{B}(k)$. The matrix $\hat{\mathbf{B}}(k)$ is dependent on the estimated values of the features' depth $\hat{Z}_s^{(j)}(k)$ $((j) \in \{(1),(2),(3)\})$ and the coordinates of the features' image projections. In particular, the matrix $\hat{\mathbf{B}}(k)$ is defined as follows:

$$\mathbf{B}(k) = \begin{bmatrix} \hat{\mathbf{B}}_F^{(1)}(k) \\ \hat{\mathbf{B}}_F^{(2)}(k) \\ \hat{\mathbf{B}}_F^{(3)}(k) \end{bmatrix}$$

where $\hat{\mathbf{B}}_F^{(j)}(k)$ is given by:

$$\hat{\mathbf{B}}_F^{(j)}(k) = T \begin{bmatrix} \frac{-1}{\hat{Z}_s^{(j)}(k)} & 0 & \frac{x^{(j)}(k)}{\hat{Z}_s^{(j)}(k)} & x^{(j)}(k) y^{(j)}(k) & -(1 + (x^{(j)}(k))^2) & y^{(j)}(k) \\ 0 & \frac{-1}{\hat{Z}_s^{(j)}(k)} & \frac{y^{(j)}(k)}{\hat{Z}_s^{(j)}(k)} & (1 + (y^{(j)}(k))^2) & -x^{(j)}(k) y^{(j)}(k) & -x^{(j)}(k) \end{bmatrix}$$

4.2 Estimation Techniques

The estimation of the feature's depth $Z_s^{(j)}(k)$ with respect to the camera frame can be done in multiple ways. In this section, we present some of these algorithms. Let's define the inverse of the depth $Z_s^{(j)}(k)$ as $\zeta_s^{(j)}(k)$.

Then, the equations (18)-(19) of ith feature point can be rewritten as $(n_F^{(i)}(k) \sim N(0, N^{(i)}(k))$:

$$y_F^{(i)}(k) = A_F^{(i)}(k-1)\, y_F^{(i)}(k-1) + \zeta_z^{(i)}(k-d)\, B_{F1}^{(i)}(k-d)\, T(k-d) + + B_{F2}^{(i)}(k-d)\, R(k-d) + n_F^{(i)}(k) \quad (33)$$

where $B_{F1}^{(i)}(k)$ and $B_{F2}^{(i)}(k)$ are given by:

$$B_{F1}^{(i)}(k) = T \begin{bmatrix} -1 & 0 & x^{(i)}(k) \\ 0 & -1 & y^{(i)}(k) \end{bmatrix},$$

$$B_{F2}^{(i)}(k) = T \begin{bmatrix} x^{(i)}(k)\, y^{(i)}(k) & -[1+(x^{(i)}(k))^2] & y^{(i)}(k) \\ [1+(y^{(i)}(k))^2] & -x^{(i)}(k)\, y^{(i)}(k) & -x^{(i)}(k) \end{bmatrix}.$$

By defining $u_t^{(i)}(k)$ and $u_r^{(i)}(k)$ as $u_t^{(i)}(k) = B_{F1}^{(i)}(k)\, T(k)$ and $u_r^{(i)}(k) = B_{F2}^{(i)}(k)\, R(k)$, respectively, equation (33) is transformed into:

$$y_F^{(i)}(k) = A_F^{(i)}(k-1)\, y_F^{(i)}(k-1) + \zeta_z^{(i)}(k-d)\, u_t^{(i)}(k-d) + + u_r^{(i)}(k-d) + n_F^{(i)}(k) \quad (34)$$

A last transformation of equation (34) is done by using the vector $\Delta y_F^{(i)}(k)$ which is defined as:

$$\Delta y_F^{(i)}(k) = y_F^{(i)}(k) - y_F^{(i)}(k-1) - u_r^{(i)}(k-d)$$

The new form of the equation (34) is:

$$\Delta y_F^{(i)}(k) = \zeta_z^{(i)}(k-d)\, u_t^{(i)}(k-d) + n_F^{(i)}(k) \quad (35)$$

