in *Proc. of the 1992 IEEE Int. Conf. on Control Applications*, pp. 1128-1133, 1992.

Increasing the Tracking Region of an Eye-in-Hand System Using Controlled Active Vision

Brad Nelson[1]   P.K. Khosla[2]

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

This paper presents a control strategy which allows a manipulator/camera system to track objects with planar motion while simultaneously avoiding kinematic singularities by moving in directions along which the tracking task space is unconstrained. The projection of a cartesian manipulability gradient on these directions of motion is used to determine the magnitude and direction of the required motion for singularity avoidance, while the controlled active vision paradigm is used for tracking. The algorithms developed have been experimentally verified on an eye-in-hand system. Results demonstrate the effectiveness of the method by showing that the tracking region of a manipulator tracking objects with planar motion can be increased by a factor of between two and three. The overall accuracy of the tracking system is also improved.

1. Introduction

Robotic assembly typically requires precise calibration of the entire assembly workcell, 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. An alternative to precise calibration is to use sensor feedback during the assembly process. Force feedback is sometimes incorporated into the assembly workcell, however force feedback can be difficult to use due to instabilities that arise when contact is made and due to the poor signal-to-noise ratio that force sensors tend to provide. In addition, the use of force feedback requires that parts be brought near one another before final parts mating occurs, thus the system must still be calibrated to a relatively high degree. Visual feedback can help overcome these problems, because a vision sensor is non-contact and can provide information on a much larger area of the workcell than a force sensor provides. The effective use of visual feedback combined with force feedback during the assembly process can be used to create robotic assembly workcells that have the ability to perform precision assemblies in imprecisely calibrated workcells. This can dramatically reduce the cost and setup time of robotic assembly systems.

An important component of a visually servoed assembly system is the visual tracking of objects during assembly. Visual tracking has several other areas of application as well, including parts inspection on moving conveyors and satellite docking in outer space. Although several researchers have studied visual tracking, for example, [1], [3], [4], [5], [8], [9], and [17], these visual servoing techniques are difficult to extend beyond their particular application. The controlled active vision paradigm, first presented in [12], provides a unique framework that has been used to track objects with full 3-D motion using a single camera [13], which no other visual servoing framework has demonstrated. The ability to extend this framework is why we chose to use it for tracking objects during assembly. By taking advantage of the properties of the controlled active vision framework, we have created an algorithm that can overcome an important problem that all manipulator/camera visual servoing systems, whether hand-eye systems or static sensor systems, encounter, the avoidance of manipulator kinematic singularities during operation. We have successfully demonstrated that the avoidance of manipulator kinematic singularities using our proposed technique greatly increases the tracking region of an eye-in-hand system.

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 be 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 maintains 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 is unknown or imprecisely known. This requires that the manipulator should not be near singularities, as well.

Typically, when visual tracking takes place it is necessary to constrain the allowable tracking region to regions of the workspace where there is no danger that the manipulator passes through internal or external kinematic singularities. Consider the visual tracking system shown in Figure 1. As the object continues moving in the -X direction, the manipulator reaches an external singularity and the object is lost. Had the hand-eye system been initially started from a different configuration, the external singularity would have been reached either sooner, if the camera was initially placed nearer the object, or later, if the camera started farther from the object. In addition, during tracking an internal singularity might have been reached depending on the manipulator, the initial placement of the tracking plane, and the motion of the object. However, if the system employs knowledge of the manipulator's configuration in order to improve the configuration using any redundancies that may exist, the tracking region can be greatly extended.

1. Robotics Ph.D. Program and the Robotics Institute.
2. Department of Electrical and Computer Engineering and the Robotics Institute.

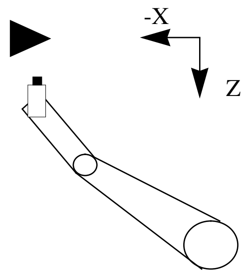

Figure 1. Overhead view of a SCARA-type manipulator visually tracking. If the object continues to move in the -X direction, the manipulator will soon reach a singularity and the system will fail.

