A Tutorial on Visual Servo Control

Seth Hutchinson, *Member, IEEE,* Gregory D. Hager, *Member, IEEE,* and Peter I. Corke, *Member, IEEE*

Abstract—**This article provides a tutorial introduction to visual servo control of robotic manipulators. Since the topic spans many disciplines our goal is limited to providing a basic conceptual framework. We begin by reviewing the prerequisite topics from robotics and computer vision, including a brief review of coordinate transformations, velocity representation, and a description of the geometric aspects of the image formation process. We then present a taxonomy of visual servo control systems. The two major classes of systems, position-based and image-based systems, are then discussed in detail. Since any visual servo system must be capable of tracking image features in a sequence of images, we also include an overview of feature-based and correlation-based methods for tracking. We conclude the tutorial with a number of observations on the current directions of the research field of visual servo control.**

I. Introduction

THE VAST majority of today's growing robot population operate in factories where the environment can be contrived to suit the robot. Robots have had far less impact in applications where the work environment and object placement cannot be accurately controlled. This limitation is largely due to the inherent lack of sensory capability in contemporary commercial robot systems. It has long been recognized that sensor integration is fundamental to increasing the versatility and application domain of robots, but to date this has not proven cost effective for the bulk of robotic applications, which are in manufacturing. The "frontier" of robotics, which is operation in the everyday world, provides new impetus for this research. Unlike the manufacturing application, it will not be cost effective to re-engineer "our world" to suit the robot.

Vision is a useful robotic sensor since it mimics the human sense of vision and allows for noncontact measurement of the environment. Since the early work of Shirai and Inoue [1] (who describe how a visual feedback loop can be used to correct the position of a robot to increase task accuracy), considerable effort has been devoted to the visual control of robot manipulators. Robot controllers with fully integrated vision systems are now available from a number of vendors. Typically visual sensing and manipulation are combined in an open-loop fashion, "looking" then "moving". The accuracy of the resulting operation depends directly on the accuracy of the visual sensor and the robot end-effector.

An alternative to increasing the accuracy of these subsystems is to use a visual-feedback control loop that will increase the overall accuracy of the system—a principal concern in most applications. Taken to the extreme, machine vision can provide closed-loop position control for a robot end-effector—this is referred to as *visual servoing*. This term appears to have been first introduced by Hill and Park [2] in 1979 to distinguish their approach from earlier "blocks world" experiments where the system alternated between picture taking and moving. Prior to the introduction of this term, the less specific term *visual feedback* was generally used. For the purposes of this article, the task in visual servoing is to use visual information to control the *pose* of the robot's end-effector relative to a target object or a set of target features. The task can also be defined for mobile robots, where it becomes the control of the vehicle's pose with respect to some landmarks.

Since the first visual servoing systems were reported in the early 1980s, progress in visual control of robots has been fairly slow, but the last few years have seen a marked increase in published research. This has been fueled by personal computing power crossing the threshold that allows analysis of scenes at a sufficient rate to "servo" a robot manipulator. Prior to this, researchers required specialized and expensive pipelined pixel processing hardware. Applications that have been proposed or prototyped span manufacturing (grasping objects on conveyor belts and part mating), teleoperation, missile tracking cameras, and fruit picking, as well as robotic ping-pong, juggling, balancing, car steering, and even aircraft landing. A comprehensive review of the literature in this field, as well the history and applications reported to date, is given by Corke [3] and includes a large bibliography.

Visual servoing is the fusion of results from many elemental areas including high-speed image processing, kinematics, dynamics, control theory, and real-time computing. It has much in common with research into *active vision* and *structure from motion*, but is quite different from the often described use of vision in hierarchical task-level robot control systems. Many of the control and vision problems are similar to those encountered by active vision researchers who are building "robotic heads". However the task in visual servoing is to *control* a robot to manipulate its environment using vision as opposed to just *observing* the environment.

Given the current interest in visual servoing it seems both appropriate and timely to provide a tutorial introduction to this topic. Our aim is to assist others in creating visually servoed systems by providing a consistent terminology and nomenclature, and an appreciation of possible applications.

Manuscript received March 24, 1995; revised January 19, 1996. G. D. Hager was supported by ARPA grant N00014-93-1-1235, Army DURIP grant DAAH04-95-1-0058, National Science Foundation grant IRI-9420982, and by funds provided by Yale University. This paper was recommended for publication by Associate Editor J. Funda and Editor S. E. Salcudean upon evaluation of reviewers' comments.

S. Hutchinson is with the Department of Electrical and Computer Engineering, The Beckman Institute for Advanced Science and Technology, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA.

G. D. Hager is with the Department of Computer Science, Yale University, New Haven, CT 06520-8285 USA.

P. I. Corke is with the CSIRO Division of Manufacturing Technology, Kenmore, Australia, 4069.

Publisher Item Identifier S 1042-296X(96)07366-1.

To assist newcomers to the field we will describe techniques which require only simple vision hardware (just a digitizer), freely available vision software [4], and which make few assumptions about the robot and its control system. This is sufficient to commence investigation of many applications where high control and/or vision performance are not required.

One of the difficulties in writing such an article is that the topic spans many disciplines that cannot be adequately addressed in a single article. For example, the underlying control problem is fundamentally nonlinear, and visual recognition, tracking, and reconstruction are fields unto themselves. Therefore we have concentrated on certain basic aspects of each discipline, and have provided an extensive bibliography to assist the reader who seeks greater detail than can be provided here. Our preference is always to present those ideas and techniques that we have found to function well in practice and that have some generic applicability. Another difficulty is the current rapid growth in the vision-based motion control literature, which contains solutions and promising approaches to many of the theoretical and technical problems involved. Again we have presented what we consider to be the most fundamental concepts, and again refer the reader to the bibliography.

The remainder of this article is structured as follows. Section II reviews the relevant fundamentals of coordinate transformations, pose representation, and image formation. In Section III, we present a taxonomy of visual servo control systems (adapted from [5]). The two major classes of systems, position-based visual servo systems and image-based visual servo systems, are then discussed in Sections IV and V respectively. Since any visual servo system must be capable of tracking image features in a sequence of images, Section VI describes some approaches to visual tracking that have found wide applicability and can be implemented using a minimum of special-purpose hardware. Finally, Section VII presents a number of observations regarding the current directions of the research field of visual servo control.

II. Background and Definitions

In this section we provide a very brief overview of some topics from robotics and computer vision that are relevant to visual servo control. We begin by defining the terminology and notation required to represent coordinate transformations and the velocity of a rigid object moving through the workspace (Sections II-A and II-B). Following this, we briefly discuss several issues related to image formation (Sections II-C and II-D), and possible camera/robot configurations (Section II-E). The reader who is familiar with these topics may wish to proceed directly to Section III.

A. Coordinate Transformations

In this paper, the task space of the robot, represented by $\mathcal{T}$, is the set of positions and orientations that the robot *tool* can attain. Since the task space is merely the configuration space of the robot tool, the task space is a smooth m-manifold (see, e.g., [6]). If the tool is a single rigid body moving arbitrarily in a three-dimensional workspace, then $\mathcal{T} = \mathrm{SE}^3 = \Re^3 \times \mathrm{SO}^3$, and $m = 6$. In some applications, the task space may be restricted to a subspace of SE^3. For example, for pick and place, we may consider pure translations ($\mathcal{T} = \Re^3$, for which $m = 3$), while for tracking an object and keeping it in view we might consider only rotations ($\mathcal{T} = \mathrm{SO}^3$, for which $m = 3$).

Typically, robotic tasks are specified with respect to one or more coordinate frames. For example, a camera may supply information about the location of an object with respect to a camera frame, while the configuration used to grasp the object may be specified with respect to a coordinate frame attached to the object. We represent the coordinates of point $\boldsymbol{P}$ with respect to coordinate frame x by the notation ${}^x\boldsymbol{P}$. Given two frames, x and y, the rotation matrix that represents the orientation of frame y with respect to frame x is denoted by ${}^x\boldsymbol{R}_y$. The location of the origin of frame y with respect to frame x is denoted by the vector ${}^x\boldsymbol{t}_y$. Together, the position and orientation of a frame specify a *pose*, which we denote by ${}^x\boldsymbol{x}_y$. If the leading superscript, x, is not specified, the world coordinate frame is assumed.

We may also use a pose to specify a *coordinate transformation*. We use function application to denote applying a change of coordinates to a point. In particular, if we are given ${}^y\boldsymbol{P}$ (the coordinates of point $\boldsymbol{P}$ relative to frame y), and ${}^x\boldsymbol{x}_y$, we obtain the coordinates of $\boldsymbol{P}$ with respect to frame x by applying the coordinate transformation rule

$$
\begin{aligned}
{}^x\boldsymbol{P} &= {}^x\boldsymbol{x}_y({}^y\boldsymbol{P}) && (1)\\
&= {}^x\boldsymbol{R}_y\,{}^y\boldsymbol{P} + {}^x\boldsymbol{t}_y. && (2)
\end{aligned}
$$

In the sequel, we will use the notation ${}^x\boldsymbol{x}_y$ to refer either to a coordinate transformation, or to a pose that is specified by a rotation matrix and translation, ${}^x\boldsymbol{R}_y$ and ${}^x\boldsymbol{t}_y$, respectively. Likewise, we will use the terms *pose* and *coordinate transformation* interchangeably. In general, there should be no ambiguity between the two interpretations of ${}^x\boldsymbol{x}_y$[1].

Often, we must compose multiple coordinate transformations to obtain a desired change of coordinates. For example, suppose that we are given poses ${}^x\boldsymbol{x}_y$ and ${}^y\boldsymbol{x}_z$. If we are given ${}^z\boldsymbol{P}$ and wish to compute ${}^x\boldsymbol{P}$, we may use the composition of coordinate transformations

$$
\begin{aligned}
{}^x\boldsymbol{P} &= {}^x\boldsymbol{x}_y({}^y\boldsymbol{P}) && (3)\\
&= {}^x\boldsymbol{x}_y({}^y\boldsymbol{x}_z({}^z\boldsymbol{P}) && (4)\\
&= ({}^x\boldsymbol{x}_y \circ {}^y\boldsymbol{x}_z)({}^z\boldsymbol{P}) && (5)\\
&= {}^x\boldsymbol{x}_z({}^z\boldsymbol{P}). && (6)
\end{aligned}
$$

As seen here, we represent the composition of coordinate transformations by ${}^x\boldsymbol{x}_z = {}^x\boldsymbol{x}_y \circ {}^y\boldsymbol{x}_z$, and the corresponding coordinate transformation of the point ${}^z\boldsymbol{P}$ by $({}^x\boldsymbol{x}_y \circ {}^y\boldsymbol{x}_z)({}^z\boldsymbol{P})$. The corresponding rotation matrix and translation are given by

$$
\begin{aligned}
{}^x\boldsymbol{R}_z &= {}^x\boldsymbol{R}_y\,{}^y\boldsymbol{R}_z && (7)\\
{}^x\boldsymbol{t}_z &= {}^x\boldsymbol{R}_y\,{}^y\boldsymbol{t}_z + {}^x\boldsymbol{t}_y. && (8)
\end{aligned}
$$

[1] We have not used more common notations based on homogeneous transforms because over parameterizing points makes it difficult to develop some of the machinery needed for control.

Some coordinate frames that will be needed frequently are referred to by the following superscripts/subscripts:

e	The coordinate frame attached to the robot end effector
t	The coordinate frame attached to the target
0	The base frame for the robot
c_i	The coordinate frame of the ith camera

When $\mathcal{T} = \mathrm{SE}^3$, we will use the notation $\boldsymbol{x}_e \in \mathcal{T}$ to represent the pose of the end-effector coordinate frame relative to the world frame. In this case, we often prefer to parameterize a pose using a translation vector and three angles, (e.g., roll, pitch and yaw [7]). Although such parameterizations are inherently local, it is often convenient to represent a pose by a vector $\boldsymbol{r} \in \Re^6$, rather than by $\boldsymbol{x}_e \in \mathcal{T}$. This notation can easily be adapted to the case where $\mathcal{T} \subseteq \mathrm{SE}^3$. For example, when $\mathcal{T} = \Re^3$, we will parameterize the task space by $\boldsymbol{r} = [x, y, z]^T$. In the sequel, to maintain generality we will assume that $\boldsymbol{r} \in \Re^m$, unless we are considering a specific task.

B. The Velocity of a Rigid Object

In visual servo applications, we are often interested in the relationship between the velocity of some object in the workspace (e.g., the manipulator end-effector) and the corresponding changes that occur in the observed image of the workspace. In this section, we briefly introduce notation to represent velocities of objects in the workspace.

Consider the robot end-effector moving in a workspace with $\mathcal{T} \subseteq \mathrm{SE}^3$. In base coordinates, the motion is described by an angular velocity $\Omega(t) = [\omega_x(t), \omega_y(t), \omega_z(t)]^T$ and a translational velocity $\boldsymbol{T}(t) = [T_x(t), T_y(t), T_z(t)]^T$. The rotation acts about a point which, unless otherwise indicated, we take to be the origin of the base coordinate system. Let $\boldsymbol{P}$ be a point that is rigidly attached to the end-effector, with base frame coordinates $[x, y, z]^T$. The time derivatives of the coordinates of $\boldsymbol{P}$, expressed in base coordinates, are given by

$$\dot{x} = z\omega_y - y\omega_z + T_x \tag{9}$$

$$\dot{y} = x\omega_z - z\omega_x + T_y \tag{10}$$

$$\dot{z} = y\omega_x - x\omega_y + T_z \tag{11}$$

which can be written in vector notation as

$$\dot{\boldsymbol{P}} = \Omega \times \boldsymbol{P} + \boldsymbol{T}. \tag{12}$$

This can be written concisely in matrix form by noting that the cross product can be represented in terms of the *skew-symmetric matrix*

$$sk(\boldsymbol{P}) = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$$

allowing us to write

$$\dot{\boldsymbol{P}} = -sk(\boldsymbol{P})\Omega + \boldsymbol{T}. \tag{13}$$

Together, $\boldsymbol{T}$ and Ω define what is known in the robotics literature as a velocity screw

$$\dot{\boldsymbol{r}} = \begin{bmatrix} T_x \\ T_y \\ T_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}.$$

Note that $\dot{\boldsymbol{r}}$ also represents the derivative of $\boldsymbol{r}$ when the rotation matrix, $\boldsymbol{R}$, is parameterized by the set of rotations about the coordinate axes.