The vectors $\Delta y_F^{(i)}(k)$ and $u_t^{(i)}(k-d)$ are known every instant of time, while the scalar $\zeta_z^{(i)}(k)$ is continuously estimated. It is assumed that an initial estimate $\hat{\zeta}_z^{(i)}(0)$ of $\zeta_z^{(i)}(0)$ is given and $p^{(i)}(0) = E[(\zeta_z^{(i)}(0) - \hat{\zeta}_z^{(i)}(0))^2]$ is a positive scalar p_0. $p^{(i)}(0)$ can be interpreted as a measure of the confidence that we have in the initial estimate $\hat{\zeta}_z^{(i)}(0)$. Accurate knowledge of the scalar $\zeta_z^{(i)}(k)$ corresponds to a small covariance scalar p_0. In our examples, $N^{(i)}(k)$ is a constant predefined matrix. In addition, for simplicity in notation, $h(k)$ is used instead of $u_t^{(i)}(k)$.

The estimation equations are (the superscript '-' denotes the predicted value of a variable while the superscript '+' denotes its updated value) [9]:

$$^{-}\hat{\zeta}_z^{(i)}(k) = {}^{+}\hat{\zeta}_z^{(i)}(k-1) \quad (36)$$

$$^{-}p^{(i)}(k) = {}^{+}p^{(i)}(k-1) + s^{(i)}(k-1) \quad (37)$$

$$^{+}p^{(i)}(k) = [\,\{^{-}p^{(i)}(k)\}^{-1} + h^T(k-d)\,\{N^{(i)}(k)\}^{-1} h(k-d)]^{-1} \quad (38)$$

$$k^T(k) = {}^{+}p^{(i)}(k)\, h^T(k-d)\,\{N^{(i)}(k)\}^{-1} \quad (39)$$

$$^{+}\hat{\zeta}_z^{(i)}(k) = {}^{-}\hat{\zeta}_z^{(i)}(k) + k^T(k)\,[\Delta y_F^{(i)}(k) - {}^{-}\hat{\zeta}_z^{(i)}(k)\, h(k-d)] \quad (40)$$

where $s^{(i)}(k)$ is a covariance scalar which corresponds to the white noise that characterizes the transition between the states. The depth related parameter $\zeta_z^{(i)}(k)$ is a time-varying variable since the camera translates along its optical axis and rotates along the X and Y axis. The estimation scheme of equations (36)-(40) can compensate for the time-varying nature of $\zeta_z^{(i)}(k)$ because it is designed under the assumption that the estimated variable undergoes a random change. One problem is to keep the covariance scalar $p^{(i)}(k)$ finite. Solutions for this type of problem can be found in [5]. In addition, we implement some other estimation techniques which deal with time-varying parameters. The first implemented technique is called *exponential data weighting* [5]. In this case, we assume that the most recent data contains more information than past data and, therefore, old data is exponentially discarded. A second useful technique is *covariance resetting*. In this case, the covariance scalar $p^{(i)}(k)$ is reset when the estimated variable is drastically changed.

Matthies et al. [10] proposed the use of a more accurate form for the state update of $\zeta_z^{(i)}(k)$. This form is based on an equation [10] which provides the change in the feature's depth $Z_z^{(i)}(k)$ between two time instances given the feature's image coordinates and the camera motion. This equation can be written as (computational delays are included):

$$Z_z^{(i)}(k) = Z_z^{(i)}(k-1) - \{\, T_z(k-d) + \{\, R_x(k-d)\, y^{(i)}(k-d) - R_y(k-d)\, x^{(i)}(k-d)\,]\, Z_z^{(i)}(k-d)\}\, T \quad (41)$$

By inverting the terms of the previous equation (41), the following equation is derived:

$$\zeta_z^{(i)}(k) = \zeta_z^{(i)}(k-1) / \{\, 1 - T[\, T_z(k-d)\, \zeta_z^{(i)}(k-1) + + \{\, R_x(k-d)\, y^{(i)}(k-d) - R_y(k-d)\, x^{(i)}(k-d)\,] \frac{\zeta_z^{(i)}(k-1)}{\zeta_z^{(i)}(k-d)}]\} \quad (42)$$

