Six DOF Visual Control of Relative Position

Gregory D. Hager

Research Report YALEU/DCS/RR-1023
June 1994

YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Six DOF Visual Control of Relative Position

Gregory D. Hager
Department of Computer Science
Yale University
New Haven, CT 06520-8285

Phone: 203 432-6432
Email: hager@cs.yale.edu

June 2, 1994

Abstract

This paper describes a family of feature-based control algorithms for performing relative positioning using stereo vision. The vision and control algorithms have the following properties: they are simple and can be implemented in realtime on using no specialized hardware, they make minimal modifications to or assumptions about the existing robot control system, and they are task-independent.

These methods operate by tracking both the robot end-effector and visual features used to define goal positions. Error signals based on the visual distance between the end-effector and its target are defined and control laws that move the robot to drive this error to zero are derived. These control laws have been integrated into a system that performs tracking and stereo control on a single processor with no special purpose hardware at real-time rates. Experiments with the system have shown that the controller is so robust to calibration error that the cameras can be moved several centimeters and rotated several degrees while the system is running with no adverse effects.

In Preparation for Submission to IEEE Transactions on Robotics and Automation.

1 Introduction

Over the last several years, a great deal of research has been devoted to using vision to guide robotic systems. Despite these efforts vision-based robotic systems are still considered to be slow, unreliable, difficult to calibrate, expensive, and time consuming to build. One reason is the difficulty of the vision problem. Enormous quantities of visual data must be processed quickly and accurately enough to be incorporated into a servo-loop that must execute at millisecond rates. This performance can usually only be achieved by using specialized, task-specific algorithms and or specialized vision processing hardware. A related problem is that such solutions usually do not readily generalize to other problems, so each visual servoing problem must be engineered from scratch. Another difficulty arises from the fact that the accuracy of many visual servoing systems depends on the accuracy to which the transformation from visual coordinates to robot coordinates—the *hand-eye* transformation—is known. Due to the complexity and nonlinearity of the hand-eye transform parameterization, its computation can be computationally intensive, time-consuming and error prone.

An effective, commercially viable visual servoing system must be portable, inexpensive and accurate. Furthermore, it must be constructed so that a large variety of tasks can be solved by composing a small set of basic operations. We have begun the construction of such a set of operations, which we term *hand-eye skills*. A hand-eye skill can be thought of as a vision-based control loop that performs a specific type of geometric motion or maintains a geometric constraint. Example skills would be positioning, alignment, straight-line motion between two reference points, guarded motion between two visual reference features, and so forth. As noted above, it is particularly important that hand-eye skills be:

Robust to hand-eye calibration error, system time delays, and other disturbances;

Task Independent so that they can be reused in many different situations without change; and

Accurate enough to perform high-precision operations that are difficult to perform using an open-loop system.

In order to be generic and easy to support, hand-eye skills must also require only modest, easily acquired types of visual information. Ideally, hand-eye skills should be simple to combine with position and/or force control to form hybrid control systems.

In order to further illustrate the difference between vision-based positioning and the closed-loop visual control inherent in hand-eye skills, consider building a tower of blocks using the system shown in Figure 1(a). The major components of the system are two video cameras on pan-tilt units, a robot arm, and computers that perform image-processing, low-level control operations, and other interface-related functions.

Since stereo cameras are available, it is possible to estimate the geometry of a scene from its stereo projection. Given this geometric information, building the tower of blocks could be programmed in the usual way—that is, by explicitly commanding the robot to move to the appropriate geometric positions and opening and closing its gripper. However, the accuracy of the reconstructed positions depends on the above-mentioned hand-eye transformation. In this system, the hand-eye

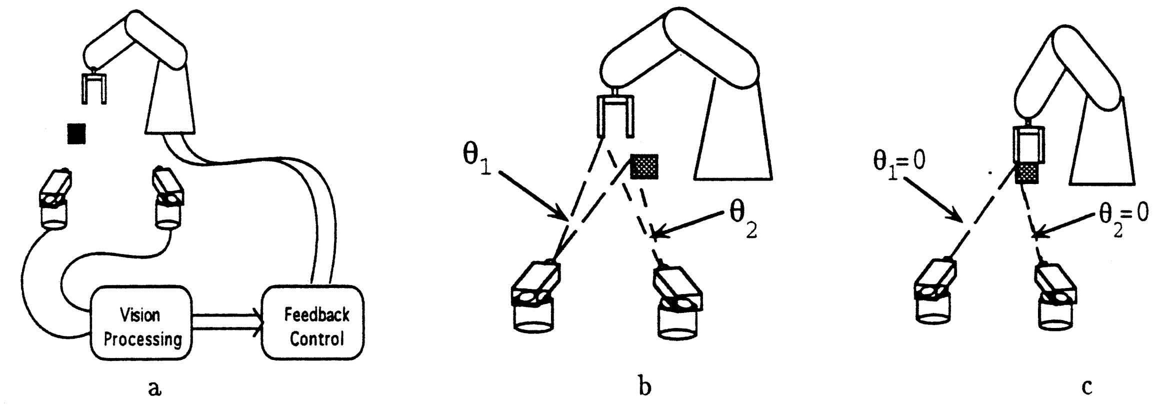


Figure 1. The figure on the left shows a visual servoing system consisting of two cameras on pan-tilt heads connected via a vision-based controller to a robot arm. The middle and right figure demonstrate positioning by reducing visual disparity to zero.

transformation includes the intrinsic parameters (scale factors, distortion coefficients, lens centers and camera constants) and extrinsic (positional) parameters for both cameras. It also includes parameters describing the kinematics of the pan-tilt heads and the kinematics of the robot arm itself. Because of the large number of calibration parameters, modeling errors, and mechanical backlash, even the most careful and intensive calibration process is unlikely to produce an extremely accurate hand-eye transformation. Consequently, it is difficult to predict the accuracy of reconstructed workspace positions or to guarantee the reliability of the system.

The hand-eye skills described in this article control the robot based on observations of *both* the manipulator and a target position during motion. With this information, it is possible to perform extremely accurate vision-based positioning despite calibration errors. Figures 1(b) and 1(c) illustrate the basic principle. In (b), the cameras observe the angles θ_1 and θ_2 between a point on the gripper and the corner of a block. If both θ_1 and θ_2 were zero, the manipulator and the target would be at the same point in space. This statement is true independent of the locations of the cameras as long as the target point and the camera centers are not collinear. Thus, if the robot can be reliably controlled so that $\theta_1 = \theta_2 = 0$, positioning precision depends *only* on how accurately the task-relevant features of the block and the manipulator can be observed. If that positioning precision is enough to guarantee the block will be grasped, then the system is reliable. Likewise, placing one block on top of the other could be expressed as a motion to a position above the stack, alignment of the visible faces of a held block and a block on the stack, and a motion downward while maintaining that constraint. If these operations can also be performed using closed-loop control, similar statements about accuracy and reliability follow. Thus, it seems that if the geometric motions for a task can be described in terms of coordinated motions of features in both camera images, the task can be performed more reliably than by using image-based reconstruction and open-loop positioning.

This article describes a family of methods for relative positioning and alignment based on stereo visual feedback. The input to these control algorithms are generic features—the projections of points and contours—that can be easily and efficiently acquired using window-based visual tracking [10]. It is believed that this approach provides a promising basis for general-purpose vision-based manipulation for the following reasons:

- Since the system uses closed-loop feedback, the accuracy of the system is independent of calibration. In practice, it has never been necessary to perform an accurate calibration of the system.
- The use of stereo makes it possible to position and orient in three dimensions without using prior geometric structure about objects or features involved.
- Target positions for the manipulator are defined using the same visual tracking mechanism used for the manipulator itself. Consequently, task specifications are immune to changes in position or attitude of the target station during execution.
- Since visual feedback is implemented in the servo loop, it is possible to implement hybrid control modes that combine vision with force or position feedback.
- It appears that *adaptive* feedback controllers that tune the calibration while the system runs can be developed.

The remainder of this paper is organized as follows. The next section discusses some of the relevant literature visual servoing literature, and defines two vision-based relative positioning problems. Section 3 describes the solutions to these problems. Section 4 examines the sensitivity of these algorithms to calibration error and projection singularities. Section 5 describes an implemented system and presents several experiments devoted to determine the accuracy and stability of the servoing algorithms. The final section describes work currently in progress on the system.

2 Background and Problem Definition

Visual servoing has been an active area of research over the last 30 years with the result that a large variety of experimental systems have been built (see [4] for an extensive review and [9, 13] for a recent collection of articles). Nearly all systems employ what is referred to as a "look-and-move" approach [27]. This means that the internal robot joint encoders are used to stabilize the mechanism itself, while visual servoing supplies velocities of position offsets expressed in terms Cartesian or joint positions. All of the systems referenced below as well as those presented in this paper are of this type.

A majority of the recently constructed visual servoing systems employ a single camera, typically mounted on the arm itself *e.g.* [5, 18, 23, 24, 25]. Mounting the camera on the arm has two main advantages. First, since the camera is rigidly fixed near the manipulator, the hand-eye transformation is relatively simple to calculate with high accuracy. Second, many operations can be programmed using a ego-centric motions. For example, an object might be grasped by moving "toward" it until contact and then closing the hand. Because it is not possible to recover the three-dimensional structure of a point or line feature from a single image, nearly all of these systems either restrict the degrees of freedom of the manipulator, restrict the location of setpoints to a subspace of the workspace, or use prior knowledge about the two or three-dimensional geometry of a set of features. Systems are also differentiated based on whether they are *position-based* or *image-based.* The former define servoing error in a Cartesian reference frame (using vision-based pose estimation) while the latter compute feedback directly from errors measured in the camera image. Generally, image-based algorithms have been found to be superior to position-based algorithms. They are less expensive computationally [5] and they do not subject the control system to estimator dynamics as do position-based systems. The major disadvantage of an image-based system is that the control problem is nonlinear.

A few systems employ arm-mounted stereo cameras, *e.g.* [15], to overcome the difficulties associated with monocular projection. The major disadvantages of arm-mounted cameras are the extra weight they add to the arm, the fact that mechanical modifications are often required to mount them, the fact that the camera is subject to vibration and danger of collision, and the problems associated with the arm occluding the camera field of view. Most authors who have investigated stereo vision do not mount them on the arm. For example, Allen *et.al.* [1], Rizzi [26], and Anderson [2] describe real-time stereo-vision-based systems. Stereo vision has the obvious advantage that it can provide accurate three-dimensional information about features in an image. For this reason, stereo-based systems are typically constructed as position-based rather than image-based systems—the vision component provides the Cartesian trajectory of a moving object in robot coordinates. These positions are used as a reference trajectory for a separate robot control system.

Most visual servoing systems are only as accurate as their hand-eye calibration. This is easily seen for the stereo systems mentioned above—an error in the estimated hand-eye calibration will cause the generated reference trajectory to be incorrect. It is also true for most arm-mounted systems because most such systems do not mount the camera in the manipulator, but near it. Achieving a particular image-plane feature configuration places the *camera* at a specific point in space; the position of the manipulator is only indirectly governed through its kinematic relationship with the camera. Thus, minor changes in the camera position relative to the manipulator or alterations in the relationship of the features to the object lead directly manipulator positioning error.

The example given in the introduction was based on the idea that *both* the manipulator and the target were observed. Such systems will be referred as "endpoint-closed-loop" (ECL) systems indicating that the control error is based on direct observation of the manipulator. Conversely, the systems described will be referred to as "endpoint-open-loop" (EOL) systems. As described in the introduction, the major advantage of ECL systems is that they can define a positioning error that is *independent* of the hand-eye calibration and thus perform with an accuracy that is independent of hand-eye calibration error.

Few ECL systems have been reported in the literature. Wijesoma *et.al.* [29] describe an ECL monocular hand-eye system for planar positioning using image feedback. In earlier work [8] we described a similar monocular visual servoing system and examined the use of adaptive control to compensate for calibration error. Hollinghurst and Cipolla [16] describe an ECL stereo visual servoing system. The system is based on using an affine approximation to the perspective transformation to reconstruct the position and orientation of planes on an object and on a robot manipulator. This allows them to define a position-based servo algorithm for aligning and positioning the gripper relative to the object. They note that the use of the affine model means that their system calibration is only locally valid which in turn restricts the operating range of their system. A similar system was recently described by Chen *et.al.* [3]. In contrast, the work presented here describes a family of stereo, image-based, look-and-move algorithms for relative positioning problems. No approximations to the perspective model are employed. Furthermore, the methods use generic features and do not require any prior knowledge of object structure or feature positions in order to define a manipulator setpoint.