Define the 3×6 matrix $\boldsymbol{A}(\boldsymbol{P}) = [I_3 | -sk(\boldsymbol{P})]$ where I_3 represents the 3×3 identity matrix. Then (13) can be rewritten in matrix form as

$$\dot{\boldsymbol{P}} = \boldsymbol{A}(\boldsymbol{P})\dot{\boldsymbol{r}}. \tag{14}$$

Suppose now that we are given a point expressed in end-effector coordinates, ${}^e\boldsymbol{P}$, and we wish to determine the motion of this point in base coordinates as the robot is in motion. Combining (1) and (14), we have

$$\dot{\boldsymbol{P}} = \boldsymbol{A}(\boldsymbol{x}_e({}^e\boldsymbol{P})\dot{\boldsymbol{r}}. \tag{15}$$

Occasionally, it is useful to transform velocity screws among coordinate frames. For example, suppose that ${}^e\dot{\boldsymbol{r}} = [{}^e\boldsymbol{T};\ {}^e\Omega]^T$ is the velocity of the end-effector in end-effector coordinates. Then the equivalent screw in base coordinates is

$$\dot{\boldsymbol{r}} = \begin{bmatrix} \boldsymbol{T} \\ \Omega \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_e{}^e\boldsymbol{T} - \boldsymbol{R}_e{}^e\Omega \times \boldsymbol{t}_e \\ \boldsymbol{R}_e{}^e\Omega \end{bmatrix}.$$

C. Camera Projection Models

To control the robot using information provided by a computer vision system, it is necessary to understand the geometric aspects of the imaging process. Each camera contains a lens that forms a 2D projection of the scene on the image plane where the sensor is located. This projection causes direct depth information to be lost so that each point on the image plane corresponds to a ray in 3D space. Therefore, some additional information is needed to determine the 3D coordinates corresponding to an image plane point. This information may come from multiple cameras, multiple views with a single camera, or knowledge of the geometric relationship between several feature points on the target. In this section, we describe three projection models that have been widely used to model the image formation process: perspective projection, scaled orthographic projection, and affine projection. Although we briefly describe each of these projection models, throughout the remainder of the tutorial we will assume the use of perspective projection.

For each of the three projection models, we assign the camera coordinate system with the x- and y-axes forming a basis for the image plane, the z-axis perpendicular to the image plane (along the optical axis), and with origin located at distance λ behind the image plane, where λ is the focal length of the camera lens. This is illustrated in Fig. 1.

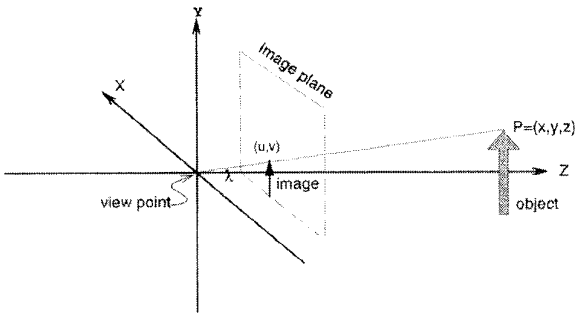

Fig. 1. The coordinate frame for the camera/lens system.

1) Perspective Projection: Assuming that the projective geometry of the camera is modeled by perspective projection (see, e.g., [8]), a point, ${}^c\boldsymbol{P} = [x, y, z]^T$, whose coordinates are expressed with respect to the camera coordinate frame, c, will project onto the image plane with coordinates $\boldsymbol{p} = [u, v]^T$, given by

$$\pi(x, y, z) = \begin{bmatrix} u \\ v \end{bmatrix} = \frac{\lambda}{z} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{16}$$

If the coordinates of $\boldsymbol{P}$ are expressed relative to coordinate frame x, we must first perform the coordinate transformation ${}^c\boldsymbol{P} = {}^c\boldsymbol{x}_x({}^x\boldsymbol{P})$.

2) Scaled Orthographic Projection: Perspective projection is a nonlinear mapping from Cartesian to image coordinates. In many cases, it is possible to approximate this mapping by the linear scaled orthographic projection. Under this model, image coordinates for point ${}^c\boldsymbol{P}$ are given by

$$\begin{bmatrix} u \\ v \end{bmatrix} = s \begin{bmatrix} x \\ y \end{bmatrix} \tag{17}$$

where s is a fixed scale factor.

Orthographic projection models are valid for scenes where the relative depth of the points in the scene is small compared to the distance from the camera to the scene, for example, an airplane flying over the earth, or a camera with a long focal length lens placed several meters from the workspace.

3) Affine projection: Another linear approximation to perspective projection is known as affine projection. In this case, the image coordinates for the projection of a point ${}^c\boldsymbol{P}$ are given by

$$\begin{bmatrix} u \\ v \end{bmatrix} = \boldsymbol{A}{}^c\boldsymbol{P} + \boldsymbol{c} \tag{18}$$

where $\boldsymbol{A}$ is an arbitrary 2×3 matrix and $\boldsymbol{c}$ is an arbitrary 2-vector.

Note that scaled orthographic projection is a special case of affine projection. Affine projection does not correspond to any specific imaging situation. Its primary advantage is that it is a good local approximation to perspective projection that accounts for both the external geometry of the camera (*i.e.,* its position in space), and the internal geometry of the lens and CCD (*i.e.,* the focal length, and scaling and offset to pixel coordinates). Since the model is purely linear, $\boldsymbol{A}$ and $\boldsymbol{c}$ are easily computed using linear regression techniques [9], and the camera calibration problem is greatly simplified.

D. Image Features and the Image Feature Parameter Space

In the computer vision literature, an *image feature* is any structural feature than can be extracted from an image (e.g., an edge or a corner). Typically, an image feature will correspond to the projection of a physical feature of some object (e.g., the robot tool) onto the camera image plane. A good feature point is one that can be located unambiguously in different views of the scene, such as a hole in a gasket [10] or a contrived pattern [11], [12]. We define an *image feature parameter* to be any real-valued quantity that can be calculated from one or more image features.[2] Some of the feature parameters that have been used for visual servo control include the image plane coordinates of points in the image [11], [14]–[19], the distance between two points in the image plane and the orientation of the line connecting those two points [10], [20], perceived edge length [21], the area of a projected surface and the relative areas of two projected surfaces [21], the centroid and higher order moments of a projected surface [21]–[24], the parameters of lines in the image plane [11], and the parameters of an ellipse in the image plane [11]. In this tutorial we will restrict our attention to point features whose parameters are their image plane coordinates.

Given a set of k image feature parameters, we can define an image feature parameter vector $\boldsymbol{f} = [f_1 \cdots f_k]^T$. Since each f_i is a (possibly bounded) real valued parameter, we have $\boldsymbol{f} = [f_1 \cdots f_k]^T \in \mathcal{F} \subseteq \Re^k$, where $\mathcal{F}$ represents the *image feature parameter space.*

The mapping from the position and orientation of the end-effector to the corresponding image feature parameters can be computed using the projective geometry of the camera. We will denote this mapping by $\boldsymbol{F}$, where

$$\boldsymbol{F}: \mathcal{T} \rightarrow \mathcal{F}. \tag{19}$$

For example, if $\mathcal{F} \subseteq \Re^2$ is the space of u, v image plane coordinates for the projection of some point $\boldsymbol{P}$ onto the image plane, then, assuming perspective projection, $\boldsymbol{f} = [u, v]^T$, where u and v are given by (16). The exact form of (19) will depend in part on the relative configuration of the camera and end-effector as discussed in the next section.

E. Camera Configuration

Visual servo systems typically use one of two camera configurations: end-effector mounted, or fixed in the workspace.

The first, often called an eye-in-hand configuration, has the camera mounted on the robot's end-effector. Here, there exists a known, often constant, relationship between the pose of the camera(s) and the pose of the end-effector. We represent this relationship by the pose ${}^e\boldsymbol{x}_c$. The pose of the target[3] relative to the camera frame is represented by ${}^c\boldsymbol{x}_t$. The relationship between these poses is shown in Fig. 2.

The second configuration has the camera(s) fixed in the workspace. In this case, the camera(s) are related to the base coordinate system of the robot by ${}^0\boldsymbol{x}_c$ and to the object by

[2] Jang [13] provides a formal definition of what we term feature parameters as image functionals.

[3] The word *target* will be used to refer to the object of interest, that is, the object that will be tracked.

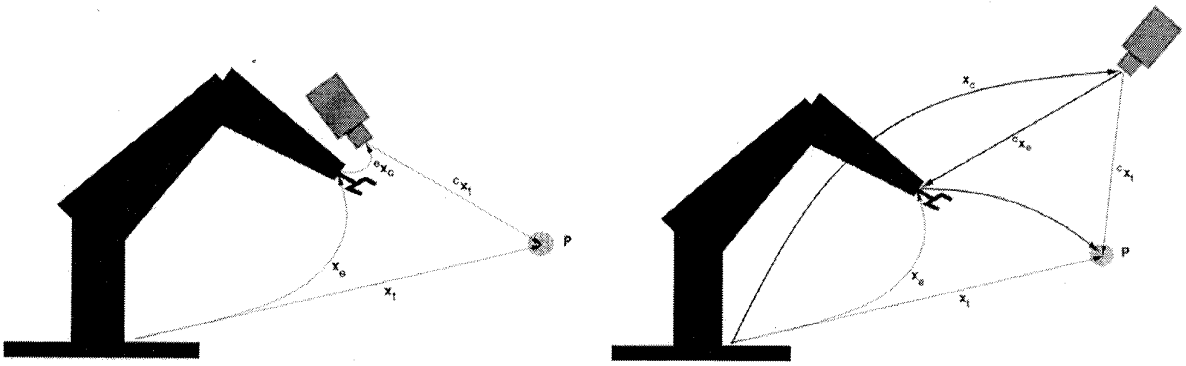

Fig. 2. Relevant coordinate frames (world, end-effector, camera and target) for end-effector mounted, and fixed, camera configurations.

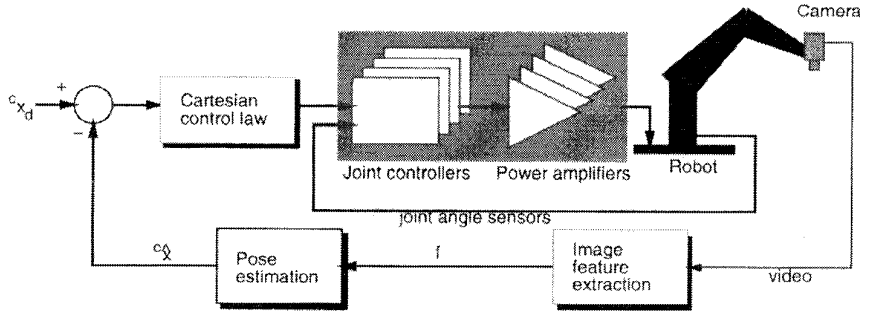

Fig. 3. Dynamic position-based look-and-move structure.

${}^c\boldsymbol{x}_t$. In this case, the camera image of the target is, of course, independent of the robot motion (unless the target is the end-effector itself). A variant of this is for the camera to be agile, mounted on another robot or pan/tilt head in order to observe the visually controlled robot from the best vantage [25].

For either choice of camera configuration, prior to the execution of visual servo tasks, camera calibration must be performed in order to determine the intrinsic camera parameters such as focal length, pixel pitch and the principal point. A fixed camera's pose, ${}^0\boldsymbol{x}_c$, with respect to the world coordinate system must be established, and is encapsulated in the extrinsic parameters determined by a camera calibration procedure. For the eye-in-hand case the relative pose, ${}^e\boldsymbol{x}_c$, must be determined and this is known as the hand/eye calibration problem. Calibration is a long standing research issue in the computer vision community (good solutions to the calibration problem can be found in a number of references, e.g., [26]–[28]).

III. Servoing Architectures

In 1980, Sanderson and Weiss [5] introduced a taxonomy of visual servo systems, into which all subsequent visual servo systems can be categorized. Their scheme essentially poses two questions:

1) Is the control structure hierarchical, with the vision system providing set-points as input to the robot's joint-level controller, or does the visual controller directly compute the joint-level inputs?
2) Is the error signal defined in 3D (task space) coordinates, or directly in terms of image features?

The resulting taxonomy, thus, has four major categories, which we now describe. These fundamental structures are shown schematically in Figs. 3–6.

If the control architecture is hierarchical and uses the vision system to provide set-point inputs to the joint-level controller, thus making use of joint feedback to internally stabilize the robot, it is referred to as a *dynamic look-and-move* system. In contrast, *direct visual servo*[4] eliminates the robot controller entirely replacing it with a visual servo controller that directly computes joint inputs, thus using vision alone to stabilize the mechanism.

For several reasons, nearly all implemented systems adopt the dynamic look-and-move approach. Firstly, the relatively low sampling rates available from vision make direct control of a robot end-effector with complex, nonlinear dynamics an extremely challenging control problem. Using internal feedback with a high sampling rate generally presents the visual controller with idealized axis dynamics [29]. Secondly, many robots already have an interface for accepting Cartesian velocity or incremental position commands. This simplifies the construction of the visual servo system, and also makes the methods more portable. Thirdly, look-and-move separates the kinematic singularities of the mechanism from the visual controller, allowing the robot to be considered as

[4] Sanderson and Weiss used the term "visual servo" for this type of system, but since then this term has come to be accepted as a generic description for any type of visual control of a robotic system. Here we use the term "direct visual servo" to avoid confusion.

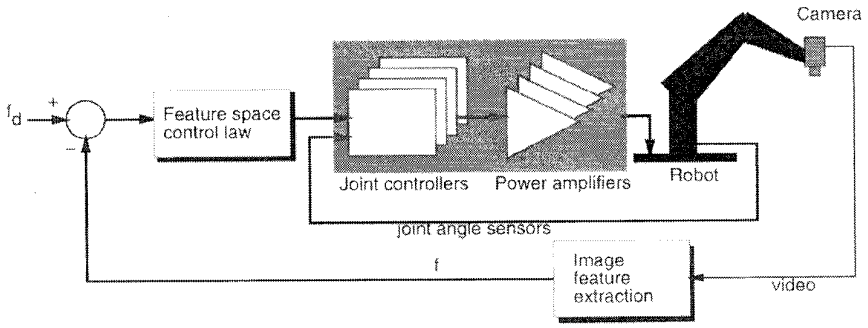

Fig. 4. Dynamic image-based look-and-move structure.

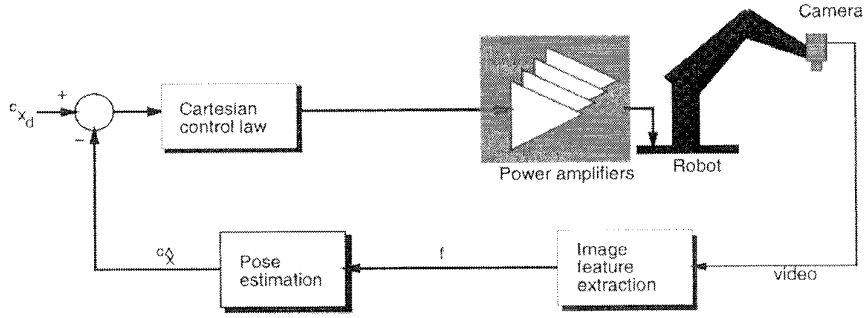

Fig. 5. Position-based visual servo (PBVS) structure as per Weiss.

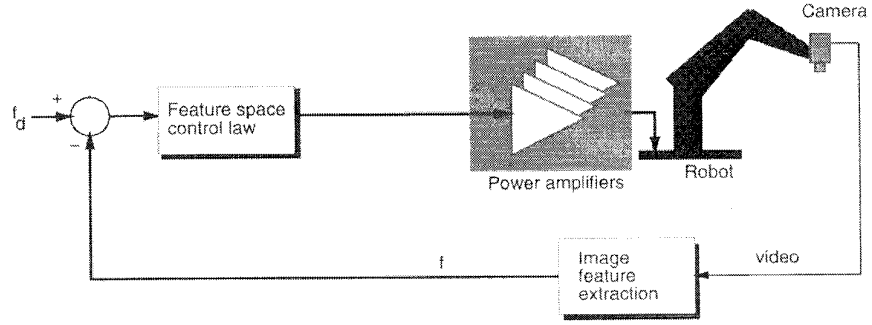

Fig. 6. Image-based visual servo (IBVS) structure as per Weiss.

an ideal Cartesian motion device. Since many resolved rate [30] controllers have specialized mechanisms for dealing with kinematic singularities [31], the system design is again greatly simplified. In this article, we will utilize the look-and-move model exclusively.

