Feature-Based Visual Servoing and its Application to Telerobotics

Gregory D. Hager

Research Report YALEU/DCS/RR-1010
January 1994

Feature-Based Visual Servoing and its Application to Telerobotics

Gregory D. Hager

Department of Computer Science
P.O. Box 208285 Yale Station
Yale University
New Haven, CT 06520-8285

Gerhard Grunwald and Gerd Hirzinger

DLR German Aerospace Research Establishment
Institute of Robotics and System Dynamics
Münchenerstr. 20
D-82234 Oberpfaffenhofen

January 14, 1994

Abstract

Recent advances in visual servoing theory and practice now make it possible to accurately and robustly position a robot manipulator relative to a target. Both the vision and control algorithms are extremely simple, however they must be initialized on task-relevant features in order to be applied. Consequently, they are particularly well-suited to telerobotics systems where an operator can initialize the system but round-trip delay prohibits direct system feedback during motion. This paper describes the basic theory behind feature-based visual servoing, and discusses the issues involved in integrating visual servoing into the ROTEX space teleoperation system.

Submitted as a **Long paper** *to Intelligent Robots and Systems, 1994. Correspondence should be addressed to the first author at the address listed above, or via electronic mail to hager@cs.yale.edu*

1 Introduction

Automation and robotics are rapidly becoming extremely attractive areas within space technology. They hold the promise of assembly, servicing, and repair with a minimal number of expensive manned missions [7]. However, unlike a factory environment, space operations require the ability to work in an environment which is unstructured. For this reason, some amount of autonomous behavior is necessary to perform complex, diverse tasks. This level of autonomy will relay heavily on sensors such as vision, depth, force, torque and tactile, as well as advanced planning and decision capabilities. Unfortunately, our current lack of understanding in sensor data interpretation, robot motion control and artificial intelligence makes the prospect of complete autonomy for complex tasks unlikely in the near future [2]. One way out of this dilemma is sensor-based telerobotics. In particular, space operation tasks for which telerobotics could play an increasingly large role include inspection, assembly, servicing, and repair [10].

When teleoperating in space, relay satellites and computer networks insert a large delay into round-trip communication. Hence sensor data such as video images cannot be used as direct, real-time feedback by a remote operator. One approach to overcoming this problem is to use a predictive computer graphics system as was successfully demonstrated in the German space robot experiment ROTEX. In ROTEX, the operator handles a control device—a 6-dof control ball—based on a predictive graphics model of the robot and its environment. The control commands issued to the robot simulator are sent to both the local and remote robot. If the simulation is properly calibrated to the real environment, the behavior of the remote system will exactly mimic that of the simulation.

Clearly, the crucial problem in this system is to provide an extremely accurate simulation. Teleoperation based on the predictive graphics will fail if the world model does not correspond with reality. This may happen if objects are deformed or damaged, for example the solar panel of the Hubble Space Telescope, or if an object is freely floating in space. Another

disadvantage of relying on a known world model is the need for a precise calibration of the entire operational space. This is problematic as the mechanics are put under extreme pressure by liftoff and the subsequent extreme temperature differences between the parts facing towards the sun and those facing away.

One way of lessening the dependence on prior geometric knowledge is to make remote operations sensor-based. Experience has shown that proper use of closed-loop control algorithms operating in orbit can significantly enhance the capabilities of the teleoperation system [6]. To date, there has been little progress in the use of visual feedback to support teleoperation. One reason is that most classical work on visual servoing relies heavily on a calibrated hand-eye system. As noted above, maintaining accurate calibration in orbit can be difficult. A second reason is that most vision algorithms are too complex to execute on hardware that is space-qualified.

In [5], an approach to feature-based visual servoing using closed-loop control was developed. The main advantages of the approach are that it is extremely robust to calibration error and it uses simple visual features that are computationally simple to track in an image. It can be shown that the accuracy of these visual servoing algorithms is, in fact, independent of calibration error. These properties make the technique well-suited to the teleoperation problem since the operator can choose features and servoing operations using a single static image, and the system can then, without further intervention, autonomously execute the operation with high accuracy and reliability.

In this paper we discuss a family of visual servoing techniques including a previously unreported extension to perform full six degree-of-freedom relative positioning. We then describe some of the issues involved in integrating visual servoing into a teleoperation environment, discuss a preliminary design for such a system, and illustrate its use on an example problem. The remainder of this paper is organized as follows. The next section describes the visual servoing problem and our solution to it. In section 3, we describe the integration of

visual tracking and the telerobotic system and we discuss the operator interface. In section 4, we describe our current progress at implementing this system.

