Uncalibrated Eye-in-Hand Visual Servoing

Jenelle Armstrong Piepmeier
U.S. Naval Academy
Annapolis, MD 21402

Ben A. Gumpert Harvey Lipkin
Georgia Institute of Technology
Atlanta, GA 30332

Abstract

This paper presents 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 and maintain the desired features. 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.

1 Introduction

This paper develops 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 knowing the robot kinematic model and without a calibrated camera model. Error velocity estimation is done simultaneously with Jacobian estimation in a partitioned matrix method. The control method is independent of the type of robot and camera or number of cameras.

Much work has been done in developing visual servoing systems for robot control resulting in a plethora of approaches to the servoing problem. The majority of the resulting methods, however, require a priori knowledge of the system including kinematic structure and link and 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. Underwater, toxic, and outer space settings are some specific examples of workspaces in which calibrations are inherently difficult. To our knowledge, this is the first method that addresses the uncalibrated eye-in-hand visual servoing problem for a moving target.

2 Background

Benefits of using eye-in-hand camera arrangements include improved target recognition and inspection resulting from localization described by Chaumette et al. [1]. Jang and Bien [2] 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 et al. [3].

2.1 Use of Eye-In-Hand Camera

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

Hashimoto and Noritsugu [4] developed a linearized observer to estimate target velocity in their model dependent controller. Allotta and Colombo [5] also use linear approximations in an affine camera-object interaction model. Again, the robot kinematics model is assumed to be known. Baeten and De Schutter [6] 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 used 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 [3]. In each method, control output is fed to a Cartesian positioning system.

Yoshimi and Allen [7] 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 [8] develop a control algorithm for tracking a moving target with no model of the image transformation. The robot system, however, consists of a camera with only two degrees of freedom, 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 [9] build upon the work done by Crétual et al. [8] to create a visual servo controller for a six degree of freedom 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.

Piepmeier et al. [10] builds on work done by Jagersand to develop a dynamic quasi-Newton method of visual servo control. A scheme is presented which allows for a moving target, and the implementation of a recursive least squares algorithm for Jacobian estimation provides 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.

3 Uncalibrated Control for Eye-in-Hand Visual Servoing

This section develops a nonlinear least squares optimization method for a time-varying, coupled system. Specifically presented are:

1. A partitioned Broyden's method for error velocity estimation.

2. Recursive and nonrecursive forms of a Gauss-Newton method implementing the partitioned Broyden update.

3. A correction scheme which produces a more accurate error velocity estimation than the partitioned Broyden update.

The partitioned Broyden update implements a simultaneous recursive least squares estimation of a dynamic Jacobian and error velocity. The error velocity estimation scheme 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 [11] required that the target position y^* be independent of robot joint position θ, and only a function of time. In the moving camera case, this is no longer valid, so $y^* = y^*(\theta, t)$. In successive iterations, the target image position may change due to camera movement, regardless of actual target motion. 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. The partitioned Broyden update is derived based on the affine model of the error function. For brevity, the development is based on the theoretical work already presented in [10] and [11]. The affine model of the error function gives the kinematic relation

$$\Delta\hat{J}h_\theta + \left(\hat{f}_t\right)_k h_t = \Delta f - \hat{J}_{k-1}h_\theta$$

where J is the Jacobian, h_θ is the required joint increment, h_t is the time increment, f is the error between the target and robot locations, f_t is the time partial derivative of the error, Δ indicates a change in successive values, and $\hat{(\cdot)}$ denotes an estimated quantity. A partitioned matrix $\tilde{J}$ and a partitioned vector $\tilde{h}$ are now introduced where

$$\begin{aligned} \tilde{J} &= \begin{bmatrix} \hat{J} & \hat{f}_t \end{bmatrix} \\ \tilde{h} &= \begin{bmatrix} h_\theta & h_t \end{bmatrix}^T \end{aligned}$$