2.1 Problem Definitions

The following nomenclature is used throughout the remainder of this article. All positions, orientations and feature coordinates are expressed relative to an arbitrary base coordinate system $\mathcal{W}$. A coordinate frame is represented by a pair (t, R), $t \in \Re(3)$, $R \in SO(3)$. Camera positions are represented by the frames $C_1 = (c_1, R_1)$ and $C_2 = (c_2, R_2)$. It is assumed that $c_1 \neq c_2$. A rotation matrix, R_i, is composed of three rows represented by the unit vectors $\vec{x}_i$, $\vec{y}_i$, and $\vec{z}_i$. The infinite line containing both camera positions is referred to as the *baseline* of the system. [1] A plane containing the baseline is referred to as an *epipolar plane,* and the intersection of an epipolar plane with the camera imaging plane is referred to as an *epipolar line.*

It is assumed that estimates of camera intrinsic parameters (parameters describe the mapping from camera pixel coordinates to metric units) and camera extrinsic parameters (the spatial position of the cameras relative to the manipulator coordinate system) are available. To simplify the exposition, all observed values are scaled to metric units for a camera lens with unit focal length (details of computing observed values are given in Appendix A). The units for linear and angular quantities are millimeters or degrees, respectively, unless otherwise specified. Finally, when dealing with vector or matrix quantities, the notation $(a; b)$ denotes column concatenation of the vectors a and b and $(a \mid b)$ denotes row concatenation of a and b.

Points in $\Re(3)$ will be written in capital letters. The projection of a point $P = (P_x, P_y, P_z)^T$ to a homogeneous vector $p_i = (u, v, 1)^T$ in camera image i is given by

$$\begin{aligned} P' &= R_i(P - c_i) \\ p_i &= (u, v, 1)^T = \frac{P'}{P'_z}. \end{aligned} \tag{1}$$

In vector form, this is written $p_i = g_i(P)$.

An arbitrary line, L, is parameterized by a six-tuple $(L_d; L_v)$ where $L_d \in \Re(3)$ is fixed point on the line and $L_v \in \Re(3)$ is a unit vector representing the direction of the line. The vector l_i parameterizing the projection of L in camera image i is

$$L' = R_i\left(L_v \times (L_d - c_i)\right) \tag{2}$$

$$l_i = \frac{L'}{\sqrt{L'^2_x + L'^2_y}} \tag{3}$$

In vector form, this is written $l_i = h_i(L)$.

[1] This differs somewhat from use in *e.g.* [17] where the baseline is the line *segment* defined by the two center points.

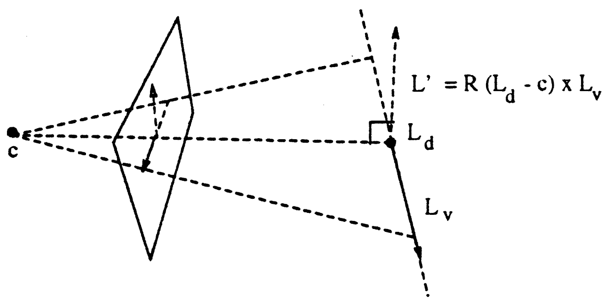


Figure 2. The geometry of line projection

Geometrically, L' is normal to the plane passing through the center of projection of the camera containing the point L_d and the vector L_v. The projection of L in a camera image is the intersection of this plane with the imaging plane. By normalizing the projection as shown, the first two components of l_i encode the normal to the projection of the line, and the final component is the distance from the line to the image origin. This geometry is illustrated in Figure 2.

For any homogeneous vector p_i in the image, it is easy to show that $p_i \cdot l_i$ is the distance between the point and the projection of the line in the image plane. It follows that a homogeneous vector p_i in camera image i lies on the line projection l_i if and only if $p_i \cdot l_i = 0$. Note that line projection is not defined when L_v is parallel to the viewing direction $(L_d - c_i)$.

The position of a robot to be controlled is represented by a coordinate frame $\mathcal{F} = (a, \Omega)$. Suppose the point P and the line $L = (L_d; L_v)$ are rigidly attached to this frame. If $\mathcal{F}$ is moving with translational velocity u and rotational velocity ω expressed in world coordinates, the motions of P and L are given by:

$$\dot{P} = \omega \times (P - a) + u \tag{4}$$
$$\dot{L}_d = \omega \times (L_d - a) + u \tag{5}$$
$$\dot{L}_v = \omega \times L_v \tag{6}$$

In the sequel, the instantaneous motion of the robot is represented by the vector $q = (u; \omega)$.

A *relative positioning skill* is a control system or strategy that adjusts the position of a robot so that a particular geometric relationship between features in the world and features rigidly attached to a robot is attained. Although there is a potentially large set of such skills, most can be defined as combinations of or variations on the following two problems:

Problem Definition 1 (Relative Position): Given a reference point P fixed with respect to $\mathcal{W}$ and a reference point S rigidly attached to $\mathcal{F}$, develop a regulator that positions $\mathcal{F}$ so that $P = S$ using p_i and s_i, $i = 1, 2$.

Problem Definition 2 (Relative Position and Orientation): Given three non-collinear, reference points P, S and T rigidly attached to $\mathcal{W}$ and two non-parallel

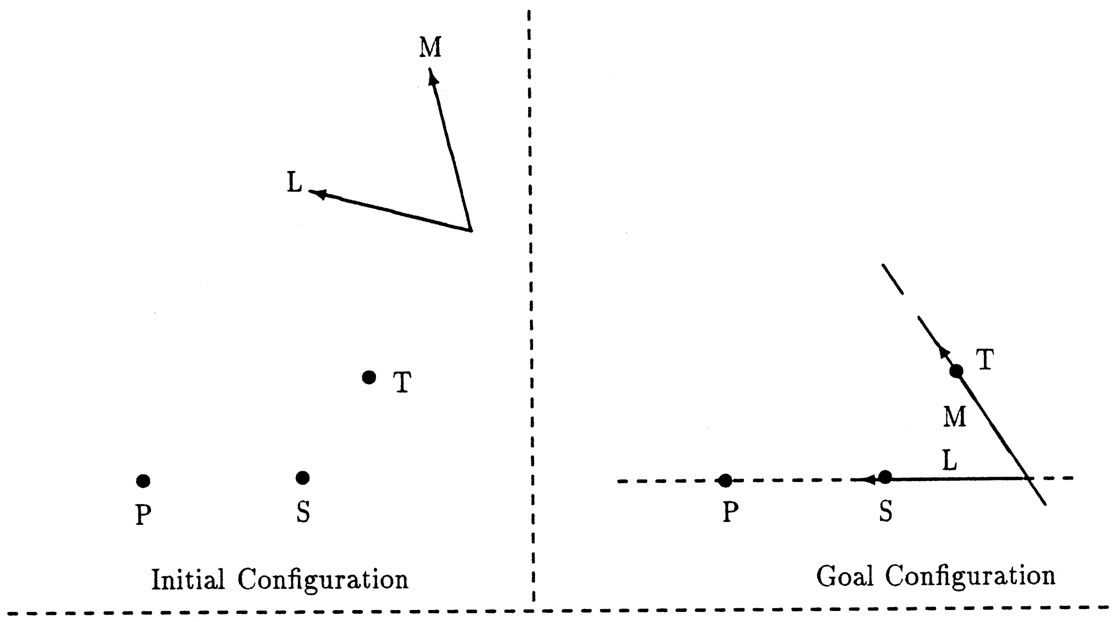


Figure 3. The geometry of the positioning and alignment problem. The lines L and M define a coordinate system as do the points P, S and T. The conditions $P \in L$, $S \in L$ and $T \in M$ fully constrain the relationship between those coordinates systems.

reference lines L and M rigidly attached to $\mathcal{F}$, develop a regulator that positions $\mathcal{F}$ so that $P \in L$, $S \in L$, and $T \in M$ using p_i, s_i, t_i, l_i, and m_i, $i = 1,2$ (see Figure 3).

Clearly, the first problem constrains only three degrees of freedom and so the natural approach is to fix the orientation of the robot and control only position. The second problem constrains exactly six degrees of freedom. To see this, note that positioning the points P and S on L requires two degrees of rotational freedom and two degrees of translational freedom. When $P \in L$ and $S \in L$, satisfying $T \in M$ can be accomplished by first rotating about the line L until $((T-M_d)\times M_v)\cdot L_v = 0$. L_v is now parallel to the plane defined by M and T, so it is possible to translate along L until $T \in M$.

3 Solutions

As noted in Section 2, there are two possibilities for constructing a control algorithm for these problems. One possibility is to develop a position-based system. This involves reconstructing the positions of points and lines in the robot coordinate frame, and then defining a positioning error in terms of reconstructed positions. The main advantage of this approach is that the control problem is simplified (it is often linear). However, the disadvantage is that the dynamics of the estimation system affect the dynamics of the control system. In order to avoid the latter problem, the control systems described here are all image-based—control errors are defined directly in terms of feature observations. Since projection of both lines and points is nonlinear, any image-based control formulation is also nonlinear. Consequently, it would be difficult to design an optimal closed-loop regulator for either problem. A practical alternative is to employ position-integral-derivative (PID) control formulations [6]. For the problems described above, the theory underlying feedback control algorithms of this type can be summarized as follows.

The robot is modeled as a Cartesian positioning device with negligible dynamics. The control inputs are translational velocities u and rotational velocities ω expressed relative to $\mathcal{W}$ and applied about the point a. With no dynamics the robot appears as an integrator to the control system. As noted above, this assumption is generally reasonable for a look-and-move system since the arm is stabilized by internal encoder feedback.

Suppose that $y = f(x)$, is a nonlinear mapping from a robot configuration space to an output (sensor) space, both of dimension n. Given a desired setpoint y^*, define $e_y = y - y^*$ and introduce a new variable z such that $\dot{z} = e_y$. Taking time derivatives of the error yields the system:

$$\begin{aligned} \dot{e_y} &= J_f(x)\dot{x} \\ \dot{z} &= e_y. \end{aligned} \tag{7}$$

where J_f is the Jacobian of f. Because f is nonlinear J_f is a function of the system state, x. If the system state is not directly available, it must be estimated from sensor data.

Define $u = \dot{x}$ to be the control input to the system. Provided J_f is full rank, u is computed as

$$u = -J_f(x)^{-1}(k_1 e_y + k_2 z), \quad k_1 > 0,\ k_2 \geq 0 \tag{8}$$

where k_1 and k_2 are constants set by the designer based on desired system performance and other practical considerations. If the system setpoint y^* is static, and the motion system is of the look-and-move type, then it is possible to set $k_2 = 0$. This version of the control system is sometimes referred to as a *resolved rate* system [28]. When $k_2 \neq 0$, the presence of the integrator, z, ensures convergence in the presence of external disturbances, in particular when the system setpoint is a linear function of time. Although not done here, it is common to add a derivative term to increase stability of the system and offset the tendency to oscillate under the influence of the integrator.

3.1 Solutions

The construction of a visual servoing system for a problem depends on the definition of an appropriate error term, derivation of the system Jacobian, construction of estimators for unknown quantities in the Jacobian, and finally a tuning process for choosing k_1 and k_2 and any other parameters of the system. Applying this process to the problems defined above yields the following solutions.

Problem 1 The following well-known property formalizes the example in the introduction:

Lemma 3.1 Any arbitrary points P and S not on the camera baseline are coincident in space if and only if $s_1 = p_1$ and $s_2 = p_2$.

