in *Visual Servoing-Real-Time Control of Robot Manipulators Based on Visual Sensory Feedback*, ed. K. Hashimoto, World Scientific Publishing Co. Pte. Ltd., River Edge, NJ, pp. 139-164, 1993.

VISUAL SERVOING FOR ROBOTIC ASSEMBLY

Brad Nelson,[†] N.P. Papanikolopoulos,[‡] and P.K. Khosla[‡]

The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

ABSTRACT

Visual feedback has traditionally been used in the assembly process to a very limited extent. With the advent of effective visual servoing techniques, visual feedback can become an integral part of the assembly process by complementing the use of force feedback to accomplish precision assemblies in imprecisely calibrated robotic assembly workcells. In this paper we present some of the issues pertaining to the introduction of visual servoing techniques into the assembly process and solutions we have demonstrated to these problems.

1. Introduction

Geometric modeling systems are rapidly replacing manual drafting techniques for defining the geometry of mechanical parts and assemblies during the design process. This makes possible the development of software tools to aid the manufacturing engineer in setting up assembly workcells, thus, integrating design and manufacturing. Robotic assembly workcells have been created which can be automatically programmed, including the automatic generation of real-time code, to perform mechanical assemblies based on geometric models of the assembly.[9,15] An important benefit of these systems is that by automatically generating assembly plans and real-time code for performing mechanical assemblies, assembly workcell setup times can be drastically reduced.

When programming robotic assembly workcells automatically, however, the accuracy as well as the repeatability of the robots' motion must be extremely high, so that parts that are to be assembled can be placed at positions and orientations within thousands of an inch of their desired position and orientation. For this reason, it is critical that the entire robotic assembly workcell be precisely calibrated for most assemblies to be successfully accomplished, but precisely calibrating the entire workcell is tedious and expensive.

An alternative to precise calibration is to use sensor feedback during the assembly

†Robotics Ph.D. Program and the Robotics Institute, Carnegie Mellon University.
‡Department of Electrical and Computer Engineering and the Robotics Institute, Carnegie Mellon University.

process. Force feedback is sometimes incorporated into assembly systems, however, force feedback is often difficult to use due to instabilities that arise when contact is made and due to the poor signal-to-noise ratio that force/torque sensors provide. Partly because of these reasons, the use of force feedback in industry has been limited.

Other types of sensors may be used in conjunction with force sensors for assembly, so that problems encountered with force feedback can be overcome. Tactile and visual sensors are well developed sensor technologies that have met with some success in assembly. Tactile sensors are prone to many of the same problems as force sensors, though, because the feedback is very localized, and noise is often difficult to distinguish from the information the sensor provides. Vision sensors have only been used within a limited framework. Traditionally, the feedback provided to the assembly process by vision sensors has been incorporated into the assembly process outside of the manipulator control loop. Incorporating sensor feedback in this fashion means that highly nonlinear systems, such as camera-lens systems, have to be carefully modeled and calibrated in order to determine inverse transformations from which the locations of objects in the workcell can be inferred, but this is difficult to do accurately.

If the vision sensor is placed within the feedback loop of the manipulator, however, we can take advantage of many different research developments in control theory in order to reduce our reliance on precise calibration of the camera-lens system, and the entire assembly workcell as well. For these reasons, we propose an assembly system which uses visual feedback at the servo level during the entire assembly process in order to effectively reduce the assembly system's reliance on precise calibration. We are creating such an assembly system by combining visual servoing with force servoing in order to close the loop of an assembly workcell throughout the assembly process, and, thus, the benefits that a feedback control system provides can be realized during the entire assembly process.

Using visual feedback effectively in the control loop of an assembly process presents challenges quite different from those presented by force feedback. The large amount of data collected by a visual sensor causes the sampling rate to be relatively small, and introduces large delays in the control loop. Since noise exists in visual sensors and the sampling rate is low, robust feature trackers and control algorithms must be used. Unless expensive and even slower 3D range finders are used, the data from a typical visual sensor, i.e. a camera, is 2D, and presents several problems due its nonlinear nature and the difficulty in deriving 3D information, which is necessary in a three dimensional workcell.

This paper presents three specific solutions designed to overcome the problems of incorporating visual feedback into the control loop of an assembly system. In Section 2, we consider how the static camera-robot system should be modeled and controlled in 3-D. The model is time-varying, multi-variable, and nonlinear, thus, control strategies which can be successfully applied to these types of systems must be used. In Section 3, we discuss how to determine where the vision sensor needs to be placed in the system. A static sensor is easier to model and less prone to noise, but occlusions can be more difficult to avoid, the parts may be out of the camera's depth-of-field or field-of-view, or the spatial resolution of the object's projection onto the camera's image plane may be inadequate. A movable camera can solve these problems, but this implies that real-time sensor placement issues be considered. Finally, in Section 4 we consider how adaptive control techniques can be effectively introduced into the system in order to further reduce the reliance of the system on precise calibration. Our proposed solutions have all been implemented on the Rapid Assembly System, a system of three Puma560's with force sensing wrists and a vision system. Experimental results demonstrating the effectiveness of our solutions to these problems are presented in each of the three sections. The paper is summarized in Section 5.

2. Static Camera Servoing in 3-D

2.1 Previous Work

Several researchers have proposed various techniques for introducing visual feedback into the control loop of manipulators. Visual servoing systems can be distinguished by two different system configurations, the eye-in-hand configuration (camera mounted at a manipulator's end-effector),[3,4,6,8,13,14] and the static camera configuration.[1,10,18] For assembly, we want to servo the parts to be assembled towards one another so that the assembly task can be accomplished. To do this, we first need to study the servoing of the end-effector of a manipulator based on a camera's observations of the assembly workcell, thus, we consider the static camera servoing case. The controlled active vision framework,[13] though originally used for an eye-in-hand system, is also useful for developing a model and control strategy for a static camera visual servoing system.

2.2 Modeling and Control

To model the 3-D visual servoing problem, it is assumed that the image plane is parallel to the X-Y plane of the tool frame of the manipulator, the depth of the tool frame from the camera frame is roughly known initially, and a cartesian robot controller exists. A pin-hole model for the camera as shown in Figure 1 is assumed,[7] so that a feature at

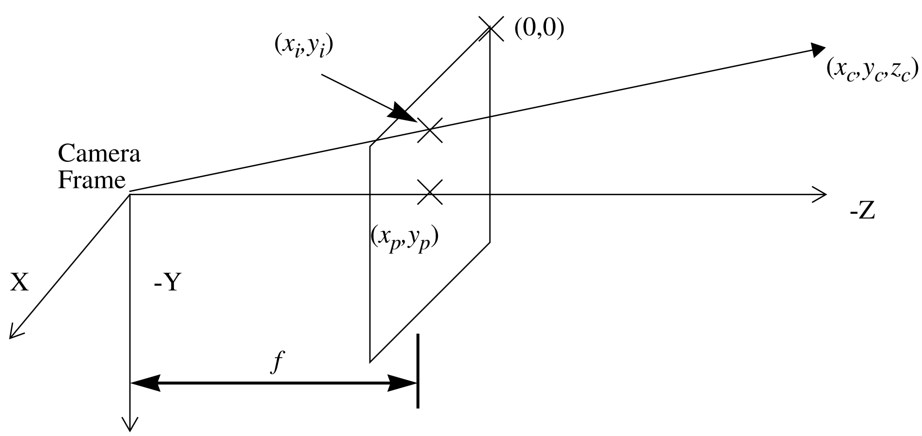

Figure 1. The pin-hole camera model with the image plane moved in front of the focal plane to simplify signs in the equations.

(x_c, y_c, z_c) in the camera frame projects onto the camera's image plane at

$$x_i = \frac{f x_c}{z_c s_x} + x_p \tag{1}$$

$$y_i = \frac{f y_c}{z_c s_y} + y_p \quad , \tag{2}$$

where (x_i, y_i) are the image coordinates of the feature, f is the focal length of the lens, s_x and s_y are the horizontal and vertical dimensions of the pixels on the CCD array, and (x_p, y_p) is the piercing point of the optical axis on the CCD. The optical flow of a feature $(u(t), v(t))$ on the image plane induced by motion of the feature in the tool frame is obtained by differentiating Eqs.1 and 2 with respect to time and is

$$u(t) = \dot{x}_i(t) = \frac{f \dot{x}_c(t)}{z_c(t) s_x} - \frac{f x_c(t) \dot{z}_c(t)}{z_c^2(t) s_x} \tag{3}$$

$$v(t) = \dot{y}_i(t) = \frac{f \dot{y}_c(t)}{z_c(t) s_y} - \frac{f y_c(t) \dot{z}_c(t)}{z_c^2(t) s_y} \quad . \tag{4}$$

Solving Eqs. 1 and 2 for x_c and y_c, respectively, and substituting them into Eqs. 3 and 4 results in the following optical flow equations

$$u(t) = \frac{f \dot{x}_c(t)}{z_c(t) s_x} - \frac{(x_i(t) - x_p) \dot{z}_c(t)}{z_c(t)} \tag{5}$$

$$v(t) = \frac{f \dot{y}_c(t)}{z_c(t) s_x} - \frac{(y_i(t) - y_p) \dot{z}_c(t)}{z_c(t)} \quad . \tag{6}$$

The benefit of putting the optical flow equations in this form is that the X-Y position of the object in world coordinates does not appear. It is still necessary to know the Z distance, or depth, of the object from the camera. The optical flow at time $t=kT$ (T is the sampling period of the vision system) can be approximated as

$$u(k) = \frac{x_i(k+1) - x_i(k)}{T} \tag{7}$$

$$v(k) = \frac{y_i(k+1) - y_i(k)}{T} \quad , \tag{8}$$

where kT is represented as k for simplicity and without any loss of generality. This can be rewritten as

$$x_i(k+1) = x_i(k) + T u(k) \tag{9}$$

$$y_i(k+1) = y_i(k) + T v(k) \quad . \tag{10}$$

If the inputs to the system are considered to be $[\dot{x}_c \ \ \dot{y}_c \ \ \dot{z}_c]^T$, the velocity of the end-effector, we can use Eqs. 5 and 6 to rewrite Eqs. 9 and 10 as

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) \quad , \end{aligned} \tag{11}$$

where $\boldsymbol{x}(k) = [(x_i(k)\text{-}x_p) \ (y_i(k)\text{-}y_p)]^T$, $\boldsymbol{u}(k) = [\dot{x}_c \ \ \dot{y}_c \ \ \dot{z}_c]^T$, $\boldsymbol{y}(k) = [(x_i(k)\text{-}x_p) \ (y_i(k)\text{-}y_p)]^T$, $\boldsymbol{A} = \boldsymbol{C}$