Making these substitutions and taking the transpose of each side yields

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

This is the constraint equation under which the term $\Delta\tilde{J}$ is determined such that the Frobenius norm is at a minimum. The Frobenius norm is defined as $\left|\left|\Delta\tilde{J}\right|\right|_F = \left(\sum_{i,j}\left(\Delta\tilde{J}\right)_{ij}^2\right)^{\frac{1}{2}}$ where $\left(\Delta\tilde{J}\right)_{ij}$ is an element of $\Delta\tilde{J}$. Applying the minimum norm solution gives the Broyden update of the partitioned matrix,

$$\Delta\tilde{J} = \frac{\left(\Delta f - \tilde{J}_{k-1}\tilde{h}\right)\tilde{h}^T}{\tilde{h}^T\tilde{h}} \tag{1}$$

An algorithm for the combination of the modified Broyden update with the recursive scheme given in (1) is as follows:

Algorithm 1 *Gauss-Newton Controller with Partitioned Broyden's Method*

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

$\tilde{h} = \begin{bmatrix} h_\theta & h_t \end{bmatrix}^T$

$$\tilde{J}_{k-1} = \begin{bmatrix} \hat{J}_{k-1} & \left(\hat{f}_t\right)_{k-1} \end{bmatrix}$$
$$\tilde{J}_k = \tilde{J}_{k-1} + \frac{\left(\Delta f - \tilde{J}_{k-1}\tilde{h}\right)\tilde{h}^T P_{k-1}}{\lambda + \tilde{h}^T P_{k-1}\tilde{h}}$$
$$P_k = \frac{1}{\lambda}\left(P_{k-1} - \frac{P_k \tilde{h}^T \tilde{h} P_{k-1}}{\lambda + \tilde{h}^T P_{k-1}\tilde{h}}\right)$$
$$\theta_{k+1} = \theta_k - \left(\hat{J}_k^T \hat{J}_k\right)^{-1}\left(\hat{J}_k^T f_k + \hat{J}_k^T \left(\hat{f}_t\right)_k h_t\right)$$

The partitioned Broyden's method uses recursive least squares 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 a recursive least squares (RLS) formulation with exponential weighting. Gumpert [12] describes a recursive method for updating the desired joint position for target tracking. An algorithm combining this scheme with the partitioned Broyden's method given in the previous is as follows.

Algorithm 2 *Recursive Gauss-Newton Method (RGN)*

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

$$\tilde{h} = \begin{bmatrix} h_\theta & h_t \end{bmatrix}^T$$
$$\tilde{J}_{k-1} = \begin{bmatrix} J_{k-1} & \left(\hat{f}_t\right)_{k-1} \end{bmatrix}$$
$$\tilde{J}_k = \tilde{J}_{k-1} + \frac{\left(\Delta f - \tilde{J}_{k-1}\tilde{h}\right)\tilde{h}^T P_{k-1}}{\lambda + \tilde{h}^T P_{k-1}\tilde{h}}$$
$$P_k = \frac{1}{\lambda}\left(P_{k-1} - \frac{P_k \tilde{h}^T \tilde{h} P_{k-1}}{\lambda + \tilde{h}^T P_{k-1}\tilde{h}}\right)$$
$$\theta_{k+1} = 2\theta_k - \theta_{k-1} - \frac{Q_{k-1}\hat{J}_k^T\left(f_k + \left(\hat{f}_t\right)_k\right)}{\hat{J}_k Q_{k-1}\hat{J}_k^T + \gamma}$$
$$Q_k = \frac{1}{\lambda}\left(Q_{k-1} - \frac{Q_k \tilde{h}^T \tilde{h} Q_{k-1}}{\lambda + \tilde{h}^T Q_{k-1}\tilde{h}}\right)$$

This method minimizes the sum of the squared affine error models for each iteration of $(\theta, t) = (\theta_{i-1}, t_{i-1})$ and the new joint positions are being solved for rather than solving for the changes in the Jacobian and error velocity. 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 give approximations for the Jacobian and the error velocity values. The averaging process of this method, however, may not produce accurate error velocity values. The error velocity values given by the Broyden update can be directly used in a joint update scheme as in Algorithms 1 and 2, or the estimated Jacobian can be used to calculate new values of the error velocity through use of the total time derivative,

$$f_t = \frac{df}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial \theta}\frac{d\theta}{dt}$$

where $\frac{df}{dt} \cong \Delta f / h_t$, the total change in error, and $\frac{\partial f}{\partial \theta}\frac{d\theta}{dt} \cong \left(J h_\theta\right)/h_t$, the change in the error due to end effector motion. The term $\frac{\partial f}{\partial t}$ represents the partial of the change in error with respect to time, or the change due to target motion, since the end effector position is instantaneously fixed. Substituting these values for the kth increment and rearranging yields

$$\left(\hat{f}_t\right)_k = \left(\Delta f - \hat{J}_k h_\theta\right)/h_t \tag{2}$$