If the values $\zeta_z^{(i)}(k)$ are substituted by their estimates, equation (42) will be transformed into:

$$^{-}\dot{\zeta}_z^{(i)}(k) = {}^{+}\dot{\zeta}_z^{(i)}(k-1) / \{\, 1 - T[\, T_z(k-d)\, {}^{+}\dot{\zeta}_z^{(i)}(k-1) + + \{\, R_x(k-d)\, y^{(i)}(k-d) - R_y(k-d)\, x^{(i)}(k-d)\,] \frac{{}^{+}\dot{\zeta}_z^{(i)}(k-1)}{{}^{+}\dot{\zeta}_z^{(i)}(k-d)}]\} \quad (43)$$

In addition, equation (37) should be modified to incorporate the new equation for the updates of states. In the experiments, the improvement in the accuracy of the estimated values from the use of the complex form (43) is minimal. Thus, the majority of the experiments are performed by using the estimation equations (36)-(40). These equations require the estimation of one parameter per feature-point and therefore, the real-time implementation of the estimation scheme is feasible. In addition, we implement an estimation scheme that computes two parameters per feature point. This scheme is a variation of the previous estimation scheme and separately estimates the depth related parameter $\zeta_z^{(i)}(k)$ in the X and Y directions on the image plane. In theory, this formulation can handle the estimation of the depth related parameters with more accuracy. The subscript i denotes the X or Y direction. The estimation equations for each feature point are:

$$^{-}\dot{\zeta}_{zi}^{(i)}(k) = {}^{+}\dot{\zeta}_{zi}^{(i)}(k-1) \quad i = 1, 2 \quad (44)$$

$$^{-}p_i^{(i)}(k) = {}^{+}p_i^{(i)}(k-1) + s_i^{(i)}(k-1) \quad i = 1, 2 \quad (45)$$

$$^{+}p_i^{(i)}(k) = [\,\{^{-}p_i^{(i)}(k)\}^{-1} + h_i(k-d)\,\{n_i^{(i)}(k)\}^{-1} h_i(k-d)]^{-1} \quad i = 1, 2 \quad (46)$$

$$\kappa_i(k) = {}^{+}p_i^{(i)}(k)\, h_i(k-d)\,\{n_i^{(i)}(k)\}^{-1} \quad i = 1, 2 \quad (47)$$

$$^{+}\dot{\zeta}_{zi}^{(i)}(k) = {}^{-}\dot{\zeta}_{zi}^{(i)}(k) + \kappa_i(k)\,[\Delta y_{Fi}^{(i)}(k) - {}^{-}\dot{\zeta}_{zi}^{(i)}(k)\, h_i(k-d)] \quad i = 1, 2 \quad (48)$$

where $\Delta y_{Fi}^{(i)}(k)$ and $h_i(k)$ denote the X or Y components of the vectors $\Delta y_F^{(i)}(k)$ and $h(k)$, respectively. In practice, the experimental results from the implementation of this estimation scheme prove to be comparable with the results of the first estimation scheme. Some researchers [11] have proposed the use of an adaptive scheme that estimates all the elements of the block matrix $B(k)$ on-line. This approach is computationally expensive and not necessary.

4.3. Stability Analysis

In this section a stability analysis is presented for the proposed algorithms. We investigate the conditions under which the servoing error $(e(k) = y(k) - y^*(k))$ asymptotically goes to zero while the system input vector $u(k)$ and the system output vector $y(k)$ remain bounded. In 1980, Goodwin et al. [12] dealt with the stability analysis of adaptive algorithms for discrete-time deterministic time-invariant MIMO systems. Using Goodwin's work as a base, we present an outline of the stability analysis for our discrete-time stochastic nonlinear slowly time-varying MIMO system. In this analysis, the 2-norm of a matrix Γ, often called the *spectral* norm, is used. This norm is defined as follows:

$$\|\Gamma\|_2 = \max \frac{\|\Gamma x\|_2}{\|x\|_2} = (\textit{maximum eigenvalue of } \Gamma^T\Gamma)^{\frac{1}{2}} \quad (x \neq 0) \quad (49)$$