The second major classification of systems distinguishes *position-based* control from *image-based* control. In *position-based* control, features are extracted from the image and used in conjunction with a geometric model of the target and the known camera model to estimate the pose of the target with respect to the camera. Feedback is computed by reducing errors in estimated pose space. In *image-based* servoing, control values are computed on the basis of image features directly. The image-based approach may reduce computational delay, eliminate the necessity for image interpretation and eliminate errors due to sensor modeling and camera calibration. However it does present a significant challenge to controller design since the plant is nonlinear and highly coupled.

One of the typical applications of visual servoing is to position an end-effector relative to a target. For example, many authors use an end-effector mounted camera to position a robot arm for grasping. In most cases, the control algorithm is expressed in terms of moving the *camera* to a pose defined in terms of the image of the object to be grasped. The position of the end-effector relative to the object is determined only indirectly by its known kinematic relationship with the camera. Errors in this kinematic relationship lead to positioning errors which cannot be observed by the system. Observing the end-effector directly makes it possible to sense and correct for such errors. In general, there is no guarantee on the positioning accuracy of the system unless control points on *both* the end-effector and target can be observed [9], [32],

[33]. To emphasize this distinction, we refer to systems that only observe the target object as *endpoint open-loop* (EOL) systems, and systems that observe both the target object and the robot end-effector as *endpoint closed-loop* (ECL) systems. The differences between EOL and ECL systems will be made more precise in subsequent discussions.

It is usually possible to transform an EOL system to an ECL system simply by including direct observation of the end-effector or other task-related control points. Thus, from a theoretical perspective, it would appear that ECL systems would always be preferable to EOL systems. However, since ECL systems must track the end-effector as well as the target object, the implementation of an ECL controller often requires solution of a more demanding vision problem and places field-of-view constraints on the system that cannot always be satisfied.

IV. Position-Based Visual Servo Control

We begin our discussion of visual servoing methods with position-based visual servoing. As described in the previous section, in position-based visual servoing, features are extracted from the image and used to estimate the pose of the target with respect to the camera. Using these values, an error between the current and the desired pose of the robot is defined in the task space. In this way, position-based control neatly separates the control issues, namely the the computation of the feedback signal, from the estimation problems involved in computing position or pose from visual data.

We now formalize the notion of a positioning task as follows:

Definition 4.1: A *positioning task* is represented by a function $\boldsymbol{E}$: $\mathcal{T} \rightarrow \Re^m$. This function is referred to as the *kinematic error function.* A positioning task is *fulfilled* with the end-effector in pose $\boldsymbol{x}_e$ if $\boldsymbol{E}(\boldsymbol{x}_e) = \boldsymbol{0}$.

If we consider a general pose $\boldsymbol{x}_e$ for which the task is fulfilled, the error function will constrain some number, $d \leq m$, degrees of freedom of the manipulator. The value d will be referred to as the *degree* of the constraint. As noted by Espiau *et al.* [11], [34], the kinematic error function can be thought of as representing a *virtual kinematic constraint* between the end-effector and the target.

Once a suitable kinematic error function has been defined and the parameters of the functions are instantiated from visual data, a regulator is defined that reduces the estimated value of the kinematic error function to zero. This regulator produces at every time instant a desired end-effector velocity screw $\boldsymbol{u} \in \Re^6$ that is sent to the robot control subsystem. For the purposes of this article, we use simple proportional control methods for linear and linearized systems to compute $\boldsymbol{u}$ [35]. Although there are formalized methods for developing such control laws, since the kinematic error functions are defined in Cartesian space, for most problems it is possible to develop a regulator through geometric insight. The process is to first determine the relative motion that would fulfill the task, and then to write a control law that would produce that motion.

The remainder of the section presents various example problems that we have chosen to provide some insight into ways of thinking about position-based control, and that will also provide useful comparisons when we consider image-based control in the next section. Section IV-A introduces several simple positioning primitives, based on directly observable feature points, which can be compounded to achieve more complex positioning tasks. Next, Section IV-B describes positioning tasks based on the explicit estimation of the target object's pose. Finally, in Section IV-C, we briefly describe how point position and object pose can be computed using visual information from one or more cameras—the visual reconstruction problem.

A. Point-Feature Based Motions

We begin by considering a positioning task in which some point on the robot with end-effector coordinates, ${}^e\boldsymbol{P}$, is to be brought to a fixed stationing point, $\boldsymbol{S}$, visible in the scene. We refer to this as *point-to-point positioning*. In the case where the camera is fixed, the kinematic error function may be defined in base coordinates as

$$\boldsymbol{E}_{pp}(\boldsymbol{x}_e; \boldsymbol{S}, {}^e\boldsymbol{P}) = \boldsymbol{x}_e({}^e\boldsymbol{P}) - \boldsymbol{S}. \tag{20}$$

Here, as in the sequel, the argument before the semicolon is the value to be controlled (in all cases, manipulator position) and the values after the semicolon parameterize the positioning task.

$\boldsymbol{E}_{pp}$ defines a three degree of freedom kinematic constraint on the robot end-effector position. If the robot workspace is restricted to be $\mathcal{T} = \Re^3$, this task can be thought of as a rigid link that fully constrains the pose of the end-effector relative to the target. When $\mathcal{T} \subseteq \mathrm{SE}^3$, the constraint defines a *virtual spherical joint* between the object and the robot end-effector.

Let $\mathcal{T} = \Re^3$. We first consider the case in which one or more cameras calibrated to the robot base frame furnish an estimate, ${}^c\hat{\boldsymbol{S}}$, of the stationing point coordinates with respect to a camera coordinate frame. Using the estimate of the camera pose in base coordinates, $\hat{\boldsymbol{x}}_c$, from off-line calibration and (1), we have $\hat{\boldsymbol{S}} = \hat{\boldsymbol{x}}_c({}^c\hat{\boldsymbol{S}})$.

Since $\mathcal{T} = \Re^3$, the control input to be computed is the desired robot translational velocity, which we denote by $\boldsymbol{u}_3$ to distinguish it from the more general end-effector screw. Since (20) is linear in $\boldsymbol{x}_e$, it is well known that in the absence of outside disturbances, the proportional control law

$$\boldsymbol{u}_3 = -k\, \boldsymbol{E}_{pp}(\hat{\boldsymbol{x}}_e; \hat{\boldsymbol{x}}_c({}^c\hat{\boldsymbol{S}}), {}^e\boldsymbol{P}) = -k(\hat{\boldsymbol{x}}_e({}^e\boldsymbol{P}) - \hat{\boldsymbol{x}}_c({}^c\hat{\boldsymbol{S}})) \tag{21}$$

will drive the system to an equilibrium state in which the value of the error function is zero [35]. The value $k > 0$ is a proportional feedback gain. Note that we have written $\hat{\boldsymbol{x}}_e$ in the feedback law to emphasize the fact that this value is also subject to errors.

The expression (21) is equivalent to open-loop positioning of the manipulator using vision-based estimates of geometry. Variations on this scheme are used by [36], [37]. In our simplified dynamics, the manipulator is stationary when $\boldsymbol{u}_3 = 0$. Since the right hand side of the equation includes estimated quantities, it follows that errors in $\hat{\boldsymbol{x}}_e$, $\hat{\boldsymbol{x}}_c$ or ${}^c\hat{\boldsymbol{S}}$

(robot kinematics, camera calibration and visual reconstruction respectively) can lead to positioning errors of the end-effector.

Now, consider the situation when the cameras are mounted on the robot and calibrated to the end-effector. In this case, we can express (20) in end-effector coordinates

$$ {}^{e}\boldsymbol{E}_{pp}(\boldsymbol{x}_e; \boldsymbol{S}, {}^{e}\boldsymbol{P}) = {}^{e}\boldsymbol{P} - {}^{e}\boldsymbol{x}_0(\boldsymbol{S}). \tag{22} $$

The camera(s) furnish an estimate of the stationing point, ${}^{c}\hat{\boldsymbol{S}}$, which can be combined with information from the camera calibration and robot kinematics to produce $\hat{\boldsymbol{S}} = (\hat{\boldsymbol{x}}_e \circ {}^{e}\hat{\boldsymbol{x}}_c)({}^{c}\hat{\boldsymbol{S}})$. We now compute

$$ \begin{aligned} {}^{e}\boldsymbol{u}_3 &= -k\, {}^{e}\boldsymbol{E}_{pp}(\hat{\boldsymbol{x}}_e; (\hat{\boldsymbol{x}}_e \circ {}^{e}\hat{\boldsymbol{x}}_c)({}^{c}\hat{\boldsymbol{S}}), {}^{e}\boldsymbol{P}) \\ &= -k({}^{e}\boldsymbol{P} - ({}^{e}\boldsymbol{x}_0 \circ {}^{0}\hat{\boldsymbol{x}}_e \circ {}^{e}\hat{\boldsymbol{x}}_c)({}^{c}\hat{\boldsymbol{S}})) \\ &= -k({}^{e}\boldsymbol{P} - {}^{e}\hat{\boldsymbol{x}}_c({}^{c}\hat{\boldsymbol{S}})). \end{aligned} \tag{23} $$

Notice that the terms involving $\hat{\boldsymbol{x}}_e$ have dropped out. Thus (23) is not only simpler, but positioning accuracy is also independent of the accuracy of the robot kinematics—a fundamental benefit of visual servoing.

All of the above formulations presume prior knowledge of ${}^{e}\boldsymbol{P}$ and are therefore EOL systems. To convert them to ECL systems, we suppose that ${}^{e}\boldsymbol{P}$ is directly observed and estimated by the camera system. In this case, (21) and (23) can be written

$$ \boldsymbol{u}_3 = -k\, \boldsymbol{E}_{pp}(\hat{\boldsymbol{x}}_e; \hat{\boldsymbol{x}}_c({}^{c}\hat{\boldsymbol{S}}), {}^{e}\hat{\boldsymbol{x}}_c({}^{c}\hat{\boldsymbol{P}})) = -k\, \hat{\boldsymbol{x}}_c({}^{c}\hat{\boldsymbol{P}} - {}^{c}\hat{\boldsymbol{S}}) \tag{24} $$

$$ {}^{e}\boldsymbol{u}_3 = -k\, {}^{e}\boldsymbol{E}_{pp}(\hat{\boldsymbol{x}}_e; \hat{\boldsymbol{x}}_c({}^{c}\hat{\boldsymbol{S}}), {}^{e}\hat{\boldsymbol{x}}_c({}^{c}\hat{\boldsymbol{P}})) = -k\, {}^{e}\hat{\boldsymbol{x}}_c({}^{c}\hat{\boldsymbol{P}} - {}^{c}\hat{\boldsymbol{S}}) \tag{25} $$

respectively. We now see that $\boldsymbol{u}_3$ (respectively ${}^{e}\boldsymbol{u}_3$) does not depend on $\hat{\boldsymbol{x}}_e$ and is homogeneous in $\hat{\boldsymbol{x}}_c$ (respectively ${}^{e}\hat{\boldsymbol{x}}_c$). Hence, if ${}^{c}\hat{\boldsymbol{S}} = {}^{c}\hat{\boldsymbol{P}}$, then $\boldsymbol{u}_3 = \boldsymbol{0}$, *independent of errors in the robot kinematics or the camera calibration.* This is an important advantage for systems where a precise camera/end-effector relationship is difficult or impossible to determine off-line.

Consider now the full Cartesian problem where $\mathcal{T} \subseteq \mathrm{SE}^3$, and the control input is the complete velocity screw $\boldsymbol{u} \in \Re^6$. Since, the error functions presented above only constrain 3 degrees of freedom, the problem of computing $\boldsymbol{u}$ from the estimated error is under-determined. One way of proceeding is as follows. Consider the case of free standing cameras. Then in base coordinates we know that $\dot{\boldsymbol{P}} = \boldsymbol{u}_3$. Using (14), we can relate this to the end-effector velocity screw as follows:

$$ \dot{\boldsymbol{P}} = \boldsymbol{u}_3 = \boldsymbol{A}(\boldsymbol{P})\boldsymbol{u}. \tag{26} $$

Thus, if we could "solve for" $\boldsymbol{u}$ in the above equation, we could effectively use the three-dimensional solution to arrive at the full Cartesian solution. Unfortunately, $\boldsymbol{A}$ is not square and therefore can cannot be inverted to solve for $\boldsymbol{u}$. However, recall that the matrix right inverse for an $m \times n$ matrix $\boldsymbol{M}, n > m$ is defined as $\boldsymbol{M}^{+} = \boldsymbol{M}^{T}(\boldsymbol{M}\boldsymbol{M}^{T})^{-1}$. The right inverse computes the minimum norm vector which solves the original system of equations. Hence, we have

$$ \boldsymbol{u} = \boldsymbol{A}(\boldsymbol{P})^{+}\boldsymbol{u}_3 \tag{27} $$

for free-standing cameras. Similar manipulations yield

$$ {}^{e}\boldsymbol{u} = \boldsymbol{A}({}^{e}\boldsymbol{P})^{+}\, {}^{e}\boldsymbol{u}_3 \tag{28} $$

for end-effector mounted cameras. Substituting the appropriate expression for $\boldsymbol{u}_3$ or ${}^{e}\boldsymbol{u}_e$ from the previous discussion leads to a form of proportional regulation for the Cartesian problem.

As a second example of feature-based positioning, consider that some point on the end-effector, ${}^{e}\boldsymbol{P}$, is to be brought to the *line joining* two fixed points $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ in the world. The shortest path for performing this task is to move ${}^{e}\boldsymbol{P}$ toward the line joining $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ along the perpendicular to the line. The error function describing this trajectory in base coordinates is:

$$ \begin{aligned} &\boldsymbol{E}_{pl}(\boldsymbol{x}_e; \boldsymbol{S}_1, \boldsymbol{S}_2, {}^{e}\boldsymbol{P}) \\ &\quad = (\boldsymbol{S}_2 - \boldsymbol{S}_1) \times ((\boldsymbol{x}_e({}^{e}\boldsymbol{P}) - \boldsymbol{S}_1) \times (\boldsymbol{S}_2 - \boldsymbol{S}_1)). \end{aligned} \tag{29} $$

Notice that although $\boldsymbol{E}_{p}l$ is a mapping from $\mathcal{T}$ to $\Re^3$, placing a point on a line is a constraint of degree 2. From the geometry of the problem and the previous discussion, we see that defining

$$ \boldsymbol{u} = -k\boldsymbol{A}(\hat{\boldsymbol{x}}_e({}^{e}\boldsymbol{P}))^{+}\boldsymbol{E}_{pl}(\hat{\boldsymbol{x}}_e; \widehat{\boldsymbol{S}}_1, \widehat{\boldsymbol{S}}_2, {}^{e}\boldsymbol{P}) $$

is a proportional feedback law for this problem.