2 Feature-Based Visual Tracking and Servoing

Vision is an extremely rich and precise sense. In many ways, it is an ideal candidate for sensor-based motion and manipulation. However, in order to combine vision and robotics, two major problems must be addressed. First, vision problems that involve scene interpretation tend to be extremely difficult, and most of the vision algorithms used to solve them require specialized hardware in order to operate in real-time. Second, in order to relate visual information to a robot manipulator, the spatial relationship between the manipulator and the camera(s) must be known. Determining and maintaining this relationship, the *hand-eye calibration,* with high accuracy is extremely difficult. Here we describe an approach to visual servoing that obviates these difficulties. The method uses locally defined visual features, primarily image contours, that can be tracked in real-time by standard workstations or PCs. Corresponding features in two cameras comprise the input to a feedback control system. The feedback control system will provide accurate positioning despite calibration errors provided control gains are chosen to ensure stability.

2.1 Visual Feedback Control

In this paper, we describe two systems for visual feedback control: one that controls only position, and one that controls both position and orientation. Figure 1(a) illustrates the underlying principle. The left figure is a schematic depiction of our visual servoing architecture which includes two video cameras, a robot arm, and computers that perform image-processing, low-level control operations, and other interface-related functions. Any attempt to accurately calibrate this system is limited by system modeling errors, mechanical

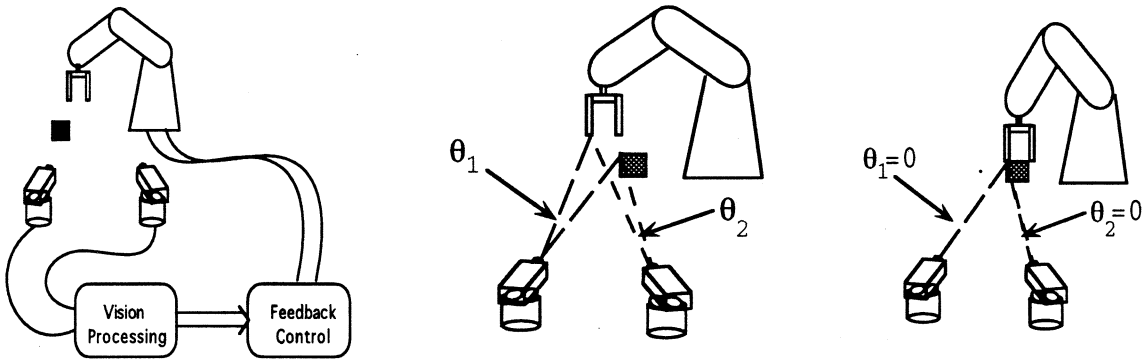

Figure 1. The figure on the left shows a visual servoing system consisting of two cameras on pan-tilt heads connected via a vision-based controller to a robot arm. The middle and right figure demonstrate positioning by reducing visual disparity to zero.

backlash, control inaccuracy, and so forth. Consequently, a system that reconstructs absolute point coordinates using vision and positions the robot based on that information will be extremely imprecise. However, as illustrated in Figures 1(b) and 1(c) the robot manipulator can be positioned extremely accurately *relative to the observed target.* In (b), the cameras observe the visual disparity θ_1 and θ_2 between a point on the manipulator and the corner of the box. We know the following property: *zero disparity between the manipulator and the goal in both images means that they are at the same point in space.* This property is true independent of where the cameras are located relative to the robot.[1] Thus, if we have a stable controller to achieve zero visual disparity, we can position accurately, even if the system is badly calibrated.

The theory underlying a feedback control algorithm for this problem can be summarized as follows. Suppose that $y = f(x)$, is a mapping from a robot configuration space to an output (sensor) space, both of dimension n. Given a desired setpoint y^*, define $e_y = y - y^*$

[1]More precisely this statement is true modulo configurations where the goal or robot and the cameras are collinear.

and introduce a new variable z such that $\dot{z} = e_y$. Taking time derivatives of the system, we see that

$$\dot{e_y} = J_f \dot{x}. \tag{1}$$

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

We take $u = \dot{x}$ to be the control input to the system and, presuming J_f is full rank, compute

$$u = -J_f^{-1}(k_1 e_y + k_2 z)$$

where k_1 and k_2 are constants set by the designer. To analyze the behavior of this control method, we substitute this expression for u into (1) yielding:

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

Or,

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

It is well known that proper choices of k_1 and k_2 in differential equations of this form lead to an error, e, that asymptotically approaches zero. The presence of the integrator, z, ensures this behavior in the presence of external disturbances or if the system model, f, is in error provided the errors are not large enough to destabilize the system.

