Eye-In-Hand Robotic Tasks In Uncalibrated Environments

Christopher E. Smith[1] **Scott A. Brandt**[2] **Nikolaos P. Papanikolopoulos**[1]

[1]Department of Computer Science
University of Minnesota
4-192 EE/CS Building
200 Union Street SE
Minneapolis, MN 55455

[2]Department of Computer Science
Campus Box 430
University of Colorado
Boulder, CO 80309-0430

Keywords: Active and real-time vision; Experimental computer vision; Systems and applications; Vision-guided robotics.

Submitted to IEEE Transactions on Robotics and Automation
(Short Paper)

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.

For future correspondence, contact Prof. N. Papanikolopoulos, University of Minnesota, phone number: (612) 625 0163, e-mail address: npapas@cs.umn.edu.

Eye-In-Hand Robotic Tasks In Uncalibrated Environments

ABSTRACT

Flexible operation of a robotic agent in an uncalibrated environment requires the ability to recover unknown or partially known parameters of the workspace through sensing. Of the sensors available to a robotic agent, visual sensors provide information that is richer and more complete than other sensors. In this paper we present robust techniques for the derivation of depth from feature points on a target's surface and for the accurate and high-speed tracking of moving targets. We use these techniques in a system that operates with little or no *a priori* knowledge of object- and camera-related parameters to robustly determine such object-related parameters as velocity and depth. Such determination of extrinsic environmental parameters is essential for performing higher level tasks such as inspection, exploration, tracking, grasping, and collision-free motion planning. For both applications, we use the Minnesota Robotic Visual Tracker (a single visual sensor mounted on the end-effector of a robotic manipulator combined with a real-time vision system) to automatically select feature points on surfaces, to derive an estimate of the environmental parameter in question, and to supply a control vector based upon these estimates to guide the manipulator.

Keywords: Active and real-time vision; Experimental computer vision; Systems and applications; Vision-guided robotics.

1 Introduction

In order to be effective, robotic agents in uncalibrated environments must operate in a flexible and robust manner. The computation of unknown parameters (e.g., the velocity of objects and the depth of object feature points) is essential information for the accurate execution of many robotic tasks, such as manipulation, inspection, and exploration. The determination of such parameters has traditionally relied upon the accurate knowledge of other related environmental parameters. For instance, early efforts in depth recovery concentrated upon stereo camera pairs with known geometries to derive an estimate of the depth of a feature based upon its projection in each image plane and triangulation [24]. Much of this work owes its origins to the field of photogrammetry [44].

Other methods used what is commonly referred to as *shape-from-X* or *structure-from-X* where "*X*" may be shading, texture, or motion. These techniques typically rely upon stringent lighting models (single source, orthogonal sources, etc.), surfaces that possess a known quality (ideal Lambertian surfaces, spherical objects, prior knowledge of texture maps, etc.), or multiple, known viewpoints. Prior work in *shape-from-motion* [18] often relied upon the incidental motion of the object or the manipulator to provide the disparity needed for triangulation and depth extraction [25][33][38]. Many traditional approaches to the problem of depth recovery [11][25] have assumed that extremely accurate measurements of the camera parameters and the camera system geometry are provided *a priori*, making these methods useful in only a limited number of situations.

Similarly, previous approaches to visual tracking assumed known and accurate measures of camera parameters, camera positioning, manipulator positioning, target depth, target orientation, and environmental conditions [5][14][21][22][40]. The initial research efforts in this area assumed a static camera (as opposed to one mounted on the robot itself) [2][14][19] and some performed only simulations [41]. Virtually no work took into account the dynamics of the robot in computing the control signal. Since then, various researchers have expanded the problem to incorporate robot dynamics, moving camera models, or both, and several have focused on using vision

information in the dynamic feedback loop [8][39]. Robotic visual tracking techniques applicable to slow-moving objects were developed wherein the centroid of the object is determined [17]. Several others researchers [15][28] attempted to apply neural networks to the problem of robotic visual tracking while others [11][12][13] concentrated on the control aspects of the problem. In many of these cases, explicit knowledge of camera, manipulator and environmental parameters were assumed.