Thus, define the joint projection function $g(x) = (g_1(x); g_2(x))$, the setpoint $y^* = g(P)$, the error $\alpha = g(S) - g(P)$, and consider the robot configuration space to be only translation. Defining $D_i = (S - c_i) \cdot \vec{z}_i$, we have

$$J_\alpha(S) = \begin{bmatrix} \frac{\vec{x}_1^T D_1 - \vec{z}_1^T((S-c_1)\cdot\vec{x}_1)}{D_1^2} \\ \frac{\vec{y}_1^T D_1 - \vec{z}_1^T((S-c_1)\cdot\vec{y}_1)}{D_1^2} \\ 0 \\ \frac{\vec{x}_2^T D_2 - \vec{z}_2^T((S-c_2)\cdot\vec{x}_2)}{D_2^2} \\ \frac{\vec{y}_2^T D_2 - \vec{z}_2^T((S-c_2)\cdot\vec{y}_2)}{D_2^2} \\ 0 \end{bmatrix} \tag{9}$$

There is a slight difficulty due to the fact that g maps three values—the Cartesian position of a point—into six values—the homogeneous camera image locations of the projections of the point. The two constant values can be ignored, but one of the remaining four nonconstant values is redundant. Expression (8) can be modified to accommodate overdetermined systems as follows:

$$u = -(J_\alpha(S)^T J_\alpha(S))^{-1} J_\alpha(S)^T (k_1\alpha + k_2 z), \quad k_1 > 0,\ k_2 \geq 0 \tag{10}$$

Since the error vector has four components and the control vector only three, this control law only guarantees convergence of a projection of the error term. The complete error vector will be driven to zero only if the errors inhabit a specific three-dimensional manifold in $\Re(4)$. This is not a problem since the geometry of camera projection ensures that the observed error values do lie on this manifold. In fact, the redundancy will tend to suppress random observation errors.

In many cases the final redundant error can be easily removed. For example, if the cameras are arranged so that the y axes are approximately parallel, then one of the y components of the camera observations be discarded. More generally, it is possible to use the camera calibration to define the epipolar plane through the goal [17]. The observation components can be decomposed into components parallel to and perpendicular to this plane, and one of the components perpendicular to it can be discarded.

Problem 2 The error term for problem two is based on the following observation:

Lemma 3.2 Given an arbitrary line L that does not lie in an epipolar plane and a point P not on the baseline, $l_1 \cdot p_1 = l_2 \cdot p_2 = 0$ if and only if $P \in L$.

Proof: From (1) and (3), under the stated assumptions it follows by substitution that the conditions above are equivalent to:

$$\begin{aligned} [R_i(P - c_i)] \cdot [R_i((L_d - c_i) \times L_v)] &= \\ (P - c_i)^T R_i^T R_i((L_d - c_i) \times L_v) &= \\ (P - c_i) \cdot ((L_d - c_i) \times L_v) &= 0, \quad i = 1, 2. \end{aligned}$$

This expression indicates that point P lies in the plane defined by the viewing direction $L_d - c_i$ and the direction vector L_v. In order to satisfy this constraint for both cameras simultaneously P must lie at the intersection of the two viewing planes. But unless the planes for both cameras are identical, in which case the plane is an epipolar plane, this intersection is exactly the line L and hence $P \in L$. ■

Define a positioning error $\beta \in \Re(6)$ as follows:

$$\beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \end{bmatrix} = \begin{bmatrix} p_1 \cdot l_1 \\ p_2 \cdot l_2 \\ s_1 \cdot l_1 \\ s_2 \cdot l_2 \\ t_1 \cdot m_1 \\ t_2 \cdot m_2 \end{bmatrix} \tag{11}$$

Based on Lemma 3.2, it follows that, modulo a set of singular configurations, $\beta = 0$ if and only if $P \in L$, $S \in L$, and $T \in M$.

In computing the time derivative of β, recall that the expression $a \times b$ can be written as $sk(a)b = sk(-b)a$ where the skew symmetric matrix is:

$$sk((x,y,z)^T) = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}.$$

The time derivative of the first component of the error, β_1, is composed of two terms: the Jacobian of the normalization operation, and the time derivatives of unnormalized projection, $m = R_1(L_v \times (L_d - c_1))$. The Jacobian of the normalization operation is:

$$N((m_x, m_y, m_z)^T) = \frac{1}{(m_x^2 + m_y^2)^{3/2}} \begin{bmatrix} m_x^2 & -m_x m_y & 0 \\ -m_x m_y & m_y^2 & 0 \\ -m_z m_x & -m_z m_x & (m_x^2 + m_y^2) \end{bmatrix}.$$

The full derivative is then

$$\begin{aligned} \dot{\beta}_1 &= p_1^T \dot{l}_1 = p_1^T N(m)\dot{m} && (12) \\ \dot{m} &= R_1 \left[sk(L_v)\, u + (sk(L_v) sk(a - L_d) + sk(L_d - c_1) sk(L_v))\, \omega \right] \\ &= R_1 \left[sk(L_v) \mid (sk(L_v) sk(a - L_d) + sk(L_d - c_1) sk(L_v)) \right] q && (13) \end{aligned}$$

If the origin of $\mathcal{F}$ is at L_d (13) simplifies to

$$\dot{m} = R_1 \left[sk(L_v) \mid sk(L_d - c_1) sk(L_v) \right] q$$

The expressions for the remaining error terms can be computed analogously. The collection of these row vectors defines the system Jacobian $J_\beta(P, S, T, L, M)$. Applying (8) using β and J_β leads to a regulator for position and orientation.

3.2 Parameter Estimation

All of the Jacobian matrices defined above depend on estimates of the Cartesian position and orientation of features used to define positioning error. This section examines the problem of estimating the direction of lines and the positions of points as well as the center of rotation, a, of the robot.

All point and line estimation is performed first deriving a closed-form solution to provide an initial estimate, and then using difference equations based on perturbation of a closed form estimation method. These methods have proved sufficient for the applications described in this paper. However, more sophisticated techniques, *e.g.* the extended Kalman Filter [7] could easily be substituted if more is known about the noise characteristics of the system.

Points For an arbitrary point P, the projection equations for C_i can be written in the form:

$$A_i(p_i)P = b_i(p_i)$$

where $p_i = (u_i, v_i, 1)$ is the observation of P in camera i, A_i is:

$$A_i = \begin{bmatrix} \vec{z}_i u_i - \vec{x}_i \\ \vec{z}_i v_i - \vec{y}_i \end{bmatrix} \tag{14}$$

and b_i is

$$b_i = A_i c_i. \tag{15}$$

Let $p = (p_1; p_2)$, $A(p) = (A_1(p_1); A_2(p_2))$ and $b(p) = (b_1(p_1); b_2(p_2))$. The joint system can now be written:

$$A(p)P = b(p).$$

Define the left inverse operator $M^{\#} = (M^T M)^{-1} M^T$ for any matrix M for which the expression is well-defined. Then it is possible to solve directly for P as:

$$\hat{P} = \left[A(p)^T A(p)\right]^{-1} A(p)^T b(p) = A^{\#}(p) b(p). \tag{16}$$

Given a prior estimate $\hat{P}$, let $e = A(p)\hat{P} - b(p)$. Substituting into (16)yields:

$$\hat{P}^+ = A^{\#}(A(p)\hat{P} - e) = \hat{P} - A^{\#}(p)e.$$

If P is rigidly attached to $\mathcal{F}$ this expression can be combined with feedforward prediction to yield the sequential estimation rule:

$$\dot{\hat{P}} = u + \omega \times (\hat{P} - a) - k_3 A^{\#}(p)e \tag{17}$$

where $0 < k_3 \leq 1$ is tuned based on the noise characteristics of the system. The convergence of this system requires A to have rank 3. It is straightforward to verify algebraically that it will provided P does not lie on the baseline.

Lines For an arbitary line $L = (L_v; L_d)$, L_d is chosen to be a fixed point which is observed and estimated as described above. The direction of the line is given by

$$\widehat{L_v} = R_1^T l_1 \times R_2^T l_2. \tag{18}$$

An incremental update for line direction can be derived by defining $\delta_i = h_i(\widehat{L}) \times l_i$. Let $\hat{l}_i = h_i(\widehat{L})$ and note that for small differences $l_i \approx \hat{l}_i - \delta_i \times \hat{l}_i$, $i = 1, 2$. Thus, let $e_i = \delta_i \times \hat{l}_i$. Substituting into (18) yields:

$$\begin{aligned} \widehat{L_v}^+ &= R_1^T(\hat{l}_1 + e_1) \times R_2^T(\hat{l}_2 + e_2) & (19) \\ &= \widehat{L_v} + R_1^T\hat{l}_1 \times R_2^T e_2 + R_1^T e_1 \times R_1^T \hat{l}_2 + (R_1^T e_1 \times R_2^T e_2) & (20) \end{aligned}$$

Combining with prediction, this leads to the update equation:

$$\dot{\widehat{L_v}} = \omega \times \widehat{L_v} + k_4 \left[R_1^T\hat{l}_1 \times R_2^T e_2 + R_1^T e_1 \times R_1^T \hat{l}_2 + (R_1^T e_1 \times R_2^T e_2) \right]. \tag{21}$$

An approximate rule can be derived by noting that the final term involves products of small factors. Dropping this term and rewriting using triple products leads to the simpler form:

$$\dot{\widehat{L_v}} = \omega \times \widehat{L_v} + k_4 \left[(R_1^T\delta_1 - R_2^T\delta_2)\, R_1^T\hat{l}_1 \cdot R_2^T\hat{l}_2 + (R_1^T\delta_1 + R_2^T\delta_2) \times \widehat{L_v} \right]. \tag{22}$$

The result of this estimation rule is then normalized to a unit vector.

These rules must also account for the fact that at observation time the assignment of line normals is arbitrary, and consequently so is the sign of $\widehat{L_v}$. Let θ_i denote the angle that the projection of L makes with the x axis of camera image i, define the vector $b_i = [\cos(\theta_i), \sin(\theta_i), 0]$, and finally let $s_i = b_i \cdot R_i\widehat{L_v}$. If the direction of the projection of L and its estimate agree, this quantity will be positive, otherwise it is negative. If $s_1 * s_2 < 0$, the assignment of line normals to inputs is *inconsistent* (there is no line such that the direction of the input agrees with the specified direction of the line in both images). Otherwise, the correct estimate of line direction is $\mathrm{signum}(s_1)\widehat{L_v}$. This forces the estimated direction of the line to agree with the direction assigned in the camera image.

Center of Rotation There are several possibilities for determining a, the origin of the robot coordinate system. First, estimation of a can be dispensed with if a is fixed to be the origin of $\mathcal{W}$ and an accurate kinematic transformation between $\mathcal{W}$ and $\mathcal{F}$ is always available. The primary disadvantages of this approach are: 1) the controller must have direct access to the position of the robot; and 2) rotation and translation become highly coupled which tends to destabilize the system in practice. If a is a known point on the robot arm, the value of a can computed from the robot kinematics. The disadvantage of this approach is that any error in the hand-eye calibration will immediately manifest itself in an incorrect value for a. Furthermore, this method again requires the robot position to be directly accessible to the controller.

Another approach is to estimate a using camera information. If a is observable by the cameras, then a can be calculated directly using the point estimation techniques described above. More generally, given any three non-colinear points P_1, P_2, and P_3 with known coordinates in the robot coordinate $\mathcal{F}$, and estimated quantities $\hat{P}_1$, $\hat{P}_2$, and $\hat{P}_3$ in global coordinates, it is possible to directly compute a by solving the equations:

$$\hat{P}_i = a + \Omega P_i$$

for a and Ω [12, Chapter 14].

Most robot arms support the definition of a *tool frame*. Given estimates for a and L_d, the center of tool frame can be offset by the vector $L_d - a$. This will cause rotations to occur about the point L_d which simplifies the Jacobian computation as noted in Section 3.1.

3.3 Variations

There are several useful variants on Problem 2 that allow for different feature inputs or different types of motions. The roles of points and lines in the definition of β can be interchanged provided the Jacobian matrix is adjusted to account for the modified definition. One way of defining L (or M) is by choosing two reference points, call them X and Y, and computing the line that runs through them. Since perspective preserves colinearity, the directed lines $(x_1 \rightarrow y_1)$ and $(x_2 \rightarrow y_2)$ can be used to compute l_1 and l_2. The problem formulation is also independent of whether the cameras are stationary in the environment or mounted on the manipulator itself. The only differences in the solutions are the placement of feedforward terms in the estimation equations (given below) and some minor adjustments to the system kinematics.