We propose a technique based on singularity avoidance using a cartesian manipulability gradient which allows a visual tracking manipulator to track objects with planar motion (X-Y-Θ) while avoiding internal singularities continuously and avoiding external singularities as long as possible until the projection of the object's planar path extends beyond the maximum workspace boundary of the manipulator. This method improves the manipulator's configuration while tracking by using the redundancies that a six DOF manipulator provides when tracking in a five dimensional task space. Our results indicate that the tracking region of an eye-in-hand system can be increased by a factor of between 2 and 3 for a typical system. Section 2 of this paper describes modeling and control of the 2-D visual tracking eye-in-hand problem. The vision techniques used to track features are described in Section 3, and a discussion of the cartesian manipulability gradient appears in Section 4. A description of the implementation of the visual tracking system followed by experimental results which demonstrate the effectiveness of this method are described in Sections 5 and 6 followed by the conclusion.

2. Modeling and Control

2.1. Modeling

The model of the eye-in-hand system used is based on the controlled active vision framework proposed in [12]. To model the 2-D visual servoing problem, it is assumed that the image plane is parallel to the X-Y plane in which the object is constrained to move, the object's center of rotation is about the camera's optical axis, the object is initially located at $z=Z_o$ with respect to the camera frame, and a six degree of freedom cartesian robot controller exists. A pin-hole model for the camera as shown in Figure 2 is assumed [6], so that a feature

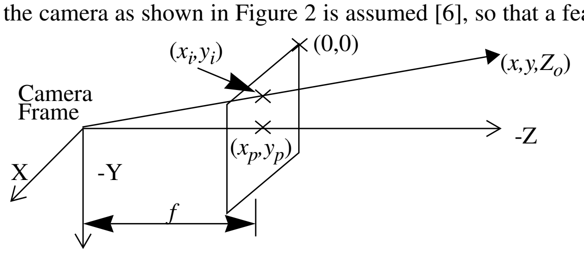

Figure 2. The pin-hole camera model with the image plane moved in front of the camera to simplify signs in the equations.

on the object with coordinates of (x,y,Z_o) in the camera frame projects onto the camera's image plane at

$$x_i = \frac{fx}{Z_o s_x} + x_p \qquad\qquad y_i = \frac{fy}{Z_o s_y} + y_p \tag{1}$$

where (x_i,y_i) are the image coordinates, f is the focal length of the lens, s_x and s_y are the horizontal and vertical dimensions of the pixels, and (x_p, y_p) is the piercing point of the optical axis on the image plane of the camera-lens system. It is assumed that Z_o is much larger than f. The optical flow observed on the image plane is the sum of the optical flow due to the motion of the object $(u_o(t),v_o(t))$ and the optical flow induced by the motion of the camera $(u_c(t),v_c(t))$. If it is assumed that the manipulator moves only along the X and Y axes and about the Z axis, then the optical flow of a feature on the image plane induced by the camera's motion $(\dot{x}_c(t), \dot{y}_c(t), \omega_c(t))$ is [7]

$$u_c(t) = -\frac{f\dot{x}_c(t)}{Z_o s_x} + (y_i(t) - y_p)\frac{s_y}{s_x}\omega_c(t) \tag{2}$$

$$v_c(t) = -\frac{f\dot{y}_c(t)}{Z_o s_y} - (x_i(t) - x_p)\frac{s_x}{s_y}\omega_c(t) \tag{3}$$

The optical flow observed at time $t=kT$ can be approximated as

$$u(k) = \frac{x_i(k+1) - x_i(k)}{T} \qquad v(k) = \frac{y_i(k+1) - y_i(k)}{T} \tag{4}$$

where kT is represented as k for simplicity and without any loss of generality, and T is the sampling period of the vision system. This can be rewritten as

$$x_i(k+1) = x_i(k) + Tu_c(k) + Tu_o(k) \tag{5}$$

$$y_i(k+1) = y_i(k) + Tv_c(k) + Tv_o(k) \tag{6}$$

The control inputs are $\dot{x}_c(t)$, $\dot{y}_c(t)$ and $\omega_c(t)$ ($\dot{x}_c(t)$ and $\dot{y}_c(t)$ for X-Y tracking), i.e., the velocity of the end-effector. The optical flow due to motion of the object is considered an exogenous disturbance that can be suppressed by camera motion. After introducing the exogenous disturbance vector $\boldsymbol{d}$ into the state equations for a single feature on the object, the system can be written as