Using these concepts, we can construct visual servoing systems to perform different types of positioning and motion by defining the appropriate error term. The inputs available for defining errors are contours and fixed visual reference points observed in two cameras. The following is a brief review of camera imaging geometry for these features. Camera positions are represented by the frames $\mathcal{C}_1 = (c_1, \Sigma_1)$ and $\mathcal{C}_2 = (c_2, \Sigma_2)$. Points and lines in $\Re(3)$ will be written in capital letters. The projection of a point P or line L in camera i will be written p_i and l_i respectively.

The projection of a point $P = (P_x, P_y, P_z)^T$ to a homogeneous vector $p_i = (u, v, 1)^T$ is given by

$$\begin{aligned} P' &= \Sigma_i (P - c_i) \\ p_i &= (u, v, 1)^T = P'/P'_z \end{aligned} \tag{2}$$

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

Following the approach of Taylor and Kriegman [14], an arbitrary infinite line, L, is parameterized by a tuple (L_d, L_v) where L_d is fixed point on the line and L_v is the direction of the line. The vector l_i parameterizing the projection of L in camera i is

$$l_i = \Sigma_i \left(L_v \times (L_d - c_i) \right). \tag{3}$$

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

The Jacobian of g is a function of the position of the observed point, and the Jacobian of h is a function of the fixed point on the line. As described in [3], it is possible to estimate the parameters of both lines and points using relatively straightforward techniques. We refer the reader to that publication for the technical details of estimation and control and proceed to describe the structure of two representative controllers.

Problem Definition: Given a fixed reference point, P, on a manipulator and a point, R, not on the manipulator, servo the manipulator so that $P = R$ using inputs p_i and r_i, $i = 1, 2$.

Define $g(P) = (g_1(P); g_2(P))$ and $y^* = (r_1; r_2)$. Note that g maps three values—the Cartesian position of a point—into six values—the homogeneous camera image locations of the projections of the point. Two values are constant and can be discarded. Of the remaining four, one is redundant and should also be discarded. As described in [5], it is possible to compute a 3 by 6 matrix E which depends on camera calibration parameters such that the product $Eg(P)$ provides three independent observations of P. Define $J(P) = EJ_g(P)$ and $e = E(y^* - g(P))$. Applying the methods described above yields a controller for relative positioning. The controller has been implemented and tested as described in [5].

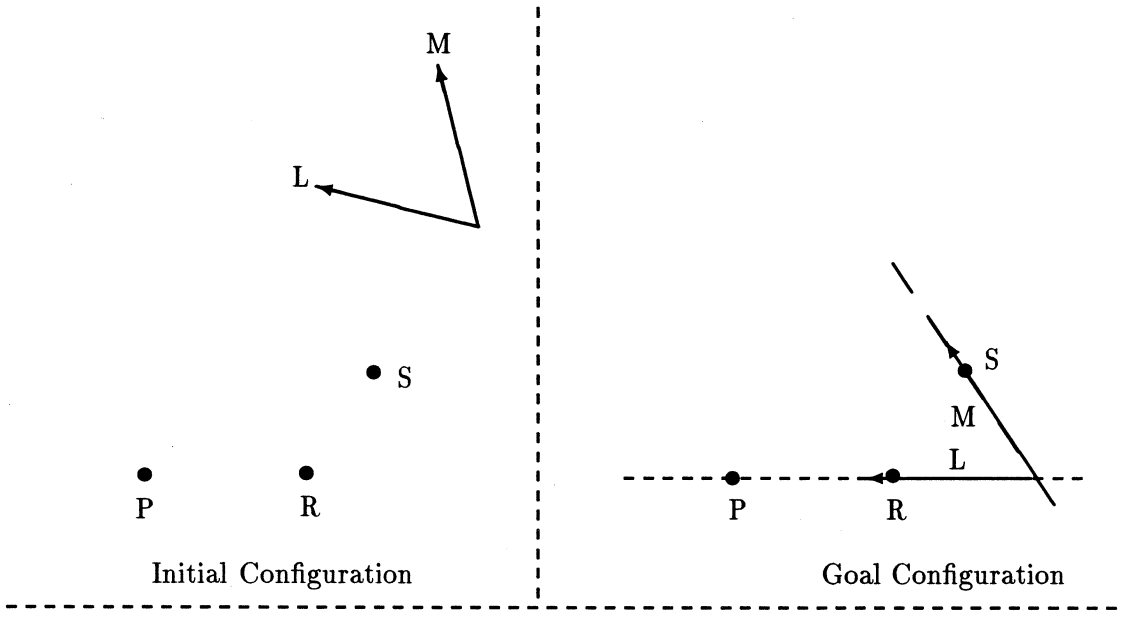

Figure 2. The geometry of the six degree of freedom servoing problem.