Other types of regulation can be defined by suppressing or altering parts of β. Define an error κ to be β without components involving M and T. Since the Jacobian, J_κ, is nonsquare in this case, the control vector is computed as:

$$u = -J_\kappa^T (J_\kappa J_\kappa^T)^{-1}(k_1 \kappa + k_2 z).$$

This defines an *alignment* hand-eye skill. The regulator aligns two points to an axis, but leaves rotation about the axis and translation along the axis free. Figure 4 illustrates how this positioning primitive could be used to align a screwdriver with a screw. Tracking the sides of the screwdriver shaft and the screw define the centerline of both. The centerline of the shaft defines the line L and the intersection of the centerline of the screw with its head and the surface define the points P and S. Once the alignment is accomplished, the alignment skill can be combined with a force along the alignment axis and a rotation about the alignment axis to turn the screw.

Choosing an additional setpoint, x_i, along l_i in *only one* of the camera images and adding the error term $\|x_i - p_i\|$ to κ constrains one more degree of translational freedom leaving only rotations about the line free. Figure 4 shows how this variation can be used to move a floppy disk to a disk drive slot. The corners of the disk to define two points P and S. The drive slot defines the line L and the edge of the drive slot provides a setpoint X along L at which to place P.

The modified error term

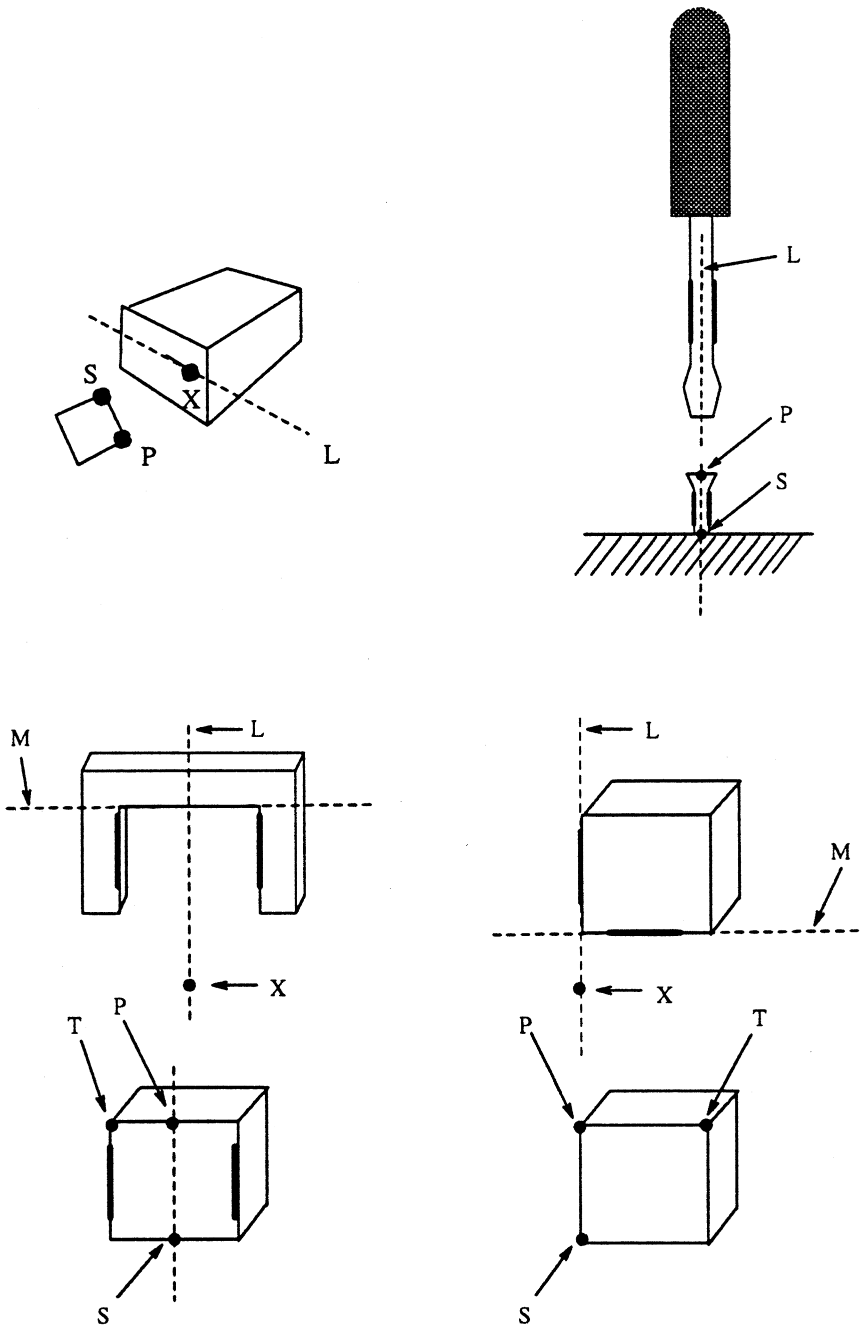


Figure 4. Examples of tasks using visual positioning. Upper left: positioning a floppy disk at a disk drive opening. Upper right: aligning a screwdriver with a screw. Lower left: setpoints for moving a gripper above a block and then down onto it. Lower right: similar setpoints for stacking the block.

$$\gamma = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 - \beta_1 \\ \beta_4 - \beta_2 \\ \beta_5 \\ \beta_6 \end{bmatrix} = \begin{bmatrix} p_1 \cdot l_1 \\ p_2 \cdot l_2 \\ (s_1 - p_1) \cdot l_1 \\ (s_2 - p_2) \cdot l_2 \\ t_1 \cdot m_1 \\ t_2 \cdot m_2 \end{bmatrix} \tag{23}$$

defines an error that permits independent control of alignment to the line L (γ_3 and γ_4) and positioning on L (γ_1 and γ_2.) That is, the third and fourth components can be used to define a regulator that controls two rotations for alignment leaving all other degrees of freedom uncontrolled. This can be combined with any positioning regulator (*e.g.* the one defined above based on the error term α) to provide an alternative formulation for positioning and alignment. The floppy disk problem could have also been solved using this formulation.

Figure 4 shows how these primitives could be combined for the block stacking problem. Using centerline constructions as discussed above to define L, P, S, and X, the manipulator can be moved into a grasping position above a block. Using L, M, P, S, and T the manipulator can be moved onto the block. Similar constructions can be used to move the block above an existing stack, and then to place the block onto the stack with the proper alignment.

4 Analysis

In the absence of noise, the systems defined above are guaranteed to be stable provided they do not move through configurations in which the Jacobian matrix is singular. However, since the control problems are nonlinear, it is extremely difficult to analyze the paths the controllers generate. Implementations of these algorithms have shown them to be extremely stable, even when exposed to fairly radical errors in system calibration. These issues are discussed further below.

4.1 Sensitivity to Calibration Error

One of the major concerns in visual servoing is sensitivity of servoing error to errors in the hand-eye calibration. Although, the nonlinearity of the controller makes it difficult to derive results on general system stability, a description of the set of the locally stable points in the workspace can be derived in certain special cases. In general, the behavior of the continuous-time closed-loop control system follows by substituting (8) into (7) yielding

$$\dot{e_y} = J_f^{-1}(J_f(k_1 e_y + k_2 z)) = k_1 e_y + k_2 \int_0^t e_y$$

Or, equivalently

$$\ddot{e_y} = k_1 \dot{e_y} + k_2 e_y.$$

It is well known that solutions to this system of equations are *always* stable provided the Jacobian matrix is nonsingular for all points along the path. If the integral gain, k_2, is set 0, errors decrease asymptotically with time regardless of k_1. These results assume no parameter (calibration) error.

Of primary interest in any stereo system is the effect of errors in the relative positions of the cameras on the servoing system. To this end, consider relative positioning confined to the $x - z$ plane with $k_2 = 0$. Assume observations are noise-free. The location of two cameras in the plane is parameterized by the length of the baseline, l, and two angles θ_1 and θ_2 describing the orientation of each camera relative to the baseline. An angle of zero indicates the optical axis of the camera is perpendicular to the baseline.

The origin of the coordinate system will be chosen as the center of the baseline, so camera 1 is located at $(-l/2, 0)$ and camera 2 is located at $(l/2, 0)$. Let $C^* = (l^*, \theta_1^*, \theta_2^*)$ denote the true (physical) system configuration, and $C = (l, \theta_1, \theta_2) = (Kl^*, \delta\theta_1 + \theta_1^*, \delta\theta_2 + \theta_2^*)$ be the estimated system calibration parameters. Under these assumptions, the control value is computed as

$$\hat{S} = A^{-1}(g(S, C^*), C)\, b(g(S, C^*), C) \tag{24}$$
$$u = -J_g^{-1}(\hat{S}, C)\,(g(P, C^*) - g(S, C^*)) \tag{25}$$

where $g(\cdot,\cdot)$, $A(\cdot,\cdot)$, $b(\cdot,\cdot)$ and $J_g(\cdot,\cdot)$ are based on the 2-D versions of the quantities described in expressions (1), (14), (15) and (9) with the calibration parameters represented explicitly.

In this case, the closed-loop behavior of the system is governed by the equation

$$\dot{e} = -J_g(S, C^*) J_g^{-1}(\hat{S}, C) e = M(S, C, C^*) e. \tag{26}$$

For a given configuration, C^*, and estimated calibration, C, this system will be stable in the neighborhood of P if $\mathrm{Det}(M(P, C, C^*)) < 0$ and $\mathrm{Tr}(M(P, C, C^*)) > 0$. Substituting (24) into (26) and solving the equation $\mathrm{Det}(M(P, C, C^*)) = 0$ for P_z as a function of P_x yields the solutions:

$$P_z = 0 \tag{27}$$
$$P_z = \tan(\delta\theta_1)(P_x - 1) \tag{28}$$
$$P_z = -\tan(\delta\theta_2)(P_x + 1) \tag{29}$$
$$(P_z - \cot(\delta\theta_1 - \delta\theta_2))^2 + P_x^2 = \csc(\delta\theta_1 - \delta\theta_2)^2 \tag{30}$$

The first equation reiterates that points on the baseline are always unstable. The next two equations define a cone-like region that is bounded by lines forming angles $\delta\theta_1$ and $\delta\theta_2$ with the baseline. The final equation defines a circle which is in front of the cameras when the physical cameras point more toward each other than the calibration parameterization indicates ($\delta\theta_1 - \delta\theta_2 > 0$) and behind the cameras otherwise. It is easy to show that the center of the circle is stable when it lies in front of the cameras.

Solving $\mathrm{Tr}(M(S, C, C^*)) = 0$ yields a bicubic equation in P_x and P_z. Although it is possible to obtain general solutions to this form using computer algebra packages, more insight can be gained by considering the simpler case where the calibration assumes the cameras point straight ahead ($\theta_1 = \theta_2 = 0$) and the physical cameras verge in a coordinated fashion ($-\theta_1^* = \theta_2^*$). In this case, the solutions are:

$$P_z = \tan(\delta\theta_1) \tag{31}$$
$$(P_z - \cot(\delta\theta_1))^2 + P_x^2 = \csc(\delta\theta_1)^2 \tag{32}$$