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{E}\boldsymbol{d}(k) \tag{7}$$

where $\boldsymbol{x}(k) = [x(k)\ y(k)]^T = [(x_i(k)-x_p)\ (y_i(k)-y_p)]^T$, $\boldsymbol{A} = \mathbf{I}_2$, $\boldsymbol{u}(k) = [\ \dot{x}_c(t)\ \dot{y}_c(t)\ \omega_c(t)\ ]^T$ ($\boldsymbol{u}(k) = [\ \dot{x}_c(t)\ \dot{y}_c(t)\ ]^T$ for X-Y tracking), $\boldsymbol{d}(k) = [u_o(t)\ v_o(t)]^T$, $\boldsymbol{E} = T\mathbf{I}_2$, and

$$\boldsymbol{B}_{xy\theta}(k) = T\begin{bmatrix} -\frac{f}{Z_o s_x} & 0 & y(k)\frac{s_y}{s_x} \\ 0 & -\frac{f}{Z_o s_y} & -x(k)\frac{s_x}{s_y} \end{bmatrix} \qquad \boldsymbol{B}_{xy} = T\begin{bmatrix} -\frac{f}{Z_o s_x} & 0 \\ 0 & -\frac{f}{Z_o s_y} \end{bmatrix} \tag{8}$$

where $\boldsymbol{B}_{xy\theta}(k)$ is used for tracking objects with both translational and rotational motion, and $\boldsymbol{B}_{xy}$ is for tracking objects with only translational motion. Since the control input requires three arguments for X-Y-Θ tracking, but only two equations exist for a single feature, at least two features on the object must be tracked for the control input to be determined if object rotation in the plane is allowed. For two features on a rigid object, the features' coordinates are coupled and the state vector to represent the error in the position and orientation of the object on the image plane can be written as $\boldsymbol{x}(k) = [x(k)\ y(k)\ \theta(k)]^T$. Given the coordinates of two features (n=1,2) on the object $(x_{in}(k),\ y_{in}(k))$ and their optical flow $(u_n(k),\ v_n(k))$ by the feature tracker, and assuming small rotations between measurements and that the optical axis and the object's axis of rotation are initially aligned, the new state vector can be obtained from the following equations

$$\theta(k) = \frac{T(u_1(k) - u_2(k))}{y_{i2}(k) - y_{i1}(k)}\frac{s_x}{s_y} = \frac{T(v_1(k) - v_2(k))}{x_{i1}(k) - x_{i2}(k)}\frac{s_y}{s_x} \tag{9}$$

$$x(k) = Tu_1(k) + (y_{i1}(k) - y_p)\,\theta(k)\frac{s_y}{s_x} \tag{10}$$

$$y(k) = Tv_1(k) - (x_{i1}(k) - x_p)\,\theta(k)\frac{s_x}{s_y} \tag{11}$$

In deriving equations 9, 10, and 11, it is assumed that the depth of the two features is approximately the same, and that rotations between consecutive sampling periods are small.

2.2. Kinematic Redundancies

The goal of the eye-in-hand visual tracking system is to maintain the coordinates of the features on the object being tracked at their initial (x_i, y_i) coordinates on the image plane. The object is constrained to move in a plane with motion in X-Y-Θ, and we wish to keep the optical axis of the camera perpendicular to this plane of motion. This results in a five dimensional task space, where three dimensions are needed for tracking (X, Y, and rotation about Z in the camera frame), and two rotational degrees of freedom (rotation about X and Y in the camera frame) are constrained to remain at zero to maintain the desired orientation of the optical axis with the object's plane of motion. Since it was initially assumed that a six degree of freedom cartesian robot controller is available, this leaves one degree of freedom unconstrained, which is translational motion along the optical axis, i.e., in the Z direction of the camera frame. This direction can be used to improve the configuration of the manipulator as tracking takes place in order to avoid kinematic singularities, except when the projection of the object's planar path exceeds the maximum workspace boundaries of the manipulator.