$= \mathbf{I}_2$, and

$$\boldsymbol{B} = T\begin{bmatrix} \frac{f}{z_c(k)\, s_x} & 0 & -\frac{(x_i(k) - x_p)}{z_c(k)} \\ 0 & \frac{f}{z_c(k)\, s_y} & -\frac{(y_i(k) - y_p)}{z_c(k)} \end{bmatrix} . \tag{12}$$

The system requires three control inputs. For a single feature, however, there are only two states that can be observed, thus two state equations. If it is desired to drive the coordinates of a feature to a particular position on the image plane, then there are an infinite number of positions in the world to which the object could be driven so that the feature is at the desired image coordinates. All of these positions in the world are on the line of sight extending from the CCD element at the desired feature coordinate through the focal point of the lens. If, however, two features on the object are used, then there is only one possible position (assuming the object is not allowed to rotate) in the world to which the object must be driven so that the two features are driven to their desired coordinates on the image plane. Therefore, we need to use two features on the object resulting in four state equations. The state equations for two feature points become

$$\begin{bmatrix} x_1(k+1) \\ y_1(k+1) \\ x_2(k+1) \\ y_2(k+1) \end{bmatrix} = \begin{bmatrix} x_1(k) \\ y_1(k) \\ x_2(k) \\ y_2(k) \end{bmatrix} + T\begin{bmatrix} \frac{f}{z_c(k)\, s_x} & 0 & -\frac{x_1(k)}{z_c(k)} \\ 0 & \frac{f}{z_c(k)\, s_y} & -\frac{y_1(k)}{z_c(k)} \\ \frac{f}{z_c(k)\, s_x} & 0 & -\frac{x_2(k)}{z_c(k)} \\ 0 & \frac{f}{z_c(k)\, s_y} & -\frac{y_2(k)}{z_c(k)} \end{bmatrix} \begin{bmatrix} \dot{x}_c(k) \\ \dot{y}_c(k) \\ \dot{z}_c(k) \end{bmatrix} , \tag{13}$$

which is a non-linear, multi-variable, time-varying system and can be represented in standard state-space matrix form as

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \boldsymbol{B}(k)\,\boldsymbol{u}(k) . \tag{14}$$

The control law is derived from the minimization of a cost function which allows weights to be placed on the future error and the current control inputs. The vector $\boldsymbol{x}_D(k+1)$ represents the desired coordinates of the two features on the object at the next time increment, and is constant over time. The cost function is of the form ($\boldsymbol{y}(k)=\boldsymbol{x}(k)$ by Eq. 11)

$$J(k+1) = [\boldsymbol{x}(k+1) - \boldsymbol{x}_D(k+1)]^T \boldsymbol{Q}\, [\boldsymbol{x}(k+1) - \boldsymbol{x}_D(k+1)] + \boldsymbol{u}(k)^T \boldsymbol{R}\boldsymbol{u}(k) . \tag{15}$$

By substituting Eq. 14 into Eq. 15, we can see that

$$J(k+1) = [\boldsymbol{x}(k) + \boldsymbol{B}(k)\,\boldsymbol{u}(k) - \boldsymbol{x}_D(k+1)]^T \boldsymbol{Q}\, [\boldsymbol{x}(k) + \boldsymbol{B}(k)\,\boldsymbol{u}(k) - \boldsymbol{x}_D(k+1)] + \\ + \boldsymbol{u}(k)^T \boldsymbol{R}\boldsymbol{u}(k) . \tag{16}$$

Differentiating Eq. 16 with respect to $\boldsymbol{u}(k)$, we get

$$\frac{\partial}{\partial \boldsymbol{u}(k)} J(k+1) = 2\boldsymbol{B}^T(k)\,\boldsymbol{Q}\, [\boldsymbol{x}(k) + \boldsymbol{B}(k)\,\boldsymbol{u}(k) - \boldsymbol{x}_D(k+1)] + 2\boldsymbol{R}\boldsymbol{u}(k) . \tag{17}$$

Setting the gradient equal to zero and rearranging we obtain

$$0 = 2\boldsymbol{B}^T(k)\boldsymbol{Q}[\boldsymbol{x}(k) - \boldsymbol{x}_D(k+1)] + 2\boldsymbol{B}^T(k)\boldsymbol{Q}\boldsymbol{B}(k)\boldsymbol{u}(k) + 2\boldsymbol{R}\boldsymbol{u}(k) \quad . \tag{18}$$

Solving for $\mathbf{u}(k)$ results in

$$\boldsymbol{u}(k) = -\left(\boldsymbol{B}^T(k)\boldsymbol{Q}\boldsymbol{B}(k) + \boldsymbol{R}\right)^{-1}\boldsymbol{B}^T(k)\boldsymbol{Q}[\boldsymbol{x}(k) - \boldsymbol{x}_D(k+1)] \quad . \tag{19}$$

From Eq. 19 it can be seen that the control law's gain matrix is time-varying. The matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, however, are constant and must be chosen by the user. Because we want to drive the entire state vector to its desired position, $\boldsymbol{Q}$ will be chosen as a diagonal matrix with equivalent terms on the diagonal. The matrix $\boldsymbol{R}$ is 3x3 and places a cost on each of the three control inputs. An advantage of using a control law in this form is that there is no need to create a trajectory from the object's starting position to its final position. By choosing the cost matrices properly, the control input $\boldsymbol{u}(k)$ at each time instant will drive the end-effector in such a way that the error term $[\boldsymbol{x}(k) - \boldsymbol{x}_D(k+1)]$ will be reduced by the maximum amount possible, while accounting for the costs placed on the control input.

The control law described by Eq. 19 does not take into account delays which exist in the system due to the large amount of data that must be processed when performing visual servoing. Modeling these delays by introducing them into the state-space description of the system (Eq. 14) results in an equation of the form

$$\boldsymbol{x}(k) = \boldsymbol{x}(k-1) + \boldsymbol{B}(k-d)\boldsymbol{u}(k-d) \quad , \tag{20}$$

where d represents the delay factor ($d \in \{1,2,3,...\}$). Using a cost function of the form

$$J(k+d) = [\boldsymbol{x}(k+d) - \boldsymbol{x}_D(k+d)]^T\boldsymbol{Q}[\boldsymbol{x}(k+d) - \boldsymbol{x}_D(k+d)] + \boldsymbol{u}(k)^T\boldsymbol{R}\boldsymbol{u}(k) \quad , \tag{21}$$

and solving for $\boldsymbol{u}(k)$ in a manner similar to the one described for the previous control law results in a control law of the form

$$\boldsymbol{u}(k) = -\left(\boldsymbol{B}^T(k)\boldsymbol{Q}\boldsymbol{B}(k) + \boldsymbol{R}\right)^{-1}\boldsymbol{B}^T(k)\boldsymbol{Q}\left[\boldsymbol{x}(k) - \boldsymbol{x}_D(k+d) + \sum_{m=1}^{d-1}\boldsymbol{B}(k-m)\boldsymbol{u}(k-m)\right] \quad . \tag{22}$$

It may be desirable to introduce an integrating term into the control law. This can be done by modifying the cost function described by Eq. 15 to account for changes in control inputs. The cost function becomes

$$J(k+1) = [\boldsymbol{x}(k+1) - \boldsymbol{x}_D(k+1)]^T\boldsymbol{Q}[\boldsymbol{x}(k+1) - \boldsymbol{x}_D(k+1)] + \\ + \boldsymbol{u}(k)^T\boldsymbol{R}\boldsymbol{u}(k) + \Delta\boldsymbol{u}(k)^T\boldsymbol{R}_D\Delta\boldsymbol{u}(k) \quad . \tag{23}$$

The matrix $\boldsymbol{R}_D$ is diagonal and places weights on the desired change for particular controller inputs. Using Eq. 14 and Eq. 23 results in a control law of the form

$$\boldsymbol{u}(k) = -\left(\boldsymbol{B}^T(k)\boldsymbol{Q}\boldsymbol{B}(k) + \boldsymbol{R} + \boldsymbol{R}_D\right)^{-1}\left[\boldsymbol{B}^T(k)\boldsymbol{Q}\{\boldsymbol{x}(k) - \boldsymbol{x}_D(k+1)\} - \boldsymbol{R}_D\boldsymbol{u}(k-1)\right] \quad . \tag{24}$$

Finally, we can account for delays and, simultaneously, introduce an integrating term by using a cost function of the form

$$J(k+d) = [\boldsymbol{x}(k+d) - \boldsymbol{x}_D(k+d)]^T\boldsymbol{Q}[\boldsymbol{x}(k+d) - \boldsymbol{x}_D(k+d)] + \\ + \boldsymbol{u}(k)^T\boldsymbol{R}\boldsymbol{u}(k) + \Delta\boldsymbol{u}(k)^T\boldsymbol{R}_D\Delta\boldsymbol{u}(k) \quad , \tag{25}$$

in conjunction with Eq. 20 to arrive at the control law

$$\boldsymbol{u}(k) = -\left(\boldsymbol{B}^T(k)\boldsymbol{Q}\boldsymbol{B}(k) + \boldsymbol{R} + \boldsymbol{R}_D\right)^{-1} \Bigg[\boldsymbol{B}^T(k)\boldsymbol{Q} \Big\{ (d+1)\boldsymbol{x}(k) - \boldsymbol{x}_D(k+d) - d\boldsymbol{x}(k-1) - - d\boldsymbol{B}(k-d)\boldsymbol{u}(k-d) + \sum_{m=1}^{d-1} \boldsymbol{B}(k-m)\boldsymbol{u}(k-m) \Big\} - \boldsymbol{R}_D\boldsymbol{u}(k-1) \Bigg]. \tag{26}$$

Experimental results show that the most effective control law is, in general, the one given by Eq. 26. The greatest improvement in control arises from using Eq. 21, the state equation in which system delays are taken into account. When system delays are not accounted for, the visual servoing system often oscillates about the steady state position, because, otherwise, control inputs are calculated based on sensor readings that lag by two to three cycles. The addition of the integrating term into the cost function has the main benefit of smoothing the system response by not allowing large changes from one iteration to the next to occur.