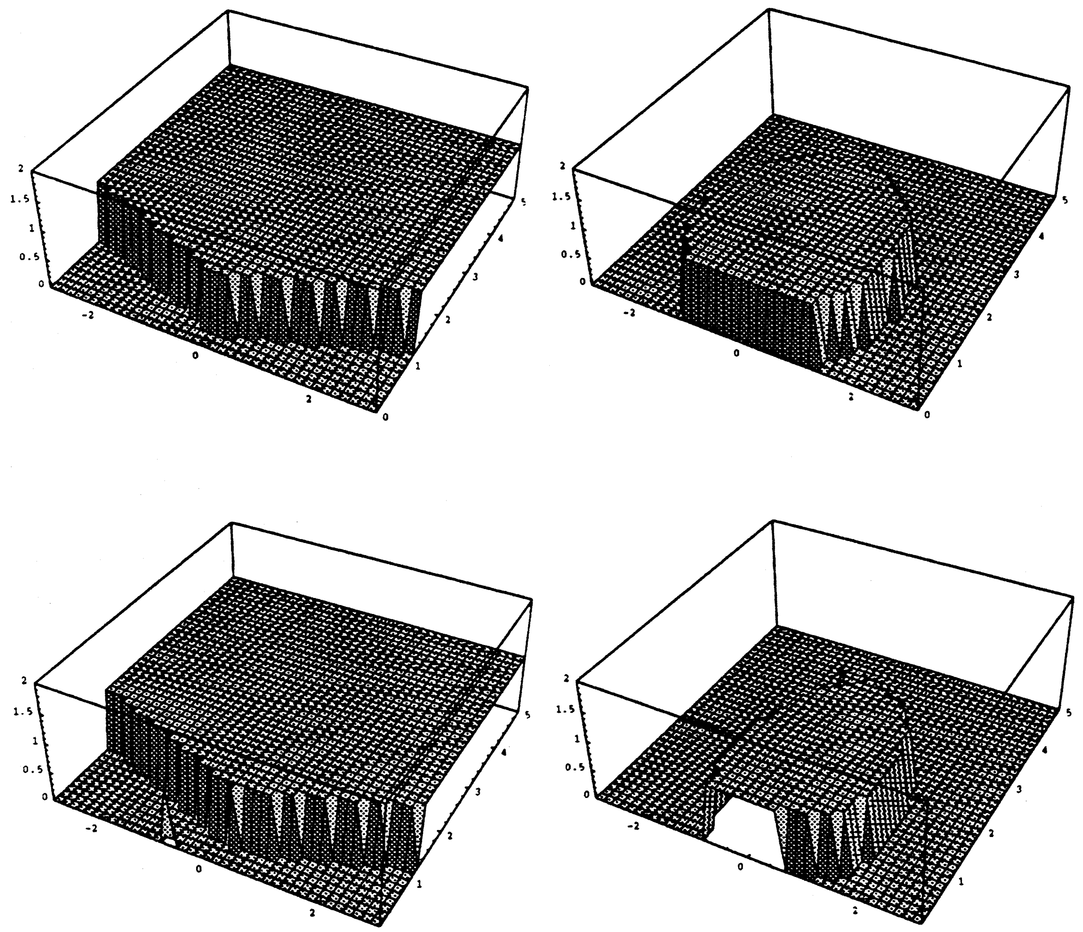


Figure 5. Stability plots for cameras pointed together 0.3 radians (left) and apart 0.3 radians (right). The plots above apply to the two degree of problem, the plots below to the six degree of freedom problem.

The first equation defines a line that is in the unstable region already defined above. The second equation again defines a circle which is in front of the cameras for positive errors in camera vergence and behind the camera otherwise. However, the the radius of the circle defined above is smaller, so it defines the boundary of the stable region when $\delta\theta_1 > 0$. When $\delta\theta_1 < 0$ the stable region is a cone bounded by the lines defined by (28) and (29). Hence, for a fixed set of estimated calibration parameters, the set of stable points shrinks quickly as the physical cameras are rotated toward each other, but relatively slowly as they are rotated away from each other.

Figure 5 shows the stability regions for $\delta\theta_1 = -15$ (cameras pointed inward 15 degrees radians) and the corresponding region for $\delta\theta_1 = 15$ radians. By way of comparison, the stability region for the six DOF problem is also shown. It appears that the same qualitative stability results also apply for that problem.

Notice that the estimated camera baseline, l does not appear in any of the stability equations. In the continuous time case, stability at a point does not depend on baseline errors. This is because the ratio l/l^* appears as a gain in the closed-loop equation. Likewise, errors in any other coefficients that enter the equations as scale factors (*e.g.* camera focal length or conversions from pixel to metric coordinates) do not affect system stability in the continuous time case.

In order to understand the effect of these parameter errors, consider a discrete version of the system with unit time delay and no calibration parameter error:

$$e_{n+1} = e_n + t\ k\ e_{n-1}$$

where k is a gain coefficient and t is the sampling time (one over the sampling rate). This system has characteristic polynomial $x^2 + x + tk = 0$ [6]. System is stability is guaranteed when $k < 1/t$. The system is overdamped if $tk < 1/4$ and underdamped if $tk > 1/4$. So, for example, if $t = 0.1$, then k must be less than 2.5 to prevent oscillation.

Since errors in the intrinsic camera calibration parameters and the system baseline appear as scale factors, their effect can be viewed as modifying the system gain. Calibration errors that increase the effective gain and tend to destabilize the system, while the opposite errors lower the effective closed-loop gain and tend to damp the system. Consequently, the effect of incorrect focal length, scale factors and baseline as well as relative camera orientation can be qualitatively summarized as follows.

When there are calibration errors, to increase system damping:

Relative Orientation: The physical camera setup should point outward from the calibration description.

Baseline: The physical camera baseline should be wider than the calibrated system baseline.

Scale Factors: The true camera scale factors (in mm/pixel) should be smaller than the calibrated scale factors.

Focal Length: The true camera focal length should be smaller than the calibrated camera focal length.

Of course, the more accurate the system calibration, the better the system performance.

4.2 Singularities

Nonlinear systems typically have singular configurations. These are cases where the system Jacobian looses rank, or equivalently the system looses a degree of freedom and ceases to be controllable. Setpoints that lie at a singular point are not achievable. For the positioning problem, the set of singular configurations is quite simple—the Jacobian is singular when the estimated position of the robot is on the baseline. Any position in the workspace of the robot is attainable provided it does not lie on this line. Likewise, point position estimation is singular when the point lies on the baseline.

The six degree of freedom positioning problem has a more complex set of singularities. These singularities are most easily described using the error term γ defined in (23). Since this is a linear variation on the original error term β, it has the same set of stable points and the same singularity structure.

As described in Section 3.2, there are two consistent assignments of direction to the observation of a line in both cameras. Two of the four possible sign combinations are inconsistent. The choice of the remaining combinations is arbitrary. To see this, suppose the system is at any nonsingular point in the space, observes l_1 and l_2, and computes line direction L_v. Changing the sign of l_1 and l_2 causes line direction, L_v to computed as $L_v' = -L_v$. Since L_v appears in every term of (13), changing the sign of L_v changes the sign of every row in which it appears. Likewise, changing the sign of l_i changes the sign of every error term in which it appears. The two sign changes cancel each other out and hence the two assignments are equivalent. This property also holds for assignment of direction to M.

The fact that line direction is arbitrary also suggests that the system may not have a unique final configuration. In fact, it is guaranteed to have four possible final configurations which are stable and have zero error. Any path between these stable configurations must pass through at least one singular configuration.

Consider a single line $L = (L_v; L_d)$. Line observation is undefined when L_v is parallel to $L_d - c_i$, $i = 1, 2$. In this case, the cross product in (3) yields the zero vector and the division operation is not well-defined. Physically, this corresponds an orientation where L is parallel to the line of sight and cannot be observed. This means that part of the error term becomes undefined any time that L or M become parallel to the line of sight of either camera.

In addition, there are two other important cases to consider:

1. When L_v lies in the epipolar plane defined by L_d, c_1 and c_2, it is easily seen from (3) that rotations about the normal to the plane do not change the line projection, and hence cannot be observed. Note that the poses for which the projection of L is undefined are all positions within an epipolar plane.

2. The error term $l_i \cdot (s_i - p_i)$ reaches its maximum value when the projection of L is perpendicular to the line through s_i and p_i in the image. Consequently, the Jacobian vanishes at that point. For a stable system this singularity never arises since the magnitude of the components of the error term are bounded by their initial value.

From item 1, it follows that line estimation cannot be performed for lines lying in an epipolar plane. This can also be seen from the estimation equation (18). When the line is in an epipolar plane, the observed normals are parallel and the cross product disappears. Likewise, the system is not fully controllable when either L or M move into the epipolar plane. Hence, final configurations that place L or M in an epipolar plane, or configurations in which any one of P, S, or T lie on the baseline are not attainable.

One issue that remains to be resolved is the problem of determining the final configuration of the system based on its starting configuration. Again, due to the nonlinearity of the system, these predictions can be difficult to make. One way of avoiding these problems is to use open-loop positioning to first place the manipulator in an approximately correct orientation and position, and then to use feedback to correct for small positioning errors. Appendix B describes how to perform a Cartesian motion to get the system near its goal configuration from visual inputs.

5 Experiments

Our visual servoing system consists of a Zebra Zero robot arm with PC controller, a Zebra pan-tilt head, a Directed Perceptions pan-tilt head, a Sony XC-77 camera with 12.5 mm lens, a Cohu camera with an 8 mm lens, and two Imaging Technologies digitizers attached to a Sun Sparc II computer via a Solflower SBus-VME adapter. The workstation and PC are connected by an ethernet link. The cameras are placed 80 to 100 centimeters from the robot along the x axis, 20 to 30 centimeters apart along the y axis, and are oriented to point back along the x axis of the robot. During the trials described below, both pan-tilt heads are inactive. Although not often employed, system calibration can be performed by tracking the manipulator as it moves to a series of positions, and applying a least-squares minimization to generate the calibration parameters. This is an automatic procedure that takes 10 to 15 seconds.

All image processing and visual control calculations are performed on the Sun workstation. Robot control is implemented using both static and dynamic servo systems. In the dynamic case, Cartesian velocities are continually sent to the PC which converts them into coordinated joint motions using a resolved-rate controller. There is no return communication from the PC to the Sun. However, as the system runs, it logs 5 minutes of joint motion information at 20 Hz which can be used to examine the dynamic behavior of the system. In the static case, the robot is commanded to move from point to point and each motion is allowed to finish before the next is sent. In this case, visual control is also a resolved-rate system in which the control velocities are multipled by a user-controlled "time granularity" parameter to construct positional offsets. In all trials below the time granularity was chosen to be 0.3 seconds. Before each motion, the robot position is stored for later analysis.

A custom tracking system written in C++ provides visual input for the controller. The system, more fully described in [11], provides extremely fast feature detection on a memory-mapped framebuffer. In addition, it supports simultaneous tracking of multiple segments, and can also enforce constraints among segments. The experiments described here are based on tracking corners formed by the intersection of two line segments. The segment length was set to be 20 pixels, and the search area around a segment is ± 10 pixels. Specifics of the tracking setup for each application are described below. Details on the conversion from pixel coordinates to the metric inputs to the control and estimation algorithms are briefly described in Appendix A.

Stereo images from the experimental setup are shown in Figure 7. The top set of images shows the system in a goal configuration where it is attempting to touch the corners of two 3.5 inch floppy disks. The disks are used as a convenient testing tool since their narrow width (approximately 2.5 mm) makes them easy to track and at the same time makes it simple to observe positioning and orientation precision. It is important to note that no specific geometric information about the disks is used other than the fact that the three corners are not colinear.

5.1 Simulation

A simulation system was constructed to test the stability and noise sensitivity of the system. In all cases, the noise is Gaussian with a variance of $(.005\text{mm})^2$. This corresponds to a unit variance of about one pixel in a CCD camera. The focal length of the lenses is taken to be 8.0mm. The camera calibration assumes the cameras are aligned to be parallel, perpendicular to the baseline, 30cm apart, about 80 centimeters from the target. The differential equations are applied as difference equations with a time granularity of 0.1 seconds. All other control and estimation constants are set to 1.0. Calibration errors are introduced by changing the position of the cameras along the baseline, and rotating the cameras about their y axes as was discussed in the analytical section above.

The following table shows the root-mean-square (RMS) positioning error for 3-D position over 1000 time steps for a variety of different camera positioning errors relative to the given calibration:

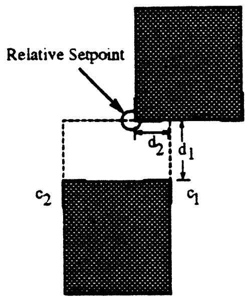
Offset		RMS Position Error		
Baseline	Rotation	(0, 0, 800)	(100, 0, 800)	(0, 100, 800)
0	0	0.628	0.695	0.555
10	0	0.514	0.407	0.516
-10	0	0.7	0.779	0.709
0	5	1.337	1.265	1.17
0	-5	0.337	0.365	0.258

As expected, moving the cameras toward each or rotating them toward each other increases the effective closed-loop gain and hence increases noise sensitivity. The oppose motions have the opposite effect. Note that the maximum RMS error is extremely small in any case—about 1.4 millimeters in the worst case.