In formulating the state equations in section 2.1, it was assumed that camera motion took place only along X-Y-Θ $(\dot{x}_c(t), \dot{y}_c(t), \omega_c(t))$, however, motion along Z, $\dot{z}_c(t)$, is now a possibility. This has an effect on the observed optical flow, and the optical flow induced by the camera $(u_c(t), v_c(t))$ becomes

$$u_c(t) = -\frac{f\dot{x}_c(t)}{Z(t)s_x} + y(t)\frac{s_x}{s_y}\omega_c + \frac{x(t)s_x\dot{z}_c(t)}{f} \tag{12}$$

$$v_c(t) = -\frac{f\dot{y}_c(t)}{Z(t)s_y} - x(t)\frac{s_y}{s_x}\omega_c - \frac{y(t)s_y\dot{z}_c(t)}{f} \tag{13}$$

The two terms due to $\dot{z}_c(t)$ in the above optical flow equations can be neglected, however, because s_x/f and s_y/f are small (0.000688pixel^{-1} and 0.000813pixel^{-1}, respectively) compared to the other terms. $Z(t)$ is known, because the initial distance of the tracking plane from the camera is assumed known, and the motion of the camera along its optical axis, Z, is a commanded input and, therefore, is known. The matrix $\boldsymbol{B}$ in the state equation now becomes

$$\boldsymbol{B}_{xy\theta}(k) = T\begin{bmatrix} -\frac{f}{Z(k)s_x} & 0 & y(k)\frac{s_y}{s_x} \\ 0 & -\frac{f}{Z(k)s_y} & -x(k)\frac{s_x}{s_y} \end{bmatrix} \tag{14}$$

$$\boldsymbol{B}_{xy}(k) = T\begin{bmatrix} -\frac{f}{Z(k)s_x} & 0 \\ 0 & -\frac{f}{Z(k)s_y} \end{bmatrix} \tag{15}$$

This model of the hand-eye tracking system, which is based on the controlled active vision paradigm [12], allows camera motion along X, Y and about Z for tracking, and also allows camera motion along Z in order to improve the manipulator configuration as tracking occurs. Camera motion about X and Y is not allowed.

2.3. Proportional-Integral Control

The objective of the tracking controller is to drive the state vector $\boldsymbol{x}(k)$ back to its initial state whenever a disturbance is introduced in the form of object motion or noise by creating the proper controller input. For proportional-integral control the control law to create the proper input for tracking is given by

$$\boldsymbol{u}(k) = \begin{bmatrix} \dot{x}_c(k) \\ \dot{y}_c(k) \\ \omega_c(k) \end{bmatrix} = \tag{16}$$

$$= T^{-1}\begin{bmatrix} -\frac{s_x Z(k)}{f} & 0 & 0 \\ 0 & -\frac{s_y Z(k)}{f} & 0 \\ 0 & 0 & 1 \end{bmatrix}\left[\boldsymbol{G}_P\boldsymbol{e}(k) + \boldsymbol{G}_I T\sum_{i=1}^{k}\boldsymbol{e}(i)\right]$$

where $\boldsymbol{G}_P$ and $\boldsymbol{G}_I$ are the constant proportional and integral gain matrices, respectively, and $\boldsymbol{e}(k) = [x(k)\ y(k)\ \theta(k)]^T$, since $x(0) = y(0) = \theta(0) = 0$. Equations 9, 10, and 11 are used to calculate $\boldsymbol{e}(k)$. For X-Y tracking this simplifies to $\boldsymbol{e}(k) = [(x_i(k)-x_i(0))\ (y_i(k)-y_i(0))]^T$, and

$$\boldsymbol{u}(k) = \boldsymbol{B}_{xy}^{-1}\left[\boldsymbol{G}_P\boldsymbol{e}(k) + \boldsymbol{G}_I T\sum_{i=1}^{k}\boldsymbol{e}(k)\right] \tag{17}$$

The control input for motion along the optical axis is determined by a simple proportional controller. This input is expressed as

$$u_{oa}(k) = T_z(k) = G((\nabla_x w)\cdot\boldsymbol{x}_{oa}) \tag{18}$$

where $\nabla_x w$ is the cartesian manipulability gradient, to be explained in Section 4, $\boldsymbol{x}_{oa}$ is a unit vector along the optical axis of the camera, and G is the proportional gain constant.