2.3 Feature Tracking

The measurement of the motion of the features on the image plane must be done continuously and quickly. The method used to measure this motion is based on optical flow techniques and is a modification of a method known as Sum-of-Squared-Differences (SSD) optical flow.[2] SSD optical flow is based on the assumption that the intensities around a feature point remain constant as that point moves across the image plane. The displacement of a point $\mathbf{p}_a = (x, y)$ at the next time increment to $\mathbf{p}_{a'} = (x + \Delta x, y + \Delta y)$, is found by finding the displacement $\Delta\boldsymbol{x} = (\Delta x, \Delta y)$ which minimizes the SSD measure

$$e(\boldsymbol{p}_a, \Delta\boldsymbol{x}) = \sum_W [I_a(x+i, y+j) - I_{a'}(x+i+\Delta x, y+j+\Delta y)]^2 \quad , \tag{27}$$

where I_a and $I_{a'}$ are the intensity functions from two successive images and W is the window centered about the feature point which makes up the feature template. For the algorithm implemented, W is 16x16 pixels, and possible displacements of up to $\Delta x = \Delta y = \pm 32$ pixels are considered. Features on the object that are to be tracked can be selected by the user, or a feature selecting algorithm can be invoked. Features with strong intensity gradients in perpendicular directions are typically the best features to select, such as corners.

In order to decrease the search space, a pyramidal search scheme (Figure 2) has been implemented which first searches a coarse resolution of the image that has 1/16 the area of the original image, using a feature template in which a W that is originally 32x32 is averaged to 8x8. After determining where the feature is in the coarse image, a finer resolution image that is 1/4 the original spatial resolution is searched with an original W of 16x16 which is averaged to 8x8 in an area centered about the location of the minimum SSD measure found in the coarse image. Finally, the full resolution image and the 16x16 feature template are used to pinpoint the location of the displaced feature.

The pyramidal scheme reduces the time required for the computation of the SSD algorithm by 89msec (from 116msec to 27msec) for a single feature over the method of computing the feature locations at the full resolution only, however, reliability can be sacrificed when the selected feature loses its tracking properties (strong perpendicular intensity gradients) at the coarser image resolutions. Since the search scheme first estimates where

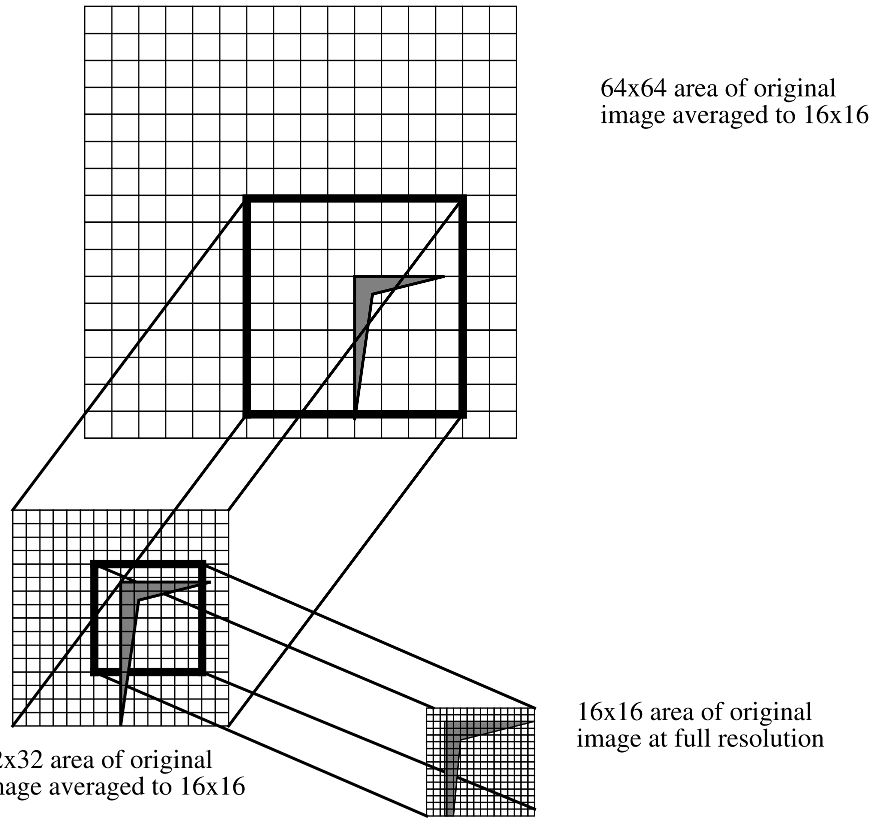

Figure 2. A pyramidal search scheme is used for the SSD optical flow algorithm in order to increase the overall sampling rate of the tracking system.

the feature is located based on the coarse image, it is critical that good features at coarse resolutions are tracked. When a user selects features, it is often not obvious that a particular feature may lose its tracking characteristics at coarse resolutions. Because of this, an automatic feature selector has been implemented which extends previous work in feature selection[17] by accounting for the different levels of resolution in the pyramidal search scheme.

2.4 Implementation

The visual tracking algorithm described previously has been implemented on a robotic assembly system consisting of three Puma560's called the TROIKABOT. One of the Puma's has a Sony XC-77RR camera mounted at its end-effector so that the camera can be easily repositioned. The camera is connected to an Innovision IDAS/150 Vision System. A VME bus with an Ironics IV-3230 (68030 CPU) running the ChimeraII real-time operating system lies between the vision system and the manipulators and handles communication between the two. The other two Puma560's can be visually servoed using the previously described tracking algorithms.

The robot cartesian controller used on the Puma is the Alter controller which allows path modifications in cartesian coordinates at control rates of 35Hz. The IV-3230 is used to pass control inputs to the Alter controller from the vision system using the serial communication protocol required by Alter. Since the control rate of Alter and the sampling rate of the vision system differ by a factor of more than three, the IV-3230 must account for the slower vision output and the faster controller input. This is done by passing a portion of the integrated control input to the Puma at each instance that Alter expects a command, such that, by the time the vision system determines the next control input, the last control input has been completely sent.

The Innovision IDAS/150 Vision System calculates the displacement of the feature point and the control inputs using the control law discussed in Section 2.3. A special floating point array processor on the IDAS/150 is used to calculate the optical flow of the feature, and a Heurikon 68030 board, also on the vision system, computes the control inputs. An image can be grabbed, displacements found for two features on an object, and control inputs along the tracking axes calculated at 6 Hz.

2.5 Experimental Results

The performance of the visual servoing algorithms is shown in Figures 3, 4, and 5. The two features on an object in the Puma560's gripper were commanded to move from their initial positions on the image plane to positions that would require them to move 20 pixels away from one another along Y in the image plane, therefore closer to the camera along Z in the camera frame. Translations of either 80 or 100 pixels in both X and Y were required. Figure 3 shows the errors in the pixel coordinates of the two features versus time. The two features on the object were chosen to have approximately the same Y coordinate in the camera frame in order to simplify the analysis of the motion, as Figure 4 shows. The depth of the object is updated at each time step based on the current control input, though it is possible to estimate the depth using a recursive least-squares estimation scheme. The performance of the system is not dependent on highly accurate knowledge of the object's depth.

Figure 5 shows the translation of the object along X, Y, and Z that was commanded in order to move the features to their desired positions on the image plane. The object being servoed began its motion from a depth of 50cm with respect to the camera frame. The overall translations were -4.7cm in X, 3.3cm in Y, and 8.0cm in Z.

Several tradeoffs must be considered when choosing the gains for the controller. For the experiments shown, the gains used are $\boldsymbol{Q}$ = diag(2.5, 2.5, 2.5, 2.5), $\boldsymbol{R}$ = diag(75.0, 75.0, 1.5), and $\boldsymbol{R_d}$ = diag(10.0, 10.0, 10.0). Large values for the diagonal terms in $\boldsymbol{Q}$ will cause initial motion of the object directly toward its final position, rather than the Z motion shown in Figure 5, but the final desired position of the object's features on the image plane cannot be achieved as the object will oscillate in X and Y about the final desired position. Another reason for the motion along Z shown in Figure 5 is due to the fact that it is necessary to place a smaller cost on the control input that represents camera motion parallel to the optical axis. This is because larger motions are required along this axis in order to change the position of the features on the image plane. Since there is a small cost in moving parallel to the optical axis, the controller can cause the object being servoed to initially move away from its final position along Z, since motion in this direction initially reduces the error in the features' position on the image plane due to the characteristics of the perspective transformation. Motion of this type is dependent on the initial position of the features on the image plane. Features which begin their motion near the piercing point on the image plane, tend to move directly toward their final positions, since motion along the optical axis has little effect on the coordinates of the feature on the image plane. Features initially located far from the piercing point, as Figure 4 illustrates for the experimental results presented, tend to move as the plot of motion in Z in Figure 5 demonstrates. Consideration of the pinhole camera model given by Eqs. 1 and 2 gives further insight into why this occurs.

A tradeoff must be made when using controllers such as the ones described in Section 2.3. Straight-line motion in cartesian space is difficult to achieve due to the nonlinearities inherent in camera-lens systems. It may be possible to create straight-line motion in cartesian space by creating nonlinear trajectories on the image plane that account for the perspective transformation of the camera-lens system and the values of the controller's gain matrices. Gain scheduling techniques may also be applied with success to improve the cartesian trajectory of the end-effector.

3. Dynamic Sensor Placement

3.1 Introduction

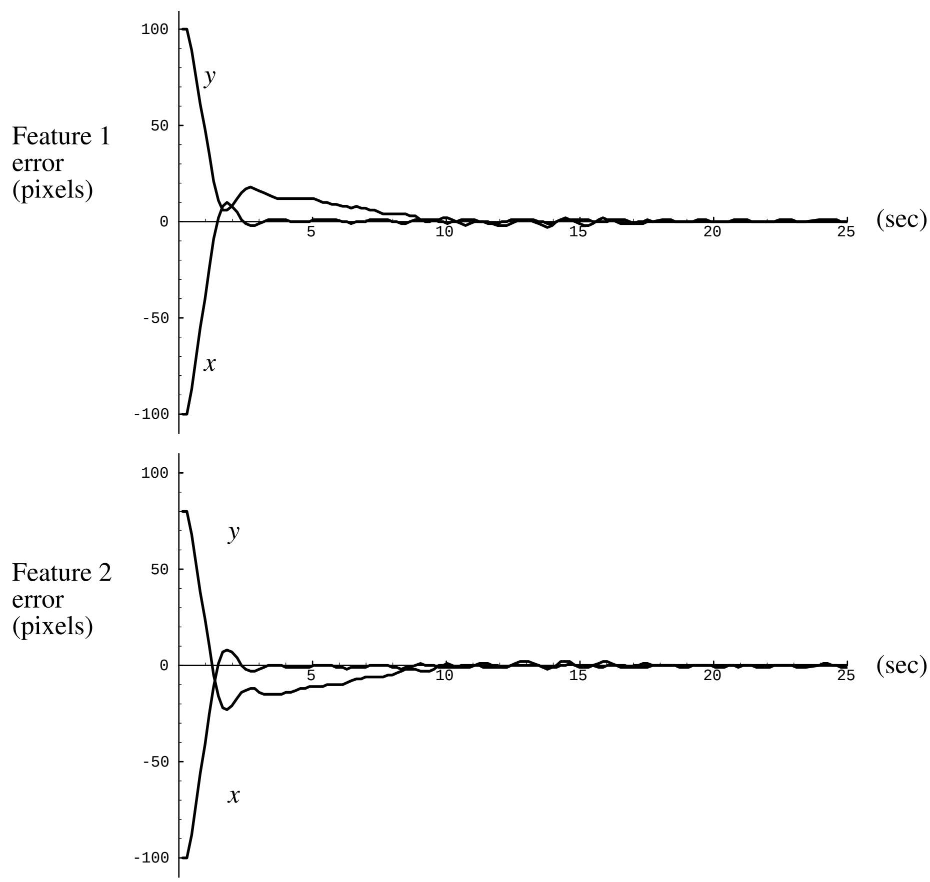

Figure 3. Errors in feature image coordinates versus time.

