Uncalibrated Eye-in-Hand Visual Servoing

Jenelle Armstrong Piepmeier and Harvey Lipkin

The International Journal of Robotics Research 2003; 22; 805
DOI: 10.1177/027836490302210002

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
OCT 2003		**00-00-2003 to 00-00-2003**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Uncalibrated Eye-in-Hand Visual Servoing	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **United States Naval Academy,Annapolis,MD,21402**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In this paper we present newuncalibrated control schemes for visionguided robotic tracking of a moving target using a moving camera. These control methods are applied to an uncalibrated robotic system with eye-in-hand visual feedback. Without a priori knowledge of the robot?s kinematic model or camera calibration, the system is able to track a moving object through a variety of motions and maintain the object?s image features in a desired position in the image plane. These control schemes estimate the system Jacobian as well as changes in target features due to target motion. Four novel strategies are simulated and a variety of parameters are investigated with respect to performance. Simulation results suggest that a Gauss? Newton method utilizing a partitioned Broyden?s method for model estimation provides the best steady-state tracking behavior.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**16**	

Jenelle Armstrong Piepmeier

U.S. Naval Academy
Annapolis, MD 21402, USA

Harvey Lipkin

Georgia Institute of Technology
Atlanta, GA 30332, USA

Uncalibrated Eye-in-Hand Visual Servoing

Abstract

In this paper we present new uncalibrated control schemes for vision-guided robotic tracking of a moving target using a moving camera. These control methods are applied to an uncalibrated robotic system with eye-in-hand visual feedback. Without a priori *knowledge of the robot's kinematic model or camera calibration, the system is able to track a moving object through a variety of motions and maintain the object's image features in a desired position in the image plane. These control schemes estimate the system Jacobian as well as changes in target features due to target motion. Four novel strategies are simulated and a variety of parameters are investigated with respect to performance. Simulation results suggest that a Gauss–Newton method utilizing a partitioned Broyden's method for model estimation provides the best steady-state tracking behavior.*

KEY WORDS—uncalibrated eye-in-hand visual servoing, nonlinear least squares, Jacobian estimation

1. Introduction

In this paper we develop a model-independent, vision-guided, robotic control method. The controller presented is a recursive Gauss–Newton method and uses nonlinear least-squares optimization. The combined camera/robot model is approximated in a dynamic Jacobian estimation strategy, allowing servo control to be applied to systems without requiring a robot kinematic model or a calibrated camera model. Error velocity estimation is done simultaneously with Jacobian estimation in a partitioned matrix method. The control method is completely independent of the type of robot, the type of camera, and number of cameras.

Much work has been done in developing visual servoing systems for robot control resulting in a variety of approaches to the servoing problem. The majority of the resulting methods, however, require a priori knowledge of the system including the kinematic structure and the camera parameters. Using a model-dependent system also demands calibration of the robot and vision system, or recalibration due to the sensitivity of the system to disturbances. These activities can be both difficult and time-consuming, or perhaps unfeasible in an unstructured or changing environment. To our knowledge, this is the first method that addresses the uncalibrated eye-in-hand visual servoing problem for a moving target.

The International Journal of Robotics Research
Vol. 22, No. 10–11, October–November 2003, pp. 805-819,

2. Background

Benefits of using eye-in-hand camera arrangements include improved target recognition and inspection resulting from localization described by Chaumette, Rives, and Espiau (1991). Jang and Bien (1991) contend that effective resolution is increased, the problem of occlusion is solved, and an image nearly free of parallax error can be obtained using an eye-in-hand camera. Static cameras can be limited in their abilities due to limited depth of field and spatial resolution, problems which can be relieved by using eye-in-hand cameras as indicated by Papanikolopoulos, Khosla, and Kanade (1993).

2.1. Use of Eye-In-Hand Camera

The majority of eye-in-hand visual servoing controllers fall under two categories which could limit their use. Either they are not model-independent with regards to either the camera or the robot, or they require that the target is static.

Hashimoto and Noritsugu (1999) developed a linearized observer to estimate target velocity in their model-dependent controller. Allotta and Colombo (1999) also use linear approximations in an affine camera–object interaction model with the robot kinematic model assumed known. Baeten and De Schutter (1999) use an eye-in-hand feedforward controller in conjunction with force control for planar contour following. Both a camera model and a robot model are required in this case.

Algorithms using a PI controller or a pole assignment controller, both in combination with a steady-state Kalman filter, are implemented by Papanikolopoulos, Khosla, and Kanade

(1993). In each method, control output is fed to a Cartesian positioning system which requires a kinematic model.

Yoshimi and Allen (1994) implement the geometric effect of rotational invariance to approximate the image Jacobian. In addition to requiring knowledge of the robot Jacobian, their algorithm inherently works for only a static target.

Crétual, Chaumette, and Bouthemy (1991) develop a control algorithm for tracking a moving target with no model of the image transformation. The robot's vision system, however, consists of a camera with only two degrees of freedom (DoF), pan and tilt, which are controlled by an image error minimization. Depth is not considered in the process, as tracking is only done in the image plane, not in Cartesian space.

Flandin, Chaumette, and Marchand (2000) build upon the work done by Crétual, Chaumette, and Bouthemy (1991) to create a visual servo controller for a 6-DoF robot. In this case, the global positioning is done by a static, global camera, and the rotational positioning is servoed by another, independent controller. These algorithms do require knowledge of the kinematic robot Jacobian.

Oh and Allen (2001) present real-time experiments tracking people and robot hands (moving targets) using a 3-DoF gantry robot with a 2-DoF pan–tilt unit. The approach partitions the axes based on dynamic performance to servo the camera. The controller uses the kinematic and dynamic attributes of each DoF to increase tracking performance.

Corke and Hutchinson (2001) develop a decoupled image-based controller to handle pathological image-based visual servoing problems for the eye-in-hand case. They also employ a potential field technique to prevent features from leaving the image plane. The tasks studied involve static targets.

Malis and Chaumette (2002) and Malis (2002) discuss a task-function approach to image-based model-free visual servoing. This work uses information from a eye-in-hand camera to servo the robot into a desired orientation with respect to a static target. In Malis (2002) the controller is able to achieve straight-line Cartesian motion of the robot-mounted camera. It is important to note that the reference to "model-free" visual servoing in these works refers to the fact that there is no known model of the target object and the camera models are uncalibrated. The method is robust with respect to camera calibration errors, but a well-calibrated robot is used.

Several methods have been developed using a rank one update estimation of a composite Jacobian, known as a Broyden estimator. The Jacobian estimate is employed as part of a Gauss–Newton method to minimize squared error in the image plane. The Broyden estimator develops an on-line estimate of the system model, relating changes in image features to changes in joint variables. This means that no kinematic *or* camera model is used. Knowledge or estimation of individual kinematic and camera parameters are not necessary. Piepmeier (1999) builds on work carried out by Jagersand, Fuentes, and Nelson (1997) and Hosada and Asada (1994) to develop a dynamic Gauss–Newton method of visual servo control for tracking a moving target. A recursive least-squares (RLS) algorithm is implemented for Jacobian estimation to provide a robust control method even in the presence of system and measurement noise. The controller does not, however, permit the application of a moving camera. This paper extends the controller developed for the fixed camera case to solve the moving camera problem.

2.2. Uncalibrated Control for Fixed Camera Visual Servoing

In this section we present the fixed camera case for reference. In Section 3 it is modified for the moving camera case. First the dynamic Gauss–Newton controller is presented followed by the dynamic Jacobian estimator.

2.2.1. Dynamic Gauss–Newton Method

The dynamic Gauss–Newton method minimizes a time-varying objective function based on errors in the image plane. If the desired behavior is simple target following, the error function in the image plane for a moving target at position $y^*(t)$ and an end-effector at position $y(\theta)$ is the residual error

$$f(\theta, t) = y(\theta) - y^*(t) \tag{1}$$

where θ represents the joint angles and t represents time. The objective function to be minimized is the squared error:

$$F(\theta, t) = \frac{1}{2} f^{\mathrm{T}}(\theta, t) f(\theta, t). \tag{2}$$

Objective function (2) can be minimized using the dynamic Gauss–Newton algorithm (Piepmeier, McMurray, and Lipkin 1999a). Let $\hat{J}_k$ represent the kth approximation to the Jacobian $\frac{\partial f_k}{\partial \theta} = J_k$. The dynamic Gauss–Newton method computes the joint angles iteratively to minimize the objective function (2)

$$\theta_{k+1} = \theta_k - \left(\hat{J}_k^{\mathrm{T}} \hat{J}_k\right)^{-1} \hat{J}_k^{\mathrm{T}} (f_k + \frac{\partial f_k}{\partial t} h_t) \tag{3}$$

where $h_t = t_k - t_{k-1}$ is the time increment. The term $\frac{\partial f_k}{\partial t} h_t$ predicts the change in the error function for the next iteration. This term is a critical addition to the work presented by Jagersand, Fuentes, and Nelson (1997) and Hosada and Asada (1994) that enables the robot to servo to and track target motion. Note that eq. (3) is the controller used to compute desired changes in joint angles. It is assumed that individual joint level controllers are used to effect the changes.