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 values given by the partitioned Broyden estimator. Thus, a Gauss-Newton controller can either implement the error velocity values output by the partitioned Broyden's method or the corrected error velocity values given by the total time derivative equation. Equation (2) can be inserted in either Algorithm 1 or 2 after the Jacobian estimation.

4 Simulation and Results

4.1 System Description

A six degree-of-freedom (DOF) system is simulated using the Robotics and Machine Vision Toolboxes developed by Corke for use in the MATLAB environment [13]. The camera is assumed to be coincident with the final frame of the Puma 560 manipulator. A sampling time of $T = 50ms$ is used.

The target consists of four planar feature points spaced in a square with 5 cm sides. The target's initial location is within the camera's field of view. 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

$$(x, y, z) = (0.65, 0.50 + 0.1 \sin(k\omega h_t), 0.1 \cos(k\omega h_t)) \ m$$

where h_t is the sampling period, k is the iteration number, and ω is the frequency. Thus, the target is translating in a circular path. Uniform image noise between ± 0.5 *pixels* 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 four controller schemes simulated here include: (NGN)-Nonrecursive Gauss-Newton controller given by Algorithm 1, (RGN)-Recursive Gauss-Newton controller given by Algorithm 2, (NGNNC)-Nonrecursive Gauss-Newton controller with the Nonrecursive error velocity Correction scheme given in Section 3.3, and (RGNNC)-Recursive Gauss-Newton controller with the Nonrecursive error velocity Correction in Section 3.3

The results of a sample simulation utilizing the NGN 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. If a static camera viewed the feature points, the speed of the target object (moving at 2.5cm/s) would correspond to an feature velocity of 63.5 pixels/s. Figure 2 shows the path the target and robot take during the simulation, and Figure 3 shows the image error. The average steady-state pixel error is 5.35 pixels.

To study the behavior of the four different controllers, the same simulation was repeated 25 times at eight different speeds for each controller. The eight different speeds correspond to a range of 0.5 cm/s to 4m/s or 37 pixels/s to 94 pixels/s (for a static camera). Figure 4 shows the average image error (in pixels) for the repeated trials using the NGN, RGN, NGNNC, and RGNNC controllers. The non-recursive Gauss-Newton controllers show a distinct advantage over the recursive controllers. The velocity correction scheme does not appear to offer any advantage. In fact, the RGNNC errors are somewhat worse than the RGN errors, and the NGNNC simulations exhibit complete loss of control. Interestingly, the NGNNC simulations perform similarly to NGN simulations when the image noise level is reduced by half.

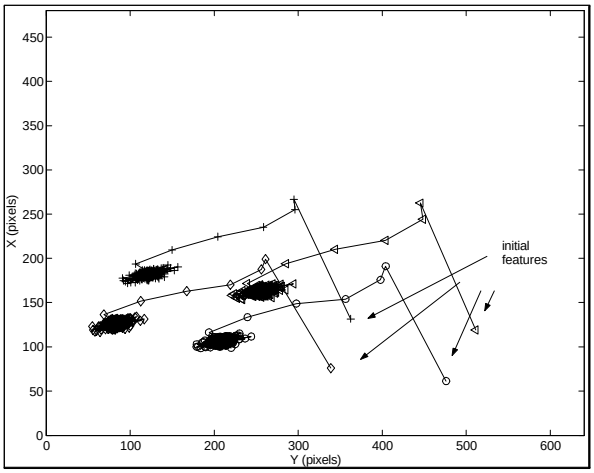

Figure 1: Image features for eye-in-hand visual servoing using an NGN controller with $\lambda = 0.98$ and $\omega = 0.25$ A moving object with four features is seen initially on the right, and the robot is controlled such that the desired image feature locations are viewed. Average steady state image error is 5.35 pixels.