The previous section described an algorithm for visually servoing a manipulator in 3-D using a statically mounted camera. In a robotic assembly workcell, however, a single camera position will almost surely be inadequate for providing sufficient visual feedback throughout the entire assembly process. Several cameras could be statically mounted at various points throughout the workspace, or a single camera could be mounted at the end-effector of a manipulator, and the camera could be servoed based on the assembly actions occurring in the workcell. Ignoring for the moment the manipulators holding the parts to be servoed, the manipulator with the camera mounted at its end-effector becomes an eye-in-hand system, and the control issues associated with these types of visual servoing systems must be addressed. In addition to eye-in-hand control issues, the placement of the camera in the work cell must be considered. Assuming a camera with a fixed focal length lens is used, the eye-in-hand system can be used to move the camera closer to the assembly task being performed in order to increase the spatial resolution of the sensor such that the static visual servoing algorithm can servo the object to a sufficient accuracy. Experimental results indicate that for objects at depths of approximately 50cm, the position of the objects along the optical axis could be resolved to approximately 4mm. Changing the viewing direction of the object by the camera would increase the resolution of the objects in directions formerly along the optical axis. Simultaneously, the object being observed must remain within the camera's field-of-view and depth-of-field and occlusions must be avoided. When servoing a camera mounted at a manipulator's end-effector, it is also important that the manipulator holding the camera maintains a good configuration far enough from kinematic

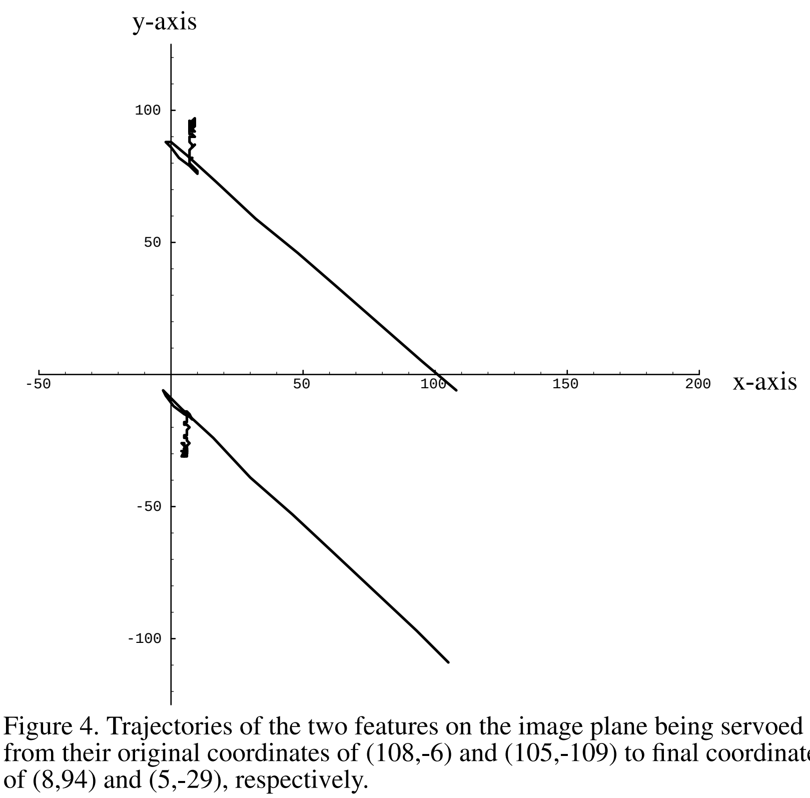

Figure 4. Trajectories of the two features on the image plane being servoed from their original coordinates of (108,-6) and (105,-109) to final coordinates of (8,94) and (5,-29), respectively.

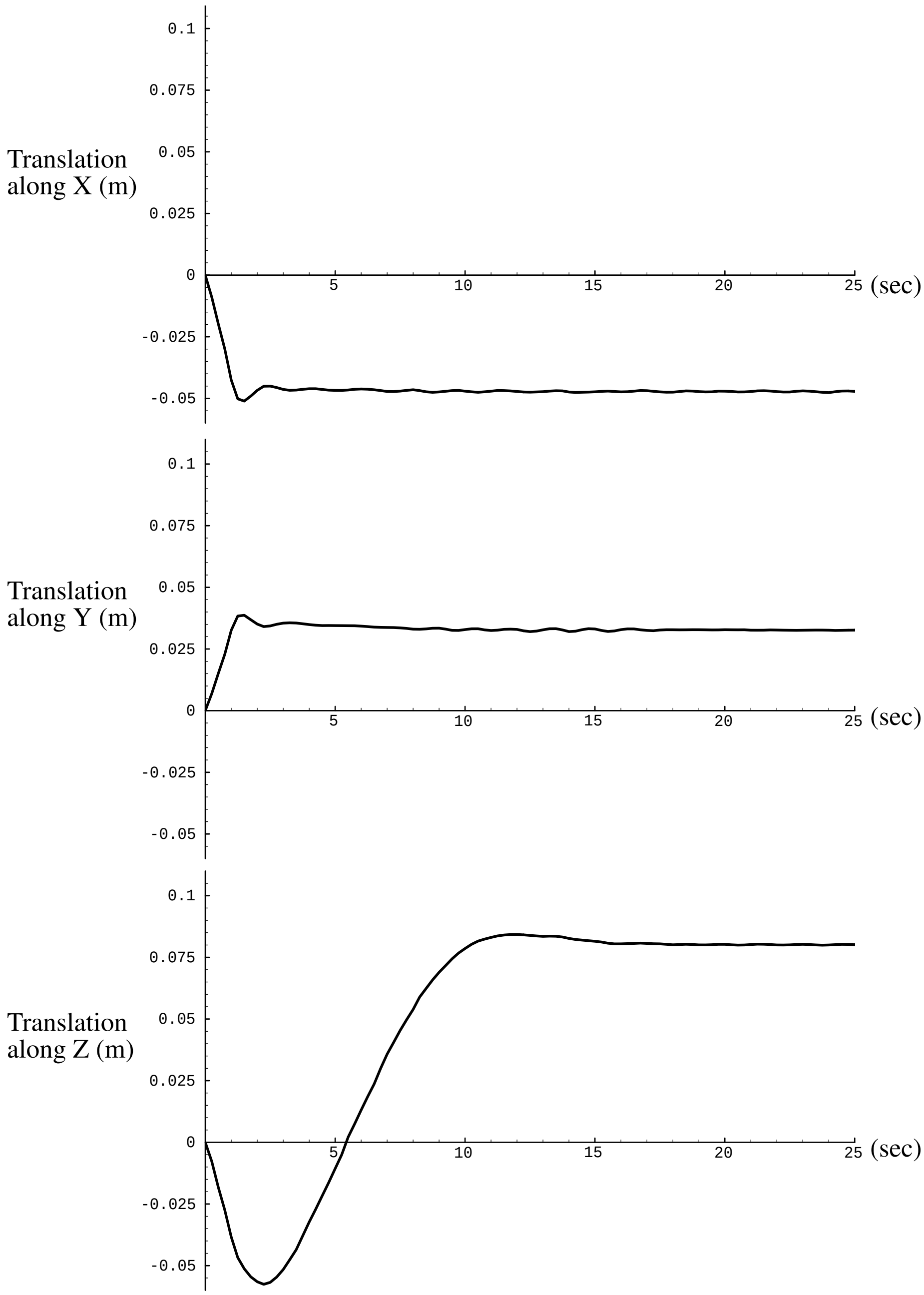

Figure 5. Object translation in the camera frame versus time.

singularities so that the manipulator cartesian control algorithms are properly conditioned.

3.2 Previous Work

In the past, camera placement has been determined by accounting for occlusions, field-of-view, depth-of-field, and/or camera resolution.[5,16,19] In none of these cases, however, is the camera actually servoed based on visual data. For assembly, the camera must move in real-time, so the placement of the camera must be determined quickly, and visual tracking algorithms can be effectively applied. The configuration of the manipulator must also be taken into account in the control strategy, unless the working region of the workcell can be severely constrained. Another factor which must be considered is the maximum optical flow which the visual tracking algorithms can determine, since the optical flow of features moving at a constant non-zero speed increases as depth decreases.

3.3 Sensor Placement/Visual Tracking Hybrid Controller

The controlled active vision framework[13] is used to model the dynamic sensor placement/ tracking problem. Since the camera is mounted at the end-effector of a manipulator that can control camera motion in the six cartesian directions, a hybrid controller can be designed. The visual tracking problem can be considered a 2-D problem, where the two cartesian directions in the camera frame (see Figure 1) along which tracking takes place is along and about the Y axis. This keeps the object in the center of the image and ensures that the field-of-view constraint on camera motion will be satisfied, assuming the camera is not too near the object. Motion along the X axis can be used to obtain a desired view of the object in order to increase spatial resolution along a particular direction, or to help avoid occlusions. Motion about the X and Z axes is constrained to be zero. This keeps the camera's optical axis perpendicular to the object as the camera is servoed about the object. This leaves motion along the Z axis unconstrained. It is motion along Z which will be used to ensure that: (a) the entire object appears within the field-of-view of the camera; (b) the object is within the maximum and minimum depth-of-field limits of the sensor; (c) the object can be viewed with a sufficient spatial resolution; and (d) the manipulator holding the camera remains in a non-singular configuration. We should also consider the feature tracking ability of our optical flow algorithm. If the camera is too near the object, relatively low servoing speeds of the object can cause the features to be lost because the features' optical flow will be high. This will place an additional constraint on motion along the Z axis. For the experiments performed, however, our depth-of-field constraint and the maximum allowable commanded cartesian velocities ensured that the optical flow of the features was not beyond the tracking capabilities of our SSD optical flow algorithm.