We define the maximum eigenvalue of the matrix Γ as $\lambda_{max}(\Gamma)$ while the minimum eigenvalue of the matrix Γ is defined as $\lambda_{min}(\Gamma)$. For a deterministic version of our model (white noise is ignored) and for a system delay $d = 1$, the error equation is ($y^*(k)$ is known a priori and is constant over time):

$$e(k+1) = [I_6 - B(k)\, M(k)]\, e(k) \quad (50)$$

where $M(k)$ is defined as:

$$M(k) = [\hat{B}^T(k) Q \hat{B}(k) + L]^{-1} \hat{B}^T(k) Q . \quad (51)$$

The servoing error goes asymptotically to zero if the following condition holds:

$$\| I_6 - B(k) M(k) \|_2 < 1 \quad (52)$$

The previous condition can be rewritten as:

$$\lambda_{max}(I_6 - B(k) M(k) - B^T(k) M^T(k) + B^T(k) M^T(k) B(k) M(k)) < 1 \quad (53)$$

After some simple matrix computations, the previous condition is transformed to:

$$\lambda_{min}(B(k) M(k) + B^T(k) M^T(k) - B^T(k) M^T(k) B(k) M(k)) > 0 \quad (54)$$

Therefore, the matrix $B(k) M(k) + B^T(k) M^T(k) - B^T(k) M^T(k) B(k) M(k)$ should be strictly positive definite. In the case that $L = 0$, the following condition should hold:

$$B(k) \hat{B}^{-1}(k) + B^T(k) [\hat{B}^T(k)]^{-1} - B^T(k) [\hat{B}^T(k)]^{-1} B(k) \hat{B}^{-1}(k) > 0 \quad (55)$$

In continuous time, the condition becomes simpler. Chaumette states [13] that the matrix $B(t) \hat{B}^{-1}(t)$ should be positive definite. For $d > 1$, the error equation is more complex than the case of unit delay because previous control input vectors are included. The new error equation is given by:

$$e(k+1) = [I_6 - B(k+1-d) M(k+1-d)] e(k) + B(k+1-d) M(k+1-d) \sum_{m=1}^{m=d-1} [B(k+1-d-m) - \hat{B}(k+1-d-m)] u(k+1-d-m) \quad (56)$$

The $M(k)$ is again given by (51). For $L = 0$, $M(k)$ is equal to $\hat{B}^{-1}(k)$. This implies that if $\hat{B}(k)$ asymptotically goes to $B(k)$, then the servoing error asymptotically goes to zero. This can be concluded from the error equation (56). In order for $\hat{B}(k)$ to converge to $B(k)$ [5], the input signal $u(k)$ should be *Persistently Exciting* (PE). Goodwin [5] proposed several methods for generating *persistently exciting* input signals.

More complex stability proofs can be created by using discrete Lyapunov functions and the properties of the estimation scheme. In this way, we can guarantee stability of the adaptive control algorithms under weaker conditions. The proper selection of initial and target feature points as well as the selection of L, along with the careful design of the estimation scheme can guarantee continuous nonsingularity of the matrix $\hat{B}^T(k) Q \hat{B}(k) + L$ as described in the experimental section of this paper.

5. Experiments

The theory was verified by performing a number of experiments on the CMU DD Arm II (Direct-Drive Arm II) robotic system. A detailed description of the hardware configuration of CMU DD Arm II is given in [3]. The camera is mounted on the end-effector. The real images are 510x492 and are quantized to 256 gray levels. The focal length of the camera is 7.5 mm and the objects are static (the initial depth of the objects' center of mass with respect to the camera frame Z_c is varying from 500 mm to 1000 mm). The camera's pixel dimensions are: s_x=0.01278 mm/pixel and s_y=0.00986 mm/pixel. The maximum permissible translational velocity of the end-effector is 10 cm/sec, and each one of the components (roll, pitch, yaw) of the end-effector's rotational velocity must not exceed 0.05 rad/sec.