Problem Definition: Given a three non-collinear, fixed reference points P, R and S and two non-parallel reference lines L and M that are rigidly attached to a manipulator, develop a regulator that positions the manipulator so that $P \in L$, $R \in L$, and $S \in M$ using p_i, r_i, s_i, l_i, and m_i, $i = 1, 2$. This is illustrated in Figure 2.

A homogeneous vector p_i in camera image i lies on the line projection l_i if and only if $p_i \cdot l_i = 0$. Modulo a set of singular configurations, it can be shown that for any arbitrary line L and a point P, $l_1 \cdot p_1 = l_2 \cdot p_2 = 0$ if and only if $P \in L$. This motivates the definition

of a positioning error $e \in \Re(6)$ as:

$$e = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \end{bmatrix} = \begin{bmatrix} p_1 \cdot l_1 \\ p_2 \cdot l_2 \\ r_1 \cdot l_1 \\ r_2 \cdot l_2 \\ s_1 \cdot m_1 \\ s_2 \cdot m_2 \end{bmatrix} = \begin{bmatrix} g_1(P) \cdot h_1(L) \\ g_2(P) \cdot h_2(L) \\ g_1(R) \cdot h_1(L) \\ g_2(R) \cdot h_2(L) \\ g_1(S) \cdot h_1(M) \\ g_2(S) \cdot h_2(M) \end{bmatrix} = H(P, R, S, L, M) \tag{4}$$

Based on the remarks above, it follows that, modulo a set of singular configurations, $e = 0$ if and only if $P \in L$, $R \in L$, and $S \in M$. The system Jacobian J_H is square, and depends on five vector quantities that can be estimated from sensor data. Applying the methods described above produces a regulator that can control all six degrees of freedom of a robot manipulator. For details on the form of the Jacobian and the estimation of unknown values, we refer the reader to [3].

2.1.1 Variations On These Systems

Note that the assignment of features in the problem above was arbitrary. That is, the roles of points and lines can be interchanged without changing the final result. Also, one way of defining L and M is by choosing two reference points and computing the line that runs through them. The problem formulation is independent of whether the cameras are stationary in the environment or mounted on the manipulator itself. Thus this basic control formulation can operate on a wide variety of systems with different types of input.

The controller can also be simplified to control fewer degrees of freedom. Dropping components involving M and S from the error term leads to a system that controls 4 degrees of freedom. It aligns two points to an axis, but leaves rotation about the axis and translation along the axis free. Since the Jacobian, J, is nonsquare in this case, the control vector is computed by

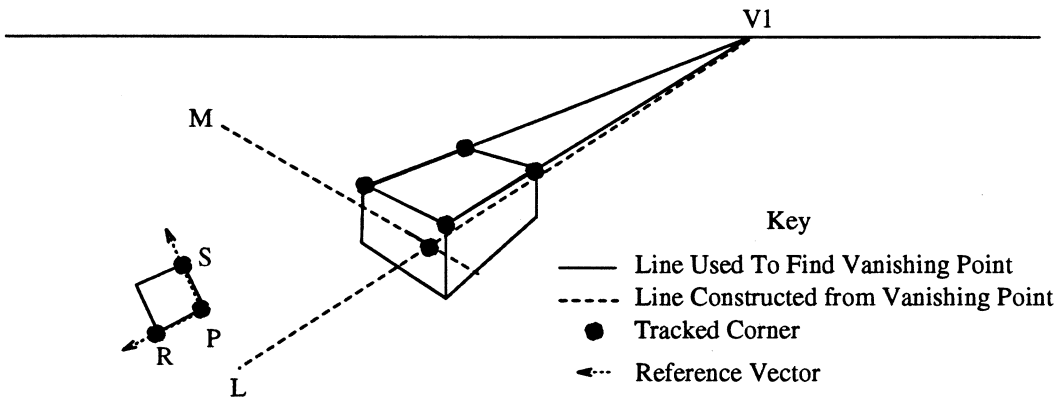

Figure 3. The geometric constructions for inserting a floppy disk into a drive unit.

$$u = -(JJ^T)^{-1}J^T(k_1 e_y + k_2 z).$$

With this formulation, it is possible to impose the remaining two degrees of freedom arbitrarily by chosing a rotational velocity and translation velocity about and along the estimated value of the line L, respectively. This provides for shared control between the control system and an external agent, *e.g.* an operator. Although not discussed here, other related variations are possible.