This type of detailed information is not always available or, when it is available, not always accurate. Inaccuracies are introduced by positioning, path constraints, changes in the robotic system, and changes in the operational environment. In addition, camera calibration and the determination of camera parameters can be computationally expensive, time consuming, and error prone. To be effective in uncalibrated environments, the robotic agent must perform under a variety of situations when only simple estimates of parameters (e.g., depth, focal length, pixel size, etc.) are used and with little or no *a priori* knowledge about the target, the camera, or the environment.

In this paper, we present the proposed techniques for performing eye-in-hand robotic tasks in uncalibrated environments. We first discuss the equations for visual measurements, including an enhanced SSD surface construction strategy, optimizations, and an alternative search technique, and we present the feature point selection scheme. Next, we present the control and measurement equations used in the adaptive controller and elaborate on the selection of the features' trajectories for the application of depth recovery. Finally, we discuss results from experiments in both of the selected applications using the Minnesota Robotic Visual Tracker (MRVT).

2 Proposed Solution

One solution to the problems that arise from relying heavily on *a priori* information can be found under the Controlled Active Vision framework [30][31]. This framework provides the flexibility necessary to operate under dynamic conditions when many environmental and target-related factors are unknown and possibly changing. The framework is based upon adaptive controllers that utilize Sum-of-Squared Differences (SSD) optical flow measurements [3] as inputs to the control loop. The SSD algorithm is used to measure the displacements of feature points in a

sequence of images where the displacements may be induced by manipulator motion, target motion, or both. The measured displacements are compared to predicted displacements that are derived using the current parameter estimates in the adaptive controller. The errors from these comparisons are then used, in conjunction with previous measured displacements, to update parameter estimates and to produce the next control input. The control input is derived such that the amount of the error in the next iteration will be minimized given environmental- and manipulator-specific constraints. This technique is useful under a variety of situations, including the application areas we have selected: depth recovery and robotic visual tracking.

We propose a controlled exploratory motion that provides identifiability of the depth parameter rather than an accidental motion of the eye-in-hand system commonly used in depth extraction techniques [25][33][38]. To reduce the influence of workspace-, camera-, and manipulator-specific inaccuracies, an adaptive controller is utilized to provide accurate and reliable information regarding the depth of an object's feature points. This information may then be used to guide operations such as tracking, inspection, and manipulation [11][30].

Additionally, we propose a visual tracking system that does not rely upon accurate measures of environmental and target parameters. An adaptive controller is used to track feature points on a target's surface in spite of the unconstrained motion of the target, possible occlusion of feature points, and changing target and environmental conditions. High-speed targets are tracked with only rough operating parameter estimates and no explicit target models. Tracking speeds are nearly twelve times faster and have similar accuracy to those reported by Papanikolopoulos [30].

While this work is based upon that in [30][31], it is different in several significant ways. First, the method for the derivation of depth is a unique formulation under the Controlled Active Vision framework. Second, a new technique for automatic feature selection is presented that addresses the aperture problem in a direct and purposeful way. Third, optimizations to the vision processing are introduced that enhance overall performance by at least one order of magnitude. Additionally, the vision processing includes a dynamic pyramiding technique to resolve speed/accuracy trade-offs. Finally, all the presented work is implemented on the Minnesota Robotic

Visual Tracker (MRVT), an active vision testbed that integrates a traditional robotic manipulator (a Puma 560) with a state-of-the-art vision system to provide a unique and flexible sensor based robotic system.

3 Visual Measurements

Our depth recovery and robotic visual tracking applications both use the same basic visual measurements that are based upon a simple camera model and the measure of optical flow in a temporal sequence of images. The visual measurements are combined with search-specific optimizations and a dynamic pyramiding technique in order to enhance the visual processing from frame to frame and to optimize the performance of the system in our selected applications.

3.1 Camera Model and Optical Flow

We assume a pinhole camera model with a world frame, R_W, centered on the optical axis. We also assume a focal length f. A point $\mathbf{P} = (X_W, Y_W, Z_W)^T$ in $\mathbf{R}_W$, projects to a point $\mathbf{p}$ in the image plane with coordinates (x, y). We can define two scale factors, s_x and s_y, to account for camera sampling and pixel size, and include the center of the image coordinate system (c_x, c_y) given in frame F_A [30]. The derived model is precisely the same as the model presented by Papanikolopoulos [30] and results in the following equations (assuming camera motion where T_x, T_y, T_z are the components of the translational velocity and R_x, R_y, R_z are the components of the rotational velocity):