Our objective is to move the manipulator so that the image projections of features of the object move to desired positions in the image. The objects used in the servoing examples are books, pencils, items with distinct features (Fig. 2). The user, by using the mouse, proposes to the system some of the object's features. Then, the system evaluates on-line the quality of the features based on the confidence measures described in [3]. The same operation can be done automatically by a computer process that runs once for approximately 2 to 3 secs, depending on the size of the interest operators which are used. The three (this is the minimum number of required features) best features are selected and used for the robotic visual servoing task. The size of the windows is 10x10. The experimental results are presented in Fig. 3-5. The gains for the controllers are $Q = I_6$ and $L = diag\{0.025, 0.025, 0.25, 2x10^5, 2x10^5, 2x10^5\}$. The delay factor d is 2. The computation of the $[\hat{B}^T(k) Q \hat{B}(k) + L]^{-1}$ matrix is done on a Heurikon 68030 board. We use two different techniques for its computation. The first technique performs a Singular Value Decomposition (SVD) of the 6x6 matrix based on a routine given by Forsythe [14]. This routine uses techniques such as the Householder reduction to bidiagonal form and diagonalization by the QR method. The computation of the inverse is based on the results of the SVD routine. The computational time for the first technique is 30 ms. The second technique is based on the partition of the matrix $K(k) = \hat{B}^T(k) Q \hat{B}(k) + L$ into four submatrices [6] [5]:

$$K(k) = \begin{bmatrix} K_{11}(k) & K_{12}(k) \\ K_{21}(k) & K_{22}(k) \end{bmatrix}$$

Goodwin [5] shows that the inverse of $K(k)$ is given by:

$$K^{-1}(k) = \begin{bmatrix} N_2^{-1}(k) & -N_2^{-1}(k) K_{12}(k) K_{22}^{-1}(k) \\ -N_1^{-1}(k) K_{21}(k) K_{11}^{-1}(k) & N_1^{-1}(k) \end{bmatrix}$$

where $N_1(k) = K_{22}(k) - K_{21}(k) K_{11}^{-1}(k) K_{12}(k)$, and $N_2(k) = K_{11}(k) - K_{12}(k) K_{22}^{-1}(k) K_{21}(k)$.

We can reduce the complexity of matrix inversions by using simple matrix algebra and the matrix inversion lemma [5]. Therefore, we can derive two new forms for $K^{-1}(k)$ which require only the inversion of two 3x3 matrices. The first form is [6]:

$$K^{-1}(k) = \begin{bmatrix} K_{11}^{-1}(k) [I_3 + K_{12}(k) N_1^{-1}(k) K_{21}(k) K_{11}^{-1}(k)] & -K_{11}^{-1}(k) K_{12}(k) N_1^{-1}(k) \\ -N_1^{-1}(k) K_{21}(k) K_{11}^{-1}(k) & N_1^{-1}(k) \end{bmatrix}$$

The second form is similar to the first form and is given by:

$$K^{-1}(k) = \begin{bmatrix} N_2^{-1}(k) & -N_2^{-1}(k) K_{12}(k) K_{22}^{-1}(k) \\ -K_{22}^{-1}(k) K_{21}(k) N_2^{-1}(k) & K_{22}^{-1}(k) [I_3 + K_{21}(k) N_2^{-1}(k) K_{12}(k) K_{22}^{-1}(k)] \end{bmatrix}$$

By using the previous forms, we are able to reduce the computational time of the 6x6 matrix inversion to 10 ms. Thus, the total computation time (image processing and control calculations) is approximately 200 ms. The inversion of the two 3x3 submatrices is done based on the assumption that the submatrices are invertible. Thus, the singularity of the submatrices should be checked every period T. The initial and the estimated values of the coefficients of the ARX models are given in the Table 1. In the experimental results (Fig. 3-5), we check the efficiency of the various proposed estimation and control schemes. The experimental results present a small steady-state error which is due to image noise and strict constraints on the rotational motion of the manipulator-camera system.