A dynamic RLS estimation is used to provide on-line estimates of the Jacobian J_k as discussed in Piepmeier, McMurray, and Lipkin (1999b) and as follows.

2.2.2. Dynamic Jacobian Estimation: Recursive Least-Squares

Since the robot model is assumed to be unknown, a RLS algorithm is used to estimate the system Jacobian. This is accomplished by minimizing a weighted sum of the changes in the affine model of the error in the image plane. The affine model[1] of the error function $f(\theta, t)$ is denoted as $m(\theta, t)$ and expansion about the kth data point gives

$$m_k(\theta, t) = f(\theta_k, t_k) + \hat{J}_k(\theta - \theta_k) + \frac{\partial f_k}{\partial t}(t - t_k). \quad (4)$$

As shown in Haykin (1991) an exponentially weighted RLS algorithm that minimizes a weighted sum of the changes in the affine model over time can be used to recursively estimate the system Jacobian.

$$\min \mathcal{E}_k = \sum_{i=0}^{k-1} \lambda^{k-i-1} \|\Delta m_{ki}\|^2 \quad (5)$$

$$\begin{aligned} \Delta m_{ki} &= m_k(\theta_i, t_i) - m_i(\theta_i, t_i) \quad (6) \\ &= \left[f_k - f_i - \frac{\partial f_k}{\partial t}(t_k - t_i) \right] - \hat{J}_k h_{ki} \end{aligned}$$

where

$$h_{ki} = (\theta_k - \theta_i)$$

and where the weighting factor is $0 < \lambda \leq 1$ and the unknown variables are the elements of $\hat{J}_k$. The problem is solved in Appendix A to yield

$$\begin{aligned} \hat{J}_k &= \hat{J}_{k-1} + \left(\Delta f - \hat{J}_{k-1} h_\theta - \frac{\partial f_k}{\partial t} h_t \right) \\ &\quad \left(\lambda + h_\theta^{\mathrm{T}} P_{k-1} h_\theta \right)^{-1} h_\theta^{\mathrm{T}} P_{k-1} \quad (7) \\ P_k &= \frac{1}{\lambda} \left(P_{k-1} - P_{k-1} h_\theta \left(\lambda + h_\theta^{\mathrm{T}} P_{k-1} h_\theta \right)^{-1} h_\theta^{\mathrm{T}} P_{k-1} \right) \quad (8) \end{aligned}$$

where $h_\theta = \theta_k - \theta_{k-1}$, $\Delta f = f_k - f_{k-1}$. Equations (7) and (8) define the recursive update for $\hat{J}_k$. At each iteration, the calculation of $\hat{J}_k$ using eqs. (7) and (8) solves the minimization of eq. (5). The Jacobian estimate is used in the dynamic Gauss–Newton method of Section 2.2.1 to determine the joint angles θ_k that track the target. The parameter λ can be tuned to average in more or less of the previous data in the minimization. The approximation $\frac{1}{1-\lambda}$ is often used to estimate the effective number of terms being minimized. A λ close to 1 results in a filter with a longer memory.

Equation (7) bears some similarity to Broyden's method of Jacobian estimation as employed by Jagersand, Fuentes, and Nelson (1997). Broyden's method (Dennis and Schnabel 1983) is a Jacobian estimation scheme that satisfies the secant condition, i.e. the change in the m_{k-1} and m_k affine models is zero at the $(k-1)$th data point. Note that by minimizing a weighted sum of the changes in the affine error models, the algorithm will generally not satisfy the secant condition as in Broyden's method (Dennis and Schnabel 1983). Hosoda and Asada (1994) and Hosoda, Igarashi, and Asada (1998) implement a RLS Jacobian estimation scheme similar to Broyden's method with addition of a forgetting factor. In Piepmeier, McMurray, and Lipkin (1999b) it has been shown that this method is a RLS technique, a type of Kalman filter. The RLS algorithm displays greater stability than Broyden's method for Jacobian estimation by considering data over a period of time instead of just the previous iteration.

The Jacobian estimation in eq. (7) requires the (partial) velocity error term $\frac{\partial f_k}{\partial t}$, which can be directly estimated from the target image feature vector when a static camera is used with, for example, first-order differencing:

$$\frac{\partial f_k}{\partial t} = \left(\frac{\partial (y(\theta) - y^*(t))}{\partial t} \right)_k \cong -\left(\frac{y_k^* - y_{k-1}^*}{h_t} \right).$$

However, in the moving camera case the image target features are a function of θ and t, $y^* = y^*(\theta, t)$. This makes a similar direct calculation of the time partial derivative of error unfeasible. This is resolved using the partitioned recursive Jacobian estimation scheme in the following section.

3. Uncalibrated Control for Eye-in-Hand Visual Servoing

With an eye-in-hand system, changes in image features of a target object may be due either to target motion or robot motion. In this section we develop a nonlinear least-squares optimization method for a time-varying, coupled system and we introduce:

1. a partitioned Broyden's update for the (partial) error velocity estimation;
2. recursive and non-recursive forms of a Gauss–Newton method implementing the partitioned Broyden update;
3. a correction scheme for the error velocity estimation using the partitioned Broyden result as a predictor.

The partitioned Broyden update implements a simultaneous dynamic RLS estimation of the Jacobian and the error velocity. The error velocity estimation corrector is based on the total time derivative of the error function. Various controllers are created in the following section by combining the two Gauss–Newton forms with the error velocity correction.

3.1. A Partitioned Broyden's Method

The dynamic Broyden's method presented in Piepmeier, McMurray, and Lipkin (1999a) requires that the target's image position y^* be independent of robot joint position θ, and only a function of time. In the moving camera case, this is no longer

1. An affine model is a linear model that does not necessarily pass through the origin.

valid, because $y^* = y^*(\theta, t)$. In successive iterations, the target image position may change due to camera movement, even if the actual target is stationary. This makes estimation of the error velocity necessary.

The estimation of the error velocity can be incorporated into a Broyden Jacobian estimation method using a partitioned matrix to rewrite eq. (6) as

$$\Delta m_{k(i-1)} = [f_k - f_{i-1}] - \tilde{J}_k \tilde{h}_{k(i-1)} \tag{9}$$

where

$$\begin{aligned} \tilde{J}_k &= \begin{bmatrix} \hat{J}_k & \frac{\partial f_k}{\partial t} \end{bmatrix} \\ \tilde{h}_{k(i-1)} &= \begin{bmatrix} (\theta_k - \theta_{i-1}) \\ (t_k - t_{i-1}) \end{bmatrix} \end{aligned}$$

so the Jacobian is augmented by the error velocity and the joint angle differences are augmented by a time difference. Noting the similarity of eqs. (6) and (9), the RLS solution for the minimization of $\mathcal{E}_k$ in eq. (5) is given by

$$\tilde{J}_k = \tilde{J}_{k-1} + \left(\Delta f - \tilde{J}_{k-1}\tilde{h}\right)\left(\lambda + \tilde{h}^{\mathrm{T}}\tilde{P}_{k-1}\tilde{h}\right)^{-1}\tilde{h}^{\mathrm{T}}\tilde{P}_{k-1} \tag{10}$$

$$\tilde{P}_k = \frac{1}{\lambda}\left(\tilde{P}_{k-1} - \tilde{P}_{k-1}\tilde{h}\left(\lambda + \tilde{h}^{\mathrm{T}}\tilde{P}_{k-1}\tilde{h}\right)^{-1}\tilde{h}^{\mathrm{T}}\tilde{P}_{k-1}\right) \tag{11}$$

where

$$\tilde{h} = \begin{bmatrix} (\theta_k - \theta_{k-1}) \\ (t_k - t_{k-1}) \end{bmatrix}.$$

While eqs. (7) and (8) are inherently similar to eqs. (10) and (11), it is important to note the difference between $\hat{J}_k$ used in eq. (7) and the partitioned $\tilde{J}_k$ used in eq. (10) which incorporates both $\hat{J}_k$ and $\frac{\partial f_k}{\partial t}$ to be used by the controller. An algorithm for the combination of the partitioned Broyden update with the (non-recursive) Gauss–Newton controller given by eq. (3) in Section 2.2.1 is as follows:

Algorithm 1: Gauss–Newton Controller with Partitioned Broyden's Method (NGN)

Given $f : \mathbb{R}^n \to \mathbb{R}^m$; $\theta_0, \theta_1 \in \mathbb{R}^n$; $\hat{J}_0 \in \mathbb{R}^{m\times n}$; $\left(\hat{f}_t\right)_0 \mathbb{R} \in^{m\times 1}$; $P_0 \in \mathbb{R}^{n+1\times n+1}$; $\lambda \in (0, 1)$

Do for $k = 1, 2, \ldots$

$\Delta f = f_k - f_{k-1}$, $h_\theta = \theta_k - \theta_{k-1}$, $h_t = t_k - t_{k-1}$

$$\tilde{h} = \begin{bmatrix} (\theta_k - \theta_{k-1}) \\ (t_k - t_{k-1}) \end{bmatrix}$$

$$\tilde{J}_{k-1} = \begin{bmatrix} \hat{J}_{k-1} & \left(\hat{f}_t\right)_{k-1} \end{bmatrix}$$

$$\tilde{J}_k = \tilde{J}_{k-1} + \left(\Delta f - \tilde{J}_{k-1}\tilde{h}\right)\left(\lambda + \tilde{h}^{\mathrm{T}}\tilde{P}_{k-1}\tilde{h}\right)^{-1}\tilde{h}^{\mathrm{T}}\tilde{P}_{k-1}$$

$$\tilde{P}_k = \tfrac{1}{\lambda}\left(\tilde{P}_{k-1} - \tilde{P}_{k-1}\tilde{h}\left(\lambda + \tilde{h}^{\mathrm{T}}\tilde{P}_{k-1}\tilde{h}\right)^{-1}\tilde{h}^{\mathrm{T}}\tilde{P}_{k-1}\right)$$

$$\theta_{k+1} = \theta_k - \left(\hat{J}_k^{\mathrm{T}}\hat{J}_k\right)^{-1}\left(\hat{J}_k^{\mathrm{T}} f_k + \hat{J}_k^{\mathrm{T}}\left(\hat{f}_t\right)_k h_t\right)$$

End for

The term $\left(\hat{f}_t\right)_k$ is the estimated/predicted value of the error velocity $\frac{\partial f_k}{\partial t}$. This partitioned Broyden's method uses RLS to minimize a weighted sum of the squared differences between the current and previous affine models for each iteration of $(\theta, t) = (\theta_{i-1}, t_{i-1})$ to determine new Jacobian and error velocity approximations. Varying the parameter λ between 0 and 1 changes the memory of the scheme, with values closer to 1 resulting in longer memory, implying that a greater number of previous Jacobian and error velocity values have a significant effect on the new values.

3.2. Recursive Gauss–Newton Method

In order to provide a filtering action to the joint change calculation, the Gauss–Newton method is extended to an RLS formulation with exponential weighting. In this case the weighted sum being minimized uses the affine error models evaluated at the next data point

$$\min \mathcal{G}_k = \sum_{i=0}^{k} \gamma^{k-i} \left\| m_i\left(\theta_{k+1}, t_{k+1}\right)\right\|^2 \tag{12}$$

$$m_i\left(\theta_{k+1}, t_{k+1}\right) = f_i + \hat{J}_i(\theta_{k+1} - \theta_i) + \frac{\partial f_i}{\partial t}\left(t_{k+1} - t_i\right) \tag{13}$$

$$= \left[f_i + \frac{\partial f_i}{\partial t}\left(t_{k+1} - t_i\right) + \hat{J}_i\left(\theta_k - \theta_i\right)\right] - \left(-\hat{J}_i\right)\left(\theta_{k+1} - \theta_k\right) \tag{14}$$

which must be solved for θ_{k+1} given t_{k+1}. This can be rewritten as

$$\min \mathcal{G}_k = \min e^{\mathrm{T}} W e$$
$$e = b - A x_k$$

where

$$e = \begin{bmatrix} m_0\left(\theta_{k+1}, t_{k+1}\right) \\ \vdots \\ m_k\left(\theta_{k+1}, t_{k+1}\right) \end{bmatrix}, \quad W = \begin{bmatrix} \gamma^0 I & & \\ & \ddots & \\ & & \gamma^{k-1} I \end{bmatrix},$$

$$b = \begin{bmatrix} f_0 + \frac{\partial f_0}{\partial t}\left(t_{k+1} - t_0\right) + \hat{J}_0\left(\theta_k - \theta_0\right) \\ \vdots \\ f_k + \frac{\partial f_k}{\partial t}\left(t_{k+1} - t_k\right) + \hat{J}_k\left(\theta_k - \theta_k\right) \end{bmatrix}$$

$$A = \begin{bmatrix} -\hat{J}_0 \\ \vdots \\ -\hat{J}_k \end{bmatrix}, \quad x_k = \left(\theta_{k+1} - \theta_k\right) = h_\theta.$$

The RLS solution for the minimization of $\mathcal{G}_k$ in eq. (12) is

given by

$$\theta_{k+1} = 2\theta_k - \theta_{k-1} - Q_{k-1}\hat{J}_k^{\mathrm{T}}\left(\gamma I + \hat{J}_k Q_{k-1}\hat{J}_k^{\mathrm{T}}\right)^{-1} \left(f_k + \frac{\partial f_k}{\partial t}(t_{k+1} - t_k) + \hat{J}_k(\theta_k - \theta_{k-1})\right) \quad (15)$$

$$Q_k = \frac{1}{\gamma}\left(Q_{k-1} - Q_{k-1}\hat{J}_k^{\mathrm{T}}\left(\gamma I + \hat{J}_k Q_{k-1}\hat{J}_k^{\mathrm{T}}\right)^{-1}\hat{J}_k Q_{k-1}\right).$$

Combining this recursive Gauss–Newton method with the partitioned Broyden's method and using the predicted value of the error velocity gives the following algorithm:

Algorithm 2: Recursive Gauss–Newton Method (RGN)

Given $f : \mathbb{R}^n \to \mathbb{R}^m$; θ_0, $\theta_1 \in \mathbb{R}^n$; $\hat{J}_0 \in \mathbb{R}^{m\times n}$; $\left(\hat{f}_t\right)_0 \in \mathbb{R}^{m\times 1}$; $P_0 \in \mathbb{R}^{n+1\times n+1}$; $Q_0 \in \mathbb{R}^{n\times n}$; $\gamma, \lambda \in (0, 1)$

Do for $k = 1, 2, \ldots$

$\Delta f = f_k - f_{k-1}$, $h_\theta = \theta_k - \theta_{k-1}$, $h_t = t_k - t_{k-1}$

$$\tilde{h} = \begin{bmatrix} (\theta_k - \theta_{k-1}) \\ (t_k - t_{k-1}) \end{bmatrix}$$

$$\tilde{J}_{k-1} = \begin{bmatrix} \hat{J}_{k-1} & \left(\hat{f}_t\right)_{k-1} \end{bmatrix}$$

$$\tilde{J}_k = \tilde{J}_{k-1} + \left(\Delta f - \tilde{J}_{k-1}\tilde{h}\right)\left(\lambda + \tilde{h}^{\mathrm{T}}\tilde{P}_{k-1}\tilde{h}\right)^{-1}\tilde{h}^{\mathrm{T}}\tilde{P}_{k-1}$$

$$\tilde{P}_k = \frac{1}{\lambda}\left(\tilde{P}_{k-1} - \tilde{P}_{k-1}\tilde{h}\left(\lambda + \tilde{h}^{\mathrm{T}}\tilde{P}_{k-1}\tilde{h}\right)^{-1}\tilde{h}^{\mathrm{T}}\tilde{P}_{k-1}\right)$$

$$\begin{bmatrix} \hat{J}_k & \left(\hat{f}_t\right)_k \end{bmatrix} = \tilde{J}_k$$

$$\theta_{k+1} = 2\theta_k - \theta_{k-1} - Q_{k-1}\hat{J}_k^{\mathrm{T}}\left(\gamma I + \hat{J}_k Q_{k-1}\hat{J}_k^{\mathrm{T}}\right)^{-1} \left(f_k + \left(\hat{f}_t\right)_k (t_{k+1} - t_k) + \hat{J}_k(\theta_k - \theta_{k-1})\right)$$

$$Q_k = \frac{1}{\gamma}\left(Q_{k-1} - Q_{k-1}\hat{J}_k^{\mathrm{T}}\left(\gamma I + \hat{J}_k Q_{k-1}\hat{J}_k^{\mathrm{T}}\right)^{-1}\hat{J}_k Q_{k-1}\right)$$

End for

Again, there is a memory term in the formulation, γ, similar in function to λ, which can be tuned to vary the effect of previous information.