3. Measurement of Feature Positions

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 the method proposed in [2]. This technique is known as a Sum-of-Squares-Differences (SSD) optical flow, and 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

$$e(\boldsymbol{p}_a, \Delta\boldsymbol{x}) = \sum_W [I_a(x+i, y+j) - I_{a'}(x+i+\Delta x, y+j+\Delta y)]^2 \tag{19}$$

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 Δx=Δy=±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 3) 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

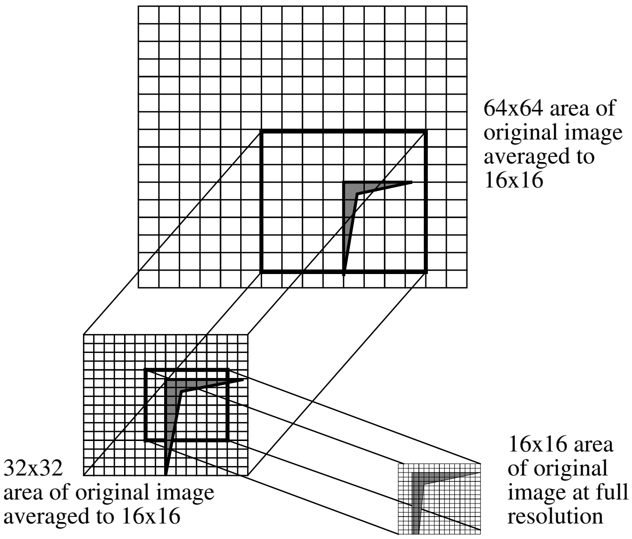

Figure 3. A pyramidal search scheme is used for the SSD optical flow in order to increase the overall sampling rate of the system.

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 alone, 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 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 times not obvious that a particular feature may lose its tracking characteristics at coarse resolutions. Because of this, an automatic feature selector has been implemented based on [14] which accounts for the different levels of resolution in the pyramidal search scheme.

For singularity avoidance, the tracking manipulator is allowed to move along the Z axis of the camera, i.e., perpendicular to the image plane, in order to maximize the distance the manipulator is from a singularity. This slowly changes the size of the feature template based on the projection equation. In order to account for this change, the feature template can be periodically updated by using the matched feature window from a recent image as the new feature template.

4. Singularity Avoidance using the Cartesian Gradient of Manipulability

Although many researchers have proposed solutions to the singularity avoidance problem, all of these solutions have important drawbacks for the visual tracking algorithm presented in this paper. This is because, (a) the path of the object being tracked is not known a priori, so the manipulator's path may not be planned in advance to avoid singularities as is proposed in [15], and (b) pseudo-inverse and singularity-robust inverse [10] or damped least-squares methods [16] of cartesian control allow motion to continue near and even at singularities by approximation of the cartesian path, but higher than usual joint velocities can still occur, and the loss of movement in one or more directions still exists. An efficient and effective method has been developed that keeps the manipulator as far from singularities as possible while visual tracking occurs.

To avoid singularities we use a version of the cartesian singularity avoidance scheme proposed in [11]. This scheme computes the cartesian gradient of Yoshikawa's manipulability measure [18], $\nabla_x w$, quickly and efficiently from the explicitly formulated gradient of manipulability in joint space $\nabla_q w$, where w represents manipulability. The method consists of first explicitly calculating $\nabla_q w$ which, for a non-redundant manipulator, is equivalent to calculating $\nabla_q |det(J(\boldsymbol{q}))|$ $= \partial |det(J(\boldsymbol{q}))| / \partial \boldsymbol{q}$. The homogeneous transformation matrix of the end-effector resulting from motion along this gradient is determined by simulating the motion of the manipulator along $\nabla_q |det(J(\boldsymbol{q}))|$ in joint space. Let

$$ {}^{b}_{e}T = \Psi(\boldsymbol{q}) \tag{20} $$

where ${}^{b}_{e}T$ represents the 4x4 homogeneous transformation matrix of the end-effector frame with respect to the base, and $\Psi(\boldsymbol{q})$ is the forward kinematic solution of the manipulator and is a function of $\boldsymbol{q}$, the manipulator's joint angles. The differential transformation 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

$$ \Delta = \left({}^{b}_{e1}T\right)^{-1} \cdot {}^{b}_{e2}T = \Psi(\boldsymbol{q} - \nabla_q w)^{-1} \cdot \Psi(\boldsymbol{q} + \nabla_q w) \tag{21} $$

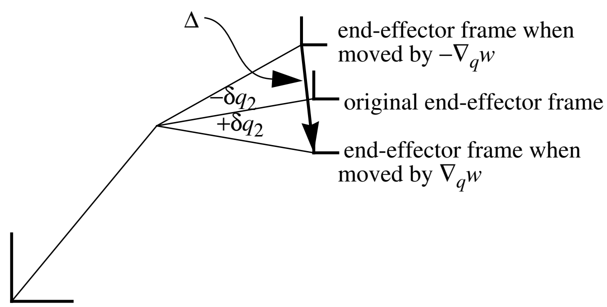

Figure 4. The cartesian manipulability gradient $\nabla_x w$ is represented by the differential transformation matrix Δ.