The accuracy of six DOF control was also simulated for the three configurations of lines and points. In all cases, the lines were intersecting with the parameters given below:

Label	L_d and M_d	L_v	M_v	P	S	T
A	(0, 0, 800)	(1, 1, 0)	(-1, 1, 0)	(0, 0, 800)	(100, 100, 800)	(-100, 100, 800)
B	(0, 0, 800)	(1, 1, 0)	(0, 1, 0)	(0, 0, 800)	(100, 100, 800)	(0, 100, 800)
C	(0, 0, 800)	(1, 1, 1)	(-1, 1, -1)	(0, 0, 800)	(100, 100, 900)	(-100, 100, 700)

The geometry of these configurations has been arranged so that the lines will be positioned so that the intersection point is placed at P. Consequently, positioning error was measured as the distance from the intersection of the two lines to P. Rotation error was measured as the angle between L and the line through P and S and the angle between M and the line through P and T. The following table shows the RMS positioning error for 1000 time steps for the same camera configurations as above:



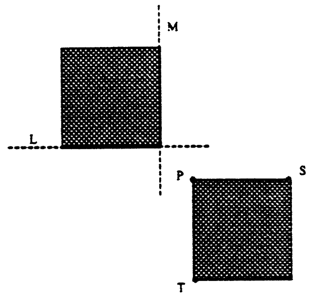


Figure 6. Left, a diagram showing how the positioning visual trajectory was defined, and right, a diagram showing how positioning and alignment setpoint was defined.

Offset		RMS Position Error			RMS Angle Error		
Baseline	Rotation	A	B	C	A	B	C
0	0	1.135	1.629	1.823	0.073, 0.847	0.759, 0.815	0.555, 0.536
10	0	0.931	1.123	1.335	0.804, 0.782	0.906, 0.816	0.504, 0.551
-10	0	1.571	2.009	2.202	0.821, 0.889	0.763, 0.893	0.515, 0.558
0	5	5.604	7.918	5.149	1.189, 1.187	1.249, 1.216	0.654, 0.604
0	-5	0.590	0.687	0.819	0.573, 0.511	0.477, 0.659	0.472, 0.485

Note that the positioning error is larger than the previous trial by about a factor of two. In all trials, the qualitative stability results of the previous section are apparent. For trials where the cameras farther apart than given in the calibration, the system response is damped. Likewise, pointing the cameras away from each other tends to damp the system.

5.2 Real System — Positioning

To test positioning accuracy and repeatability, the robot is guided along a square trajectory defined by the sides and top of a target disk. The robot manipulator begins by moving the left corner of its disk to touch the right corner of the target disk. Next, it moves to a position where the left corner of its disk is 1/2 disk length (approximately 1.75 inches) above the right corner of the target disk. It then moves the right corner of its disk to the corresponding stationing point above the left corner of the target disk. Finally, it descends to touch the right corner of its disk to the left corner of the target disk. The complete procedure is then reversed, and the entire cycle executed repeatedly.

This trajectory is described visually as follows. A tracker for two corners of a floppy disk is defined. The tracker has a state variable that depends on two constants, d_1 and d_2 as well as the positions of the two corners which are denoted c_1 and c_2. Let p denote a vector of length $\|c_2 - c_1\|$ perpendicular to the line joining c_1 and c_2—that is, $p \cdot (c_2 - c_1) = 0$. Then position of the setpoint, s, is given by

$$s = c_1 + d_2(c_2 - c_1) + d_1 p.$$

(see Figure 6). A pair of these trackers are used to track the bottom of the floppy disk held by the robot in each image. This provides observations of robot visual position. A second pair tracks the

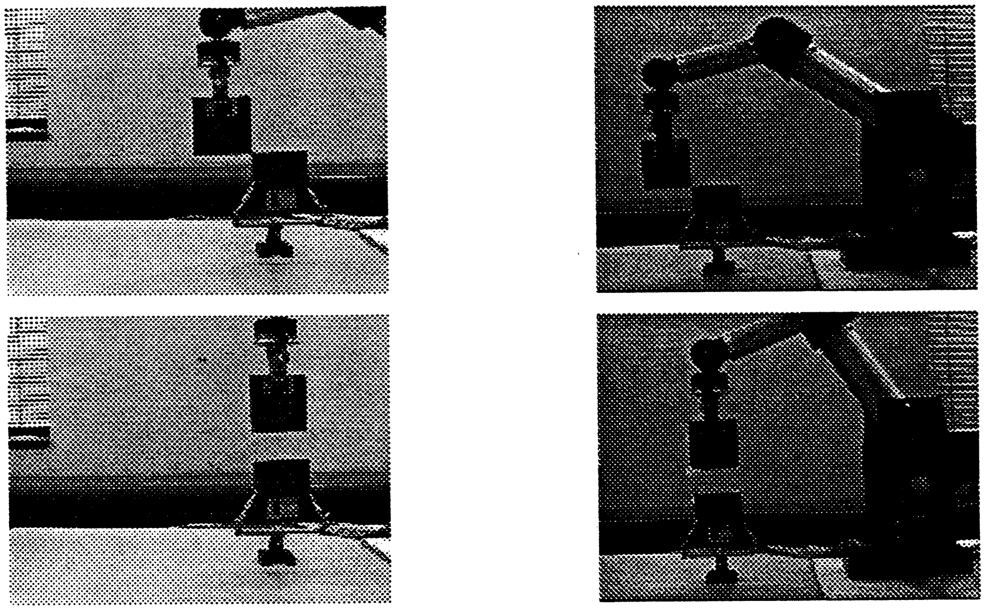


Figure 7. Example stereo images from two stages of a feature-based trajectory. The top shows the system touching the corners of two floppy disks. The bottom shows the visual cues used to define a position directly above a corner.

target floppy in the stand, providing a setpoint. Thus, this trial involves tracking a total of sixteen primitive features (four corners composed of two contours in two images). Executing the test trajectory requires changing the setpoint values of both the robot disk tracker and the target disk tracker though eight different motion segments. The velocity along the trajectory is constant—no attempt is made to incorporate acceleration constraints.

The entire visual control system (including tracking and control signal computation) runs at a rate of up to 20 Hz, however due to the computational and architectural limitations of the PC-based control system, velocities are only sent to the PC at a maximum rate of 10 Hz. At 20 Hz, the robot can be tracked at velocities covering up to 200 pixels/sec. which, with a 12.5 mm lens, converts to maximum velocities of approximately 12 cm/sec. perpendicular to the camera optical axis at 1 meter. For these trials robot velocities were limited to a maximum of 5 cm/sec. The total time lag in the system (from images to robot motion) is estimated as follows: the maximum frame lag (1/30 sec.) plus processing time (1/20 sec.) plus send time (estimated send time 1/100 sec.) plus maximum wait on the PC side (1/20 sec.) yielding a maximum of 0.14 sec. worst case delay time. The best case is 0.06 seconds. This suggests that one time step delay model (a delay of 0.1 sec.) is a reasonable model for the system.

The gain values k_1 and k_2 are set as described below. The system is insensitive to values the choice of estimation coefficients. In all trials these coefficients are set to 1.0, yielding a deadbeat estimation system.

The expected positioning accuracy of the system depends on the error in edge localization. One camera pixel has a width of approximately 0.01mm. At 80 cm. with 12.5 mm focal length lenses on both cameras, the expected vertical and horizontal positioning accuracy is ±0.32 mm, and the expected accuracy in depth is ±1.75 mm. Consequently, the system should be able to reliably position the corners of the disks so that they nearly touch one another.

Several trials were performed under varying conditions and with various control gains. In nearly all cases, the system was able to position the disks so that the corners touched. Occasionally the system failed due to systematic aliasing problems in the edge tracking system. These aliasing problems arise due to "blooming" effects in the CCD cameras. These only appear when the contrast across an edge becomes excessive. The following details the experimental results:

Stability: The major destabilizing factors in the system are time lag, discretization effects, unmodeled robot dynamics and calibration error. Unmodeled dynamics of the robot and calibration error do not appear to play a major role. When $k_2 = 0$, empirical tests have shown that $k_1 < 2$ leads to critically-damped response at endpoint velocities of 5.0 cm/sec. Higher values lead to oscillations. This is approximately in accord with the discrete time model which would predict underdamped response at $k_1 = 2.5$.

Setting $k_2 < 0.05$ maintains acceptable performance with less than 2mm overshoot at each setpoint. Higher values lead to oscillation and large overshoot.

Accuracy: As noted above, if the edge detection is accurate to one pixel, we expect the manipulator to position the disks so that they overlap. Within the operating range described above we have found this to always be the case. While at the station, the manipulator continues to perform small corrective motions of about 1mm due to noise in the edge tracker. However, in spite of these motions the manipulator maintains it's position within the expected 2.5mm error interval. Measurements indicate a maximum error of about $\pm$1mm. Figure 8 shows the robot motions while at a setpoint position. Note that the maximum range of error is less and 1mm in all coordinate directions.

Tracking Trajectories Because the target disk is nearly parallel to both camera imaging planes, the stereo trajectory defined above should cause the robot to move nearly in the plane of the target disk from side to side. Figure 9 shows the manipulator motion when executing one cycle of the corner-to-corner trajectory. The slight deviation in front of and behind the target disk occurs due to the distance between the setpoint delivered from tracking and the position of the robot. Increasing the integral gain improves tracking performance slightly, but causes overshoot at both endpoints (Figure 9). This is not unexpected since the trajectory as parameterized would require infinite acceleration.

Calibration Sensitivity: In practice, the system insensitive to calibration as expected. During operation it is possible to reorient the cameras and/or move them about without seriously effecting the performance of the system. The system is often run without prior calibration. The most noticeable effect is that positioning errors due to tracking lag are magnified. Naturally, if the cameras are moved sufficiently far, the system becomes unstable.

5.3 Real System—Position and Orientation

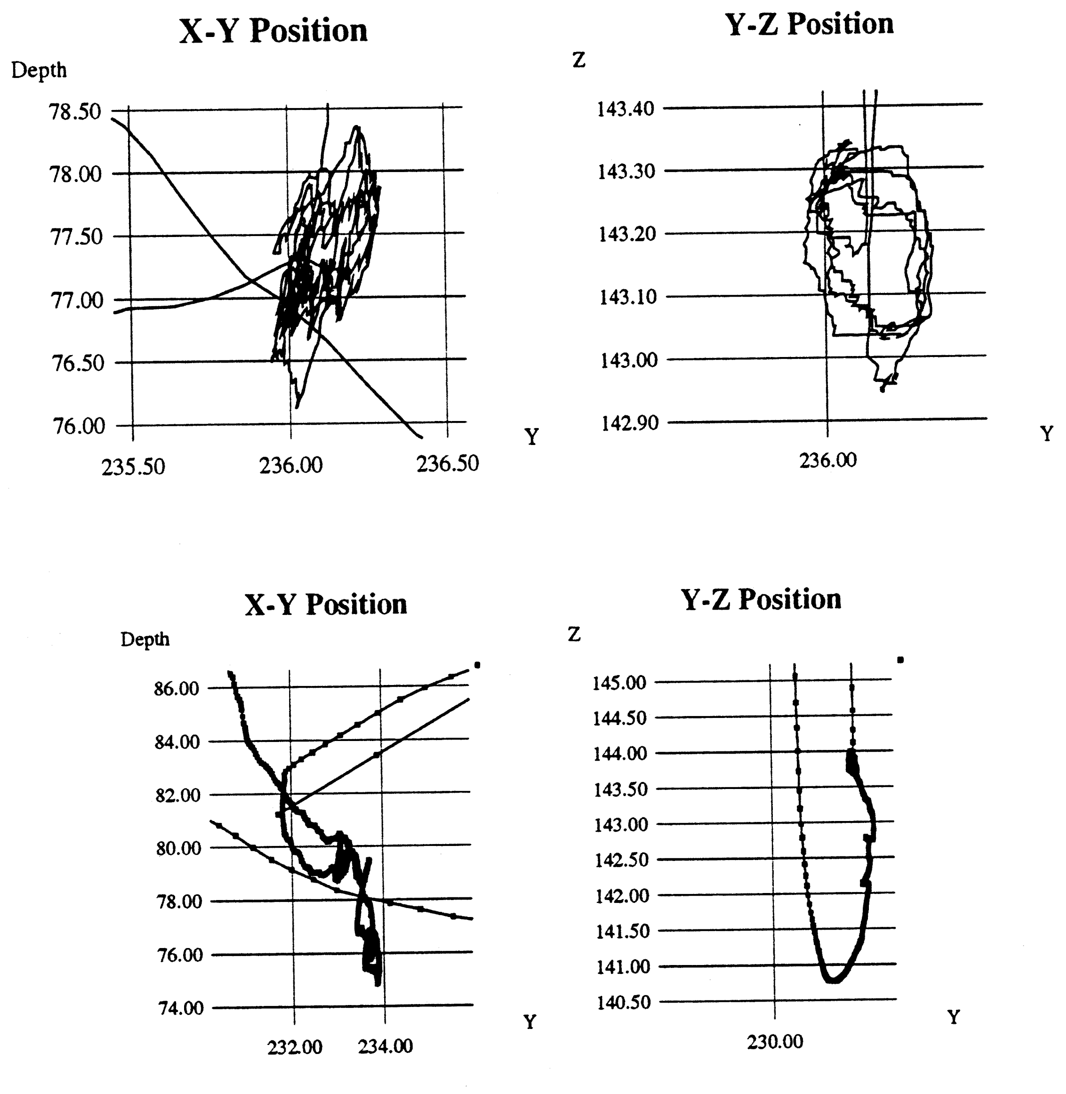


Figure 8. Above the x/y and y/z projections at stable point with unit feedback and no integrator. Below the same test with integral gain of 0.1.