Suppose that now we apply this constraint to two points on the end-effector

$$ \boldsymbol{E}_{ppl}(\boldsymbol{x}_e; \boldsymbol{S}_1, \boldsymbol{S}_2, {}^{e}\boldsymbol{P}_1, {}^{e}\boldsymbol{P}_2) = \begin{bmatrix} \boldsymbol{E}_{pl}(\boldsymbol{x}_e; \boldsymbol{S}_1, \boldsymbol{S}_2, {}^{e}\boldsymbol{P}_1) \\ \boldsymbol{E}_{pl}(\boldsymbol{x}_e; \boldsymbol{S}_1, \boldsymbol{S}_2, {}^{e}\boldsymbol{P}_2) \end{bmatrix}. $$

$\boldsymbol{E}_{ppl}$ now defines a four degree of freedom positioning constraint that aligns the points on the end-effector with those in target coordinates, and again no unique motion satisfies this kinematic error function. A geometrically straightforward solution is to compute a translation, $\boldsymbol{T}$, which moves ${}^{e}\boldsymbol{P}_1$ to the line through $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$. Simultaneously, we can choose a rotation, $\boldsymbol{R}$ which rotates ${}^{e}\boldsymbol{P}_2$ about ${}^{e}\boldsymbol{P}_1$ so that the line through ${}^{e}\boldsymbol{P}_1$ and ${}^{e}\boldsymbol{P}_2$ becomes parallel to that through $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$.

In order to compute a velocity screw $\boldsymbol{u} = (\boldsymbol{T}, \Omega)$, we first note that the end-effector rotation matrix $\boldsymbol{R}_e$ can be represented as a rotation through an angle θ about an axis defined by a unit vector $\boldsymbol{k}$ [7]. In this case, the axis of rotation is

$$ \boldsymbol{k} = \overline{(\boldsymbol{S}_2 - \boldsymbol{S}_1)} \times \overline{[\boldsymbol{R}_e({}^{e}\boldsymbol{P}_2 - {}^{e}\boldsymbol{P}_1)]} $$

where the bar over expressions on the right denotes normalization to a unit vector. Hence, a natural feedback law for the rotational portion of the velocity screw is

$$ \Omega = -k_1\boldsymbol{k}. \tag{30} $$

Note the expression on the right hand size is the zero vector if the lines joining associated points are parallel as we desire.

The only complication to computing the translation portion of the vector is to realize that rotation introduces translation of points attached to the end-effector. Hence, we need to move ${}^{e}\boldsymbol{P}_1$ toward the goal line while compensating for the motion introduced by rotation. Based on the discussion above, we know the former is given by $-\boldsymbol{E}_{pl}(\boldsymbol{x}_e; \boldsymbol{S}_1, \boldsymbol{S}_2, {}^{e}\boldsymbol{P}_1)$ while

from (12) the latter is simply $\Omega \times \boldsymbol{x}_e({}^e\boldsymbol{P}_1)$. Combining these two expressions, we have

$$\boldsymbol{T} = -k_2(\boldsymbol{S}_2 - \boldsymbol{S}_1) \times ((\hat{\boldsymbol{x}}_e({}^e\boldsymbol{P}_1) - \boldsymbol{S}_1) \times (\boldsymbol{S}_2 - \boldsymbol{S}_1)) - \Omega \times (\hat{\boldsymbol{x}}_e({}^e\boldsymbol{P}_1)). \quad (31)$$

Note that we are still free to choose translations along the line joining $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ as well as rotations about it. Full six degree-of-freedom positioning can be attained by enforcing another point-to-line constraint using an additional point on the end-effector and an additional point in the world. Similar geometric arguments can be used to define a proportional feedback law.

These formulations can be adjusted for end-effector mounted camera and can be implemented as ECL or EOL systems. We leave these modifications as an exercise for the reader.

B. Pose-Based Motion

In the previous section, positioning was defined in terms of directly observable point features. When working with *a priori* known objects, it is possible to recover the *pose* of the object, $\boldsymbol{x}_t$, and to define stationing points with respect to object pose.

The methods of the previous section can be easily applied when object pose is available. For example, suppose ${}^t\boldsymbol{S}$ is an arbitrary stationing point in a target object's coordinate system, and that we can compute ${}^e\hat{\boldsymbol{x}}_t$ using end-effector mounted camera(s). Then using (1) we can compute ${}^e\hat{\boldsymbol{S}} = {}^e\hat{\boldsymbol{x}}_t({}^t\boldsymbol{S})$. This estimate can be used in any of the end-effector based feedback methods of the previous section in both ECL and EOL configurations. Similar remarks hold for systems utilizing free-standing cameras.

Given an object pose, it is possible to directly define positioning tasks in terms of that object pose. Let ${}^t\boldsymbol{x}_e^*$ be a desired stationing pose (rather than point as in the previous section) for the end-effector, and suppose the system employs free-standing cameras. We can define a positioning error

$$\boldsymbol{E}_{rp}(\boldsymbol{x}_e; {}^t\boldsymbol{x}_e^*, \boldsymbol{x}_t) = {}^e\boldsymbol{x}_e^* = {}^e\boldsymbol{x}_0 \circ \boldsymbol{x}_t \circ {}^t\boldsymbol{x}_e^*. \quad (32)$$

(Note that in order for this error function to be in accord with our definition of kinematic error we must select a parameterization of rotations which is 0 when the end-effector is in the desired position.)

Using feature information and the camera calibration, we can directly estimate $\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_c \circ {}^c\hat{\boldsymbol{x}}_t$. If we again represent the rotation in terms of a unit vector ${}^{e^*}\boldsymbol{k}_e$ and rotation angle ${}^{e^*}\theta_e$, we can define

$$\Omega = k_1 {}^{e^*}\hat{\theta}_e {}^{e^*}\hat{\boldsymbol{k}}_e \quad (33)$$

$$T = k_2 {}^{e^*}\hat{\boldsymbol{t}}_e - \boldsymbol{t}_e \times \Omega \quad (34)$$

where $\boldsymbol{t}_e$ is the origin of the end-effector frame in base coordinates.

If we can also observe the end-effector and estimate its pose, ${}^c\hat{\boldsymbol{x}}_e$ we can rewrite (32) as follows:

$${}^e\hat{\boldsymbol{x}}_{e^*} = ({}^e\hat{\boldsymbol{x}}_c \circ {}^c\hat{\boldsymbol{x}}_0) \circ ({}^0\hat{\boldsymbol{x}}_c \circ {}^c\hat{\boldsymbol{x}}_t) \circ {}^t\boldsymbol{x}_{e^*} = {}^e\hat{\boldsymbol{x}}_c \circ {}^c\hat{\boldsymbol{x}}_t \circ {}^t\boldsymbol{x}_{e^*}.$$

Once again we see that for an ECL system, both the robot kinematic chain and the camera pose relative to the base coordinate system have dropped out of the error equation. Hence, these factors do not affect the positioning accuracy of the system.

The modifications of pose-based methods to end-effector based systems are completely straightforward and are left for the reader.

C. Estimation

A key issue in position-based visual servo is the estimation of the quantities used to parameterize the feedback. In this regard, position-based visual servoing is closely related to the problem of recovering scene geometry from one or more camera images. This encompasses problems including structure from motion, exterior orientation, stereo reconstruction, and absolute orientation. Unfortunately, space does not permit a complete coverage of these topics here and we have opted to provide pointers to the literature, except in the case of point estimation for two cameras, which has a straightforward solution. A comprehensive discussion of these topics can be found in a recent review article [38].

1) Estimation with a Single Camera: As noted previously, it follows from (16) that a point in a single camera image corresponds to a line in space. Although it is possible to perform geometric reconstruction using a single moving camera, the equations governing this process are often ill-conditioned, leading to stability problems [38]. Better results can be achieved if target image features have some internal structure, or the image features come from a known object. Below, we briefly describe methods for performing both point estimation and pose estimation with a single camera assuming such information is available.

a) Single Points: Clearly, extra information is needed in order to reconstruct the Cartesian coordinates of a point in space from a single camera projection. This may come from additional measurable attributes, for example, in the case of a circular opening with known diameter d the image will be an ellipse. The ellipse can be described by five image feature parameters from which can be derived distance to the opening, and orientation of the plane containing the hole.

b) Object Pose: Object pose can be estimated if the vision system observes multiple point features on a known object. This is referred to as the *pose estimation* problem in the vision literature, and numerous methods for its solution have been proposed. These can be broadly divided into analytic solutions and least-squares solutions. Analytic solutions for three and four points are given by [39]–[43], and unique solutions exist for four coplanar, but not collinear, points. Least-squares solutions can be found in [44]–[50]. Six or more points always yield unique solutions and allow the camera calibration matrix to be computed. This can then be decomposed [48] to yield the target's pose.

The general least-squares solution is a nonlinear optimization problem which has no known closed-form solution. Instead, iterative optimization techniques are generally employed. These techniques iteratively refine a nominal pose value using observed data (see [51] for a recent review). Because of the sensitivity of the reconstruction process to

noise, it is often a good idea to incorporate some type of smoothing or averaging of the computed pose parameters, at the cost of some delay in response to changes in target pose. A particularly elegant formulation of this updating procedure results by application of statistical techniques such as the extended Kalman filter [52]. This approach has been recently demonstrated by Wilson [53] for six DOF control of end-effector pose. A similar approach was recently reported in [54].

2) Estimation with Multiple Cameras: Multiple cameras greatly simplify the reconstruction process and many systems utilizing position-based control with stereo vision from free-standing cameras have been demonstrated. For example, Allen [36] shows a system that can grasp a toy train using stereo vision. Rizzi [37] demonstrates a system which can bounce a ping-pong ball. All of these systems are EOL. Cipolla [9] describes an ECL system using free-standing stereo cameras. One novel feature of this system is the use of the affine projection model (Section II-C) for the imaging geometry. This leads to linear calibration and control at the cost of some system performance. The development of a position-based stereo eye-in-hand servoing system has also been reported [55].

a) Single Points: Let ${}^a\boldsymbol{x}_{c1}$ represent the pose of a camera relative to an arbitrary base coordinate frame a. By inverting this transformation and combining (1) and (16) for a point ${}^a\boldsymbol{P} = [x, y, z]^T$ we have

$$\boldsymbol{p}_1 = \begin{bmatrix} u_1 \\ v_1 \end{bmatrix} = \frac{\lambda}{\boldsymbol{z}\,{}^a\boldsymbol{P} + t_z} \begin{bmatrix} \boldsymbol{x}\,{}^a\boldsymbol{P} + t_x \\ \boldsymbol{y}\,{}^a\boldsymbol{P} + t_y \end{bmatrix} \tag{35}$$

where $\boldsymbol{x}, \boldsymbol{y}$ and $\boldsymbol{z}$ are the rows of ${}^{c1}\boldsymbol{R}_a$ and ${}^{c1}\boldsymbol{t}_a = [t_x, t_y, t_z]^T$. Multiplying through by the denominator of the right-hand side, we have

$$A_1(\boldsymbol{p}_1)\,{}^a\boldsymbol{P} = b_1(\boldsymbol{p}_1). \tag{36}$$

where

$$A_1(\boldsymbol{p}_1) = \begin{bmatrix} \lambda\boldsymbol{x} - u_1\boldsymbol{z} \\ \lambda\boldsymbol{y} - v_1\boldsymbol{z} \end{bmatrix} {}^a\boldsymbol{P} \quad \text{and} \quad b_1(\boldsymbol{p}_1) = \begin{bmatrix} u_1 t_z - \lambda t_x \\ v_1 t_z - \lambda t_y \end{bmatrix}.$$

Given a second camera at location ${}^a\boldsymbol{x}_{c2}$ we can compute $A_2(\boldsymbol{p}_2)$ and $b_2(\boldsymbol{p}_2)$ analogously. Stacking these together results in a matrix equation

$$\begin{bmatrix} A_1(\boldsymbol{p}_1) \\ A_2(\boldsymbol{p}_2) \end{bmatrix} {}^a\boldsymbol{P} = \begin{bmatrix} b_1(\boldsymbol{p}_1) \\ b_2(\boldsymbol{p}_2) \end{bmatrix}.$$

which is an over-determined system that can be solved for ${}^a\boldsymbol{P}$. Note the same approach can be used to provide estimates from three or more cameras.

b) Object Pose: As seen above, given two or more cameras, it is straightforward to estimate their camera relative coordinates. Given observations of three or more points in known locations with respect to an object coordinate system, it is relatively straightforward to solve the *absolute orientation* problem which relates camera coordinates to object coordinates. The solution is based on noting that the centroid of a rigid set of points is invariant to rotation. By exploiting this observation, it is possible to first isolate rotation as the only unknown in the system. The corresponding least-squares problem can either be solved explicitly for rotation (see [56]–[58]), or solved incrementally using linearization. Given an estimate for rotation, the computation of translation is a standard linear least squares problem.

D. Discussion

The principle advantage of position-based control is that it is possible to describe tasks in terms Cartesian pose as is common in robotics. It's primary disadvantage is that feedback is computed using estimated quantities that are a function of the system calibration parameters. Hence, in some situations, position-based control can become extremely sensitive to calibration error. Endpoint closed-loop systems are demonstrably less sensitive to calibration. However, particularly in stereo systems, small errors in computing the orientation of the cameras can still lead to reconstruction errors that impact the positioning accuracy of the system.

Pose-based methods for visual servoing seem to be the most generic approach to the problem, as they support arbitrary relative position with respect to the object. An often cited disadvantage of pose-based methods is the computation time required to solve the relative orientation problem. However recent results show that solutions can be computed in only a few milliseconds even using iteration [51] or Kalman filtering [53]. In general, given the rapid advances in microprocessor technology, computational considerations are becoming less of an issue in the design of visual servoing systems. Another disadvantage of pose-based approaches is the fact that they inherently depend on having an accurate model of the target object—a form of calibration. Hence, feature-based approaches tend to be more appropriate to tasks where there is no prior model of the geometry of the task, for example in teleoperation applications [59]. Generally speaking, since feature-based methods rely on less prior information (which may be in error), they can be expected to perform more robustly on comparable tasks.

Another approach to position-based visual servoing which has not been discussed here is to use an active 3D sensor. For example, active 3D sensors based on structured lighting are now compact and fast enough to use for visual servoing. If the sensor is small and mounted on the robot the depth and orientation information can be used directly for position-based visual servoing [60]–[62].

V. Image-Based Control

As described in Section III, in image-based visual servo control the error signal is defined directly in terms of image feature parameters (in contrast to position-based methods that define the error signal in the task space coordinates). Thus, we posit the following definition.

Definition 5.1: An *image-based visual servoing task* is represented by an image error function $\boldsymbol{e}$: $\mathcal{F} \rightarrow \Re^l$, where $l \leq k$ and k is the dimension of the image feature parameter space.

As described in Section II-E, the system may use either a fixed camera or an eye-in-hand configuration. In either case, motion of the manipulator causes changes to the image

observed by the vision system. Thus, the specification of an image-based visual servo task involves determining an appropriate error function $\boldsymbol{e}$, such that when the task is achieved, $\boldsymbol{e} = \mathbf{0}$. This can be done by directly using the projection equations (16), or via a "teach by showing" approach in which the robot is moved to a goal position and the corresponding image is used to compute a vector of desired image feature parameters, $\boldsymbol{f}_d$. If the task is defined with respect to a moving object, the error, $\boldsymbol{e}$, will be a function, not only of the pose of the end-effector, but also of the pose of the moving object.