2.2 Geometry and Visual Servoing

The controllers described above do not have to be based on directly observed features. Various types of geometric constructions can be used to define "virtual" lines and points. However, in order to maintain the accuracy of servoing, these constructions must rely on image-level geometric constructions.

For example, Figure 3 describes the use of geometric constructions to place a floppy disk into a disk drive. Geometrically, the strategy is to move the disk into the plane of the disk

drive unit, align perpendicular to the slot, and then move toward the slot until the disk slides in. In the visual domain, we can perform this task if we know that the sides of the disk drive unit are parallel to the floppy disk drive. In a projected image, parallel lines meet at a point, so we first construct vanishing point $V1$ by tracking the indicated corners of the drive unit. The edge of the floppy slot and the vanishing point are used to construct the line L. The line M is constructed from the image of the slot itself. The three corners of the floppy are used to define the reference points P, R, and S. Servoing using this information will place the floppy disk at the mouth of the drive slot, oriented parallel to it.

As this example illustrates, a variety of geometric constructions can be used to provide increased functionality for visual serving. As with the basic servoing routines, these constructions rely on tracking specific features in an image, and computing values based on those features.

2.3 Feature Tracking

In [4], a feature-based tracking system was described. The tracking system is based on two central ideas: window-based image processing, and state-based programming of networks of tracked features. A window is an image defined by its height, width, position, and orientation in device (framebuffer) coordinates. Tracking a feature means that image-processing operations are used to maintain a fixed relationship between window coordinates and the underlying feature. The low-level features currently available in the system include solid or broken contrast edges detected using convolutions, and general grey-scale patterns tracked using SSD methods [1, 15]. The entire tracking system is designed to run on a standard CPU and framegrabber with no additional hardware support. For example, tracking single contours on a Sun Sparc 2 with an Imaging Technologies 100 series framegrabber requires 1.5ms for a 20 pixel contour searching $\pm$ 10 pixels using a mask 15 pixels wide. Tracking arbitrary patterns using SSD methods requires 70 ms to localize a 20 by 20 window.

Basic features can be easily composed into more complex configurations using feature networks. Every feature or image property in our system can be characterized in terms of a state vector. For *basic features*—those that operate directly on images—the state of the feature tracker is usually the position and orientation of the feature relative to the framebuffer coordinate system. We define *composite features* to be features that compute their state from other basic and composite features. A *feature network* is defined as a set of basic and composite featuers connected by two types of directed arcs referred to as *up-links* and *down-links*. The *up-links* provide image information for higher levels of abstraction. For example, a cluster of image features that all lie in a plane may contribute information to a composite feature that has, as state, the slant and tilt of the plane. The *down-links* provide constraints from higher levels of abstraction. For example, if something is known about the motion of a rigid object, this information can be propagated down to the feature level to predict positions of image features in the subsequent image.

This ability to "package" features into higher-level primitives is particularly useful for providing visual constructions such as those used in the last section. Each of these constructions can be defined as an object that requires certain initialization information and provides a particular type of image information. These objects can be composed to form more objects, or can be used directly to provide input to visual servoing routines.

3 Teleoperation and Visual Servoing

Visual servoing provides the means to achieve a particular geometric configuration reliably and accurately. However, some intelligence external to the servoing algorithm must translate a geometric task into the visual operations and choose the visual features needed to perform those operations. In this section, we explore the issues that arise when integrating visual servoing into a teleoperation system similar to ROTEX.

The information available to the human operator consists of a menu driven programming

environment, two camera images of the remote site, and a graphics simulation of the known aspects of the remote site. The completeness of the latter depends on the availability of prior object knowledge and/or sensor reconstruction methods. The system interface offers the operator a list of the visual servoing operations, *e.g. positioning* or *alignment*, that are available in the current execution context. The operator must select an operation and, based on the selection, he or she must choose image features that initialize the tracking system for the chosen operation. These may be image features, or features defined using visual constructions. In the latter case, the system requests further initialization information until all operations are fully instantiated with image-level features. Note that the initialization of feature tracking must take place in both camera images.