Figure 4 illustrates this method for a simple two-link manipulator. The differential transformation is with respect to the tool frame and can be decomposed into six components which represent the cartesian gradient of manipulability $\nabla_x w = \nabla_x |det(J(\boldsymbol{q}))|$ for a non-redundant manipulator. By calculating the cartesian gradient in this fashion, the inversion of the 6x6 Jacobian matrix is avoided when calculating the gradient. Motion of the manipulator's tool frame in this direction 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.

5. 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. 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 manipulator and handles communication between the two, as is shown in Figure 5. A sec-

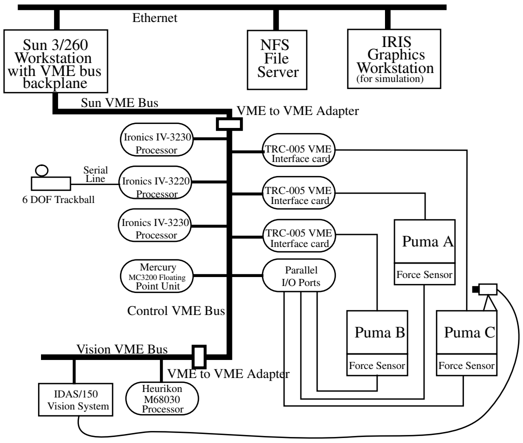

Figure 5. The TROIKABOT system architecture.

ond Puma560 is used as a target. The eye-in-hand Puma has no prior knowledge of the target Puma's path other than that the object being tracked on the target Puma moves in a plane approximately parallel to the eye-in-hand Puma's tool frame.

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. 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 7Hz for X-Y-Θ tracking and 10Hz for X-Y tracking.

The cartesian manipulability gradient is calculated on a Sun3 workstation and the control input for singularity avoidance is sent to the Alter controller at 2Hz. Though this control input could be determined at higher control rates, 2Hz is sufficient for avoiding singularities as tracking occurs.

6. Experimental Results

Experimental results demonstrate the effectiveness of the algorithm. Figure 6 shows an object being tracked that moves along the camera's X axis without singularity avoidance. The objective of the tracking manipulator is to keep the features on the object at the features' original X and Y positions in the camera frame. The error in a feature's position relative to its original position in the camera frame is calculated by recording the joint angles of the tracking Puma and the target Puma, and, then, solving the kinematic chain off-line for the transformation between end-effectors as a function of time. The vertical axis on Figures 6 and 7 represents the deviation of the X component of the transformation between the end-effectors from the transformation at time $t=t_o$. The horizontal axis represents the dis-

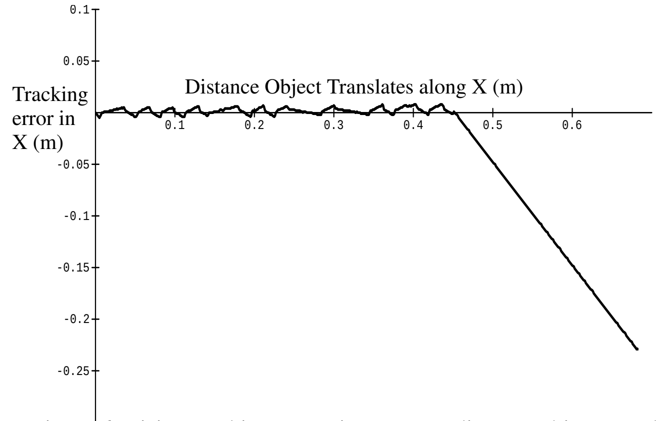

Figure 6. Vision tracking errors in X versus distance object travels along X from its starting position without singularity avoidance along the optical axis of camera (Experimental).