3.4 Visual Tracking Model and Control

For the dynamic sensor placement model, the pin-hole camera model in Figure 1 still applies, however, instead of the feature moving with a velocity of $[\dot{x}_c \;\; \dot{y}_c \;\; \dot{z}_c]^T$, the camera now moves with this translational velocity $\boldsymbol{T} = [\dot{x}_c \;\; \dot{y}_c \;\; \dot{z}_c]^T$ and a rotational velocity $\boldsymbol{R} = [\omega_{xc} \; \omega_{yc} \; \omega_{zc}]^T$. The equations of optical flow for this tracking model are obtained by first changing the signs of Eqs. 3 and 4 (since the camera is moving instead of the feature). The result is combined with the velocity induced by camera motion of a point $\boldsymbol{P}$ in the camera frame

$$\frac{d\boldsymbol{P}}{dt} = -\boldsymbol{T} - \boldsymbol{R} \times \boldsymbol{P} . \qquad (28)$$

If $x(t) = (x_i(t) - x_p)$ and $y(t) = (y_i(t) - y_p)$ are the projections of $\boldsymbol{P}$ on the image plane, and z_c is the Z-coordinate of $\boldsymbol{P}$ in the camera frame, then the optical flow equations become[15]

$$u = -\frac{f\dot{x}_c}{z_c(t)\,s_x} + \frac{x(t)\,\dot{z}_c}{z_c(t)} + \frac{x(t)\,y(t)\,s_y\omega_{xc}}{f} - \left(\frac{f}{s_x} + \frac{x^2(t)\,s_x}{f}\right)\omega_{yc} + \frac{y(t)\,s_y}{s_x}\omega_{zc} \tag{29}$$

$$v = -\frac{f\dot{y}_c}{z_c(t)\,s_y} + \frac{y(t)\,\dot{z}_c}{z_c(t)} + \left(\frac{f}{s_y} + \frac{y^2(t)\,s_y}{f}\right)\omega_{xc} - \frac{x(t)\,y(t)\,s_x\omega_{yc}}{f} - \frac{x(t)\,s_x}{s_y}\omega_{zc} \quad . \tag{30}$$

For the model previously described, $\omega_{xc} = \omega_{zc} = 0$. The control inputs for tracking are the camera velocities along and about Y. The camera velocities along Z and X are assumed to be known since they are based on the manipulator configuration and sensor placement criteria. Using Eqs. 9 and 10, we can write the state equations as

$$x(k+1) = x(k) - T\frac{f\dot{x}_c(k)}{z_c(k)\,s_x} + T\frac{x(k)\,\dot{z}_c(k)}{z_c(k)} - T\left(\frac{f}{s_x} + \frac{x^2(k)\,s_x}{f}\right)\omega_{yc}(k) \tag{31}$$

$$y(k+1) = y(k) - T\frac{f\dot{y}_c(k)}{z_c(k)\,s_y} + T\frac{y(k)\,\dot{z}_c(k)}{z_c(k)} - T\frac{x(k)\,y(k)\,s_x\omega_{yc}(k)}{f} \quad . \tag{32}$$

The above equations can be rewritten as

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \boldsymbol{F}(k) + \boldsymbol{B}(k)\,\boldsymbol{u}(k) \quad , \tag{33}$$

where $\boldsymbol{x}(k) = [x(k)\ y(k)]^T$, $\boldsymbol{u}(k) = [\dot{y}_c\ \ \omega_{yc}]^T$,

$$\boldsymbol{F}(k) = \begin{bmatrix} -T\frac{f\dot{x}_c(k)}{z_c(k)\,s_x} + T\frac{x(k)\,\dot{z}_c(k)}{z_c(k)} \\ T\frac{y(k)\,\dot{z}_c(k)}{z_c(k)} \end{bmatrix} \quad , \tag{34}$$

and

$$B(k) = \begin{bmatrix} 0 & -T\left(\frac{f}{s_x} + \frac{x^2(k)\,s_x}{f}\right) \\ -T\frac{f}{z_c(k)\,s_y} & -T\frac{x(k)\,y(k)\,s_x}{f} \end{bmatrix} \quad . \tag{35}$$

A control law similar to the ones derived in Section 2.2 can be obtained. The cost function is the same as the one in Eq. 15, however, the vector $\boldsymbol{F}(k)$ in Eq. 34 results in a control input of

$$\boldsymbol{u}(k) = -\left(\boldsymbol{B}^T(k)\,\boldsymbol{Q}\boldsymbol{B}(k) + \boldsymbol{R}\right)^{-1}\boldsymbol{B}^T(k)\,\boldsymbol{Q}\,[\boldsymbol{x}(k) + \boldsymbol{F}(k) - \boldsymbol{x}_D(k+1)] \quad . \tag{36}$$

As before, $\boldsymbol{Q}$ and $\boldsymbol{R}$ are selected by the user based on desired system performance. The features are tracked using the same SSD optical flow algorithm described in Section 2.3.

3.5 Depth-of-Field, Field-of-View, and Spatial Resolution Constraints

Since motion about X and Z is constrained to be zero in order to keep the optical axis of the camera perpendicular to the object, this leaves motion along the X and Z axes for avoiding occlusions, maintaining a good manipulator configuration, keeping the features in focus and in the field-of-view, and maintaining a sufficient camera resolution. A fixed focal length lens is used, so the depth-of-field remains constant. An object exceeds the field-of-view when the features being tracked on the object no longer fall on the image plane. Camera resolution should remain as high as possible, so the camera should be as close to the object as possible without exceeding depth-of-field and field-of-view constraints. The camera should also be far enough from the object so that the optical flow of the object remains within the tracking capabilities of the SSD algorithm.

Keeping features in focus is important to the success of the SSD optical flow algorithm. Krotkov investigated several techniques for measuring the sharpness of focus.[11] He found the most effective method to be one which employs the Tenengrad measure in which the magnitude of the intensity gradient is maximized. However, since our SSD algorithm calculates the sum-of-squared differences between a feature window and the current image scene, the grey-level variance technique can be embedded into our feature tracking algorithms in a more computationally efficient manor. In addition, the grey-level variance technique tends to be more scale-invariant than the Tenengrad measure as the camera is moved toward and away from the object being tracked. For the grey-level variance technique the variance is calculated as

$$\sigma^2 = \frac{1}{N^2}\sum_{x=1}^{N}\sum_{y=1}^{N}[I(x,y)-\mu]^2, \tag{37}$$

where $N = 16$, $I(x,y)$ is the grey-level value at (x,y) in the current camera input, and μ is the average grey-level value of the feature window in the current image. The grey-level variance is calculated only over the matched feature in the image. As the grey-level variance increases, the sharpness of focusing increases. At low variances, the feature becomes less sharp and the depth-of-field constraint is violated.

The spatial resolution constraint is necessary in robotic assembly for ensuring that two parts being visually servoed by two different manipulators (or a part in a gripper and another part in a fixture) can be brought near enough one another so that the assembly can be successfully performed. For servoing parts perpendicular to the optical axis (parallel to the image plane), the resolution of the sensor is greatest. Along the optical axis the resolution of the location of parts is smallest, therefore, motion along the camera's X axis can be used to increase the camera resolution in directions not currently in the plane of the end-effector frame parallel to the image plane. This is because as the camera moves along the camera's X axis, and visual servoing occurs about the Y axis, the depth of the feature remains constant, but the plane of the end-effector frame perpendicular to the optical axis rotates as the camera servos around the object. The control law used to servo the object (Eq. 19) must account for the commanded change in the transformation from the camera frame to the manipulator's end-effector frame.

3.6 Singularity Avoidance

When tracking a moving object with an eye-in-hand system or when visually servoing a manipulator using a static camera, it is necessary that robot motion commands are given in cartesian space. A well-known problem with controlling a manipulator in cartesian space occurs when the manipulator passes through or near a kinematic singularity, because cartesian based control algorithms employing the Jacobian inverse become numerically unstable and unrealizable at or near singularities. There are two different types of singularities which a manipulator may encounter. An internal singularity occurs when two axes of the manipulator become aligned within the boundaries of the manipulator's workspace, and an external singularity occurs when the manipulator reaches the boundaries of its workspace. When a manipulator encounters either of these singularities, the visual tracking system will fail. When visually tracking an object it is also important that the manipulator is in a configuration that allows motion in all directions of possible object motion without requiring extremely large joint velocities from any actuator, because the future motion of the object may be unknown. This requires that the manipulator should not be near singularities, as well.

To avoid singularities we use a version of a previously proposed cartesian singularity avoidance scheme.[12] This scheme computes the cartesian gradient of Yoshikawa's manipulability measure,[20] $\nabla_x w$, quickly and efficiently from the explicitly formulated gradient of manipulability in joint space, $\nabla_\theta w$, where w represents manipulability. The method consists of first explicitly calculating $\nabla_\theta w$ which, for a non-redundant manipulator, is equivalent to calculating $\nabla_\theta |det(J(\theta))| = \partial |det(J(\theta))| / \partial\theta$. The homogeneous transformation matrix of the end-effector resulting from motion along this gradient is determined by simulating the motion of the manipulator along $\nabla_\theta |det(J(\theta))|$ in joint space. Let

$$ {}^b_eT = \Psi(\theta) , \tag{38} $$

where ${}^b_eT$ represents the 4x4 homogeneous transformation matrix of the end-effector frame with respect to the base, and $\Psi(\theta)$ is the forward kinematic solution of the manipulator and is a function of θ, the manipulator's joint angles. The differential homogeneous transformation matrix Δ that takes the manipulator's end-effector from its current position and orientation to the transformation determined from simulation is calculated according to the formula

$$ \Delta = \left({}^b_{e1}T\right)^{-1} \cdot {}^b_{e2}T = \Psi(\theta - \nabla_\theta w)^{-1} \cdot \Psi(\theta + \nabla_\theta w) \quad . \tag{39} $$

Figure 6 illustrates this method for a simple two-link manipulator. The differential

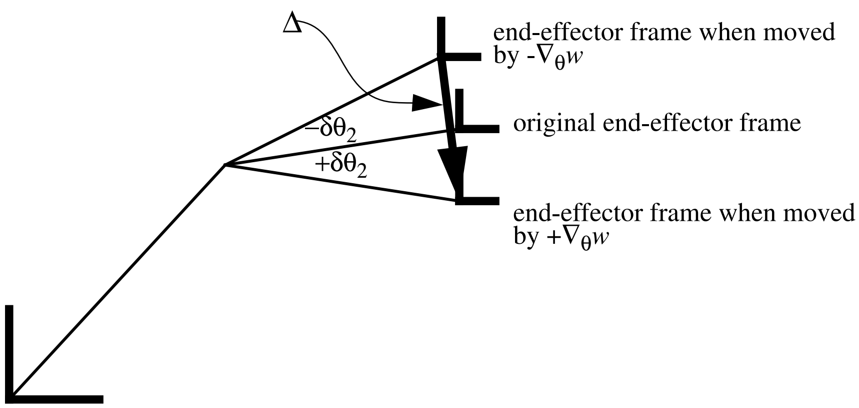

Figure 6. The cartesian manipulability gradient $\nabla_x w$ is represented by the differential transformation matrix Δ for this two-link manipulator.

transformation Δ is with respect to the tool frame and can be decomposed into six components which represent the cartesian gradient of manipulability $\nabla_x w = \nabla_x |det(J(\theta))|$ for a non-redundant manipulator. By calculating the cartesian gradient in this fashion, the inversion of the 6x6 Jacobian matrix is avoided in its calculation, although the cartesian controller must still invert the Jacobian if the particular cartesian controller requires the Jacobian inverse. Motion of the manipulator's tool frame in the direction that Δ indicates will maximally increase the manipulability of the manipulator. When visually tracking, however, the manipulator is only allowed to increase manipulability by moving along the component of the gradient which projects onto the camera's optical axis, since this is the unconstrained tracking axis. The control input for motion along the optical axis is determined by a simple proportional controller. This input is expressed as

$$u_{oa}(k) = \dot{z}_c(k) = G((\nabla_x w) \cdot \boldsymbol{x_{oa}}) \tag{40}$$

where $\boldsymbol{x}_{oa}$ is a unit vector along the optical axis of the camera, and G is the proportional gain constant.