Although the error, $\boldsymbol{e}$, is defined on the image parameter space, the manipulator control input is typically defined either in joint coordinates or in task space coordinates. Therefore, it is necessary to relate changes in the image feature parameters to changes in the position of the robot. The image Jacobian, introduced in Section V-A, captures these relationships. We present an example image Jacobian in Section V-B. In Section V-C, we describe methods that can be used to "invert" the image Jacobian, to derive the robot velocity that will produce the desired change in the image. Finally, in Sections V-D and V-E we describe how controllers can be designed for image-based systems.

A. The Image Jacobian

Let $\boldsymbol{r}$ represent coordinates of the end-effector in some parameterization of the task space $\mathcal{T}$ and $\dot{\boldsymbol{r}}$ represent the corresponding end-effector velocity (note, $\dot{\boldsymbol{r}}$ is a velocity screw, as defined in Section II-B). Let $\boldsymbol{f}$ represent a vector of image feature parameters and $\dot{\boldsymbol{f}}$ the corresponding vector of image feature parameter rates of change[5]. The image Jacobian, $\boldsymbol{J}_v$, is a linear transformation from the tangent space of $\mathcal{T}$ at $\boldsymbol{r}$ to the tangent space of $\mathcal{F}$ at $\boldsymbol{f}$. In particular

$$\dot{\boldsymbol{f}} = \boldsymbol{J}_v(\boldsymbol{r})\dot{\boldsymbol{r}} \tag{37}$$

where $\boldsymbol{J}_v \in \Re^{k\times m}$, and

$$J_v(\boldsymbol{r}) = \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{r}}\right] = \begin{bmatrix} \dfrac{\partial f_1(\boldsymbol{r})}{\partial r_1} & \cdots & \dfrac{\partial f_1(\boldsymbol{r})}{\partial r_m} \\ \vdots & & \vdots \\ \dfrac{\partial f_k(\boldsymbol{r})}{\partial r_1} & \cdots & \dfrac{\partial f_k(\boldsymbol{r})}{\partial r_m} \end{bmatrix}. \tag{38}$$

Recall that m is the dimension of the task space, $\mathcal{T}$. Thus the number of columns in the image Jacobian will vary depending on the task.

The image Jacobian was first introduced by Weiss *et al.* [21], who referred to it as the *feature sensitivity matrix*. It is also referred to as the *interaction matrix* [11] and the $\boldsymbol{B}$ matrix [16], [17]. Other applications of the image Jacobian include [10], [14], [15], [24].

The relationship given by (37) describes how image feature parameters change with respect to changing manipulator pose. In visual servoing we are interested in determining the manipulator velocity, $\dot{\boldsymbol{r}}$, required to achieve some desired value of $\dot{\boldsymbol{f}}$. This requires solving the system given by (37). We will discuss this problem in Section V-C, but first we present an example image Jacobian.

[5] If the image feature parameters are point coordinates these rates are image plane point velocities.

B. An Example Image Jacobian

Suppose that the end-effector is moving with angular velocity ${}^c\Omega_e = [\omega_x, \omega_y, \omega_z]$ and translational velocity ${}^c\boldsymbol{T}_e = [T_x, T_y, T_z]$ (as described in Section II-B) both with respect to the camera frame in a fixed camera system. Let $\boldsymbol{P}$ be a point rigidly attached to the end-effector. The velocity of the point $\boldsymbol{P}$, expressed relative to the camera frame, is given by

$${}^c\dot{\boldsymbol{P}} = {}^c\Omega_e \times {}^c\boldsymbol{P} + {}^c\boldsymbol{T}_e. \tag{39}$$

To simplify notation, let ${}^c\boldsymbol{P} = [x, y, z]^T$. Substituting the perspective projection equations (16) into (10) and (11), we can write the derivatives of the coordinates of ${}^c\boldsymbol{P}$ in terms of the image feature parameters u, v as

$$\dot{x} = z\omega_y - \frac{vz}{\lambda}\omega_z + T_x \tag{40}$$

$$\dot{y} = \frac{uz}{\lambda}\omega_z - z\omega_x + T_y \tag{41}$$

$$\dot{z} = \frac{z}{\lambda}(v\omega_x - u\omega_u) + T_z. \tag{42}$$

Now, let $\boldsymbol{f} = [u, v]^T$, as above and using the quotient rule,

$$\dot{u} = \lambda\frac{z\dot{x} - x\dot{z}}{z^2} \tag{43}$$

$$= \frac{\lambda}{z^2}\left\{z\left[z\omega_y - \frac{vz}{\lambda}\omega_z + T_x\right] - \frac{uz}{\lambda}\left[\frac{z}{\lambda}(v\omega_x - u\omega_y) + T_z\right]\right\} \tag{44}$$

$$= \frac{\lambda}{z}T_x - \frac{u}{z}T_z - \frac{uv}{\lambda}\omega_x + \frac{\lambda^2 + u^2}{\lambda}\omega_y - v\omega_z. \tag{45}$$

Similarly

$$\dot{v} = \frac{\lambda}{z}T_y - \frac{v}{z}T_z + \frac{-\lambda^2 - v^2}{\lambda}\omega_x + \frac{uv}{\lambda}\omega_y + u\omega_z. \tag{46}$$

Finally, we may rewrite these two equations in matrix form to obtain

$$\begin{bmatrix}\dot{u}\\ \dot{v}\end{bmatrix} = \begin{bmatrix} \dfrac{\lambda}{z} & 0 & -\dfrac{u}{z} & -\dfrac{uv}{\lambda} & \dfrac{\lambda^2+u^2}{\lambda} & -v \\ 0 & \dfrac{\lambda}{z} & -\dfrac{v}{z} & \dfrac{-\lambda^2 - v^2}{\lambda} & \dfrac{uv}{\lambda} & u \end{bmatrix} \begin{bmatrix} T_x \\ T_y \\ T_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{47}$$

which relates image-plane velocity of a point to the relative velocity of the point with respect to the camera. Alternative derivations for this example can be found in a number of references including [63], [64].

It is straightforward to extend this result to the general case of using $k/2$ image points for the visual control by simply stacking the Jacobians for each pair of image point coordinates; see (48), shown at the bottom of the next page.

Finally, note that the Jacobian matrices given in (47) and (48) are functions of z_i, the distance to the point being imaged. For a fixed camera system, when the target is the end-effector these z values can be computed using the forward kinematics of the robot and the camera calibration information. For an eye-in-hand system, determining z can be more difficult, and this problem is discussed further in Section V-F.

C. Using the Image Jacobian to Compute End-Effector Velocity

The results of the previous sections show how to relate robot end-effector motion to perceived motion in a camera image. However, visual servo control applications typically require the reverse—computation of $\dot{\boldsymbol{r}}$ given $\dot{\boldsymbol{f}}$ as input. There are three cases that must be considered: $k = m, k < m$, and $k > m$. We now discuss each of these.

When $k = m$ and $\boldsymbol{J}_v$ is nonsingular, $\boldsymbol{J}_v^{-1}$ exists. Therefore, in this case, $\dot{\boldsymbol{r}} = \boldsymbol{J}_v^{-1}\dot{\boldsymbol{f}}$. Such an approach has been used by Feddema [20], who also describes an automated approach to image feature selection in order to minimize the condition number of $\boldsymbol{J}_v$.

When $k \neq m$, $\boldsymbol{J}_v^{-1}$ does not exist. In this case, assuming that $\boldsymbol{J}_v$ is full rank (i.e., $\text{rank}(\boldsymbol{J}_v) = \min(k, m)$), we can compute a least squares solution, which, in general, is given by

$$\dot{\boldsymbol{r}} = \boldsymbol{J}_v^{+}\dot{\boldsymbol{f}} + (\boldsymbol{I} - \boldsymbol{J}_v^{+}\boldsymbol{J}_v)\boldsymbol{b} \tag{49}$$

where $\boldsymbol{J}_v^{+}$ is a suitable pseudoinverse for $\boldsymbol{J}_v$, and $\boldsymbol{b}$ is an arbitrary vector of the appropriate dimension. The least squares solution gives a value for $\dot{\boldsymbol{r}}$ that minimizes the norm $\|\dot{\boldsymbol{f}} - \boldsymbol{J}_v\dot{\boldsymbol{r}}\|$.

We first consider the case $k > m$, that is, there are more feature parameters than task degrees of freedom. By the implicit function theorem [65], if, in some neighborhood of $\boldsymbol{r}, m \leq k$ and $\text{rank}(\boldsymbol{J}_v) = m$ (i.e., $\boldsymbol{J}_v$ is full rank), we can express the coordinates $f_{m+1} \cdots f_k$ as smooth functions of $f_1 \cdots f_m$. From this, we deduce that there are $k - m$ redundant visual features. Typically, this will result in a set of inconsistent equations (since the k visual features will be obtained from a computer vision system and are likely to be noisy). In this case, the appropriate pseudoinverse is given by

$$\boldsymbol{J}_v^{+} = (\boldsymbol{J}_v^T\boldsymbol{J}_v)^{-1}\boldsymbol{J}_v^T. \tag{50}$$

Here, we have $(\boldsymbol{I} - \boldsymbol{J}_v^{+}\boldsymbol{J}_v) = 0$ (the rank of the null space of $\boldsymbol{J}_v$ is 0, since the dimension of the column space of $\boldsymbol{J}_v, m$, equals $\text{rank}(\boldsymbol{J}_v)$). Therefore, the solution can be written more concisely as

$$\dot{\boldsymbol{r}} = \boldsymbol{J}_v^{+}\dot{\boldsymbol{f}}. \tag{51}$$

Such approaches have been used by Hashimoto [15] and Jang [66].

When $k < m$, the system is under-constrained. In the visual servo application, this implies that we are not observing enough features to uniquely determine the object motion $\dot{\boldsymbol{r}}$, i.e., there are certain components of the object motion that can not be observed. In this case, the appropriate pseudoinverse is given by

$$\boldsymbol{J}_v^{+} = \boldsymbol{J}_v^T(\boldsymbol{J}_v\boldsymbol{J}_v^T)^{-1}. \tag{52}$$

In general, for $k < m$, $(\boldsymbol{I} - \boldsymbol{J}_v^{+}\boldsymbol{J}_v) \neq 0$, and all vectors of the form $(\boldsymbol{I} - \boldsymbol{J}_v^{+}\boldsymbol{J}_v)\boldsymbol{b}$ lie in the null space of $\boldsymbol{J}_v$ and correspond to those components of the object velocity that are unobservable. In this case, the solution is given by (49). For example, as shown in [64], the null space of the image Jacobian given in (47), is spanned by the four vectors

$$\begin{bmatrix} u \\ v \\ \lambda \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ u \\ v \\ \lambda \end{bmatrix} \begin{bmatrix} uvz \\ -(u^2 + \lambda^2)z \\ \lambda vz \\ -\lambda^2 \\ 0 \\ u\lambda \end{bmatrix} \begin{bmatrix} \lambda(u^2 + v^2 + \lambda^2)z \\ 0 \\ -u(u^2 + v^2 + \lambda^2)z \\ uv\lambda \\ -(u^2 + \lambda^2)z \\ u\lambda^2 \end{bmatrix}. \tag{53}$$

In some instances, there is a physical interpretation for the vectors that span the null space of the image Jacobian. For example, the vector $[u, v, \lambda, 0, 0, 0]^T$ reflects that the motion of a point along a projection ray cannot be observed. The vector $[0, 0, 0, u, v, \lambda]^T$ reflects the fact that rotation of a point on a projection ray about that projection ray cannot be observed. Unfortunately, not all basis vectors for the null space have such an obvious physical interpretation. The null space of the image Jacobian plays a significant role in hybrid methods, in which some degrees of freedom are controlled using visual servo, while the remaining degrees of freedom are controlled using some other modality [14].

D. Resolved-Rate Methods

The earliest approaches to image-based visual servo control [10], [21] were based on resolved-rate motion control [30], which we will briefly describe here. Suppose that the goal of a particular task is to reach a desired image feature parameter vector, $\boldsymbol{f}_d$. If the control input is defined as in Section IV to be an end-effector velocity, then we have $\boldsymbol{u} = \dot{\boldsymbol{r}}$, and assuming for the moment that the image Jacobian is square and nonsingular,

$$\boldsymbol{u} = \boldsymbol{J}_v^{-1}(\boldsymbol{r})\dot{\boldsymbol{f}}. \tag{54}$$

If we define the error function as $\boldsymbol{e}(\boldsymbol{f}) = \boldsymbol{f}_d - \boldsymbol{f}$, a simple proportional control law is given by

$$\boldsymbol{u} = \boldsymbol{K}\boldsymbol{J}_v^{-1}(\boldsymbol{r})\boldsymbol{e}(\boldsymbol{f}) \tag{55}$$

$$\begin{bmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \vdots \\ \dot{u}_{k/2} \\ \dot{v}_{k/2} \end{bmatrix} = \begin{bmatrix} \frac{\lambda}{z_1} & 0 & -\frac{u_1}{z_1} & -\frac{u_1 v_1}{\lambda} & \frac{\lambda^2 + u_1^2}{\lambda} & -v_1 \\ 0 & \frac{\lambda}{z_1} & -\frac{v_1}{z_1} & \frac{-\lambda^2 - v_1^2}{\lambda} & \frac{u_1 v_1}{\lambda} & u_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\lambda}{z_{k/2}} & 0 & -\frac{u_{k/2}}{z_{k/2}} & -\frac{u_{k/2} v_{k/2}}{\lambda} & \frac{\lambda^2 + u_{k/2}^2}{\lambda} & -v_{k/2} \\ 0 & \frac{\lambda}{z_{k/2}} & -\frac{v_{k/2}}{z_{k/2}} & \frac{-\lambda^2 - v_{k/2}^2}{\lambda} & \frac{u_{k/2} v_{k/2}}{\lambda} & u_{k/2} \end{bmatrix} \begin{bmatrix} T_x \\ T_y \\ T_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \tag{48}$$

where $\boldsymbol{K}$ is a constant gain matrix of the appropriate dimension. For the case of a nonsquare image Jacobian, the techniques described in Section V-C would be used to compute for $\boldsymbol{u}$. Similar results have been presented in [14], [15]. More advanced techniques based on optimal control are discussed in [16].

E. Example Servoing Tasks

In this section, we revisit some of the problems introduced in Section IV-A and describe image-based solutions for these problems. In all cases, we assume two fixed cameras are observing the scene.

1) Point to Point Positioning: Consider the task of bringing some point $\boldsymbol{P}$ on the manipulator to a desired stationing point $\boldsymbol{S}$. The kinematic error function was given in (20). If two cameras are viewing the scene, a necessary and sufficient condition for $\boldsymbol{P}$ and $\boldsymbol{S}$ to coincide in the workspace is that the projections of $\boldsymbol{P}$ and $\boldsymbol{S}$ coincide in each image.

If we let $[u^l, v^l]^T$ and $[u^r, v^r]^T$ be the image coordinates for the projection of $\boldsymbol{P}$ in the left and right images, respectively, then we may take $\boldsymbol{f} = [u^l, v^l, u^r, v^r]^T$. If we let $\mathcal{T} = \Re^3$, then in (19), $\boldsymbol{F}$ is a mapping from $\mathcal{T}$ to $\Re^4$.