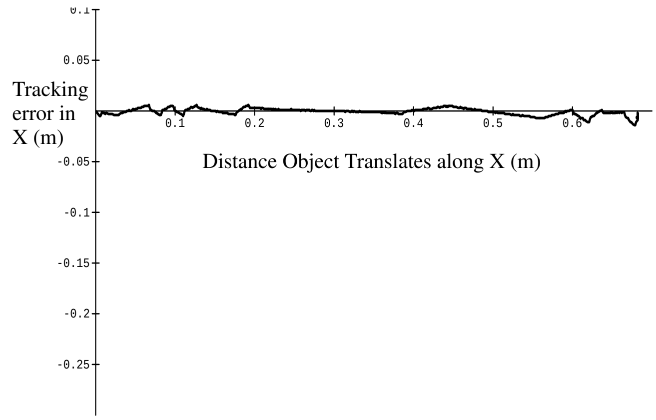

Figure 7. Vision tracking errors in X versus distance object travels along X from its starting position with singularity avoidance. (Experimental).

tance the object has moved along X with respect to the original camera frame. From Figure 6 one can see that the object is lost when the manipulator travels 0.45m along the X axis. This occurred when an external singularity was reached. When singularity avoidance is enabled, tracking along the X axis is extended by 0.25m to 0.70m, as Figure 7 shows. Assuming an approximately spherical workspace volume, this demonstrates an extension of the possible tracking area by a factor of about 2.8 when singularity avoidance is used for this particular visual tracking configuration. In fact, the tracking region is even slightly larger, however, the object being tracked was placed in the gripper of a second Puma560, which reached the limits of its workspace before the tracking manipulator reached its limits. From Figures 6 and 7 it can also be observed that the errors represented by the vertical axis are, on average, lower when singularity avoidance is enabled. This is because as the manipulator is moved away from kinematic singularities throughout the tracking process, the ability of the manipulator to respond to the VAL controller's command inputs improves due to the manipulator's increased manipulability.

Figure 8 shows the motion of the manipulator along the camera's optical axis, the optical axis being the same as the camera frame's Z axis. This is the axis along which singularity avoidance takes place. As the manipulator tracks the object 70cm along X, the manipulator moves approximately 25cm along the optical axis to better configure itself and increase its tracking region. Figure 9 shows how the motion along Z maintains a high manipulability for the manipulator throughout tracking, as compared to the trial without singularity

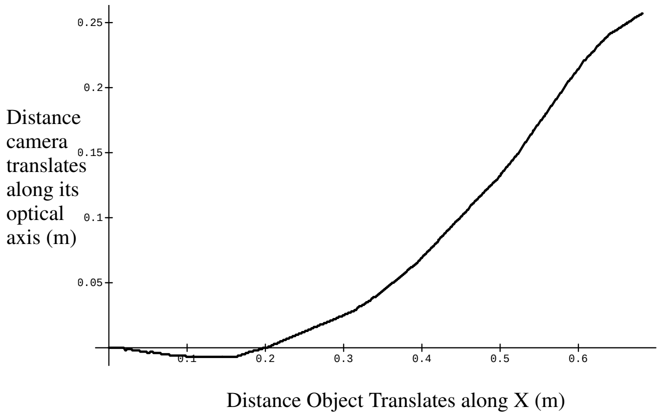

Figure 8. Translation along the camera's optical axis to avoid singularities versus distance target travels in X (Experimental).

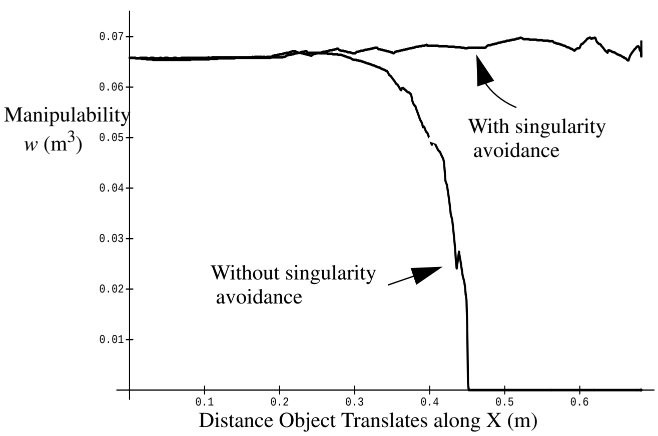

Figure 9. Manipulability w versus distance target travels with and without singularity avoidance (Experimental).

avoidance. Without singularity avoidance, as the object travels more than 30cm along X, the tracking Puma560's joints 2 and 3 come into alignment, and a kinematic singularity is soon reached.