Figure 7 shows experimental results which demonstrate that a singularity avoidance scheme of this type applied to a 2-D planar hand-eye visual tracking system can increase the tracking region significantly. The tracking distance along X increased from 45cm without singularity avoidance to over 70cm with singularity avoidance. Assuming an approximately spherical workspace, this demonstrates an extension of the possible tracking area by a factor of more than 2.8 when using the previously described singularity avoidance scheme.

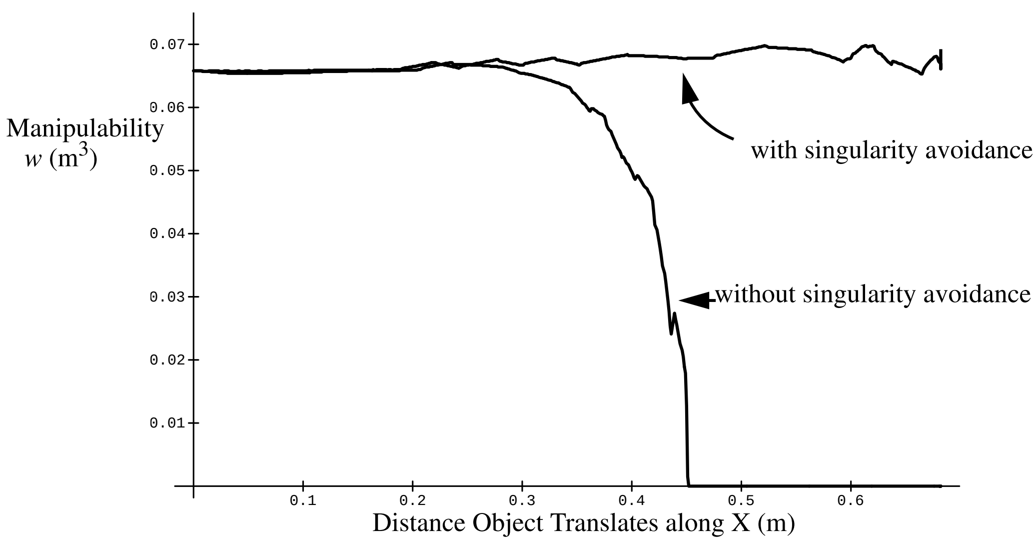

Figure 7. Manipulability *w* ("distance" from a singularity) versus distance target object travels in X with and without singularity avoidance for 2-D planar tracking.

3.7 Experimental Results

Figures 8 through 11 show experimental results obtained from a hand-eye system which tracks features on an object using the control law given by Eq. 36, while maintaining a good manipulator configuration. The focus measure previously discussed was recorded, as well as the distance between points on the object per pixel (an inverse spatial resolution measure), as determined by the known distance between features on the object. Data was collected for two trials, one with singularity avoidance enabled, and the other without considering the manipulator's configuration. Figure 8 shows the path of the manipulator around the object. Without singularity avoidance, the camera travels in a circular path around the object located on the graph at (0.0m, 0.5m). The manipulator soon reaches a singularity after travelling about 24cm, as shown in Figure 9. When singularity avoidance is enabled, the manipulator continuously repositions itself to maintain a good configuration

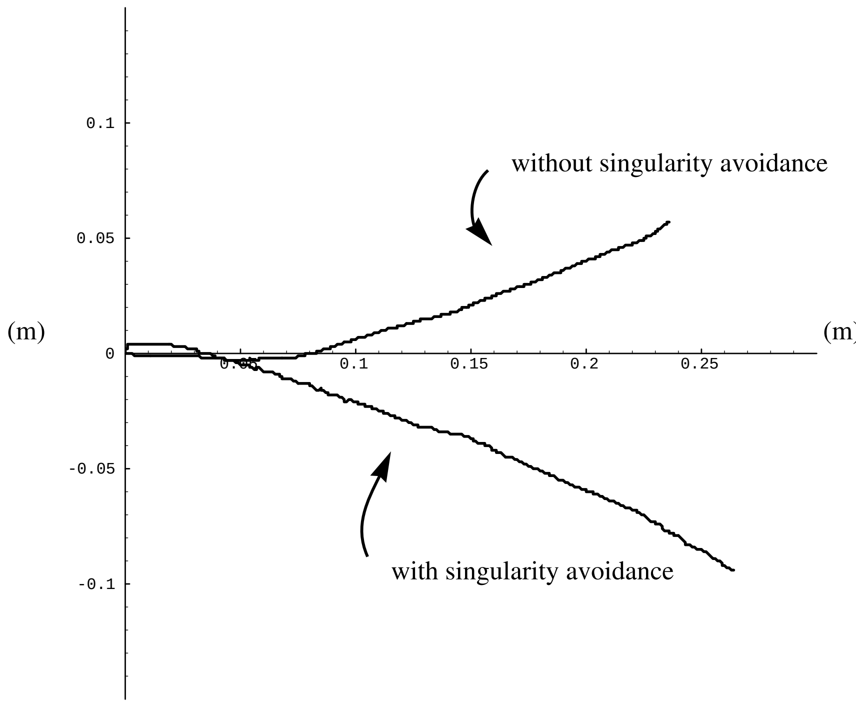

Figure 8. Path of camera in the horizontal plane with and without singularity avoidance. The object being tracked is located at (0, 0.5m).

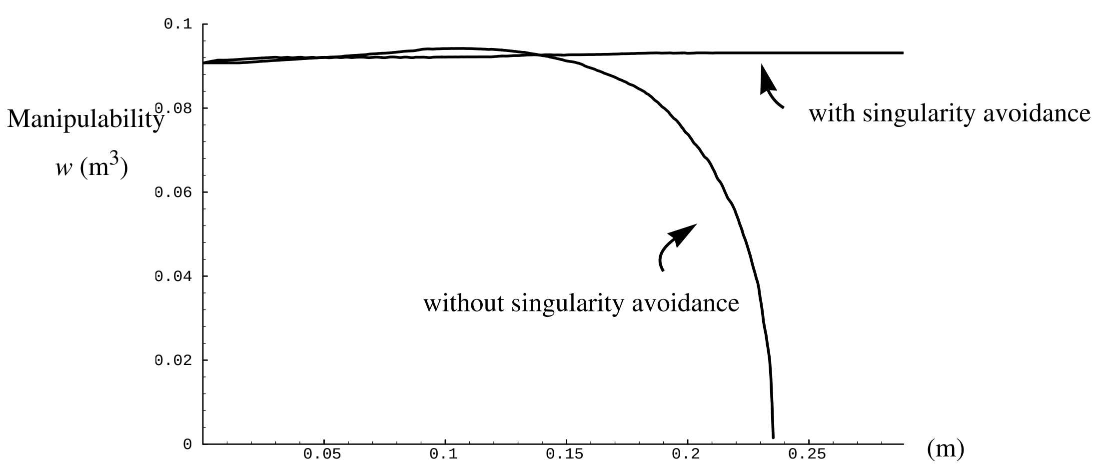

Figure 9. Manipulability w ("distance" from a singularity) versus distance camera travels with and without singularity avoidance.

throughout the tracking process.

Figures 10 and 11 demonstrate the change in focus and spatial resolution as the object is tracked. Without singularity avoidance, the distance between the object and the camera remains constant, and the measures do not change significantly, though some noise is evident, particularly in the focus measure used. When singularity avoidance is enabled the distance of the object from the camera increases. This results in a less focused image, and reduced spatial resolution. Future work is aimed toward defining an effective cost function which allows tradeoffs between focus, spatial resolution, and manipulator configuration.

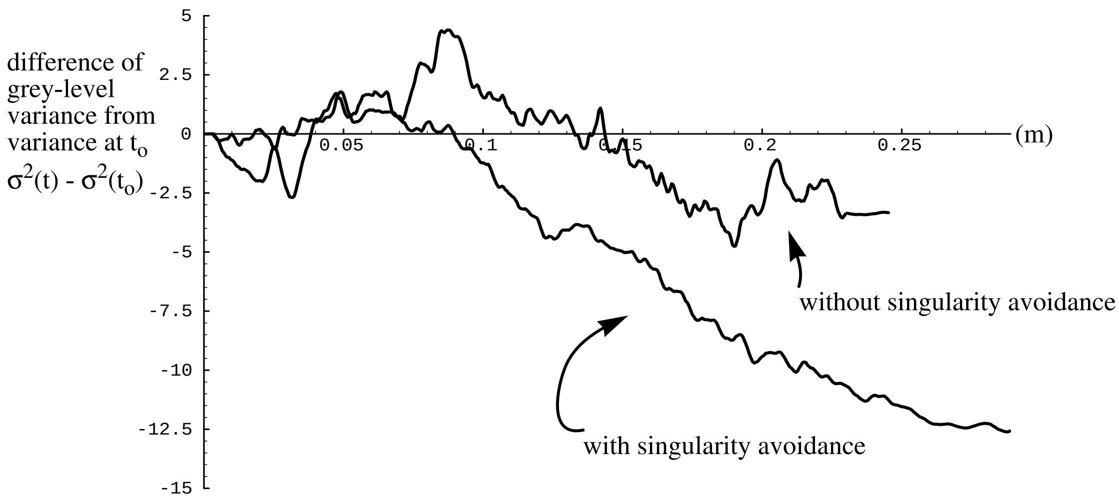

Figure 10. Focus measure (difference of grey-level variance from variance at t_o) versus distance camera travels with and without singularity avoidance.