Let the projection of $\boldsymbol{S}$ have coordinates $[u_s^l, v_s^l]$ and $[u_s^r, v_s^r]$ in the left and right images. We then define the desired feature vector to be $\boldsymbol{f}_d = [u_s^l, v_s^l, u_s^r, v_s^r]^T$, yielding

$$\boldsymbol{e}_{pp}(\boldsymbol{f}) = \boldsymbol{f} - \boldsymbol{f}_d. \tag{56}$$

The image Jacobian for this problem can be constructed by "stacking" (47) for each camera. Note, however, that a coordinate transformation must be used for each camera in order to relate the end-effector velocity screw in camera coordinates to the robot reference frame.

Unfortunately, the resulting Jacobian matrix cannot be inverted as it is a matrix with four rows and six columns which is of rank three. This is a reflection of the fact that although two cameras provide four measurements, the point observed has only three degrees of freedom. Hence, one measurement value is redundant, or equivalently the observations are constrained to lie on a three-dimensional subspace of four-dimensional measurement space. The constraint defining this subspace is known as the *epipolar constraint* in the vision literature [67].

There are a variety of methods for dealing with this problem. The simplest is to note that most stereo camera systems are arranged so that the camera x (horizontal) axes are roughly co-planar. In this case, the redundant information is largely concentrated in the y (vertical) coordinates, and so one can be discarded. Doing so removes a row from the Jacobian, and the resulting matrix has a well-defined inverse.

2) Point to Line Positioning: Consider again the task in which some point $\boldsymbol{P}$ on the manipulator end-effector is to be brought to the *line joining* two fixed points $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ in the world. The kinematic error function is given by (29).

If two cameras are viewing the workspace, it can be shown that a necessary and sufficient condition for $\boldsymbol{P}$ to be colinear with the line joining $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ is that the projection of $\boldsymbol{P}$ be colinear with the projections of the points $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ in *both* images (for nondegenerate camera configurations). The proof proceeds as follows. The origin of the coordinate frame for the left camera, together with the projections of $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ onto the left image forms a plane. Likewise, the origin of the coordinate frame for the right camera, together with the projections of $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ onto the right image forms a plane. The intersection of these two planes is exactly the line joining $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ in the workspace. When $\boldsymbol{P}$ lies on this line, it must lie simultaneously in both of these planes, and therefore, must be colinear with the the projections of the points $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ in both images.

We now turn to conditions that determine when the projection of $\boldsymbol{P}$ is colinear with the projections of the points $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, and will use the knowledge that three vectors are coplanar if and only if their scalar triple product is zero. For the left image, let the projection of $\boldsymbol{S}_1$ have image coordinates $[u_1^l, v_1^l]$, the projection of $\boldsymbol{S}_2$ have image coordinates $[u_2^l, v_2^l]$, and the projection of $\boldsymbol{P}$ have image coordinates $[u^l, v^l]$. If the three vectors from the origin of the left camera to these image points are coplanar, then the three image points are colinear. Thus, we construct the scalar triple product

$$e_{pl}^l([u^l, v^l]^T) = \left(\begin{bmatrix} u_1^l \\ v_1^l \\ \lambda \end{bmatrix} \times \begin{bmatrix} u_2^l \\ v_2^l \\ \lambda \end{bmatrix} \right) \cdot \begin{bmatrix} u^l \\ v^l \\ \lambda \end{bmatrix} \tag{57}$$

and proceeding in a similar fashion for the right image, derive e_{pl}^r from which we construct the error function

$$\boldsymbol{e}(\boldsymbol{f}) = \begin{bmatrix} e_{pl}^l([u^l, v^l]^T) \\ e_{pl}^r([u^r, v^r]^T) \end{bmatrix} \tag{58}$$

where $\boldsymbol{f} = [u^l, v^l, u^r, v^r]^T$. It is important to note that this error is a linear projection of the image coordinates of the point $\boldsymbol{P}$, and hence the Jacobian is also a linear transformation of the image Jacobian for $\boldsymbol{P}$. To make this explicit, let $\boldsymbol{J}_p^l(\boldsymbol{P})$ denote the image Jacobian for $\boldsymbol{P}$ in the left camera. Then it follows that the image Jacobian for e_{pl}^l is

$$\begin{aligned} &\boldsymbol{J}_{pl}^l(u_1^l, v_1^l, u_2^l, v_2, u^l, v^l) \\ &\quad = \left(\begin{bmatrix} u_1^l \\ v_1^l \\ \lambda \end{bmatrix} \times \begin{bmatrix} u_2^l \\ v_2^l \\ \lambda \end{bmatrix} \right)^T \begin{bmatrix} \boldsymbol{J}_p^l(\boldsymbol{P}) \\ \boldsymbol{0} \end{bmatrix}. \end{aligned} \tag{59}$$

The derivation of the image Jacobian in the right camera is similar. The full Jacobian is the "stack" consisting of $\boldsymbol{J}_{pl}^l$ and $\boldsymbol{J}_{pl}^r$ multiplied with a coordinate transformation to relate the end-effector velocity screw in robot coordinates to the equivalent motion in the camera coordinate frame. Note that given a second point on the end-effector, a four degree of freedom positioning operation can be defined by simply stacking the two "point-to-line" errors and their image Jacobians. Likewise, choosing yet another point in the world and on the manipulator, and setting up an additional independent "point-to-line" problem yields a rigid six degree of freedom positioning problem.

F. Discussion

It is interesting to note that image-based solutions to the point-to-line problem discussed above perform with an accuracy that is independent of calibration. This follows from

the fact that by construction, when the image error function is zero, the kinematic error must also be zero. Even if the hand-eye system is miscalibrated, if the feedback system is asymptotically stable, the image error will tend to zero, and hence so will the kinematic error. This is not the case with the position-based system described in Section IV [68]. Thus, one of the chief advantages to image-based control over position-based control is that the positioning accuracy of the system is less sensitive to camera calibration errors.

There are also often computational advantages to image-based control, particularly in ECL configurations. For example, a position-based relative pose solution for an ECL single-camera system must perform two nonlinear least squares optimizations in order to compute the error function. The comparable image-based system must only compute a simple image error function, an inverse Jacobian solution, and possibly a single position or pose calculation to parameterize the Jacobian. In practice, as described in Section V-B, the unknown parameter for Jacobian calculation is distance from the camera. Some recent papers present adaptive approaches for estimating this depth value [16], or develop feedback methods which do not use depth in the feedback formulation [69].

One disadvantage of image-based methods compared to position-based methods is the presence of singularities in the feature mapping function which reflect themselves as unstable points in the inverse Jacobian control law. These instabilities are often less prevalent in the equivalent position-based scheme. Returning again to the point-to-line example, the Jacobian calculation becomes singular when the two stationing points are coplanar with the optical centers of both cameras. In this configuration, rotations and translations of the setpoints in the plane are not observable. This singular configuration does not exist for the position-based solution.

In the above discussion we have referred to $\boldsymbol{f}_d$ as the desired feature parameter vector, and implied that it is a constant. If it is a constant then the robot will move to the desired pose with respect to the target. If the target is moving the system will endeavor to track the target and maintain relative pose, but the tracking performance will be a function of the system dynamics, as discussed below in Section VII. However many tasks can be described in terms of the motion of image features, for instance by aligning visual cues within the scene. Jang *et al.* [66] describe a generalized approach to servoing on image features, with trajectories specified in feature space which results in trajectories (tasks) that are independent of target geometry. Feddema [10] also uses a feature space trajectory generator to interpolate feature parameter values due to the low update rate of the vision system used. Skaar *et al.* [18] describe the example of a 1DOF robot catching a ball by observing visual cues such as the ball, the arm's pivot point, and another point on the arm. The interception task can then be specified, even if the relationship between camera and arm is not known a priori.

VI. Image Feature Extraction and Tracking

Irrespective of the control approach used, a vision system is required to extract the information needed to perform the servoing task. Hence, visual servoing pre-supposes the solution to a set of potentially difficult static and dynamic vision problems. To this end many reported implementations contrive the vision problem to be simple: *e.g.* painting objects white, using artificial targets, and so forth [10], [14], [37], [70]. Other authors use extremely task-specific clues: *e.g.* Allen [36] uses motion detection for locating a moving object to be grasped, and a fruit picking system looks for the characteristic fruit color. A review of tracking approaches used by researchers in this field is given in [3].

In less structured situations, vision has typically relied on the extraction of sharp contrast changes, referred to as "corners" or "edges", to indicate the presence of object boundaries or surface markings in an image. Processing the entire image to extract these features necessitates the use of extremely high-speed hardware in order to work with a sequence of images at camera rate. However not all pixels in the image are of interest, and computation time can be greatly reduced if only a small region around each image feature is processed. Thus, a promising technique for making vision cheap and tractable is to use *window-based* tracking techniques [4], [37], [71]. Window-based methods have several advantages, among them: computational simplicity, little requirement for special hardware, and easy reconfiguration for different applications. We note, however, that initial positioning of each window typically presupposes an automated or human-supplied solution to a potentially complex vision problem.

This section describes a window-based approach to tracking features in an image. The methods are capable of tracking a number of point or edge features at frame rate on a workstation computer and require a framestore, no specialized image processing hardware, and have been incorporated into a publicly available software "toolkit" [4]. A discussion of methods which use specialized hardware combined with temporal and geometric constraints can be found in [67]. The remainder of this section is organized as follows. Section VI-A describes how window-based methods can be used to implement fast detection of edge segments, a common low-level primitive for vision applications. Section VI-B describe an approach based on temporally correlating image regions over time. VI-C describes some general issues related to the use of temporal and geometric constraints, and Section VI-D briefly summarizes some of the issues surrounding the choice of a feature extraction method for tracking.

A. Feature Based Methods

In this section, we illustrate how window-based processing techniques can be used to perform fast detection of isolated straight edge segments of fixed length. Edge segments are intrinsic to applications where man-made parts contain corners or other patterns formed from physical edges.

Images are comprised of pixels organized into a two-dimensional coordinate system. We adopt the notation $I(\boldsymbol{x}, t)$ to denote the pixel at location $\boldsymbol{x} = [u, v]^T$ in an image captured at time t. A window can be thought of as a two-dimensional array of pixels related to a larger image by an invertible mapping from window coordinates to image coordinates. We

consider rigid transformations consisting of a translation vector $\boldsymbol{c} = [x, y]^T$ and a rotation θ. A pixel value at $\boldsymbol{x} = [u, v]^T$ in window coordinates is related to the larger image by

$$\mathcal{W}(\boldsymbol{x}; \boldsymbol{c}, \theta, t) = I(\boldsymbol{c} + \boldsymbol{R}(\theta)\boldsymbol{x}, t) \tag{60}$$

where $\boldsymbol{R}$ is a two dimensional rotation matrix. We adopt the conventions that $\boldsymbol{x} = \boldsymbol{0}$ is the center of the window, and the set $\mathcal{X}$ represents the set of all values of $\boldsymbol{x}$.

Window-based tracking algorithms typically operate in two stages. In the first stage, one or more windows are *acquired* using a nominal set of window parameters. The pixel values for all $\boldsymbol{x} \in \mathcal{X}$ are copied into a two-dimensional array that is subsequently treated as a rectangular image. Such acquisitions can be implemented extremely efficiently using line-drawing and region-fill algorithms commonly developed for graphics applications [72]. In the second stage, the windows are processed to locate image features and from their parameters a new set of window parameters, θ and $\boldsymbol{c}$, are computed. These parameters may be modified using external geometric constraints or temporal prediction, and the cycle repeats.

We consider an edge segment to be characterized by three parameters in the image plane: the u and v coordinates of the center of the segment, and the orientation of the segment relative to the image plane coordinate system. These values correspond directly to the parameters of the acquisition window used for edge detection. Let us first assume we have correct prior values $\boldsymbol{c}^- = (u^-, v^-)$ and θ^- for an edge segment. A window, $\mathcal{W}^-(\boldsymbol{x}) = \mathcal{W}(\boldsymbol{x}; \boldsymbol{c}^-, \theta^-, t)$, extracted with these parameters would then have a vertical edge segment within it.

Isolated step edges can be localized by determining the location of the maximum of the first derivative of the signal [64], [67], [73]. Let e be a 1-dimensional edge detection kernel arranged as a single row. The convolution $\mathcal{W}_1(\boldsymbol{x}) = (\mathcal{W}^- * e)(\boldsymbol{x})$ will have a response curve in each row which peaks at the location of the edge. Summing each column of $\mathcal{W}_1$ superimposes the peaks and yields a one-dimensional response curve. If the estimated orientation, θ^-, was correct, the maximum of this response curve determines the offset of the edge in window coordinates. By interpolating the response curve about the maximum value, sub-pixel localization of the edge can be achieved. Here, e is taken to be a 1-dimensional Prewitt operator [64] which, although not optimal from a signal processing point of view, is extremely fast to execute on simple hardware.

If the θ^- was incorrect, the response curves in $\mathcal{W}_1$ will deviate slightly from one another and the superposition of these curves will form a lower and less sharp aggregate curve. Thus, maximizing the maximum value of the aggregate response curve is a way to determine edge orientation. This can be approximated by performing the detection operation on windows acquired at θ^- as well as two bracketing angles $\theta^- \pm \alpha$ and performing quadratic interpolation on the maxima of the corresponding aggregate response curves. Computing the three oriented edge detectors is particularly simple if the range of angles is small. In this case, a single window is processed with the initial convolution yielding $\mathcal{W}_1$. Three aggregate response curves are computed by summing along the columns of $\mathcal{W}_1$ and along diagonals corresponding to angles of $\pm\alpha$. The maxima of all three curves are located and interpolated to yield edge orientation and position. Thus, for the price of one window acquisition, one complete 1-dimensional convolution, and three column sums, the vertical offset δo and the orientation offset $\delta\theta$ can be computed. Once these two values are determined, the state variables of the acquisition window are updated as

$$\begin{aligned} \theta^+ &= \theta^- + \delta\theta \\ u^+ &= u^- - \delta o \sin(\theta^+) \\ v^+ &= v^- + \delta o \cos(\theta^+). \end{aligned}$$

An implementation of this method [4] has shown that localizing a 20 pixel long edge using a Prewitt-style mask 15 pixels wide searching ± 10 pixels and ± 15 degrees takes 1.5 ms on a Sun Sparc II workstation. At this rate, 22 edge segments can be tracked simultaneously at 30 Hz, the video frame rate used. Longer edges can be tracked at comparable speeds by sub-sampling along the edge.

Clearly, this edge-detection scheme is susceptible to mis-tracking caused by background or foreground occluding edges. Large acquisition windows increase the range of motions that can be tracked, but reduce the tracking speed and increase the likelihood that a distracting edge will disrupt tracking. Likewise, large orientation brackets reduce the accuracy of the estimated orientation, and make it more susceptible to edges that are not closely oriented to the underlying edge.

There are several ways of increasing the robustness of edge tracking. One is to include some type of additional information about the edges being tracked such as the sign or absolute value of the edge response. For more complex edge-based detection, collections of such oriented edge detectors can be combined to verify the location and position of the entire feature. Some general ideas in this direction are discussed in Section VI-C.