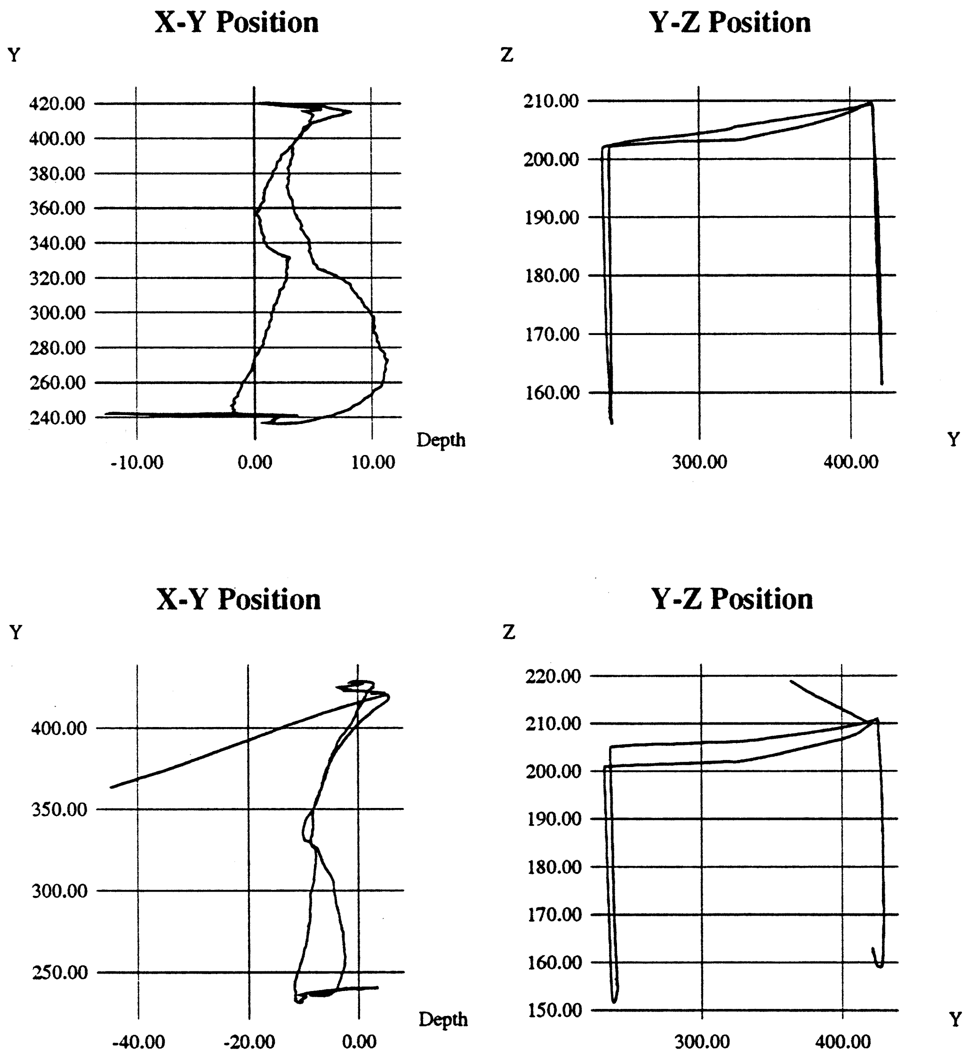


Figure 9. Above the x/y and y/z projections of path tracking with no integration. Below the same test with integral gain of 0.1.

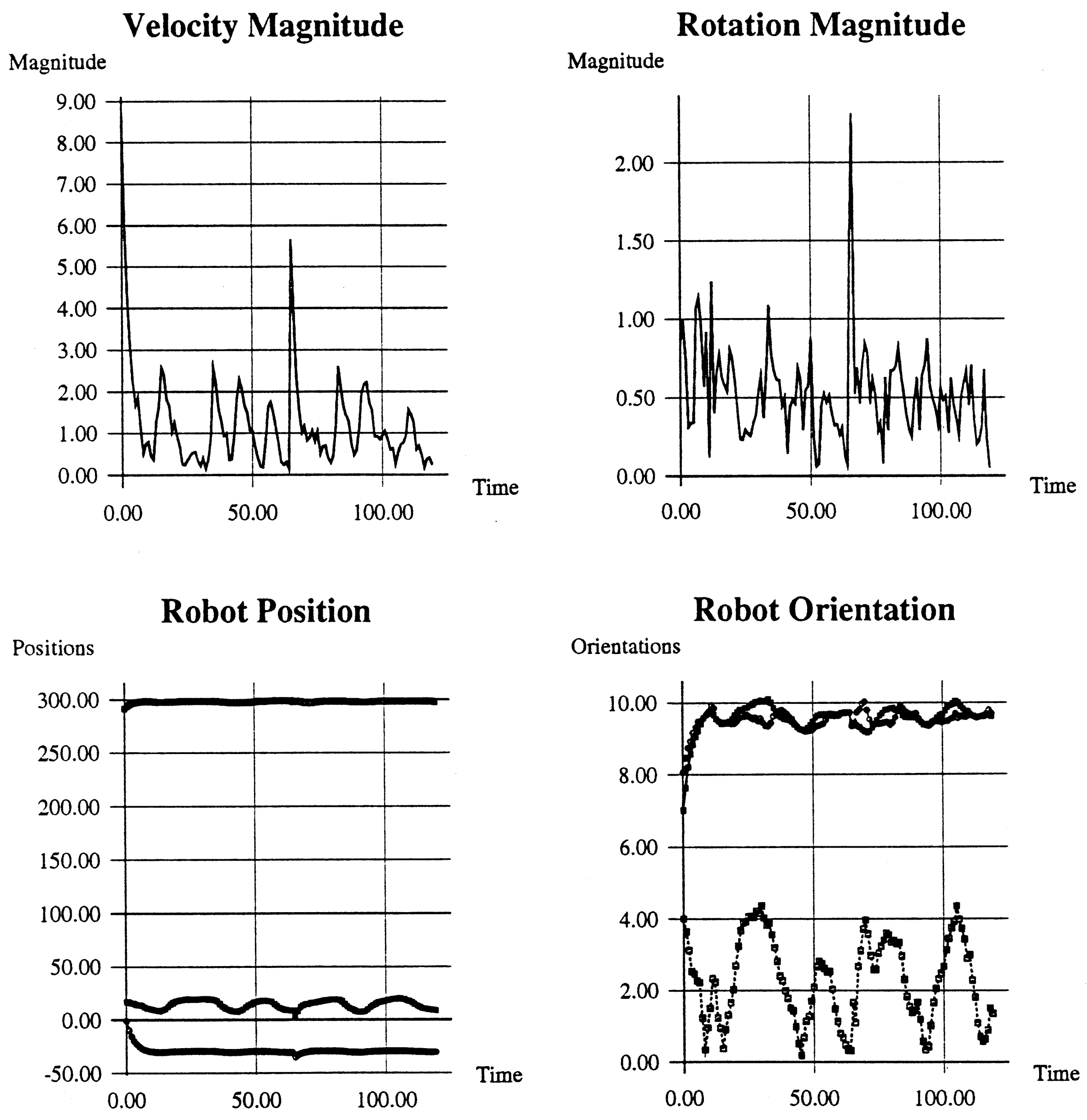
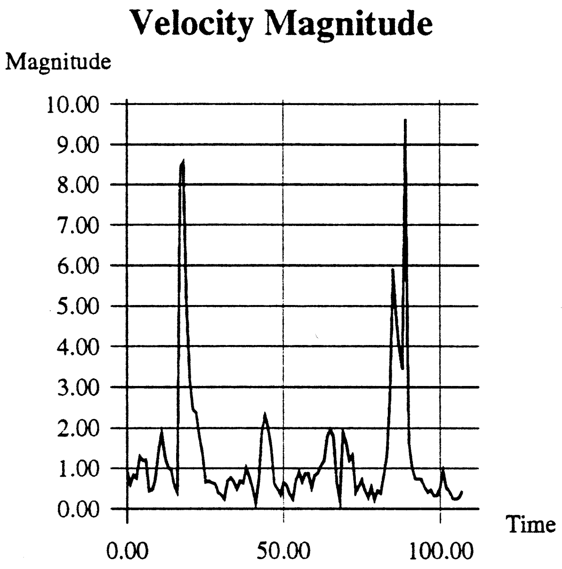
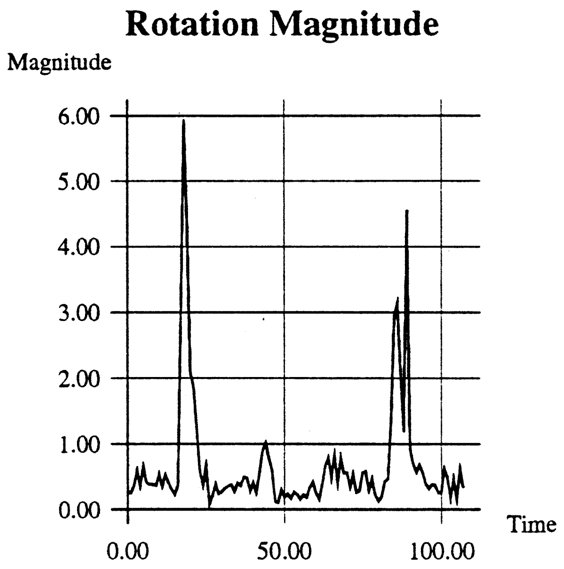
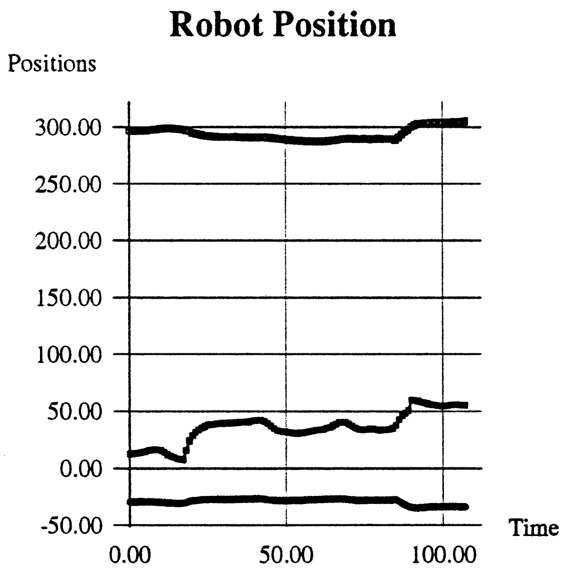


Figure 10. Velocity magnitudes (upper) and the Cartesian positions of the robot (lower) for trial one.







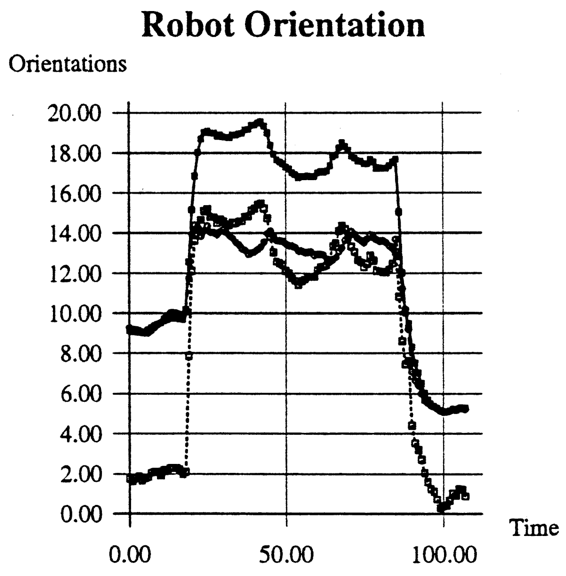


Figure 11. Velocity magnitudes (upper) and the Cartesian positions of the robot (lower) for trial two.