3.3. Estimated Error Velocity Correction

The partitioned Broyden's method uses RLS estimation to simultaneously approximate the Jacobian J and the error velocity $\frac{\partial f}{\partial t}$ because for the moving camera case they cannot be determined separately. The averaging process of this method, however, may not produce accurate error velocity values. The estimated Jacobian can also be used in a corrector to calculate new values of the error velocity through use of the total time derivative:

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial \theta}\frac{\mathrm{d}\theta}{\mathrm{d}t}.$$

The term $\frac{\partial f}{\partial t}$ represents the change in error due to the target motion since the end-effector position is instantaneously fixed. Introducing the approximations $\frac{\mathrm{d}f}{\mathrm{d}t} \cong \Delta f / h_t$, the total change in error, and $\frac{\partial f}{\partial \theta}\frac{\mathrm{d}\theta}{\mathrm{d}t} \cong (Jh_\theta)/h_t$, the change in the error due to end-effector motion, then for the kth increment

$$\left(\hat{f}_t\right)_k = \left(\Delta f - \hat{J}_k h_\theta\right)/h_t \quad (16)$$

where $\left(\hat{f}_t\right)_k$ is the estimated value of $\left(\frac{\partial f}{\partial t}\right)_k$. Using the above equation, an approximation of the error velocity values can be made for comparison with those given by the partitioned Broyden estimator. Thus, a Gauss–Newton controller (3) or the recursive Gauss–Newton controller (15) can either implement the uncorrected error velocity values output by the partitioned Broyden's method or the corrected error velocity values given by the total time derivative equation. In other words, eq. (16) can be inserted in either Algorithm 1 or 2 after the Jacobian estimation. It should be noted that all four control options compute desired changes in joint angles, and it is assumed that the robot's individual joint level controllers are able to achieve the desired changes.

4. Simulation and Results

In this section we present a series of simulations to evaluate the control schemes. First, the four controllers are compared for a simple translational, circular path for varying speeds and λ. Then with a selected controller more difficult paths are examined.

4.1. System Description

A 6-DoF system is simulated using the Robotics and Machine Vision Toolboxes developed by Corke (1996) for use in the MATLAB environment. The camera is assumed to be coincident with the final frame of the Puma 560 manipulator. The dynamics of the robot are not considered in this simulation. A sampling time of $h_t = 50$ ms is used throughout for a 20 Hz system update. While vision latencies are not explicitly modeled, it is assumed that the vision data at each update represent the changes in features for the most recently commanded motion.

The target consists of four planar feature points spaced on a square with 5 cm sides. The target's initial location is within the camera's field of view and is approximately 50 cm in front of the camera. The image features seen at this point define the target image. As the target moves, the error will be minimized as the robot servos the camera so that the camera and the target maintain constant relative positions. To start the simulation, the robot is moved away from the target. To estimate the initial Jacobian, each joint is successively moved a small amount and the change in image features is recorded.

After capturing the target image and initializing the Jacobian, the simulation is ready to begin. The target center is given a circular motion with constant orientation in the fixed frame

$$(x, y, z) = (65, 50 + 10\sin(k\omega h_t), 10\cos(k\omega h_t)) \quad (17)$$

where h_t is the sampling period, k is the iteration number, ω is the frequency, and the units are centimeters. Thus, the target is translating in a circular path. Uniform image noise between ± 0.5 pixel is added to the image features of the target object.

4.2. Controller Performance

System performance depends on the speed of the target, the type of controller used, and the memory of the controller. The simulated control systems are distinct in two ways: (1) the Gauss–Newton controller is non-recursive or recursive; and (2) the error velocity is uncorrected or corrected. All schemes use the estimated Jacobian from the partitioned Broyden update. The four controller schemes simulated here include: (NGN/UV) non-recursive Gauss–Newton controller given by Algorithm 1 with uncorrected error velocity; (RGN/UV) recursive Gauss–Newton controller given by Algorithm 2 with uncorrected error velocity; (NGN/CV) non-recursive Gauss–Newton controller with corrected error velocity from Section 3.3; (RGN/CV) recursive Gauss–Newton controller with corrected error velocity from Section 3.3.

The results of a sample simulation utilizing the NGN/UV controller ($\lambda = 0.98$) is shown in Figure 1. The four feature points are seen initially in a configuration that does not correspond to the goal positions. The feature points converge to the goal positions and the features remain at or near the goal positions even though the target object is moving in a circular path in the world coordinate frame. For a reference, if a static camera located at the initial position the moving camera, the speed of the target object (moving at 2.5 cm s^{-1}) roughly corresponds to a feature velocity of approximately 63.5 pixel s^{-1}. Figure 2 shows the initial positions of the target and the robot, and Figure 3 shows the image error. The average of the four image feature tracking RMS errors is 4.0 pixels or approximately 0.2 cm.

To study the behavior of the four different controllers, the same simulation was repeated 25 times for each controller at eight different circular speeds ω. Target velocities ranged from 0.5 to 4 cm s^{-1}, or from 37 to 94 pixel s^{-1} for a static camera. Each of the 25 simulations has somewhat different results due to the addition of random pixel noise. Figure 4 shows the average image RMS error norms (in pixels) for the repeated trials using all four controllers. Outliers such as those seen at $\omega = 0.3$ and $\omega = 0.4$ for the NGN/UV indicate simulations where the system lost track of the desired pose and the control became unstable. The NGN/UV controller shows a distinct advantage over the other controllers. The velocity correction scheme does not appear to offer any advantage. In fact, the RGN/CV errors are somewhat worse than the RGN/UV errors, and the NGN/CV simulations exhibit complete loss of control. Although not shown, it is interesting that the NGN/CV simulations perform similarly to NGN/UV simulations when the image noise level is reduced by half, indicating that it is very sensitive to system noise. This shows the beneficial filtering action of the recursive (uncorrected) error velocity calculation and the amplification of noise introduced by the first order corrector.

4.3. NGN/UV Performance

For the 6-DoF system simulated, the NGN/UV controller produces the best results. To study the effects of the forgetting factor λ, a second series of simulations was run using the NGN/UV controller and varying the circular speed ω and the forgetting factor λ. Figure 5 shows the average steady-state image error for $\omega = \{0.05, 0.1, 0.15, 0.2\}$, and $\lambda = \{0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.0\}$ for 25 simulations each. The average steady-state image RMS error for each simulation is plotted by $\times$ and the mean of the averages is plotted as a line. Clearly a long memory with $\lambda \geq 0.98$ (approximately 20 or more significant samples) gives the best overall performance. Figure 6 plots the mean of the average image error for $\lambda = \{0.98, 0.99, 1.00\}$ of λ versus the target speed determined by ω. For slower speeds, a lower value of λ produces results in better tracking whereas at higher speeds $\lambda = 0.9$ and $\lambda = 1$ result in better tracking. These results concur with similar studies performed for the fixed camera case in Piepmeier (1999).

4.4. Additional Target Motions

To further validate the capabilities of the NGN controller for uncalibrated eye-in-hand visual servoing, additional target motions are studied. The simulations in Sections 4.2 and 4.3 employed a simple circular path. This type of path results in constant Cartesian speeds for the target object. In this section two additional paths are introduced. First, we demonstrate tracking a square path (Figure 7). Next we demonstrate tracking the target object as it follows a helical path (Figure 8) involving target rotation and translation. The target is twisting and retreating from the camera. To maintain the desired pose between the target and the camera, the robot must move the camera along the robot base frame's x-axis as well as follow the object's rotation and circular motion. The square path demonstrates the ability of the controller to handle abrupt changes in target motion. The helical path demonstrates its ability to follow targets through changing depths and feature rotations. The target is a 5 cm square and the corners of the target object are the four features being tracked. For Figure 7, the four target features are moving at 3.2 cm s^{-1}, and for Figure 8 the four target points reach maximum speeds of (2.9, 1.2, 2.9, 4.0) cm s^{-1}. Figures 9 and 10 show the image features for these two paths. Extension 1 shows animations of the robotic system as it tracks the moving target. While the paths are very different in nature, the results are nearly