7. Conclusions

We have demonstrated that singularity avoidance using a cartesian gradient of manipulability is an effective method of significantly extending the tracking region of an eye-in-hand visual tracking system. By formulating the 2-D visual tracking problem as one requiring a five dimensional task space, a degree of freedom along the optical axis of the camera is unconstrained and can be used for improving the configuration of the manipulator as tracking occurs. The cartesian gradient of manipulability is projected onto this direction of unconstrained motion, the magnitude and direction of motion along this axis is determined, and the manipulator moves so that its distance from the nearest singular position is maximized while simultaneously tracking the object. The result is a significantly increased tracking region. Experimental results demonstrate the validity of this technique.

8. Acknowledgments

This research was supported by the U.S. Army Research Office through Grant Number DAAL03-91-G-0272 and by the Engineering Design Research Center, an NSF ERC. The views and conclusions contained in this document 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 Nik Papanikolopoulos for his suggestions concerning this work.

9. References

[1] P.K. Allen, "Real-Time Motion Tracking using Spatio-Temporal Filters," in *Proc. DARPA Image Understanding Workshop*, pp. 695-701, 989.

[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 IEEE Int. Conf. on Robotics and Automation*, pp. 2248-2253, April, 1991.

[4] P.I. Corke, "Video-Rate Visual Servoing for Robots," in *Lecture Notes in Control and Information Science*, eds. V. Hayward and O. Khatib, pp. 429-451, Springer-Verlag, 1989.

[5] 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.

[6] 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.

[7] B.K.P. Horn, *Robot Vision*, MIT Press, Cambridge, MA, 1986.

[8] J. Ishikawa, K. Kosuge, and K. Furuta, "Intelligent Control of Assembling Robot Using Vision Sensor," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pp. 1904-1909, April, 1990.

[9] A.J. Koivo and N. Houshangi, "Real-Time Vision Feedback for Servoing of a Robotic Manipulator with Self-Tuning Controller," in *IEEE Trans. on Systems, Man, and Cybernetics,* 21(1), pp. 134-142,1991.

[10] Y. Nakamura and H. Hanafusa, "Inverse Kinematic Solutions with Singularity Robustness for Robot Manipulator Control," in *Journal of Dynamic Systems, Measurement and Control*, 108(3), pp. 163-171, 1986.

[11] B. Nelson and M. Donath, "Optimizing the Location of Assembly Tasks in a Manipulator's Workspace," in *Journal of Robotics Systems*, 7(6), pp. 791-811, 1990.

[12] N.P. Papanikolopoulos, P.K. Khosla, and T. Kanade, "Adaptive Robotic Visual Tracking," in *Proc. of the American Control Conference*, pp. 962-967, June 1991.

[13] N.P. Papanikolopoulos, B. Nelson, and P.K. Khosla, "Full 3-D Tracking Using the Controlled Active Vision Paradigm," in *Proc. of the 1992 IEEE Int. Symp. on Intelligent Control (ISIC-92)*, August 11-13, 1992.

[14] 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.

[15] M. Uchiyama, K. Shimizu, and K. Hakomori, "Performance Evaluation of Manipulators using the Jacobian and its Application to Trajectory Planning," in *Robotics Research 2*, eds. H. Hanafusa and H. Inoue, pp. 447-454, Cambridge, MA: MIT Press, 1985.

[16] C. W. Wampler and L. J. Leifer, "Applications of Damped Least-Squares Methods to Resolved-Rate and Resolved Acceleration Control of Manipulators," in *Journal of Dynamic Systems, Measurement and Control*, 110(1), pp. 31-38, 1988.

[17] L.E. Weiss, A.C. Sanderson, and C.P. Neuman, "Dynamic Sensor-Based Control of Robots with Visual Feedback," in *IEEE Journal of Robotics and Automation* RA-3(5), pp. 404-417, October, 1987.

[18] T. Yoshikawa, "Manipulability of Robotic Mechanisms," in *Robotics Research 2*, eds. H. Hanafusa and H. Inoue, pp. 439-446, Cambridge, MA: MIT Press, 1985.