B. Area-Based Methods

Edge-based methods tend to work well in environments in which man-made objects are to be tracked. If, however, the desired feature is a specific pattern, then tracking can be based on matching the appearance of the feature (in terms of its spatial pattern of gray-values) in a series of images, and exploiting its *temporal consistency*—the observation that the appearance of small region in an image sequence changes little. Such techniques are well described in image registration literature and have been applied to other computer vision problems such as stereo matching and optical flow.

Consider only windows that differ in the location of their center. We assume some reference window was acquired at time t at location $\boldsymbol{c}$. Some small time interval, τ, later, a candidate window of the same size is acquired at location $\boldsymbol{c} + \boldsymbol{d}$. The correspondence between these two images is given by some similarity measure

$$O(\boldsymbol{d}) = \sum_{\boldsymbol{x} \in \mathcal{X}} f((\mathcal{W}(\boldsymbol{x}; \boldsymbol{c}, t)) - \mathcal{W}(\boldsymbol{x}; \boldsymbol{c} + \boldsymbol{d}, t + \tau))w(\boldsymbol{x}), \quad \tau > 0 \tag{61}$$

where $w(\cdot)$ is a weighting function over the image region and $f(\cdot)$ is a scalar function. Commonly used functions include $f(x) = |x|$ for sum of absolute values (SAD) and $f(x) = x^2$ for sum of squared differences (SSD).

The aim is to find the displacement, $\boldsymbol{d}$, that minimizes $O(\boldsymbol{d})$. Since images are inherently discrete, a natural solution is to select a finite range of values $\mathcal{D}$ and compute

$$\hat{\boldsymbol{d}} = \min_{\boldsymbol{d} \in \mathcal{D}} O(\boldsymbol{d}).$$

The advantage of a complete discrete search is that the true minimum over the search region is guaranteed to be found. However, the larger the area covered, the greater the computational burden. This burden can be reduced by performing the optimization starting at low resolution and proceeding to higher resolution, and by ordering the candidates in $\mathcal{D}$ from most to least likely and terminating the search once a candidate with an acceptably low SSD value is found [17]. Once the discrete minimum is found, the location can be refined to sub-pixel accuracy by interpolation of the SSD values about the minimum. Even with these improvements, [17] reports that a special signal processor is required to attain frame-rate performance.

It is also possible to solve (61) using continuous optimization methods [4] [74]–[76]. The solution begins by expanding $\mathcal{W}(\boldsymbol{x}; \boldsymbol{c}, t)$ in a Taylor series about $(\boldsymbol{c}, t)$ yielding

$$\begin{aligned}\mathcal{W}(\boldsymbol{x}; \boldsymbol{c} + \boldsymbol{d}, t + \tau) \\ \approx \mathcal{W}(\boldsymbol{x}; \boldsymbol{c}, t) + \mathcal{W}_x(\boldsymbol{x})\, dx + \mathcal{W}_y(\boldsymbol{x})\, dy + \mathcal{W}_t(\boldsymbol{x})\tau\end{aligned}$$

where $\mathcal{W}_x, \mathcal{W}_y$ and $\mathcal{W}_t$ are respectively the horizontal and vertical spatial, and temporal derivatives of the image computed using convolution as follows:

$$\begin{aligned}\mathcal{W}_x(\boldsymbol{x}) &= \left(\mathcal{W} * \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}\right)(\boldsymbol{x}) \\ \mathcal{W}_y(\boldsymbol{x}) &= \left(\mathcal{W} * \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}\right)(\boldsymbol{x}) \\ \mathcal{W}_t(\boldsymbol{x}) &= \left((\mathcal{W}(\cdot; \boldsymbol{c}, t + \tau) - \mathcal{W}_s(\cdot; \boldsymbol{c}, t)) * \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right)(\boldsymbol{x}).\end{aligned}$$

Substituting into (61) yields

$$O(\boldsymbol{d}) \approx \sum_{\boldsymbol{x} \in \mathcal{X}} (\mathcal{W}_x(\boldsymbol{x})\, dx + \mathcal{W}_y(\boldsymbol{x})\, dy + \mathcal{W}_t(\boldsymbol{x})\tau)^2 w(\boldsymbol{x}). \quad (62)$$

Define

$$\boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix} \mathcal{W}_x(\boldsymbol{x})\sqrt{w(\boldsymbol{x})} \\ \mathcal{W}_y(\boldsymbol{x})\sqrt{w(\boldsymbol{x})} \end{bmatrix} \quad \text{and} \quad h(\boldsymbol{x}) = \mathcal{W}_t(\boldsymbol{x})\sqrt{w(\boldsymbol{x})}.$$

Expression (62) can now be written more concisely as

$$O(\boldsymbol{d}) \approx \sum_{\boldsymbol{x} \in \mathcal{X}} (\boldsymbol{g}(\boldsymbol{x}) \cdot \boldsymbol{d} + h(\boldsymbol{x})\tau)^2. \quad (63)$$

Notice O is now a quadratic function of $\boldsymbol{d}$. Computing the derivatives of O with respect to the components of $\boldsymbol{d}$, setting the result equal to zero, and rearranging yields a linear system of equations:

$$\left[\sum_{\boldsymbol{x} \in \mathcal{X}} (\boldsymbol{g}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x})^T)\right] \boldsymbol{d} = \sum_{\boldsymbol{x} \in \mathcal{X}} h(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x}). \quad (64)$$

Solving for $\boldsymbol{d}$ yields an estimate, $\hat{\boldsymbol{d}}$ of the offset that would cause the two windows to have maximum correlation. We then compute $\boldsymbol{c}^+ = \boldsymbol{c}^- + \hat{\boldsymbol{d}}$ yielding the updated window location for the next tracking cycle. This is effectively a proportional control algorithm for the "servoing" the location of an acquisition to maintain the best match with the reference window over time.

In practice this method will only work for small motions (it is mathematically correct only for a fraction of a pixel). This problem can be alleviated by first performing the optimization at low levels of resolution, and using the result as a seed for computing the offset at higher levels of resolution. For example, reducing the resolution by a factor of two by summing groups of four neighboring pixels doubles the maximum displacement between two images. It also speeds up the computations since fewer operations are needed to compute $\hat{\boldsymbol{d}}$ for the smaller low-resolution image.

Another drawback of this method is the fact that it relies on an exact match of the gray values—changes in contrast or brightness can bias the results and lead to mistracking. Thus, it is common to normalize the images to have zero mean and consistent variance. With these modifications, it is easy to show that solving (64) is equivalent to maximizing the correlation between the two windows [74].

Continuous optimization has two principle advantages over discrete optimization. Firstly, a single updating cycle is usually faster to compute. For example, (64) can be computed and solved in less than 5 ms on a Sparc II computer [4]. Secondly, it is easy to incorporate other window parameters such as rotation and scaling into the system without greatly increasing the computation time [4], [76]. Thus, SSD methods can be used to perform template matching as well as tracking of image regions.

C. Feature Prediction

Window-based tracking implicitly assumes that the inter-frame motions of the tracked feature do not exceed the size of search window, or, in the case of continuous optimization, a few pixels from the expected location of the image region. In the simplest case, the previous location of the image feature can be used as a predictor of its current location. Unfortunately, as feature velocity increases the search window must be enlarged which adversely affects computation time.

The robustness and speed of tracking can be significantly increased with knowledge about the motion of the observed features, which may be due to the camera and/or target moving. For example, given knowledge of the image feature location $\boldsymbol{x}_t$ at time t, Jacobian $\boldsymbol{J}_v$, the end-effector velocity $\boldsymbol{u}_t$, and the inter-frame time τ, the expected location of the search windows can be computed, assuming no target motion, by the prediction

$$\boldsymbol{f}_{t+\tau} = \boldsymbol{f}_t + \tau J_v \boldsymbol{u}_t.$$

Likewise, if the dynamics of a moving object are known, then it is possible to use this information to enhance tracking performance. For example, Rizzi [37] describes the use of a Newtonian flight dynamics model to make it possible to track a

ping-pong ball during flight. Predictors based on $\alpha-\beta$ tracking filters and Kalman filters have also been used [36], [53], [67].

D. Discussion

Prior to executing or planning visually controlled motions, a specific set of visual features must be chosen. Discussion of the issues related to feature selection for visual servo control applications can be found in [20], [21]. The "right" image feature tracking method to use is extremely application dependent. For example, if the goal is to track a single special pattern or surface marking that is approximately planar and moving at slow to moderate speeds, then area-based tracking is appropriate. It does not require special image structure (*e.g.* straight lines), is robust to large set of image distortions, and for small motions can be implemented to run at frame rates.

In comparison to the edge detection methods described above, area-based tracking is sensitive to occlusions and background changes (if the template includes any background pixels). Thus, if a task requires tracking several occluding contours of an object with a changing background, edge-based methods are clearly faster and more robust.

In many realistic cases, neither of these approaches by themselves yields the robustness and performance desired. For example, tracking occluding edges in an extremely cluttered environment is sure to distract edge tracking as "better" edges invade the search window, while the changing background would ruin the SSD match for the region. Such situations call for the use of more global task constraints (*e.g.* the geometry of several edges), more global tracking (*e.g.* extended contours or snakes [77]), or improved or specialized detection methods.

To illustrate these tradeoffs, suppose a visual servoing task relies on tracking the image of a circular opening over time. In general, the opening will project to an ellipse in the camera. There are several candidate algorithms for detecting this ellipse and recovering its parameters:

1) If the contrast between the interior of the opening and area around it is high, then binary thresholding followed by a calculation of the first and second central moments can be used to localize the feature [37].
2) If the ambient illumination changes greatly over time, but the brightness of the opening and the brightness of the surrounding region are roughly constant, a circular template could be localized using SSD methods augmented with brightness and contrast parameters. In this case, (61) must also include parameters for scaling and aspect ratio [4].
3) The opening could be selected in an initial image, and subsequently located using SSD methods. This differs from the previous method in that this calculation does not compute the center of the opening, only its correlation with the starting image. Although useful for servoing a camera to maintain the opening within the field of view, this approach is probably not useful for manipulation tasks that need to attain a position relative to the center of the opening.
4) If the contrast and background are changing, the opening could be tracked by performing edge detection and fitting an ellipse to the edge locations. In particular, short edge segments could be located using the techniques described in Section VI-A. Once the segments have been fit to an ellipse, the orientation and location of the segments would be adjusted for the subsequent tracking cycle using the geometry of the ellipse.

During task execution, other problems arise. The two most common problems are occlusion of features and visual singularities. Solutions to the former include intelligent observers that note the disappearance of features and continue to predict their locations based on previously observed motion [37], or redundant feature specifications that can perform even with some loss of information. Solution to the latter require some combination of intelligent path planning and/or intelligent acquisition and focus-of-attention to maintain the controllability of the system.

It is probably safe to say that fast and robust image processing presents the greatest challenge to general-purpose hand-eye coordination. As an effort to help overcome this obstacle, the methods described above and other related methods have been incorporated into a publicly available software "toolkit." The interested reader is referred to [4] for details.

VII. Discussion

This paper has presented a tutorial introduction to robotic visual servo control, focusing on the relevant fundamentals of coordinate transformations, image formation, feedback algorithms, and visual tracking. In the interests of space and clarity, we have concentrated on presenting methods that are well-represented in the literature, and that can be solved using relatively straightforward techniques. The reader interested in a broader overview of the field or interested in acquiring more detail on a particular area is invited to consult the references we have provided. Another goal has been to establish a consistent nomenclature and to summarize important results here using that notation.

Many aspects of the more general problem of vision-based control of motion have necessarily been omitted or abbreviated to a great degree. One important issue is the choice between using an image-based or position-based system. Many systems based on image-based and position-based architectures have been demonstrated, and the computational costs of the two approaches seem to be comparable and are easily within the capability of modern computers. In many cases the motion of a target, for example an object on a conveyer, is most naturally expressed in a Cartesian reference frame. For this reason, most systems dealing with moving objects ([36], [37]) have used position-based methods. Although there has been recent progress in understanding image plane dynamics [22], the design of stable, robust image-based servoing systems for capturing moving objects has not been fully explored.

In general, the accuracy of image-based methods for static positioning is less sensitive to calibration than comparable position-based methods, however image-based methods require online computation of the image Jacobian. Unfortunately, this quantity inherently depends on the distance from the camera to the target which, particularly in a monocular

system, is difficult to compute. Many systems utilize a constant image Jacobian, which is computationally efficient, but valid only over a small region of the task space[6]. Other systems have resorted to performing a partial pose estimation [10], adaptive depth estimation [16], or image Jacobian estimation [78]. However, both add significantly to the complexity of the system design as well as introducing an additional computational load.

This issue is further complicated when dynamics are introduced into the problem. Even when the target object is not moving, it is important to realize that a visual servo system is a closed-loop discrete-time dynamical system. The sampling rate in such a system is limited by the frame rate of the camera, though many reported systems operate at a sub-multiple of the camera frame rate due to limited computational ability. Negative feedback is applied to a *plant* that generally includes a time delays due to charge integration time within the camera, serial pixel transport from the camera to the vision system, and computation time for feature parameter extraction. In addition most reported visual servo systems employ a relatively low bandwidth communications link between the vision system and the robot controller, which introduces further latency. Some robot controllers operate with a sample interval that is not related to the sample rate of the vision system, and this introduces still further delay. A good example of this is the common Unimate Puma robot whose position loops operate at a sample interval of 14 or 28 ms while vision systems operate at sample intervals of 33 or 40 ms for RS 170 or CCIR video respectively [29]. It is well known that a feedback system including delay will become unstable as the loop gain is increased. Many visual closed-loop systems are tuned empirically, increasing the loop gain until overshoot or oscillation becomes intolerable.

Simple proportional controllers are commonly used and can be shown to drive the steady state error to zero. However this implies nothing about performance when tracking a moving object, which will typically exhibit pronounced image plane error and tracking lag. If the target motion is constant then prediction (based upon some assumption of target motion) can be used to compensate for the latency, and predictors based on autoregressive models, Kalman filters, $\alpha - \beta$ and $\alpha - \beta - \gamma$ tracking filters have been demonstrated for visual servoing. However when combined with a low sample rate predictors can result in poor disturbance rejection and long reaction time to unmodeled target motion. In order for a visual-servo system to provide good tracking performance for moving targets considerable attention must be paid to modeling the dynamics of the robot, the target, and vision system and designing an appropriate control system. Other issues for consideration include whether or not the vision system should "close the loop" around robot axes which can be position, velocity or torque controlled. A detailed discussion of these dynamic issues in visual servo systems is given by Corke [29], [79].

In addition to these "low-level" considerations, other issues that merit consideration are vision-based path planning, and visual recognition. In the case of the former, although path-planning is a well-established discipline, the idea of combining image space feature path-planning with visual feedback has not been adequately explored. For a simple example of visual servoing with obstacle avoidance, see [78]. Visual recognition or interpretation is also important for any visual servoing system that is to operate without constant human intervention. These are but two of the many issues that the designer of an autonomous system that is to operate in unstructured environments must confront.

It is appropriate to note that despite the long history and intuitive appeal of using vision to guide robotic systems, the applications of this technology remain limited. To some degree this has been due to the high costs of the specialized hardware and the diverse engineering skills required to construct an integrated visually controlled robot system. Fortunately the costs of key elements such as cameras, framestores, image processing hardware and computers in general, continue to fall and appear set to do so for some time to come. Cameras are now becoming available with performance characteristics such as frame rate and image resolution beyond the limiting broadcast television standards which have constrained them for so long.