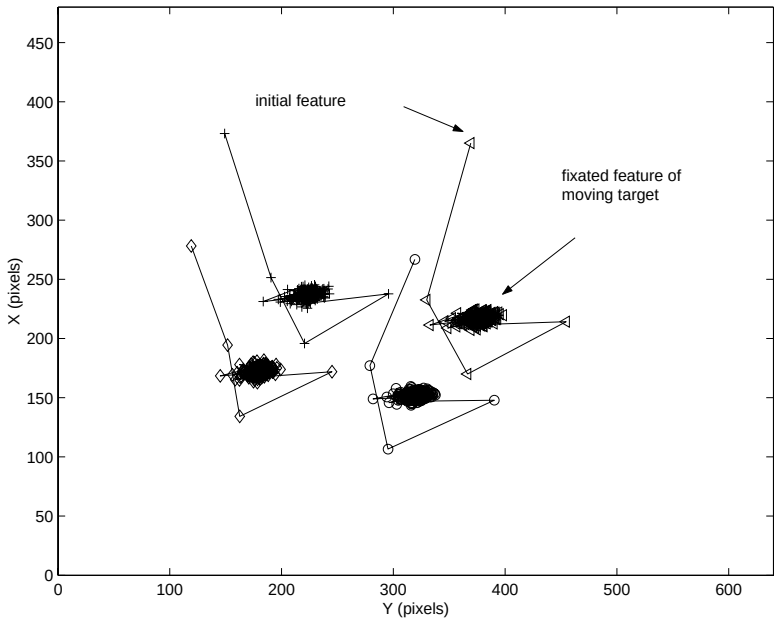

Fig. 1. Image features for eye-in-hand visual servoing using an NGN/UV controller with $\lambda = 0.98$ and $\omega = 0.25$ rad s^{-1}. A moving object with four features is seen initially on the right. These features converge on the desired pose with an average steady-state image RMS error of 4.0 pixels.

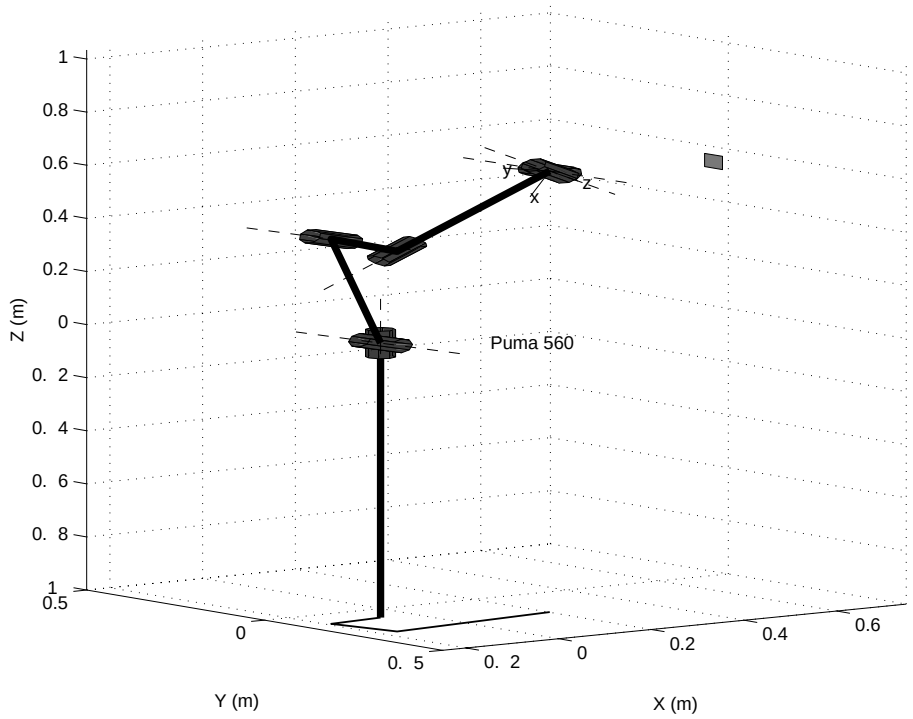

Fig. 2. Plot showing the respective positions of the target and the robot.

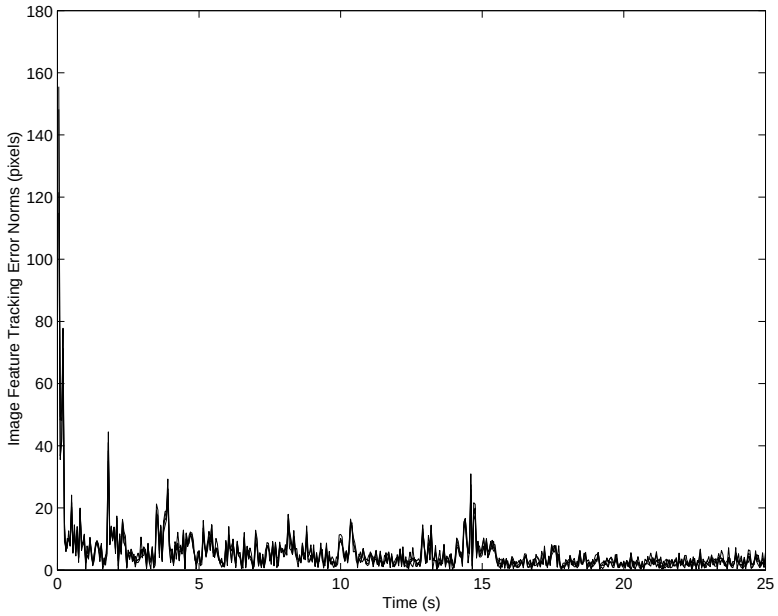

Fig. 3. Image error for using an NGN/UV controller with $\lambda = 0.98$ and $\omega = 0.25$ rad s^{-1}. The average steady-state RMS error is 4.0 pixels.

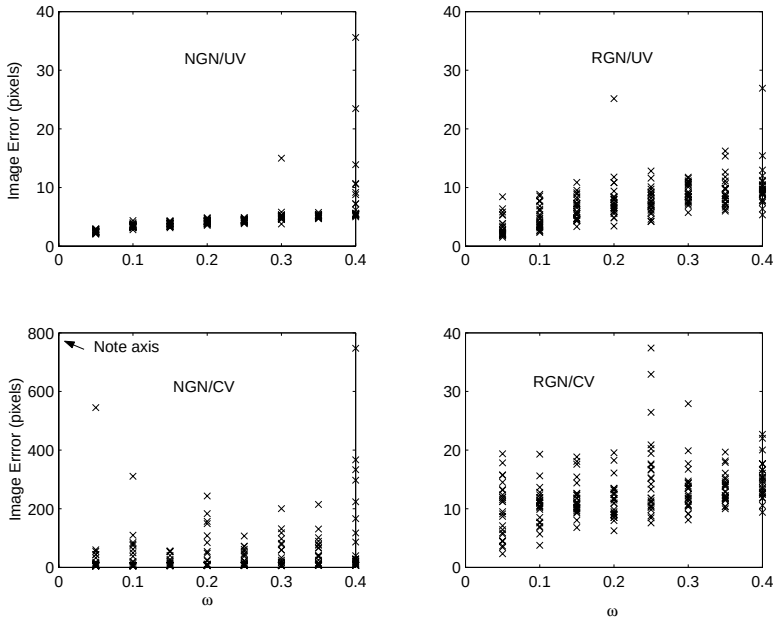

Fig. 4. A comparison of four different controllers at a range of target speeds. In general, the NGN/UV controller provides the best convergent control.

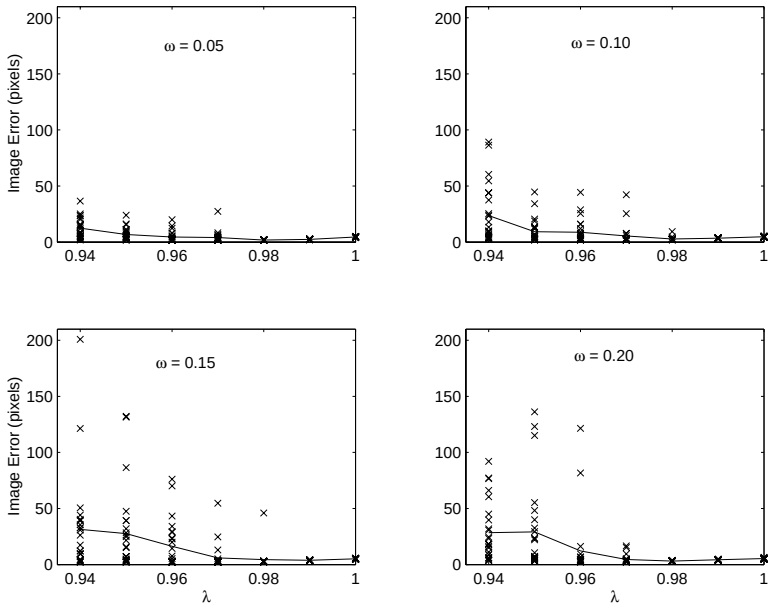

Fig. 5. A comparison of NGN/UV controller performance showing image error versus λ for four different target speeds. Individual errors are plotted for 25 simulations; the line indicates the average error.