3.1 Image Level Feature Specification

The effect of a visual control action is only meaningful if the chosen feature has physical relevance, and if the images of the same physical features are chosen in both camera images. As a consequence, the set of image-level features we consider are contours, intersections of contours, and unique surface markings. An example of the latter would be a barcode or similar artificial labeling device. We disallow arbitrary patterns that would be difficult to place in accurate correspondence in both camera images.

At the image level, the tracking system is initialized to track a feature by supplying the image coordinates (position and orientation) of the feature within a prescribed tolerance. This information may be interactively supplied by an operator "clicking" on a feature in an image, or may be indirectly supplied through model information. In both cases, the low-level image processing can check for the presence of the correct type of feature, and also ensure that there are no distracting features in the immediate vicinity that would lead to tracking ambiguity. The system can reject any feature that does not satisfy these conditions.

As noted above, it is important that the features chosen in both camera images be

physically consistent with one another. One means of ensuring this is to exploit epipolar camera geometry. Given (homogeneous) projections p_l and p_r, of a point P, there is a 3 by 3 matrix E such that $p_l^T E p_r = 0$. If the camera system is accurately calibrated, the E matrix can be computed directly from the calibration. Alternately, the E matrix can be computed from the projections of eight points in two images provided the points are arranged in a non-singular configuration [11, 9, 12]. Hence given a ninth feature point in one image, it is possible to determine the epipolar line along which the corresponding feature lies in the second image.

Note that if the point P lies on a contour in one image, the corresponding point in the second image is determined by the intersection of the corresponding contour with the epipolar line. Consequently, standard matching methods can be applied to order candidate matches along the epipolar line. The operator can then accept the system's best choice, or override it by choosing another contour. If the feature in the first image is the intersection of contour points (*e.g.* a corner) then the feature in the second image is a point where two contours and the epipolar line intersect at a single point. This has a very high probability of being unique. Similar remarks hold for special patterns.

These ideas rely on having either an accurate calibration or at least 8 corresponding points in both images. That latter condition is preferable as it is independent of camera calibration, however it may be extremely difficult to satisfy. However, the environment contains many known objects, in particular the robot itself. Consequently, a simple method for providing the required correspondences is to decorate the workspace with unique visual targets. These targets can be acquired automatically from static images *on the ground* and used for subsequent epipolar computations. Furthermore, if enough image processing bandwidth is available at the remote site, the targets can be tracked over time and do not have to be reacquired for every operation initialization.

3.2 Using Prior Environmental Knowledge

Even if the automatic correspondence mechanisms described above functioned perfectly, it would still become onerous for the operator to specify all of the features needed for numerous six degree-of-freedom positioning operations.

In many cases, image-level tracking initialization may not be necessary. Some objects in the environment will be known objects, in particular the robot manipulator. If the location and pose of these objects can be calculated using artificial markings or by tracking object features, then the operator can choose features from the model in the graphics simulation. By using the known correspondence between model and image, the tracking system can be initialized in both images automatically. If sufficient tracking bandwidth is available the operator can also register features for later use and reuse. For example, if the manipulator is holding a particular tool, *e.g.* a screwdriver, the operator may instruct the system to track the shaft and the endpoint of the screwdriver at all times. These features can then be used again and again by referencing a symbolic label, or by choosing them from an iconic representation in the graphic simulation.

Likewise, some often-used operations may implicitly select certain object features. For example, one extremely common operation is to command the manipulator to approach an object along the manipulator center axis. An operation "manipulator-aligned-approach" could be defined that automatically choses certain manipulator features to define the center axis and endpoint of the manipulator. In this case, the operator need only choose one or two features that define the goal configuration. If this can be done in the graphic simulation, visual servoing is essentially no more difficult than pointing to where the robot should go.