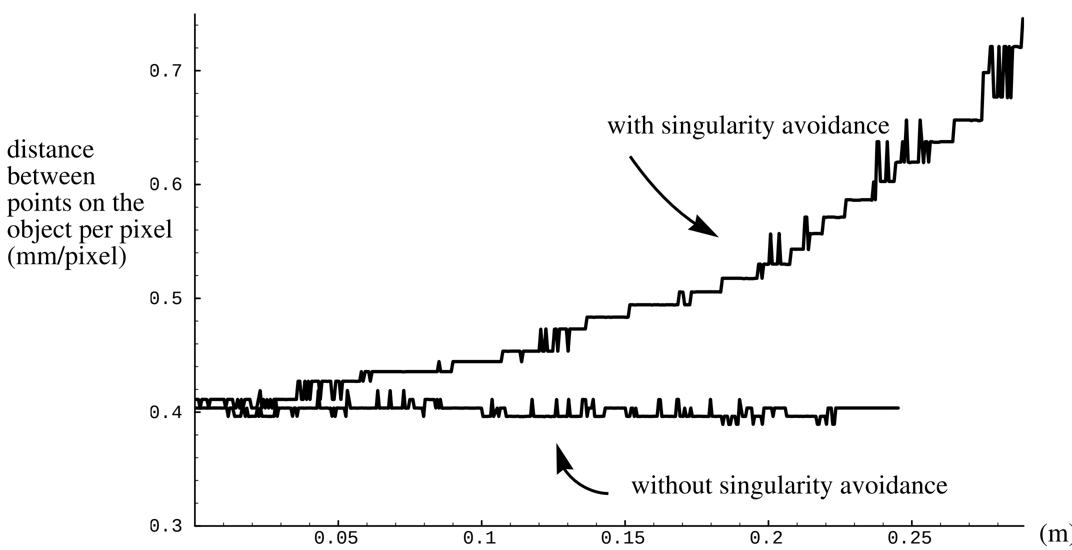

Figure 11. Distance between points on the object per pixel versus distance camera travels with and without singularity avoidance.

4. Automatic Calibration of the Camera-Object Transformation

4.1 Introduction

The most important benefit that results from integrating visual servoing techniques into robotic assembly is the increased robustness that is obtained when using a non-contact sensor to monitor a relatively large area of the workspace and provide low-level feedback to manipulators for real-time path correction. When using a vision system to servo the manipulator, it is not required that the precise transformation from the original coordinate frame of the end-effector to the destination point's frame be known. Thus, precise calibration of the entire workspace is unnecessary. It is still important, however, that the transformation from the camera's CCD array to the tool frame of the servoed manipulator is precisely known. This is necessary so that the motion of the end-effector is correctly determined in world coordinates by the visual tracking algorithm. The control commands then sent to the manipulator will result in the desired motion. In this section a recursive least-squares estimator is described that estimates this transformation on-line and adapts the visual servoing system based on the estimated transformation.

4.2 Modeling and Control

To determine the transformation parameters, the 2-D visual servoing case will be studied. The equations for the 2-D case are the same as the equations derived in Section 2.2, except that motion along Z is not considered. Therefore, Eqs. 3 and 4 simplify to

$$u(t) = \dot{x}_i(t) = \frac{f\dot{x}_c(t)}{z_c s_x} \tag{41}$$

$$v(t) = \dot{y}_i(t) = \frac{f\dot{y}_c(t)}{z_c s_y} \quad , \tag{42}$$

and the state equations become

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \tag{43}$$

where $\boldsymbol{x}(k) = [(x_i(k)-x_p)\ (y_i(k)-y_p)]^T$, $\boldsymbol{u}(k) = [\dot{x}_c \ \ \dot{y}_c\ ]^T$, and

$$\boldsymbol{B} = T\begin{bmatrix} \frac{f}{z_c s_x} & 0 \\ 0 & \frac{f}{z_c s_y} \end{bmatrix} \quad . \tag{44}$$

The control laws derived in Section 2.2 apply to this system as well.

4.3 Recursive Least-Squares Formulation

From the state equations it is apparent that the transformation from the image plane (x_i, y_i) to the tool frame plane (x_c, y_c) is represented by $\boldsymbol{B}$. This transformation can be estimated on-line by using a recursive least-squares estimator that uses the control inputs and the calculated optical flow to estimate the transformation $\boldsymbol{B}$. Using the certainty equivalence principle, the updated value of the estimate of $\boldsymbol{B}$ can be used in the control law at each sampling instance.

For the recursive least-squares formulation, an equation of the form

$$\boldsymbol{y}(k) = \boldsymbol{w}^T(k)\,\theta(k) \tag{45}$$

is desired, where $\boldsymbol{w}$ is the regressor vector, θ is the vector of parameters, the elements of $\boldsymbol{B}$ in this case, and $\boldsymbol{y}$ is the output of the system. If the state equation is first rewritten as

$$\boldsymbol{x}(k) - \boldsymbol{x}(k-1) = \boldsymbol{B}\boldsymbol{u}(k-d) \quad , \tag{46}$$

and considering that $\boldsymbol{B}$ contains four parameters and has two inputs and two outputs, the identification scheme is a multi-output-multi-input (MIMO) scheme, and can be written in the form

$$\begin{bmatrix} x(k) - x(k-1) \\ y(k) - y(k-1) \end{bmatrix} = \begin{bmatrix} x_c(k-d) & y_c(k-d) & 0 & 0 \\ 0 & 0 & x_c(k-d) & y_c(k-d) \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{12} \\ b_{21} \\ b_{22} \end{bmatrix} \quad , \tag{47}$$

where d represents the delay in the system for the control input to be realized by the vision system using the SSD algorithm's optical flow algorithm. For this MIMO system, the covariance matrix $\boldsymbol{P}$ is 4x4, thus, a 4x4 matrix must be inverted. As the main factor limiting system performance is the sampling/control rate, more efficient calculations are desired. A MIMO system of the above form can be decomposed into two MISO (multi-input-single-output) systems of the forms

$$x(k) - x(k-1) = \begin{bmatrix} x_c(k-d) & y_c(k-d) \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{12} \end{bmatrix} = \boldsymbol{w}^T(k-d)\theta_1(k) = y_1(k) \tag{48}$$

$$y(k) - y(k-1) = \begin{bmatrix} x_c(k-d) & y_c(k-d) \end{bmatrix} \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix} = \boldsymbol{w}^T(k-d)\theta_2(k) = y_2(k) \quad . \tag{49}$$

The parameter vectors can be recursively determined by the following equations (j=1,2)

$$\theta_j(k) = \theta_j(k-1) - \frac{\boldsymbol{P}(k-1)\boldsymbol{w}(k-d)\left(\theta_j^T(k-1)\boldsymbol{w}(k-d) - y_j(k)\right)}{1 + \boldsymbol{w}^T(k-d)\boldsymbol{P}(k-1)\boldsymbol{w}(k-d)} \tag{50}$$

$$\boldsymbol{P}(k) = \boldsymbol{P}(k-1) - \frac{\boldsymbol{P}(k-1)\boldsymbol{w}(k-d)\boldsymbol{w}^T(k-d)\boldsymbol{P}(k-1)}{1 + \boldsymbol{w}^T(k-d)\boldsymbol{P}(k-1)\boldsymbol{w}(k-d)} . \tag{51}$$

The entire visual servoing system is shown in block diagram form in Figure 12.

4.4 Experimental Results

Experimental results are shown in Figures 13, 14, and 15. The results were obtained using the same hardware implementation described in Section 2.4, except that sampling rates of 10Hz were achieved since only one feature had to be tracked. A sawtooth path consisting of six coordinates, each requiring x and y translations of 100 pixels in both directions for each step of the path, is used for comparing the performance of a well calibrated system with a poorly calibrated system with and without adaptation. To obtain a poorly calibrated system, the tool frame of the manipulator is rotated by 20° in a plane parallel to the

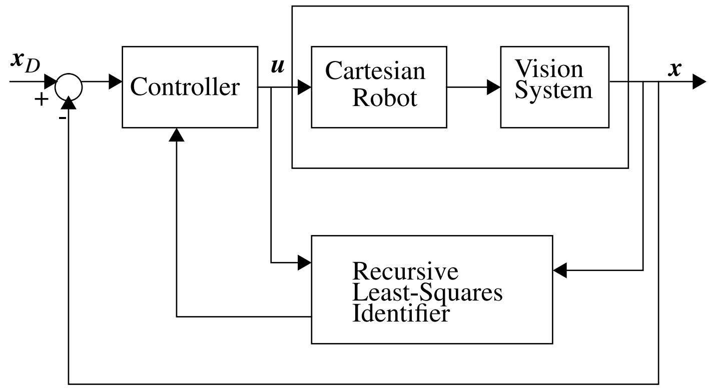

Figure 12. Block diagram of the self-tuning visual servoing system.

image plane. This simulates calibration errors in estimating the orientation of the camera or the tool frame, or errors in pixel size. Even the well calibrated system had errors in calibration, which were made apparent by the estimator. These errors were corrected to further improve the calibration of the well calibrated system for the experiments performed.

Figure 13 shows the pixel errors in x and y versus time for the well calibrated system as it traverses the sawtooth path. Approximately 4.2 seconds are required for the system to converge to each step input, 25.6 seconds overall, and overshoot is minimal. When the system is purposely uncalibrated, the performance significantly degrades as shown in Figure 14. Convergence for each step input is somewhat variable, depending on the direction of the step, and takes between 5 and 7 seconds, 33.0 seconds overall, and overshoot is

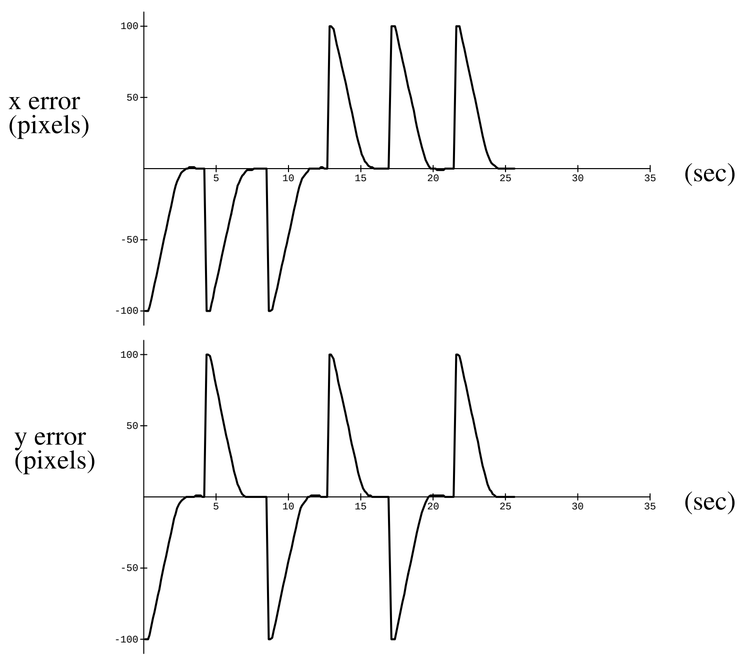

Figure 13. Positional errors of the well calibrated system when tracking a sawtooth path.