Position and orientation tests were conducted on the same experimental apparatus using only the static motion system. The visual setpoint was again defined using corners as described in Figure 6. The system runs at approximately 10 Hz including all image processing and control calculations. Because of the static nature of the motion system, no attempt was made to execute trajectories.

Figure 10 shows the steady-state system error vector and time sequence of robot positions while maintaining a single position. In general, the steady state corrections are small: within one degree of rotation and two millimeters about a nominal position. The robot encoders are attached to the motors, so due to backlash and compliance in the mechanism, rotational information includes some hysteresis effects. This results in the spike near the 60th sample. The manipulator "stuck" for a few time steps, and suddenly moved a relatively large amount causing a large correction to be applied. This is also the reason for the periodicity exhibited in the data—several time steps of motion must be applied before the mechanism actually moves. Consequently, the motions of the actual mechanism are smaller than are indicated by the data. Also, due to calibration errors in the tool frame center of rotation, rotations always induce a slight translation. These coupling effects can be seen in the graphs. Note that observation error significantly affects the y component of the system orientation. This is to be expected since the y component is most heavily dependent on stereo information.

Figure 11 shows the same data for two point-to-point motions. The behavior at each station is roughly similar. It takes between 6 and 10 time steps to move between the stations. This corresponds to 2 to 3 seconds of time for this point to point motion.

6 Discussion

We believe the visual servoing paradigm we have presented has great potential for use in commercial and scientific applications. It is simple, inexpensive, robust, portable and task-independent. The current vision processing and control computation system uses no special hardware (other than a standard framegrabber) and could be run on off-the-shelf PC's. The total control system, exclusive of the robot and pan-tilt heads could be constructed on hardware costing no more than about $3000. Furthermore, since the entire system, including image processing, runs in software, moving to a newer or more powerful system is largely a matter of recompiling. At the current rate of progress, frame-rate (60 Hz) servoing will be easily feasible in a year or two. The system is extremely accurate. As reported, the current system can easily position the end-effector to within a few millimeters relative to a target. This positioning accuracy could easily be improved by changing the camera configuration to a wider baseline, improving the image-processing to be more accurate, or increasing the focal length of the cameras.

Because the methods are insensitive to calibration errors, the methods are extremely robust and quick to reconfigure with a new camera configuration. In addition, two different methods of online calibration for positioning are currently being tested. One method uses a switching controller [21] to recover the estimate of the system Jacobian. Initial simulation tests indicate that it is stable, and rapidly calibrates the system as it moves. The second assumes an orthographic projection calibrated offline, and performs online adaptation of scale. This method has the advantage of allowing linear control techniques to be used. Such adaptive systems will make it simple to use mobile or reconfigurable camera systems.

Work is proceeding on occlusion detection and compensation. In particular, the design of motion strategies that plan an occlusion-free path offline or online are of interest. Offline vision planning using visibility models and a prior world model information is already being investigated [19]. Online motion compensation based on occlusion detection does not appear to have been considered to date.

Work is also proceeding on developing a framework for task representation and planning in the visual space. In recent work [10], it was noted that projective invariants [22] provide a basis for specifying robot positions and motion independent of geometric reconstructions, and consequently independent of camera calibration. Development of these concepts is currently underway, including both the visual tracking methods needed to compute projective invariants, and the design and implementation of vision-based motion strategies that employ invariants.

Acknowledgements The work in this paper could not have been done without the help of Sidd Puri on implementing an initial version of the tracking system, A.S. Morse for suggesting the design for the positioning feedback controller, and C.J. Taylor for suggesting the line representation used throughout the article. This research was supported by DARPA grants N00014-91-J-1577 and N00014-93-1-1235, and by National Science Foundation grants IRI-9109116 and DDM-9112458.

A System Input Conversion

Input to the estimation and control algorithms is computed from image features as follows. For points, the location of the corner in pixel coordinates is converted to a metric value in millimeters and divided by the camera focal length to arrive at a unit focal length equivalent value. For lines, the direction of the project contour, call it θ, is first extracted. At this point, there are two possible normal directions; the direction $\theta + \pi/2$ is chosen and the components $n = (n_x, n_y) = (\cos(\theta + \pi/2), \sin(\theta + \pi/2))$ are computed. Given a visual reference point with image coordinates p on the contour (converted to unit focal length metric values), the line observation vector is $l = (n_x, n_y, n \cdot p)$.

B Computing Final System Configurations

The section discusses the problem of positioning a manipulator using open-loop motions based on the estimated positions of visual setpoints. To simplify the exposition, it is assumed that the origin of the manipulator tool frame is at L_d.

The computation of setpoint position from visual inputs is relatively simple when the lines L and M intersect one another. The orientation of the coordinate system defined by L and M is given by the matrix:

$$R^T_{LM} = \left[L_v \mid \overline{L_v \times M_v} \mid \overline{L_v \times (L_v \times M_v)}\right] \tag{33}$$

where the bar indicates normalization to a unit vector. The equivalent system for the stationing point is:

$$R^T_{PST} = \left[\overline{S-P} \mid \overline{(S-P)\times(T-P)} \mid \overline{(S-P)\times((S-P)\times(T-P))}\right] \tag{34}$$

Note that there are four possible orientations depending on the choice of sign of the quantities $(S-P)$ and $(T-P)$. The rotation to align the two coordinate systems is $R = R^T_{LM}R_{PST}$.

This is combined with the translation $P - L_d$ to bring the point P onto the line L. After these motions, the coordinates of L and M are $L' = (P, RL_v)$ $M' = (R(M_d - L_d) + P, RM_v)$, respectively. The translation, d, along L' that places T on M' is computed as follows:

$$\begin{aligned} n &= (T - M'_d) - M'_v(T - M'_d)\cdot M'_v && (35)\\ d &= L'_v(n\cdot n/n\cdot L'_v). && (36)\end{aligned}$$

If L and M do not intersect, the process is more complicated. First, P and R are placed on L by applying the following rotation:

$$\begin{aligned} R &= rot(k,\theta) && (37)\\ k &= \overline{L_v \times (S-P)} && (38)\\ \theta &= \tan^{-1}(\|L_v\times(S-P)\|/(L_v\cdot(S-P))). && (39)\end{aligned}$$

Note that there are two possibilities for the final system configuration. Following this by a translation of $P - L_d$ again places L_d at P. Let the result of applying these two transformations to L and M be denoted L' and M' as above.

The next stage is to look for a rotation R' about the line L' so that $(R'M'_v \times L'_v)\cdot(R'(M'_d - L'_d) + L_d - S) = 0$. Let R_z be the rotation matrix that aligns L' with the z axis. This rotation can be computed using (37) through (39), replacing the vector $(R-P)$ with $z = (0,0,1)^T$. Define $mv = R_zM'_v$, $lv = R_zL'_v$, $md = R_zM'_d$, $ld = R_zL'_d$ and $s = R_zS$. Defining $R' = rot(z,\theta)$, leads to a quadratic equation which has the following solution:

$$\begin{aligned} a &= mv_2(ld_1 + s_1) - mv_1(s_2 + ld_2) && (40)\\ b &= mv_1(ld_1 + s_1) + mv_2(ld_2 + s_2) && (41)\\ c &= (mv_2(md_1 - ld_1) + mv_1(ld_2 - md_2) && (42)\\ x &= \frac{ac \pm b\sqrt{a^2+b^2-c^2}}{a^2+b^2} && (43)\\ \theta &= \arccos(x). && (44)\end{aligned}$$

Note there are again two choices for θ. Applying the rotation $R' = rot(L'_v,\theta)$ to M' yields the line M'' and finally applying (35) and (36) produces the translation to place T on M''. The composition of these four motions places the robot at the desired stationing point.

References

[1] P. Allen, B. Yoshimi, and A. Timcenko. Real-time visual servoing. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 851–856. 1991.

[2] R. L. Anderson. Dynamic sensing in a ping-pong playing robot. *IEEE Journal of Robotics and Automation*, 5(6):723–739, 1989.

[3] W. Chen, U. Korde, and S. Skaar. Position control experiments using vision. *Int. J. of Robot Res.*, 13(3):199–208, June 1994.

[4] P. I. Corke. Visual control of robot manipulators—a review. In K. Hashimoto, editor, *Visual Servoing*, pages 1–32. World Scientific, 1994.

[5] J. Feddema, C. Lee, and O. Mitchell. Weighted selection of image features for resolved rate visual feedback control. *IEEE Trans. on Robotics and Automation*, 7(1):31–47, February 1991.

[6] G. Franklin, J. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems.* Addison-Wesley, 2nd edition, 1991.

[7] A. Gelb, editor. *Applied Optimal Estimation.* MIT Press, Cambridge, MA, 1974.

[8] G. Hager. Some problems in adaptive visual servoing. DCS RR-948, Yale University, New Haven, CT, January 1993.

[9] G. Hager and S. Hutchinson, editors. *Proceedings of the Workshop on Visual Servoing.* IEEE, May 1994.

[10] G. D. Hager. Real-time feature tracking and projective invariance as a basis for hand-eye coordination. DCS RR-993, Yale University, New Haven, CT, November 1993. To be presented at the 1994 Int. Conf. on Computer Vision and Pattern Recognition.

[11] G. D. Hager, S. Puri, and K. Toyama. A framework for real-time vision-based tracking using off-the-shelf hardware. DCS RR-988, Yale University, New Haven, CT, September 1993.

[12] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision: Volume II.* Addison Wesley, 1993.

[13] K. Hashimoto, editor. *Visual Servoing.* World Scientific, 1994.

[14] K. Hashimoto, T. Kimoto, T. Ebine, and H. Kimura. Manipulator control with image-based visual servo. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 2267–2272. 1991.

[15] G. Hirzinger, G. Grunwald, B. Brunner, and H. Heindl. A sensor-based telerobotic system for the space robot experiment ROTEX. *2. Int. Symposium on Experimental Robotics*, 1991. Toulouse, France.

[16] N. Hollinghurst and R. Cipolla. Uncalibrated stereo hand eye coordination. Technical Report TR-126, Cambridge University, Dept. of Engineering, September 1993.

[17] B. K. Horn. *Robot Vision.* MIT Press, Cambridge, 1986.

[18] S. Hutchinson. Exploiting visual constraints in robot motion planning. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1722–1727. 1991.

[19] S. Hutchinson. Exploiting visual constraints in robot motion planning. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1722–1727. 1991.

[20] S. T. Jiang Yu Zheng, Qian Chen. Active camera guided manipulation. In *Proc IEEE Int. Conf on Robotics and Automation*, pages 632–638. Sacramento, 1991.

[21] A. Morse. Supervisory control of families of linear set-point controllers. To Appear.

[22] J. Mundy and A. Zisserman. *Geometric Invariance in Computer Vision.* MIT Press, Cambridge, Mass., 1992.

[23] B. Nelson and P. K. Khosla. Increasing the tracking region of an eye-in-hand system by singularity and joint limit avoidance. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 418–423. 1993.

[24] N. Papanikolopoulos, P. Khosla, and T. Kanade. Visual tracking of a moving target by a camera mounted on a robot: A combination of control and vision. *IEEE Trans. on Robotics and Automation*, 9(1), 1993.

[25] P. Rives, F. Chaumette, and B. Espiau. Positioning of a robot with respect to an object, tracking it and estimating its velocity by visual servoing. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 2248–2254. 1991.

[26] A. Rizzi and D. E. Koditschek. Further progress in robot juggling: The spatial two-juggle. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 919–924. 1993.

[27] L. Weiss, A. Sanderson, and C. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE J. of Robotics and Automation*, RA-3(5):404–417, Oct. 1987.

[28] D. Whitney. The mathematics of coordinated control of prosthetic arms and manipulators. *J. Dynamic Syst., Measurement, and Control*, 122:303–309, Dec. 1972.

[29] S. Wijesoma, D. Wolfe, and R. Richards. Eye-to-hand coordination for vision-guided robot control applications. *Int. J. of Robot Res.*, 12(1):65–78, 1993.