	$\hat{\xi}_0^{(1)}(k)$	$\hat{\xi}_0^{(2)}(k)$	$\hat{\xi}_0^{(3)}(k)$	$p^{(1)}(k)$	$p^{(2)}(k)$	$p^{(3)}(k)$
Initial	0.4175	0.4175	0.4175	0.1	0.1	0.1
Estimated	0.9435	0.9389	0.8594	0.0015	0.0014	0.0014

Table 1: Initial and estimated values of the parameters for visual servoing when a PD with gravity compensation cartesian robot controller is used.

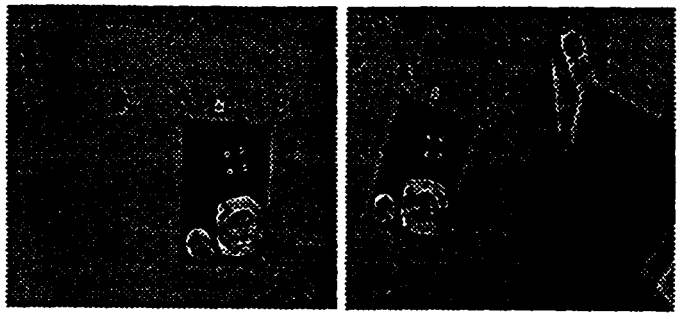
6. Conclusions

The problem of robotic visual servoing (eye-in-hand configuration) around a static target is addressed in this paper. The specific problem can be stated as "find the motion of the manipulator that will cause the image projections of certain feature points of the rigid static target to move to some desired image positions." The solution of this problem has numerous applications. Visual control can enhance the performance of industrial robots in assembly lines; improve the alignment of the object with the camera in automatic inspection systems; improve the automatic assembly of electronic devices (surface mount technology); make possible autonomous satellite docking and recovery, and finally, it can improve the efficiency of outdoor navigation techniques. This paper proposes an adaptive control scheme for an adequate solution. We claim that we should address the problem by combining vision and control techniques. The method followed includes a mathematical formulation of the problem, followed by the

introduction of adaptive control schemes for the case of inaccurate knowledge of some of the system's parameters (relative depth, noise model). Next, a stability analysis and an establishment of the minimum number of required feature points are performed. Finally, the implementation of the algorithms on our experimental testbed, the CMU DD Arm II robotic system, is presented. The real-time experiments show the feasibility and efficiency of our algorithms. Issues for future research include the introduction of the manipulator's mechanical constraints in the whole formulation, the explicit incorporation of the robot dynamics in the algorithms, the use of other features such as edges for measuring the servoing errors, the introduction and use of "snakes" for contour servoing, and the use of the 3-D target model in the servoing scheme.