In conclusion we hope that this paper has shown that visual servoing is both useful and achievable using technology that is readily available today. In conjunction with the cost trends noted above we believe that the future for visual servoing is bright and will become an important and common control modality for robot systems in the future.

[6]However recent results indicate that a visual servo system will converge despite quite significant image Jacobian errors.

ACKNOWLEDGMENT

The authors are grateful to R. Kelly and to the anonymous reviewers for their helpful comments on an earlier version of this paper.

REFERENCES

[1] Y. Shirai and H. Inoue, "Guiding a robot by visual feedback in assembling tasks," *Pattern Recognit.*, vol. 5, pp. 99–108, 1973.
[2] J. Hill and W. T. Park, "Real time control of a robot with a mobile camera," in *Proc. 9th ISIR*, Washington, D.C., Mar. 1979, pp. 233–246.
[3] P. Corke, "Visual control of robot manipulators—A review," in *Visual Servoing* K. Hashimoto, Ed. Singapore: World Scientific, 1993, pp. 1–31. (vol. 7 of *Robotics and Automated Systems*).
[4] G. D. Hager, "The "X-vision" system: A general purpose substrate for real-time vision-based robotics," in *Proc. Workshop on Vision for Robots*, 1995, pp. 56–63, 1995. Also available as Yale CS-RR-1078.
[5] A. C. Sanderson and L. E. Weiss, "Image-based visual servo control using relational graph error signals," *Proc. IEEE*, pp. 1074–1077, 1980.
[6] J. C. Latombe, *Robot Motion Planning*. Boston: Kluwer, 1991.
[7] J. J. Craig, *Introduction to Robotics*. Menlo Park: Addison-Wesley, 2nd ed., 1986.
[8] B. K. P. Horn, *Robot Vision*. Cambridge, MA: MIT Press, 1986.
[9] N. Hollinghurst and R. Cipolla, "Uncalibrated stereo hand eye coordination," *Image and Vision Computing*, vol. 12, no. 3, pp. 187–192, 1994.
[10] J. Feddema and O. Mitchell, "Vision-guided servoing with feature-based trajectory generation," *IEEE Trans. Robot. Automat.*, vol. 5, pp. 691–700, Oct. 1989.
[11] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Trans. Robot. Automat.*, vol. 8, pp. 313–326, 1992.
[12] M. L. Cyros, "Datacube at the space shuttle's launch pad," *Datacube World Review*, vol. 2, pp. 1–3, Sept. 1988. Datacube Inc., 4 Dearborn Road, Peabody, MA.

[13] W. Jang, K. Kim, M. Chung, and Z. Bien, "Concepts of augmented image space and transformed feature space for efficient visual servoing of an "eye-in-hand robot"," *Robotica*, vol. 9, pp. 203–212, 1991.
[14] A. Castano and S. A. Hutchinson, "Visual compliance: Task-directed visual servo control," *IEEE Trans. Robot. Automat.*, vol. 10, pp. 334–342, June 1994.
[15] K. Hashimoto, T. Kimoto, T. Ebine, and H. Kimura, "Manipulator control with image-based visual servo," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1991, pp. 2267–2272.
[16] N. P. Papanikolopoulos and P. K. Khosla, "Adaptive robot visual tracking: Theory and experiments," *IEEE Trans. Automat. Contr.*, vol. 38, no. 3, pp. 429–445, 1993.
[17] N. P. Papanikolopoulos, P. K. Khosla, and T. Kanade, "Visual tracking of a moving target by a camera mounted on a robot: A combination of vision and control," *IEEE Trans. Robot. Automat.*, vol. 9, no. 1, pp. 14–35, 1993.
[18] S. Skaar, W. Brockman, and R. Hanson, "Camera-space manipulation," *Int. J. Robot. Res.*, vol. 6, no. 4, pp. 20–32, 1987.
[19] S. B. Skaar, W. H. Brockman, and W. S. Jang, "Three-dimensional camera space manipulation," *Int. J. Robot. Res.*, vol. 9, no. 4, pp. 22–39, 1990.
[20] J. T. Feddema, C. S. G. Lee, and O. R. Mitchell, "Weighted selection of image features for resolved rate visual feedback control," *IEEE Trans. Robot. Automat.*, vol. 7, pp. 31–47, Feb. 1991.
[21] A. C. Sanderson, L. E. Weiss, and C. P. Neuman, "Dynamic sensor-based control of robots with visual feedback," *IEEE Trans. Robot. Automat.*, vol. RA-3, pp. 404–417, Oct. 1987.
[22] M. Lei and B. K. Ghosh, "Visually-guided robotic motion tracking," in *Proc. Thirtieth Annu. Allerton Conf. on Communication, Control, and Computing*, 1992, pp. 712–721.
[23] R. L. Andersson, *A Robot Ping-Pong Player. Experiment in Real-Time Intelligent Control.* Cambridge, MA: MIT Press, 1988.
[24] B. Yoshimi and P. K. Allen, "Active, uncalibrated visual servoing," in *Proc. IEEE Int. Conf. on Robotics and Automation*, San Diego, CA, May 1994, pp. 156–161.
[25] B. Nelson and P. K. Khosla, "Integrating sensor placement and visual tracking strategies," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1994, pp. 1351–1356.
[26] I. E. Sutherland, "Three-dimensional data input by tablet," *Proc. IEEE*, vol. 62, pp. 453–461, Apr. 1974.
[27] R. Tsai and R. Lenz, "A new technique for fully autonomous and efficient 3D robotics hand/eye calibra tion," *IEEE Trans. Robot. Automat.*, vol. 5, pp. 345–358, June 1989.
[28] R. Tsai, "A versatile camera calibration technique for high accuracy 3D machine vision m etrology using off-the-shelf TV cameras and lenses," *IEEE Trans. Robot. Automat.*, vol. 3, pp. 323–344, Aug. 1987.
[29] P. I. Corke, *High-Performance Visual Closed-Loop Robot Control.* Ph.D. Dissertation, University of Melbourne, Dept. Mechanical and Manufacturing Engineering, July 1994.
[30] D. E. Whitney, "The mathematics of coordinated control of prosthetic arms and manipulators," *J. Dyn. Syst., Meas. Control*, vol. 122, pp. 303–309, Dec. 1972.
[31] S. Chieaverini, L. Sciavicco, and B. Siciliano, "Control of robotic systems through singularities," in *Proc. Int. Workshop on Nonlinear and Adaptive Control: Issues in Robotics* C. C. de Wit, Ed. Berlin: Springer-Verlag, 1991.
[32] S. Wijesoma, D. Wolfe, and R. Richards, "Eye-to-hand coordination for vision-guided robot control applications," *Int. J. Robot. Res.*, vol. 12, no. 1, pp. 65–78, 1993.
[33] G. D. Hager, W.-C. Chang, and A. S. Morse, "Robot hand-eye coordination based on stereo vision," *IEEE Control Syst. Mag.*, vol. 15, pp. 30–39, Feb. 1995.
[34] C. Samson, M. Le Borgne, and B. Espiau, *Robot Control: The Task Function Approach.* Oxford, England: Clarendon, 1992.
[35] G. Franklin, J. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems.* Boston, MA: Addison-Wesley, 2nd ed., 1991.
[36] P. K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman, "Automated tracking and grasping of a moving object with a robotic hand-eye system," *IEEE Trans. Robot. Automat.*, vol. 9, no. 2, pp. 152–165, 1993.
[37] A. Rizzi and D. Koditschek, "An active visual estimator for dexterous manipulation," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1994.
[38] T. S. Huang and A. N. Netravali, "Motion and structure from feature correspondences: A review," *Proc. IEEE*, vol. 82, no. 2, pp. 252–268, 1994.
[39] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applicatio ns to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, June 1981.
[40] R. M. Haralick, C. Lee, K. Ottenberg, and M. Nolle, "Analysis and solutions of the three point perspective pose estimation problem," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 592–598, 1991.
[41] D. DeMenthon and L. S. Davis, "Exact and approximate solutions of the perspective-three-point problem," *IEEE Trans. Pattern Anal. Machine Intell.*, no. 11, pp. 1100–1105, 1992.
[42] R. Horaud, B. Canio, and O. Leboullenx, "An analytic solution for the perspective 4-point problem," *Comput. Vision Graphics, Image Process*, no. 1, pp. 33–44, 1989.
[43] M. Dhome, M. Richetin, J. Lapresté, and G. Rives, "Determination of the attitude of 3-D objects from a single perspective view," *IEEE Trans. Pattern Anal. Machine Intell.*, no. 12, pp. 1265–1278, 1989.
[44] G. H. Rosenfield, "The problem of exterior orientation in photogrammetry," *Photogrammetric Eng.*, pp. 536–553, 1959.
[45] D. G. Lowe, "Fitting parametrized three-dimensional models to images," *IEEE Trans. Pattern Anal. Machine Intell.*, no. 5, pp. 441–450, 1991.
[46] R. Goldberg, "Constrained pose refinement of parametric objects," *Int. J. Comput. Vision*, no. 2, pp. 181–211, 1994.
[47] R. Kumar, "Robust methods for estimating pose and a sensitivity analysis," *CVGIP: Image Understanding*, no. 3, pp. 313–342, 1994.
[48] S. Ganapathy, "Decomposition of transformation matrices for robot vision," *Pattern Recog. Lett.*, pp. 401–412, 1989.
[49] M. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting and automatic cartography," *Commun. ACM*, no. 6, pp. 381–395, 1981.
[50] Y. Liu, T. S. Huang, and O. D. Faugeras, "Determination of camera location from 2-D to 3-D line and point correspondences," *IEEE Trans. Pat. Anal. Machine Intell.*, no. 1, pp. 28–37, 1990.
[51] C. Lu, E. J. Mjolsness, and G. D. Hager, "Online computation of exterior orientation with application to hand-eye calibration," DCS RR-1046, Yale University, New Haven, CT, Aug. 1994; To appear in *Mathematical and Computer Modeling*.
[52] A. Gelb, Ed., *Applied Optimal Estimation.* Cambridge, MA: MIT Press, 1974.
[53] W. Wilson, "Visual servo control of robots using Kalman filter estimates of robot pose relative to work-pieces," in *Visual Servoing*, K. Hashimoto, Ed. Singapore: World Scientific, 1994, pp. 71–104.
[54] C. Fagerer, D. Dickmanns, and E. Dickmanns, "Visual grasping with long delay time of a free floating object in orbit," *Auton. Robots*, vol. 1, no. 1, 1994.
[55] J. Pretlove and G. Parker, "The development of a real-time stereo-vision system to aid robot guidance in carrying out a typical manufacturing task," in *Proc. 22nd ISRR*, Detroit, MI, 1991, pp. 21.1–21.23.
[56] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *J. Opt. Soc. Amer.*, vol. A-5, pp. 1127–1135, 198.
[57] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 9, pp. 698–700, 1987.
[58] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. Amer.*, vol. A-4, pp. 629–642, 1987.
[59] G. D. Hager, G. Grunwald, and G. Hirzinger, "Feature-based visual servoing and its application to telerobotics," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Jan. 1994, pp. 164–171.
[60] G. Agin, "Calibration and use of a light stripe range sensor mounted on the hand of a robot," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1985, pp. 680–685.
[61] S. Venkatesan and C. Archibald, "Realtime tracking in five degrees of freedom using two wrist-mounted laser range finders," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1990, pp. 2004–2010.
[62] J. Dietrich, G. Hirzinger, B. Gombert, and J. Schott, "On a unified concept for a new generation of light-weight robots," in *Experimental Robotics 1*, V. Hayward and O. Khatib, eds., Berling, Germany: Springer-Verlag, 1989, pp. 287–295. (vol. 139 of *Lecture Notes in Control and Information Sciences*).
[63] J. Aloimonos and D. P. Tsakiris, "On the mathematics of visual tracking," *Image and Vision Computing*, vol. 9, pp. 235–251, Aug. 1991.
[64] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision.* Reading, MA: Addison-Wesley, 1993.
[65] F. W. Warner, *Foundations of Differentiable Manifolds and Lie Groups.* New York: Springer-Verlag, 1983.
[66] W. Jang and Z. Bien, "Feature-based visual servoing of an eye-in-hand robot with improved tracking performance," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1991, pp. 2254–2260.
[67] O. Faugeras, *Three-Dimensional Computer Vision.* Cambridge, MA: MIT Press, 1993.

[68] G. D. Hager, "Calibration-free visual control using projective invariance," in *Proc. ICCV*, pp. 1009–1015, 1995. Also available as Yale CS-RR-1046.
[69] D. Kim, A. Rizzi, G. D. Hager, and D. Koditschek, "A "robust" convergent visual servoing system," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1995, vol. I, pp. 348–353.
[70] R. L. Anderson, "Dynamic sensing in a ping-pong playing robot," *IEEE Trans. Robot. Automat.*, vol. 5, no. 6, pp. 723–739, 1989.
[71] E. Dickmanns and V. Graefe, "Dynamic monocular machine vision," *Mach. Vis. Applicat.*, vol. 1, pp. 223–240, 1988.
[72] J. Foley, A. van Dam, S. Feiner, and J. Hughes, *Computer Graphics*. Reading, MA: Addison-Wesley, 1993.
[73] D. Ballard and C. Brown, *Computer Vision*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
[74] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. Int. Joint Conf. on Artificial Intelligence*, 1981, pp. 674–679.
[75] P. Anandan, "A computational framework and an algorithm for the measurement of structure from motion," *Int. J. Comput. Vis.*, vol. 2, pp. 283–310, 1989.
[76] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, 1994, pp. 593–600.
[77] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: active contour models," *Int. J. Comput. Vis.*, vol. 1, no. 1, pp. 321–331, 1987.
[78] K. Hosada and M. Asada, "Versatile visual servoing without knowledge of true jacobian," *Proc. IROS 94*, Sept. 1994, pp. 186-191.
[79] P. Corke and M. Good, "Dynamic effects in visual closed-loop systems," *IEEE Trans. Robot. Automat.*, this issue, pp. 671–683.

Seth Hutchinson (S'85–M'88), for a photograph and biography, see p. 650 of this issue.

Gregory D. Hager (S'85–M'88), for a photograph and biography, see p. 650 of this issue.

Peter I. Corke (S'82–M'83) received the B.E. (Elec), the M.Eng.Sc. and Ph.D. degrees from the University of Melbourne, Australia, where he also lectured in Electrical Engineering.

He is a Principal Research Scientist within the Industrial Automation program of the CSIRO Division of Manufacturing Technology, Kenmore, Australia. As a CSIRO overseas fellow (1988–1989) he visited the GRASP laboratory at the University of Pennsylvania, Philadelphia. At CSIRO, he has worked extensively in the areas of robotics, control, and high-speed machine vision. Projects have included robot control architectures, force controlled deblurring, development of the APA512 (a hardware unit for video rate region analysis of binary images) and applications of high-speed machine vision to the food processing, traffic monitoring and other sectors. More recently his research interests have included high-performance visually guided robot motion, and the application of robotic and vision technology to the automation of mining equipment.