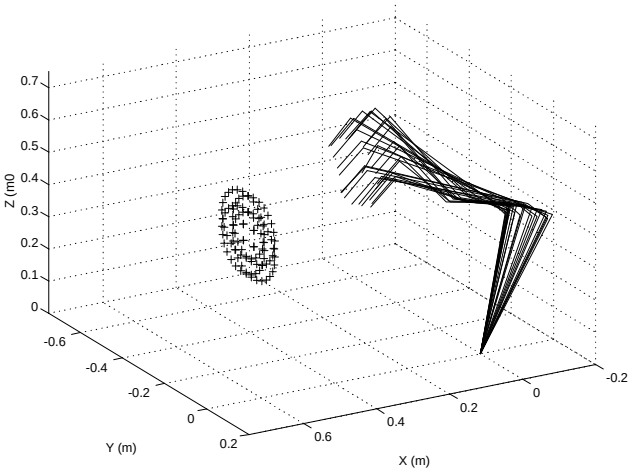

Figure 2: Plot showing the respective positions of the target motion and the robot. The robot links and target positions are drawn for every 20th iteration.

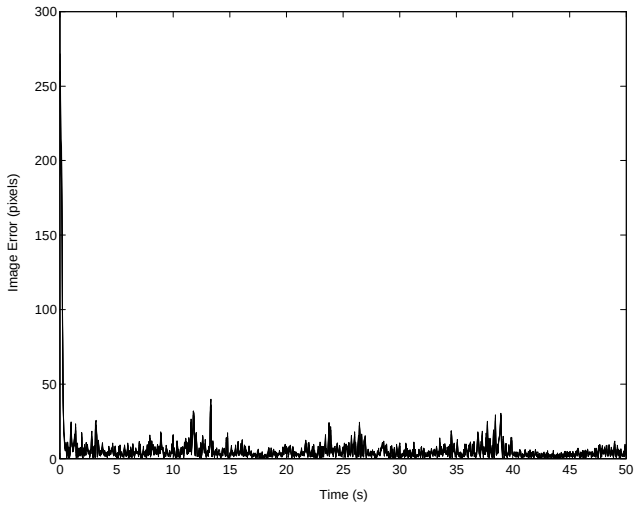

Figure 3: Image error for using an NGN controller with $\lambda = 0.98$ and $\omega = 0.25$. The averate steady state error is 5.37 pixels

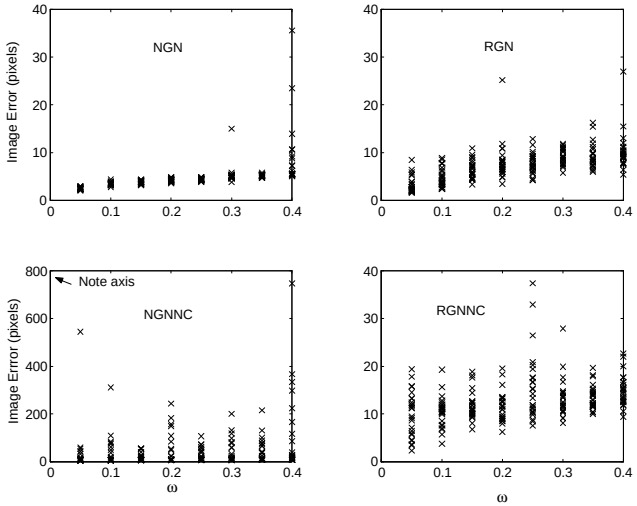

Figure 4: A comparison of four different controllers at a range of target speeds. In general, the NGN controller provides stable convergent control.