3.3 Integration into ROTEX

The ROTEX sensor-based telerobotic system [6] is based on the shared autonomy concept that distributes intelligence to man and machine [8]. Global tasks like visual servoing are

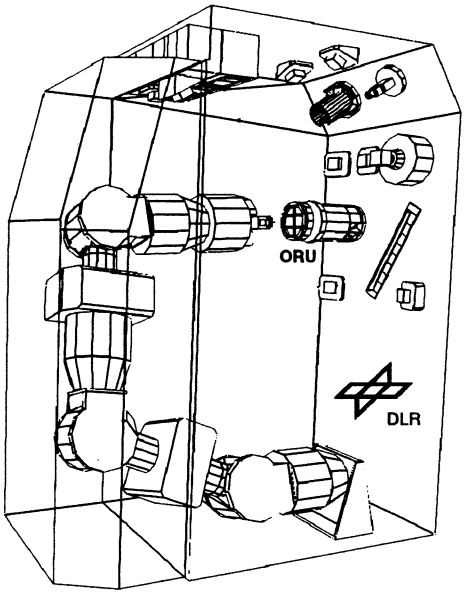

Figure 4. The ROTEX workcell.

specified interactively by a human operator and carried out by local sensory feedback loops executed by the remote robot system. Coarse task-level planning activities are performed by human intelligence, while fine path planning on manipulator level takes place on sensor-based control level [13]. The shared autonomy concept also allows shared control in which the robot is guided by local sensory feedback as well by the human operator via the teleoperational device. In context to visual servoing, this allows the specification of a visual controller constraining less than 6 degrees of freedom. The remaining degrees of freedom are controlled by the operator.

The ROTEX system already supports a three-dimensional graphics environment in which object models and the robot are represented. This virtual world allows the operator to move around and "interact" with the robot and its environment. In addition, all remote operations are mimicked in the graphical simulation. In order to support visual servoing, the functionality of the ROTEX ground station [6] will be expanded to include a menu-driven programming environment, the image processing system, and the capability of overlaying remote camera images with wireframe models of known objects. These visual aids make

the feature selection easier for the operator. The overlay not only shows the correspondence between a model and real data but also gives also a visual hint as to the validity of the system calibration. Selected visual features will appear highlighted in the camera images and, when possible, in the graphic display. Visual servoing will also be incorporated into the robot simulation so that the servo operations can be tested offline before applying them to the remote system.

Below we briefly sketch how two servoing problems would be specified in the extended ROTEX system. ROTEX contains two camera systems: a stereo system mounted on the manipulator itself, and an external camera system. The first example uses the external stereo cameras, and the second uses a single manipulator-mounted camera. We assume that some calibration information on both camera systems is available, although the correctness of the calibration is not guaranteed.

3.3.1 Examples

We first describe how the floppy insertion task could be performed using the ROTEX interface. We note this is in fact a realistic task since one of the commonly performed tasks in orbit is the removal and insertion of circuit boards. We assume that the disk drive unit is a known object registered to the graphics model. The disk itself is not an object known to the system. We assume the manipulator is holding the disk initially.

The first action of the operator would be to choose a servoing operation. Since insertion is a common operation, it would appear in the list of available operations. Once insertion is chosen, the operator is notified that he or she must must choose specific image information—in this case points and/or contours on the manipulated and target objects. The operator then chooses the three corner points of the floppy disk as the defining points for the manipulated object. When chosen, the system estimates the depth of the points (using camera calibration information) and displays them in the graphics display.

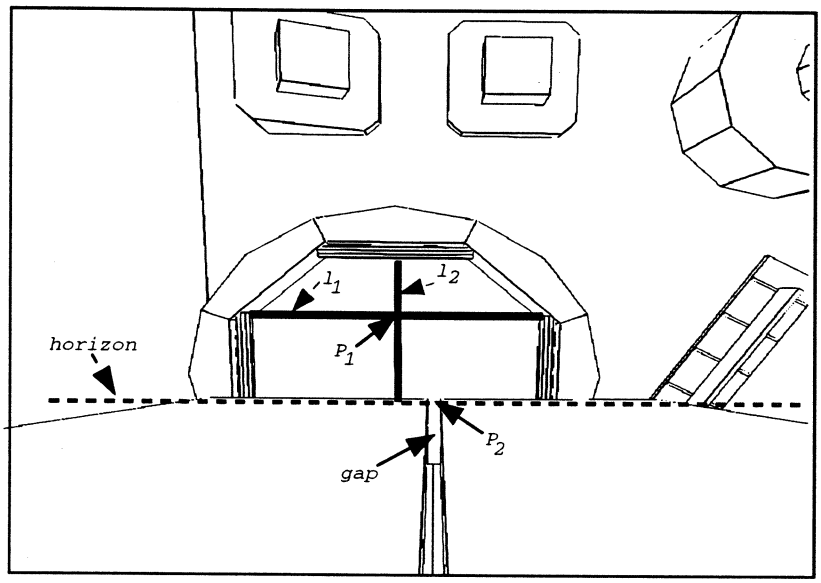

Figure 5. The view from the manipulator cameras.