$$u = \dot{x} = \left[x\frac{T_z}{Z_W} - f\frac{T_x}{Z_W s_x} \right] + \left[xy\frac{s_y R_x}{f} - \left(\frac{f}{s_x} + x^2\frac{s_x}{f} \right) R_y + y\frac{s_y}{s_x} R_z \right] \tag{1}$$

$$v = \dot{y} = \left[y\frac{T_z}{Z_W} - f\frac{T_y}{Z_W s_y} \right] + \left[\left(\frac{f}{s_y} + y^2\frac{s_y}{f} \right) R_x - xy\frac{s_x}{f} R_y - x\frac{s_x}{s_y} R_z \right]. \tag{2}$$

To measure displacements of $\mathbf{p}$ we use a matching-based technique known as the Sum-of-Squared Differences (SSD) optical flow [3]. For the point $\mathbf{p}(k-1) = (x(k-1), y(k-1))^T$ in the image $(k-1)$ where (k) denotes the kth image in a sequences of images, we want to find the point $\mathbf{p}(k) = (x(k-1) + u, y(k-1) + v)^T$. This point $\mathbf{p}(k)$ is the new position of the projec-

tion of the feature point $\mathbf{P}$ in image k. Thus, for the point $\mathbf{p}(k-1)$, the SSD algorithm selects the displacement $\Delta\mathbf{x} = (u, v)^{\mathrm{T}}$ that minimizes the SSD measure

$$e(\mathbf{p}(k-1), \Delta\mathbf{x}) = \sum_{m,\, n \in N} [I_{k-1}(x(k-1)+m, y(k-1)+n) - I_k(x(k-1)+m+u, y(k-1)+n+v)]^2 \quad (3)$$

where $u, v \in \Omega$, N is the neighborhood of $\mathbf{p}$, m and n are indices for pixels in N, and I_{k-1} and I_k are the intensity functions in images $(k-1)$ and (k).

The size of the neighborhood N must be carefully selected to ensure proper system performance. Too small an N fails to capture large displacements while too large an N increases the associated computational overhead and enhances the background. In either case, an algorithm based upon the SSD technique may fail due to inaccurate displacements. An algorithm based upon the SSD technique may also fail due to too large a latency in the system or displacements resulting from motions of the object that are too large for the method to capture accurately. We introduce search optimizations and a dynamic pyramiding technique to counter these concerns.

3.2 Search Optimizations

The primary source of latency in a vision system that uses the SSD measure is the time needed to identify $(u, v)^{\mathrm{T}}$ in equation (3). To find the true minimum, the SSD measure must be calculated over each possible $(u, v)^{\mathrm{T}}$. The time required to produce an SSD surface and to find the minimum can be greatly reduced by employing three schemes that, when combined, divide the search time significantly in the expected case.

The first optimization used is loop short-circuiting. During the search for the minimum on the SSD surface (the search for $u_{\min}$ and $v_{\min}$), the SSD measure must be calculated according to equation (3). This requires nested loops for the *m* and *n* indices. During the execution of these loops, the SSD measure is calculated as the running sum of the squared pixel value differences. If the current SSD minimum is checked against the running sum as a condition on these loops, the execution of the loops can be short-circuited as soon as the running sum exceeds the current minimum. This optimization has a worst-case performance equivalent to the original algorithm plus

the time required for the additional condition tests. This worst case occurs when the SSD surface minimum lies at the last $(u, v)^{\mathrm{T}}$ position searched. On average, this short-circuit realizes a decrease in execution time by a factor of two.

The second optimization is based upon the heuristic that the best place to begin the search for the minimum is at the point where the minimum was last found on the surface and to expand the search radially from this point. Under this heuristic, the search pattern in the (k) image is altered to begin at the point on the SSD surface where the minimum was located for the $(k-1)$ image. The search pattern then spirals out from this point, searching over the extent of u and v, in contrast to the typical indexed search pattern (see Figure 1).

Since the structure that implements the spiral search pattern contains no more overhead than the loop structures of the traditional search, worst-case performance is identical. In the general case, search time is approximately halved when combined with the loop short-circuiting.