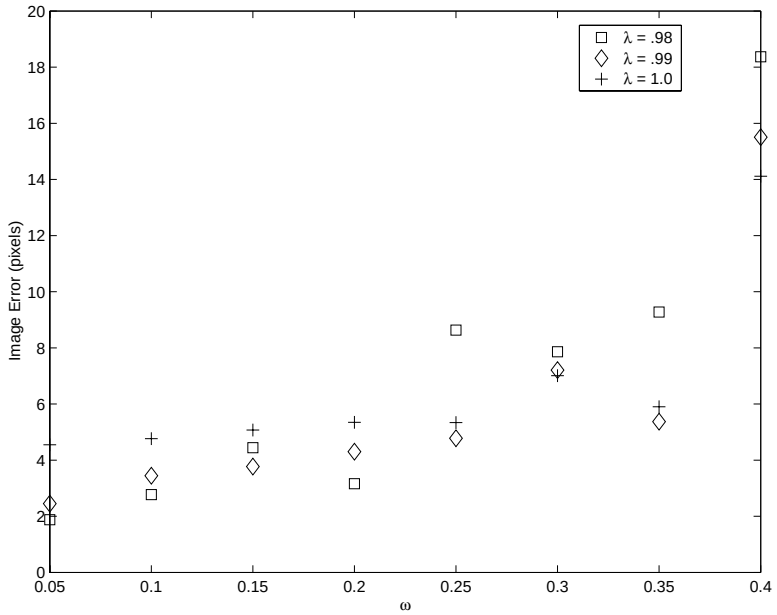

Fig. 6. The performance of the NGN/UV controller for $\lambda = \{0.98, 0.99, 1.0\}$ is shown for eight different target speeds dictated by ω.

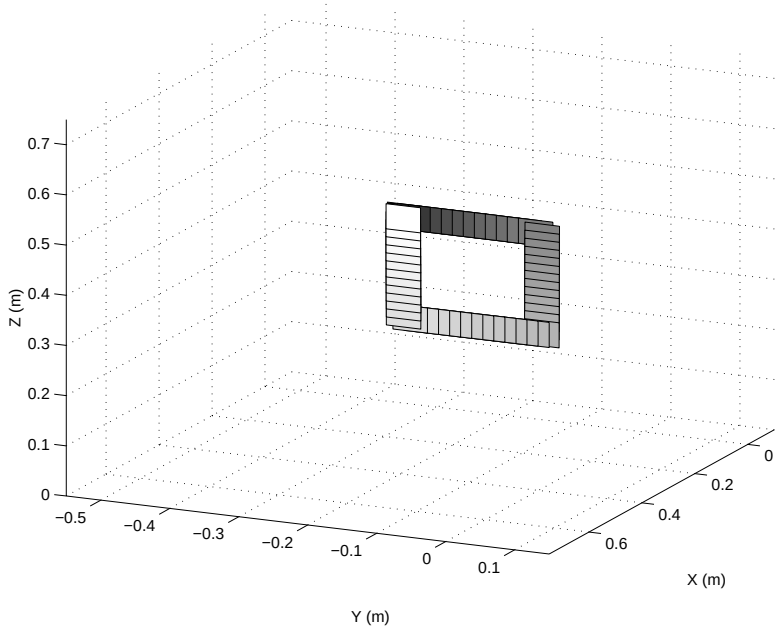

Fig. 7. Path taken by the 5 cm square target object. The object maintains a constant x position at 0.65 m while translating in the y–z plane. The initial target position is denoted by the lightest square. The darkest represents the final position.

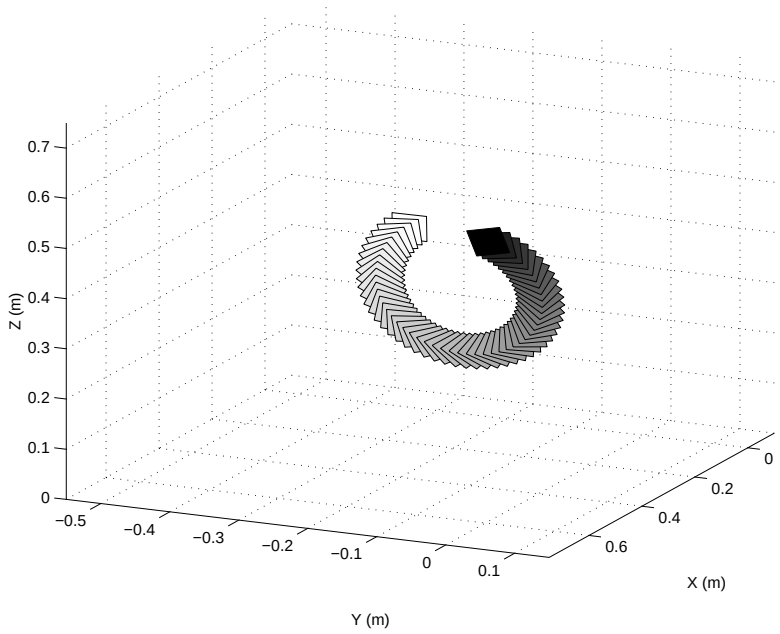

Fig. 8. Path taken by the 5 cm square target object. The object is located initially at a depth of 0.65 m along the x-axis. The object rotates as the depth increases to 0.75 m.

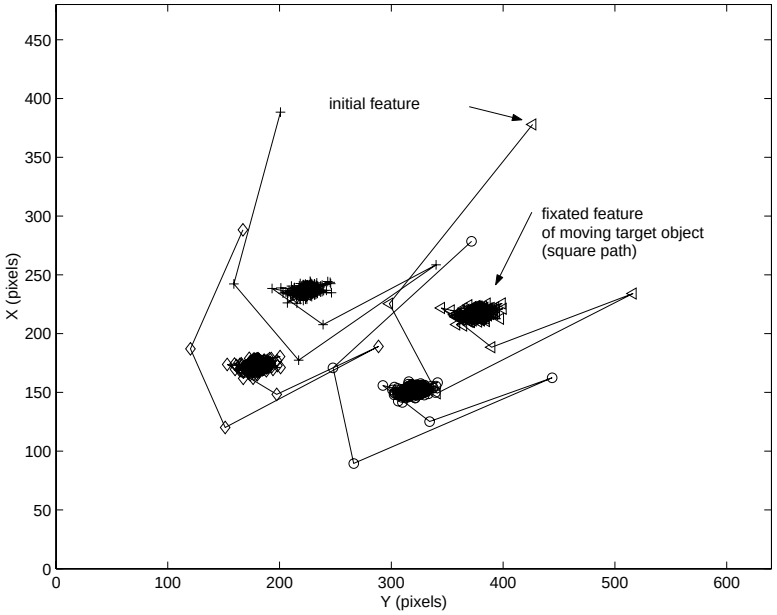

Fig. 9. Image features as seen by the eye-in-hand camera following a target object moving along the path shown in Figure 7. Average steady-state error is 4.1 pixels. The features appear stationary because the robot maintains the desired pose relative to the moving target.

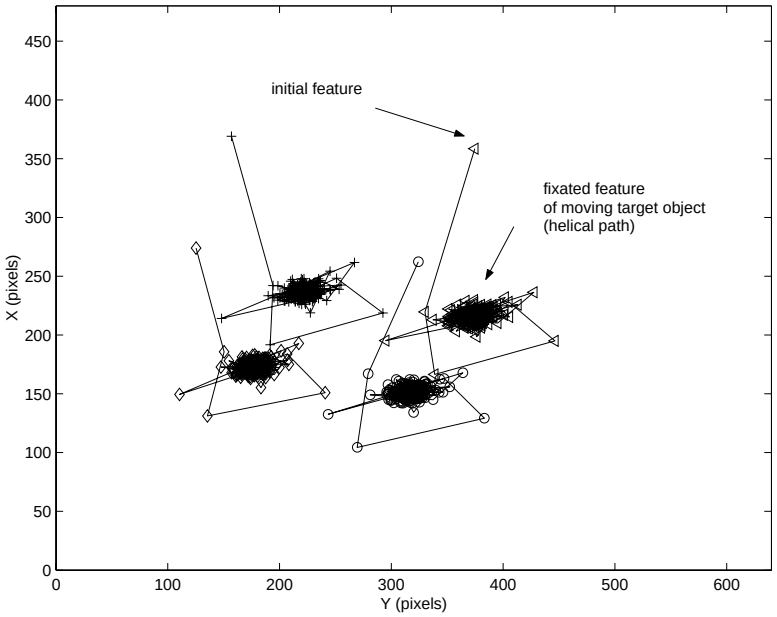

Fig. 10. Image features as seen by the eye-in-hand camera following a target object moving along the path shown in Figure 8. Average steady-state error is 6.0 pixels. The features appear stationary because the robot maintains the desired pose relative to the moving target. Note the consistency between Figures 1, 9, and 10 despite the differing target motions.

identical to each other and to the results from Figures 1–3. The average steady-state tracking error for the square path is 4.1 pixels, and the average steady-state tracking error for the helical path is 6.0 pixels. The tracking errors are shown in Figures 11 and 12. We see from Figures 9 and 11 that the NGN/UV controller can handle abrupt changes in the direction of the target's motion. From Figures 10 and 12 we see that the NGN/UV controller can handle complex paths varying in depth with respect to the image plane.