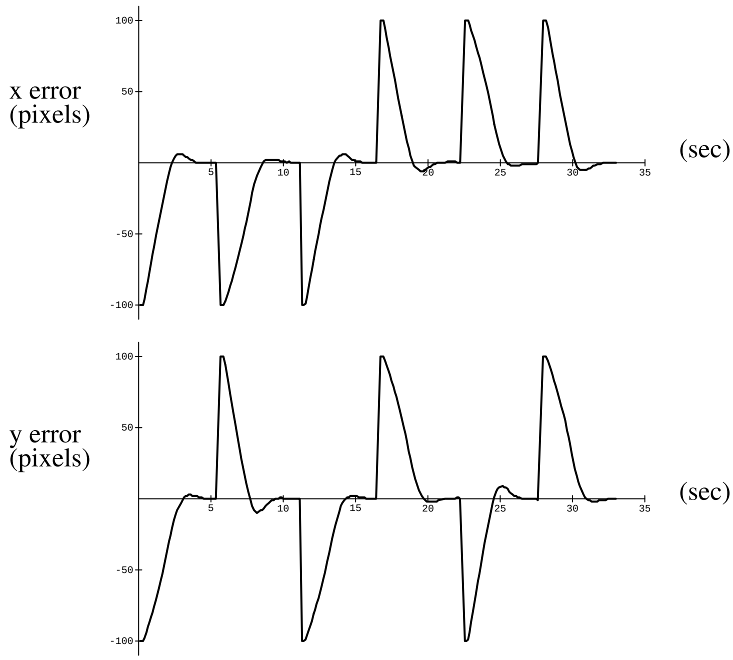

Figure 14. Positional errors of the poorly calibrated system when tracking a
a sawtooth path.

quite apparent.

When the recursive least-squares estimator is used with the poorly calibrated system, performance improves almost to the degree of the original well-calibrated system, as Figure 15 shows. The covariance matrix $\boldsymbol{P}$ is initially chosen to be diag(0.1, 0.1), and the transformation matrix $\boldsymbol{B}$ is initialized to

$$\begin{bmatrix} 4.364 & 0.0 \\ 0.0 & 3.692 \end{bmatrix} .$$

Convergence for the self-tuned system takes 5.0 seconds for the first step input, and improves for each succeeding input. The system takes 27.5 seconds to traverse the path, and overshoot is successively reduced until it is negligible.

5. Summary

In this paper we have addressed some of the important issues surrounding the integration of visual servoing into robotic assembly systems. Our motivation for employing visual servoing is to increase the robustness of robotic assembly systems that are automatically programmed based on geometric models of mechanical parts and assemblies that designers create using CAD solid modelers. The intention is to demonstrate a completely integrated design and manufacturing environment that has the capability of performing precision assemblies automatically. Towards this goal, we have successfully demonstrated the ability to visually servo objects with 3-D translational motion using a single camera. We have also shown the ability to dynamically place the sensor at various poses in the workcell

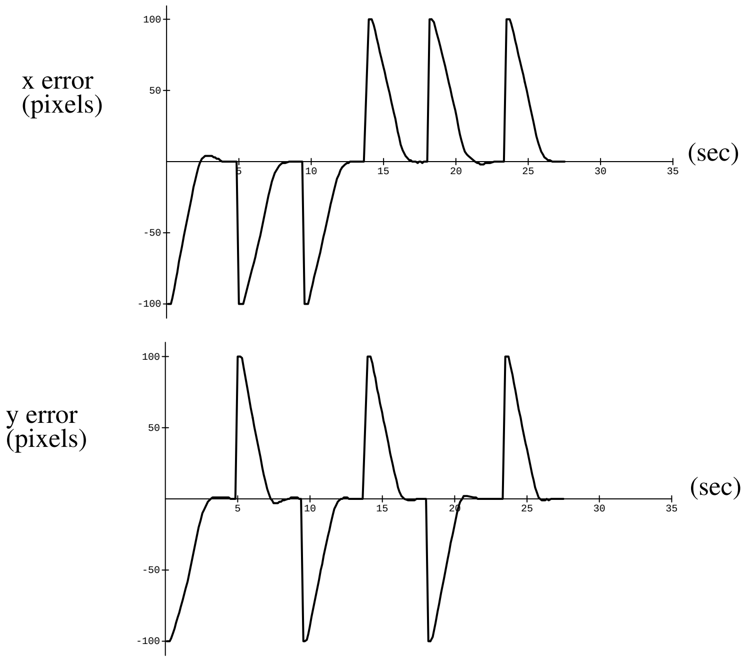

Figure 15. Positional errors of the poorly calibrated system with self-tuning when tracking a sawtooth path.

using visual tracking techniques combined with real-time sensor placement measures. Finally, we have demonstrated the use of adaptive control techniques to improve system performance by determining the camera-object transformation automatically.

Several areas of research are being pursued on the Rapid Assembly System. We are developing effective cost measures using previously developed sensor measures which can automatically choose the proper camera placement quickly, based on the assembly task and geometric models of the parts. Enhancements to our 3-D servoing control algorithms are being investigated in order to improve the trajectory of the manipulator's cartesian path. We are creating techniques for using visual servoing to perform assembly tasks, such as insertions, in imprecisely calibrated workcells. Manipulator control strategies which integrate data from both force and vision sensors are being developed. Finally, automatic programming of the entire visual/force servoing assembly system is our long term goal.

6. Acknowledgments

This research was supported by the U.S. Army Research Office through Grant Number DAAL03-91-G-0272, by the Defense Advanced Research Projects Agency though ARPA Order Number DAAA-21-89C-0001, and by the Engineering Design Research Center (an NSF ERC). The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies. We would like to thank Reg Willson and Yangsheng Xu for their suggestions concerning this work.

7. References

1. P.K. Allen.
   Real-Time Motion Tracking using Spatio-Temporal Filters.
   In *Proc. DARPA Image Understanding Workshop*, pp. 695-701, 1989.
2. P. Anandan.
   Measuring Visual Motion from Image Sequences.
   Technical Report COINS-TR-87-21, COINS Department, University of Massachusetts, 1987.
3. F. Chaumette, P. Rives, and B. Espiau.
   Positioning of a Robot with respect to an Object, Tracking it, and Estimating its Velocity by Visual Servoing.
   In *Proc. of the 1991 IEEE Int. Conf. on Robotics and Automation*, pp. 2248-2253, April, 1991.
4. P.I. Corke.
   Video-Rate Visual Servoing for Robots.
   Lecture Notes in Control and Information Science, eds. V. Hayward and O. Khatib, pp. 429-451, Springer-Verlag, 1989.
5. C.K. Cowan and A. Bergman.
   Determining the Camera and Light Source Location for a Visual Task.
   In *Proc. of the 1989 IEEE Int. Conf. on Robotics and Automation*, pp. 509-514, April, 1989.
6. J.T. Feddema and C.S.G. Lee.
   Adaptive Image Feature Prediction and Control for Visual Tracking with a Hand-Eye Coordinated Camera.
   IEEE Trans. on Systems, Man, and Cybernetics, **20(5)**, pp. 1172-1183,1990.
7. K.D. Gremban, C.E. Thorpe, and T. Kanade.
   Geometric Camera Calibration using Systems of Linear Equations.
   In *Proc. of Image Understanding Workshop*, pp. 820-825, April, 1988.
8. J. Ishikawa, K. Kosuge, and K. Furuta.
   Intelligent Control of Assembling Robot Using Vision Sensor.
   In *Proc. of the 1990 IEEE Int. Conf. on Robotics and Automation*, pp. 1904-1909, April, 1990.
9. P.K. Khosla, R.S. Mattikalli, B. Nelson, and Y. Xu.
   CMU Rapid Assembly System.
   In *Video Proc. of the 1992 IEEE Int. Conf. on Robotics and Automation*, July, 1992.
10. A.J. Koivo and N. Houshangi.
    Real-Time Vision Feedback for Servoing of a Robotic Manipulator with Self-Tuning Controller.
    IEEE Trans. on Systems, Man, and Cybernetics, **21(1)**, pp. 134-142,1991.
11. E. Krotkov.
    Focusing.
    International Journal of Computer Vision, **1**, pp. 223-237, 1987.

12. B. Nelson and M. Donath.
Optimizing the Location of Assembly Tasks in a Manipulator's Workspace.
Journal of Robotics Systems, **7(6)**, pp. 791-811, 1990.
13. N. Papanikolopoulos, P.K. Khosla, and T. Kanade.
Adaptive Robotic Visual Tracking.
In *Proc. of the American Control Conference*, pp. 962-967, June 1991.
14. N. Papanikolopoulos, B. Nelson, and P.K. Khosla.
Full 3-D Tracking Using the Controlled Active Vision Paradigm.
In *Proc. 1992 IEEE Int. Symp. on Intelligent Control*, August 11-13, 1992.
15. D.R. Strip.
Lessons from Archimedes, a System for Planning and Executing Mechanical Assemblies.
Sandia Technical Report SAND92-0520C, 1992.
16. K. Tarabanis, R.Y. Tsai, and P.K. Allen.
Automated sensor planning for robotic vision tasks.
In *Proc. of the 1991 IEEE Int. Conf. on Robotics and Automation*, pp. 76-82, April, 1991.
17. C. Tomasi and T. Kanade.
Detection and Tracking of Point Features.
Technical Report CMU-CS-91-132, Carnegie Mellon University, School of Computer Science, 1991.
18. L.E. Weiss, A.C. Sanderson, and C.P. Neuman.
Dynamic Sensor-Based Control of Robots with Visual Feedback.
IEEE Journal of Robotics and Automation **RA-3(5)**, pp. 404-417, October, 1987.
19. S. Yi, R.M. Haralick, and L.G. Shapiro.
Automatic sensor and light source positioning for machine vision.
In *Proc. of the 10th Int. Conf. on Pattern Recognition*, pp. 55-59, June, 1990.
20. T. Yoshikawa.
Manipulability of Robotic Mechanisms.
Robotics Research 2, eds. H. Hanafusa and H. Inoue, pp. 439-446, Cambridge, MA: MIT Press, 1985.