The third optimization arose from the observation that search times for feature points varied significantly — by as much as 100 percent — depending upon the shape/orientation of the feature. Applying the spiral image traversal pattern to the calculation of the SSD measure simultaneously fixes the problem and achieves additional performance improvements. Spiraling the calculation of the SSD measure yields a best-case performance that is independent of target orientation by changing the order of the SSD calculations to no longer favor one portion of the image over another. Additional speed gains are achieved because the area of greatest change in the SSD measure calculations typically occurs near the center of the feature window, which generally coin-

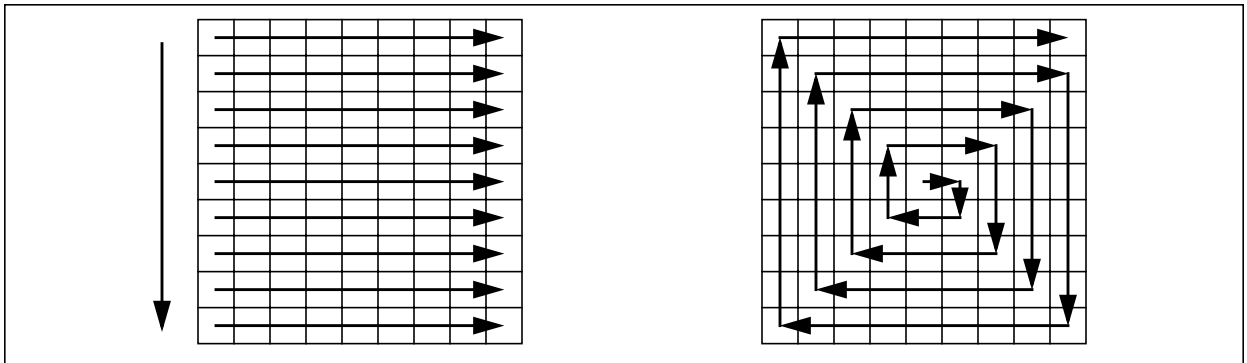


Figure 1: Traditional and spiral search patterns

cides with an edge or a corner of the target, resulting in fewer calculations before the loop is terminated in non-matching cases. Speed gains from this optimization are approximately 40 percent.

When combined, these three optimizations (the loop short-circuiting and the spiral pattern for both the search loop and the SSD calculation loop) interact cooperatively to find the minimum of the SSD surface as much as 17 times faster on average than the unmodified search.

Experimentally, the search times for the unmodified algorithm averaged 136 msec over 5000 frames under a variety of relative feature point motions. The modified algorithm with the search-loop short-circuiting alone averaged 60-72 msec search times over several thousand frames with arbitrary relative feature point motion. The combined short-circuit/spiral search algorithm produced search times that averaged 13 msec under similar tests, and the combined short-circuit, dual-spiral algorithm produced search times that averaged 8 msec. Together these optimizations allow the vision system described in Section 6.1 to track three to four features at video rates (33 msec per frame) without video under-sampling.

3.3 Dynamic Pyramiding

Dynamic pyramiding is a heuristic technique that attempts to resolve the conflict between accurate positioning of a manipulator and high-speed tracking when the displacements of the feature points are large. Previous applications have typically depended upon one preset level of pyramiding to enhance either the top tracking speed of the manipulator or the positioning accuracy of the end-effector above the target [30].

In contrast, dynamic pyramiding uses multiple levels of pyramiding. The level of the pyramiding is selected based upon the observed displacements of the target's feature points. If the displacements are small relative to the search area, the pyramiding level is reduced; if the measured displacements are large compared to the search area, then the pyramiding level is increased. The result is a system that enhances the tracking speed when required, but always biases in favor of the maximum accuracy achievable. The dynamic level switching thus allows the tracker to adjust to accelerations of the target (when displacements increase) and then to increase accuracy when the tracking adapts to the new, higher speed or when the target is at rest.

During the search process, the SSD measurements are centered upon particular pixels in the pyramided search area. Which pixel positions are selected ($\Delta\mathbf{x} = (u, v)^{\mathrm{T}}$ in equation (3)) is dependent upon which of the four levels of pyramiding is currently active. The lowest level searches every pixel a square 32×32 pixel patch of the current frame. The second level searches every other pixel in a 64×64 patch, and the third, every third pixel in a 96×96 patch. The fourth and highest level searches every fourth pixel in a 128×128 patch.