5. Conclusion

In this paper we have investigated the use of eye-in-hand visual feedback for the tracking of moving targets. The methods presented are uncalibrated and require no kinematic models or camera calibration. Thus, the same controller can be used on a wide variety of platforms. Simulation results suggest that a Gauss–Newton method utilizing a partitioned Broyden's method for model estimation provides the best steady-state tracking behavior. This work is the first presentation of an uncalibrated, eye-in-hand, vision-based control scheme for a moving target. While the eye-in-hand system has been studied here, the algorithm is generic, and is applicable when the camera is moving but not fixed to the robotic system. Furthermore, the control strategy could be applied to either a manipulator or mobile robot for uncalibrated control.

Appendix A: Derivation of RLS Jacobian Estimator

The recursive dynamic Jacobian estimator is the RLS solution to the minimization problem (5) which is repeated here

$$\begin{aligned} \min \mathcal{E}_k &= \sum_{i=0}^{k-1} \lambda^{k-i-1} \|\Delta m_{ki}\|^2 \\ \Delta m_{ki} &= m_k(\theta_i, t_i) - m_i(\theta_i, t_i) \\ &= \left[f_k - f_i - \frac{\partial f_k}{\partial t}(t_k - t_i) \right] - \hat{J}_k h_{ki} \end{aligned}$$

where $h_{ki} = (\theta_k - \theta_i)$, $0 < \lambda \leq 1$ is the weighting factor, and $\hat{J}_k$ is the estimated Jacobian to be determined.

It is direct to confirm that the problem can be re-expressed in a stacked and partitioned form

$$\min \bar{e}^{\mathrm{T}} \bar{W} \bar{e}$$
$$\bar{e} = \bar{b} - \bar{A}\bar{x}_k$$

where

$$\bar{e} = \left[\frac{\bar{e}_{k-1}}{e_k} \right] = \begin{bmatrix} \Delta m_{k0} \\ \vdots \\ \frac{\Delta m_{k(k-2)}}{\Delta m_{k(k-1)}} \end{bmatrix},$$

$$\bar{b} = \left[\frac{\bar{b}_{k-1}}{b_k} \right] = \begin{bmatrix} f_k - f_0 - \frac{\partial f_k}{\partial t}(t_k - t_0) \\ \vdots \\ \frac{f_k - f_{k-2} - \frac{\partial f_k}{\partial t}(t_k - t_{k-2})}{f_k - f_{k-1} - \frac{\partial f_k}{\partial t}(t_k - t_{k-1})} \end{bmatrix}$$

$$\bar{x}_k = \begin{bmatrix} (\hat{J}_k^{\mathrm{T}})_1 \\ \vdots \\ (\hat{J}_k^{\mathrm{T}})_n \end{bmatrix},$$

$$\bar{A} = \left[\frac{\bar{A}_{k-1}}{A_k} \right] = \begin{bmatrix} \operatorname{diag} h_{k0}^{\mathrm{T}} \\ \vdots \\ \frac{\operatorname{diag} h_{k(k-2)}^{\mathrm{T}}}{\operatorname{diag} h_{k(k-1)}^{\mathrm{T}}} \end{bmatrix},$$

$$\bar{W} = \begin{bmatrix} \lambda^{k-1} I & & & \\ & \ddots & & \\ & & \lambda^1 I & \\ & & & \lambda^0 I \end{bmatrix}$$

and where I is the identity matrix, $(\hat{J}_k^{\mathrm{T}})_r$ denotes the rth column of $\hat{J}_k^{\mathrm{T}}$ and $(\operatorname{diag} h_{ki}^{\mathrm{T}})$ means each element of a diagonal matrix contains the row vector h_{ki}^{T}. An overbar indicates a stacked and/or blocked quantity so, for example, the stacked vector $\bar{e}$ is partitioned into $k-1$ stacked vectors in $\bar{e}_{k-1}$ and one vector e_k; the matrix $\bar{W}$ has k blocks on the diagonal. Note, however, that $\bar{x}_k$ merely stacks the columns of $\hat{J}_k^{\mathrm{T}}$ into a single stacked column.

The minimization problem is now in the standard form used for a recursive solution (see, for example, Franklin, Powell, and Workman 1990)

$$\bar{x}_k = \bar{x}_{k-1} + \bar{K}_k (b_k - A_k \bar{x}_{k-1}) \tag{18}$$

where $\bar{K}_k$ is defined as

$$\bar{K}_k = \frac{\bar{P}_{k-1}}{\lambda} A_k^{\mathrm{T}} \left(A_k \frac{\bar{P}_{k-1}}{\lambda} A_k^{\mathrm{T}} + I \right)^{-1} \tag{19}$$

$\bar{P}_k$ is defined by the recursion

$$\bar{P}_k = \frac{1}{\lambda}(I - \bar{K}_k A_k) \bar{P}_{k-1} \tag{20}$$

and the initial values $\bar{x}_0$ and $\bar{P}_0$ must be supplied. However, $\bar{P}_k$ also has a non-recursive definition that is useful for determining its symmetric structure:

$$\bar{P}_k = \left(\bar{A}^{\mathrm{T}} \begin{bmatrix} \lambda^{k-1} I & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda^0 I \end{bmatrix} \bar{A} \right)^{-1}.$$

Since $\bar{A}_{k-1}$ is a stacking of block diagonal matrices, after some rearranging, $\bar{P}_k$ reduces to a block diagonal matrix with repeated elements

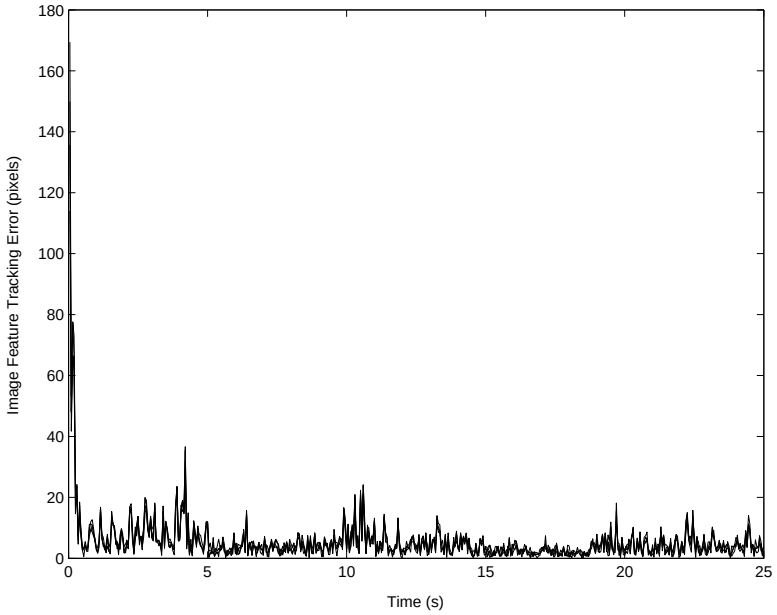

Fig. 11. Image feature tracking error using the NGN/UV controller ($\lambda = 0.98$) for a target object following the square path. Average steady-state RMS error is 4.1 pixels. The tracking is stable despite abrupt changes in the target's motion.