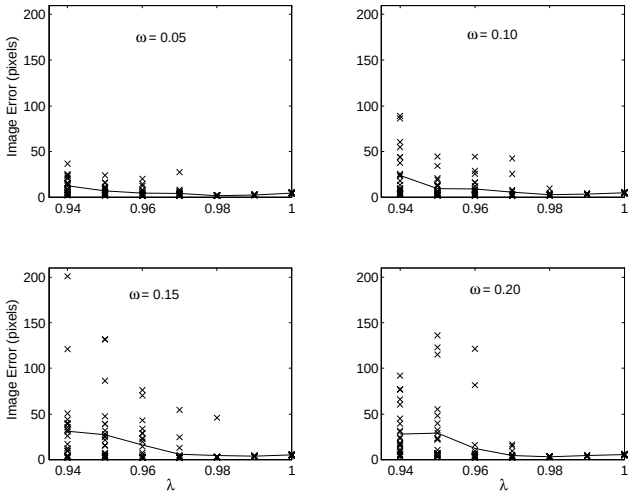

Figure 5: A comparison of NGN controller performance showing image error versus lambda for four different target speeds. Individual errors are plotted for 25 simulations; the line indicates the average error.

4.3 NGN Performance

For the 6 DOF system simulated, the NGN controller produces the best results. To study the effects of the forgetting factor λ, a second series of simulations was run using the NGN controller and varying 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 error for each simulation is plotted as an 'x' and the mean of the averages is plotted as a line. Clearly $\lambda = 0.98$, 0.99, or 1.0 give the best overall performance. Figure 6 plots the mean of the average image error for these three values 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 are concur with similar studies done for the fixed camera case in [10].

5 Conclusion

This paper has 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. Simulation results suggest that a Gauss-Newton method utilizing a partitioned Broyden's method for estimation pro-

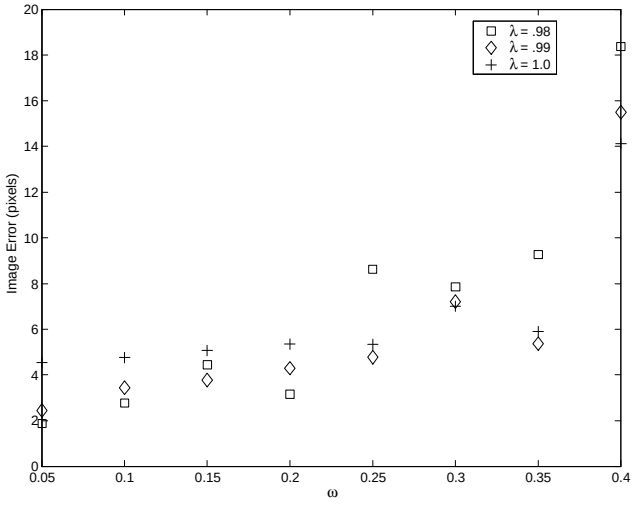

Figure 6: The performance of the NGN controller for $\lambda = \{0.98, 0.99, 1.0\}$ is shown for eight different target speeds, ω.

vides 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. This control strategy could be applied to either a manipulator or mobile robot for uncalibrated control. Future work will focus on the application of this control scheme on experimental systems.

References

[1] 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," in *IEEE International Conference on Robotics and Automation*, pp. 2248–2253, April 1991.

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

[3] T. K. N.P. Papanikolopoulos, P.K. Khosla, "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*, vol. 9, pp. 14–35, February 1993.

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

[5] B. Allotta and C. Colombo, "On the use of linear camera-object interaction models in visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 15, pp. 350–357, April 1999.

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

[7] B. H. Yoshimi and P. K. Allen, "Active, uncalibrated visual servoing," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (San Diego), pp. 156–161, 1994.

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

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

[10] J. A. Piepmeier, *A Dynamic Quasi-Newton Method for Model Independent Visual Servoing*. PhD thesis, Georgia Institute of Technology, 1999.

[11] J. A. Piepmeier, G. V. McMurray, and H. Lipkin, "A dynamic quasi-Newton method for uncalibrated visual servoing," in *IEEE International Conference on Robotics and Automation*, (Detroit, Michigan), pp. 1595–1600, May 1999.

[12] B. A. Gumpert, "A recursive Gauss-Newton method for model independent eye-in-hand visual servoing," Master's thesis, Georgia Institute of Technology, 2001.

[13] P. I. Corke, "A robotics toolbox for MATLAB," *IEEE Robotics and Automation Magazine*, vol. 3, pp. 24–32, March 1996.