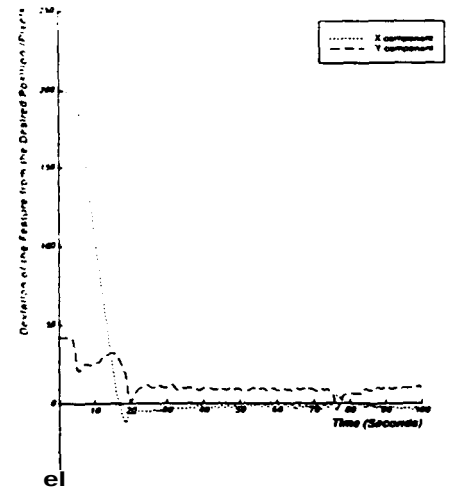
References

1. F. Chaumette and P. Rives, "Vision-based-control for robotic tasks", *Proc. of the IEEE International Worshop on Intelligent Motion Control*, 20-22 August 1990, pp. 395-400.
2. N. Papanikolopoulos, P. K. Khosla, and T. Kanade, "Adaptive robotic visual tracking", *Proc. of the American Control Conference*, June 1991, pp. 962-967.
3. N. Papanikolopoulos, P. K. Khosla, and T. Kanade, "Vision and control techniques for robotic visual tracking", *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1991, pp. 857-864.
4. P. Anandan, "Measuring visual motion from image sequences", Tech. report COINS-TR-87-21, COINS Department, University of Massachusetts, 1987.
5. G.C. Goodwin and K.S. Sin, *Adaptive filtering, prediction and control*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, Information and Systems Science Series, Vol. 1, 1984.
6. J.T. Feddema, C.S.G. Lee, and O.R. Mitchell, "Weighted selection of image features for resolved rate visual feedback control", *IEEE Trans. Robotics and Automation*, Vol. 7, No. 1, 1991, pp. 31-47.
7. N. P. Papanikolopoulos and P. K. Khosla, "Robotic visual servoing around a static target: an example of controlled active vision", Tech. report, Carnegie Mellon University, The Robotics Institute, 1991.
8. F.L. Lewis, *Optimal control*, John Wiley & Sons, New York, 1986.
9. P.S. Maybeck, *Stochastic models, estimation, and control*, Academic Press, London, 1979.
10. L. Matthies, R. Szeliski, and T. Kanade, "Kalman filter-based algorithms for estimating depth from image sequences", Tech. report 88-1, Carnegie Mellon University, The Robotics Institute, 1988.
11. J.T. Feddema, and C.S.G. Lee, "Adaptive image feature prediction and control for visual tracking with a hand-eye coordinated camera", *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 20, No. 5, 1990, pp. 1172-1183.
12. G.C. Goodwin and R.S. Long, "Generalization of results on multivariable adaptive control", *IEEE Trans. on Automatic Control*, Vol. 25, No. 6, December 1980, pp. 1241-1245.
13. F. Chaumette, P. Rives and B. Espiau, "Positioning of a robot with respect to an object, tracking it and estimating its velocity by visual servoing", *Proc. of the IEEE Int. Conf. on Robotics and Automation*, April 1991, pp. 2248-2253.
14. G.E. Forsythe, M.A. Malcolm, and C.B. Moler, *Computer methods for mathematical computations*, Prentice-Hall, Englewood Cliffs, N.J., 1977.



Initial Final

Figure 2: The initial and final images of the target.



el

Figure 3: Servoing errors for the first feature (A) in the example (PD with gravity compensation cartesian robot controller). The depth related parameter $\zeta_i^{@}(k)$ of each feature point is estimated by taking into consideration both the measurements in the X and Y directions.

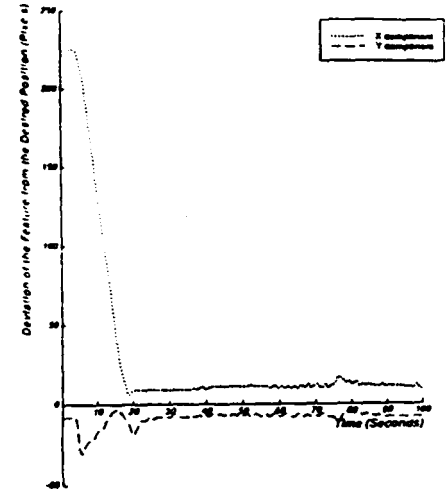


Figure 4: Servoing errors for the second feature (B) in the example (PD with gravity compensation cartesian robot controller). The depth related parameter $\zeta_i^{@}(k)$ of each feature point is estimated by taking into consideration both the measurements in the X and Y directions.

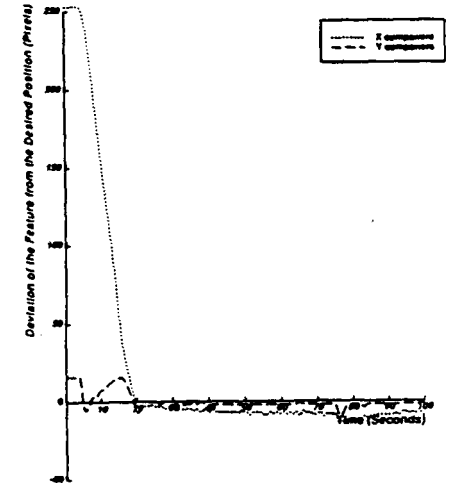


Figure 5: Servoing errors for the third feature (C) in the example (PD with gravity compensation cartesian robot controller). The depth related parameter $\zeta_i^{@}(k)$ of each feature point is estimated by taking into consideration both the measurements in the X and Y directions.