The system continues to request two lines on the target object. The operator now chooses the construction *parallel-line*. The system offers options for constructing parallel lines, in particular by choosing two parallel lines in the world and a third point. The operator chooses this option. Since the floppy drive is registered in the graphics model, the operator can directly choose the sides of the drive unit as the requested parallel lines and the edge of the floppy slot as the point defining the insertion axis. This defines the line L of Figure 3. Still needing a second line, the operator indicates the drive slot of disk unit in the graphics display. The feature specification for the task is now complete. The system analyzes the features for trackability and completeness. If the features are satisfactory, the operation proceeds in both simulation and on the remote system. When finished, the operator receives an image and checks that the operation was correctly performed. If so, the system proceeds to perform the insertion operation, presumably under position and force control.

As a second example, we describe the task of grasping the ORU (Orbit Replaceable Unit). This task is representative of many experiments which involve the manipulator moving

toward a rigidly attached object, grasping it, and moving it to some other place.

Figure 5 shows a typical view from the manipulator cameras above the ORU. The visual line markings l_1 and l_2 were added to aid the human operator in positioning. The final goal position is reached if l_1 aligns with the horizon of the fingertips, l_2 aligns with the gap between the fingers and the manipulator is perpendicular to the surface of the ORU as it touches down. In the final approach, four laser distance sensors integrated into the fingertips can provide distance and some orientation information. In this example it is assumed, that all objects are known and their models are registered in the graphics model.

Unfortunately, the configuration of the manipulator cameras is such that occlusion and visual singularities severely limit their usefulness. In particular, the "horizon" line in the figure is parallel to the baseline of the cameras. This means that it is not possible to acquire depth information on this line, or on lines parallel to it. Instead, we use visual information on the lines l_1 and l_2 to constrain the orientation of the manipulator about the center axis and translation perpendicular to the manipulator axis while the operator controls the direction and attitude of approach.

Proceeding as above, the operator begins by choosing an operation, this case the appropriate approach operation. This operation assumes that the horizon line and the center of the manipulator gap (P_2 in the figure) are the manipulator setpoints. It indicates that the operator must specify a line and point in one camera. Since the ORU is registered in the graphics simulation, the operator indicates l_1 and P_1. The system verifies that these features are visible and trackable, and then begins to graphically simulate the visual servoing operation as it is carried out remotely. The operator uses the tracking ball to move the manipulator toward the ORU, simultaneously adjusting orientation, until the laser distance sensors detect the ORU surface. At this point, the distance sensors can control manipulator attitude with respect to the surface. Vision continues to control translation parallel to the surface and rotation about the center axis. The operator continues to control distance to

the object until touchdown is registered on the force sensors.

4 Discussion and Future Research

We have described a method for feature-based visual servoing and its application to remote teleoperation. The visual servoing methods described above have been tested in stand-alone systems. We are now in the process of integrating them into a larger, more flexible servoing system. We have taken some initial steps toward including visual servoing in the ROTEX environment.

Even at this early stage, several points are clear. First, any remotely operated system will benefit from the availability of artificial visual features, even if the features are not part of a calibrated geometric model. Ensuring that eight such features are always visible is extremely useful for correspondence calculation when the camera system is not well calibrated. Second, the ability to recognize and calibrate the robot end-effector and other objects in the environment to a simulation significantly eases the burden on the operator. Careful design of the visual servoing operations can relieve the operator from the potentially onerous task of specifying visual cues. Third, the availability of a graphical simulation for the visual servoing operation is an extremely useful tool. In addition to providing a means for choosing object features, it supports operator interaction using hybrid control. The latter often simplifies the specification of a visual servoing task.

One issue we have not addressed in any detail is the issue of error handling. The visual servoing methods described above will fail if the system moves through (visually) singular configurations. In many cases, these errors can be detected and corrected for automatically. However, some errors should be flagged and returned to the operator. For example, in ROTEX is it possible for the operator to choose a set of features on the manipulator that lead to a globally singular system. This should be detected and the feature specification rejected with an appropriate explanation of the problem.

[14] C. Taylor and D. Kriegman. Structure and motion from line segments in multiple images. In *IEEE Int. Conf. on Robotics and Automation*, pages 1615–1620. 1992.

[15] C. Tomasi and T. Kanade. Shape and motion from image streams: a factorization method, full report on the orthographic case. CMU-CS 92-104, CMU, 1992.