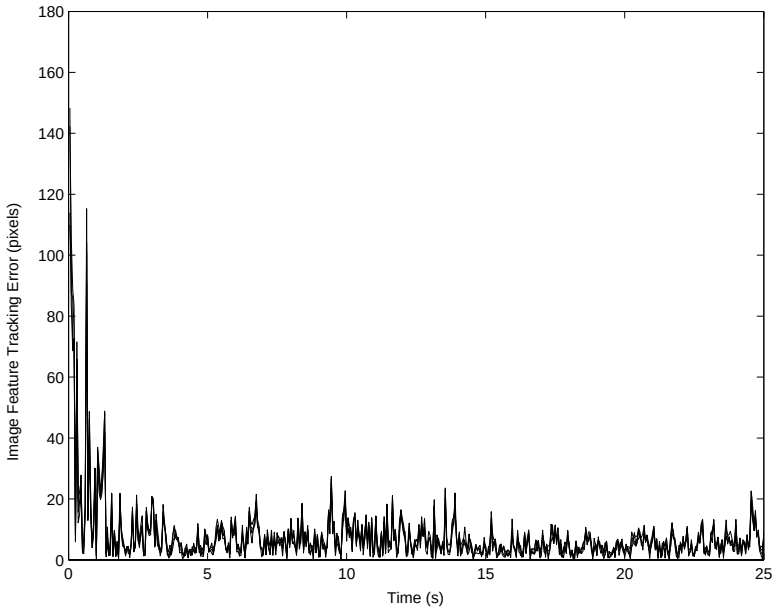

Fig. 12. Image feature tracking error using the NGN/UV controller ($\lambda = 0.98$) following a target object moving along a helical path. Average steady-state RMS error is 6.0 pixels. The similarity to Figures 3 and 11 demonstrates the consistency of the tracking behavior despite the different target motions.

$$\bar{P}_k = \begin{bmatrix} P_k & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & P_k \end{bmatrix}$$

where the non-recursive form of P_k is

$$P_k = \left(\begin{bmatrix} h_{k0} \\ \vdots \\ h_{k(k-1)} \end{bmatrix} \begin{bmatrix} \lambda^{k-1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda^0 \end{bmatrix} \begin{bmatrix} h_{k0}^{\mathrm{T}} \\ \vdots \\ h_{k(k-1)}^{\mathrm{T}} \end{bmatrix} \right)^{-1}.$$

A similar form applies to $\bar{P}_{k-1}$ which is used to also reduce $\bar{K}_k$ to a block diagonal matrix with repeated elements

$$\bar{K}_k = \begin{bmatrix} K_k & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & K_k \end{bmatrix}$$

where

$$K_k = \frac{P_{k-1}}{\lambda} h_{k(k-1)} \left(h_{k(k-1)}^{\mathrm{T}} \frac{P_{k-1}}{\lambda} h_{k(k-1)} + 1 \right)^{-1}$$

and together with eqs. (19) and (20) yields the recursive form of P_k

$$P_k = \frac{1}{\lambda} \left(I - K_k h_{k(k-1)}^{\mathrm{T}} \right) P_{k-1}. \tag{21}$$

Using these in eq. (18) and unstacking gives

$$(\hat{J}_k^{\mathrm{T}})_i = (\hat{J}_{k-1}^{\mathrm{T}})_i + K_k \left((b_k)_i - h_{k(k-1)}^{\mathrm{T}} (\hat{J}_{k-1}^{\mathrm{T}})_i \right) \quad i = 1 \ldots n$$

where $(b_k)_i$ is the ith element of b_k. Collecting the n columns $(\hat{J}_k^{\mathrm{T}})_i$ and transposing gives

$$\hat{J}_k = \hat{J}_{k-1} + \left(b_k - \hat{J}_{k-1} h_{k(k-1)} \right) K_k^{\mathrm{T}}. \tag{22}$$

Finally, substituting for K_k, using the definition $h_\theta \equiv h_{k(k-1)}$, and rearranging the λ terms in eqs. (22) and (21) yields the desired results in eqs. (7) and (8), respectively.

Appendix B: Index to Multimedia Extensions

The multimedia extension page is found at http://www.ijrr.org.

Table of Multimedia Extensions

Extension	Type	Description
1	Video	This file provides an animation of an eye-in-hand robotic system as it tracks a square target moving along a square path. The left window shows the robot and target motion. The right window shows the camera view as the robot is servoing. The blue polygon represents the desired view of the target, and the red polygon represents where the target is actually seen. The target appears as a polygon (and not a square) because the optical axis of the camera is not perpendicular to the surface of the target object.

References

Allotta, B., and Colombo, C. 1999. On the use of linear camera-object interaction models in visual servoing. *IEEE Transactions on Robotics and Automation* 15(2):350–357.

Baeten, J., and De Schutter, J. 1999. Improving force controlled planar contour following using on-line eye-in-hand vision based feedforward. In *International Conference on Advanced Intelligent Mechatronics*, September, pp. 902–907.

Chaumette, F., Rives, P., and Espiau, B. 1991. Positioning of a robot with respect to an object, tracking it, and estimating its velocity by visual servoing. In *IEEE International Conference on Robotics and Automation*, April, pp. 2248–2253.

Corke, P. I. 1996. A robotics toolbox for MATLAB. *IEEE Robotics and Automation Magazine* 3(1):24–32.

Corke, P. I., and Hutchinson, S. A. 2001. A new partitioned approach to image-based visual servo control. *IEEE Transactions on Robotics and Automation* 17(4):507–515.

Crétual, A., Chaumette, F., and Bouthemy, P. 1991. Complex object tracking by visual servoing based on 2d image motion. In *14th International Conference on Pattern Recognition*, August, pp. 1251–1254.

Dennis, J. E., and Schnabel, R. B. 1983. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ.

Flandin, G., Chaumette, F., and Marchand, E. 2000. Eye-in-hand/eye-to-hand cooperation for visual servoing. In *IEEE International Conference on Robotics and Automation*, April, pp. 2741–2746.

Franklin, G. F., Powell, J. D., and Workman, M. L. 1990. *Digital Control of Dynamic Systems*. Addison-Wesley, Reading, MA.

Hashimoto, K., and Noritsugu, T. 1999. Visual servoing with linearized observer. In *International Conference on Robotics and Automation*, May, pp. 263–268.

Haykin, S. 1991. *Adaptive Filter Theory*. Prentice-Hall, Englewood Cliffs, NJ.

Hosoda, K., and Asada, M. 1994. Versatile visual servoing without knowledge of true Jacobian. In *IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, September, Munich, Germany, pp. 186–193.

Hosoda, K., Igarashi, K., and Asada, M. 1998. Adaptive hybrid control for visual servoing and force servoing in an unknown environment. *IEEE Robotics and Automation Magazine* 5(4):39–43.

Jagersand, M., Fuentes, O., and Nelson, R. 1997. Experimental evaluation of uncalibrated visual servoing for precision manipulation. In *Proceedings of International Conference on Robotics and Automation*, April, Albuquerque, NM, pp. 2874–80.

Jang, W., and Bien, Z. 1991. Feature-based visual servoing of an eye-in-hand robot with improved tracking performance. In *IEEE International Conference on Robotics and Automation*, April, pp. 2254–2260.

Malis, E. 2002. Vision-based control invariant to camera intrinsic parameters: Stability analysis and path tracking. In *IEEE International Conference on Robotics and Automation*, pp. 217–222.

Malis, E., and Chaumette, F. 2002. Theoretical improvements in the stability analysis of a new class of model-free visual servoing methods. *IEEE Transactions on Robotics and Automation* 18(2):176–186.

Oh, P. Y., and Allen, P. K. 2001. Visual servoing by partitioning degrees of freedom. *IEEE Transactions on Robotics and Automation* 17(1):1–17.

Papanikolopoulos, N. P., Khosla, P. K., and Kanade, T. 1993. Visual tracking of a moving target by a camera mounted on a robot: A combination of control and vision. *IEEE Transactions on Robotics and Automation* 9(1):14–35.

Piepmeier, J. A. 1999. *A Dynamic Quasi-Newton Method for Model Independent Visual Servoing*. Ph.D. Thesis, Georgia Institute of Technology.

Piepmeier, J. A., McMurray, G. V., and Lipkin, H. 1999a. A dynamic quasi-Newton method for uncalibrated visual servoing. In *IEEE International Conference on Robotics and Automation*, May, Detroit, MI, pp. 1595–1600.

Piepmeier, J. A., McMurray, G. V., and Lipkin, H. 1999b. Experimental results utilizing vision-based control for uncalibrated robotic systems. In *SPIE International Symposium on Intelligent Systems and Advanced Manufacturing*, September, Boston, MA.

Yoshimi, B. H., and Allen, P. K. 1994. Active, uncalibrated visual servoing. In *Proceedings of the IEEE International Conference on Robotics and Automation*, San Diego, CA, pp. 156–161.