4 Feature Point Selection

In addition to failures due to system latency, the effect of large displacements, or technique-specific problems, an algorithm based upon the SSD technique may fail due to repeated patterns in the intensity function of the image or due to large areas of uniform intensity in the image. Both cases can provide multiple matches within a feature point's neighborhood, resulting in spurious displacement measures. In order to avoid this problem, our system automatically evaluates and selects feature points.

Feature points are selected based upon a confidence measure derived using the SSD measure and an auto-correlation technique. The neighborhood N, centered upon a prospective feature point in image I_k, is used to collect SSD measures at offsets belonging to the area Ω, also in I_k [3][30]. This produces a surface of SSD measures over the area Ω. Several possible confidence measures can be applied to the surface to measure the suitability of the potential feature point.

The selection of a confidence measure is critical since many such measures lack the robustness required by changes in illumination, intensity, etc. We utilize a two-dimensional displacement parabolic fit that attempts to fit a parabola $e(\Delta x_r) = a\Delta x_r^2 + b\Delta x_r + c$ to a cross-section of the SSD surface in several predefined directions, producing a measure of the goodness of the fit. Papanikolopoulos [30] selected a feature point if the minimum directional measure was sufficiently high, as measured by equation (3). For instance, a corner point produces an SSD surface with very good parabolic fits in all directions. Such a point is selected since the minimal directional measure will be high. A point belonging to an edge would not be selected since the fit in the direction along the edge would be poor due to multiple matches for the prospective feature point.

For the purpose of depth extraction (where a large set of feature points is desired), we extend this approach in response to the aperture problem [23]. Briefly stated, the aperture problem refers to the motion of points that lie on a feature such as an edge. Any sufficiently small local measure of motion can only retrieve the component of motion orthogonal to the edge. Any component of motion other than that perpendicular to the edge is unrecoverable due to multiple matches for any given feature point [23].

The improved confidence measure for feature points permits the selection of a feature if it has a sufficiently high correlation with the parabola in a particular direction. For instance, a feature point corresponding to an edge will be selected due to the high parabolic-fit measurement in the direction perpendicular to the edge.

This directional information is then used during the depth recovery process to constrain the motion of the manipulator. A corner feature would therefore allow manipulator motion in any direction while an edge feature would constrain motion to the direction perpendicular to the edge. This ensures that the only component of resulting motion is precisely the component that can be recovered under consideration of the aperture problem. This ideal motion is not always achievable; therefore, matches that lie closer to the ideal manipulator motion will be preferred, if multiple matches are detected by the SSD search.

Directionally constrained features are not used during the tracking of moving objects since the motion of such objects cannot be characterized until after the selection of tracking features. If used, such directional features potentially leave large portions of the motion of objects unrecoverable unless the motion at multiple points with dissimilar directional measures are used. Since this would increase overhead by at least a factor of two, only features that are determined to be omnidirectional (e.g., a corner point) should be used for tracking applications.

5 Modeling and Controller Design

Modeling of the system in question is critical to the design of a controller to perform the task at hand. In our application areas, the modeling and controller designs are similar; however they are also distinct. We have presented the depth recovery modeling and controller design previ-

ously in [36]. Papanikolopoulos *et al*. [31] provide an in-depth treatment of the modeling and controller design for robotic visual tracking applications.

Previous work in depth recovery using active vision relied upon random camera displacements to provide displacement changes in the $\mathbf{p}(k)$'s [18][26][33][38][42]. In our system, we select feature points automatically, produce feature trajectories (a different trajectory for every feature), and predict future displacements using the depth estimate [34][36]. The errors in these predicted displacements, in conjunction with the estimated depth, are included as inputs in the control loop of the system. Thus, the next calculated movement of the system produces a camera movement that will eliminate the largest possible portion of the observed error while adhering to various environmental- and manipulator-specific constraints. This purposeful movement of the visual sensor provides more accurate depth estimates and a faster convergence of the depth measures over a sequence of estimates.

For depth recovery, we use the SSD optical flow to measure displacements of feature points in the image plane. For a single feature point, **p**, we assume that the optical flow at time instant kT is $(u(kT), v(kT))$, where T is the time between two successive frames. In order to simplify the notation, the index k will be used instead of kT. By taking delays into account, eliminating the flow components due to object motion, including the inaccuracies of the model (e.g., neglected accelerations, inaccurate robot control) as white noise, and utilizing the relations

$$u(k) = \frac{x(k+1) - x(k)}{T} \qquad \text{and} \qquad v(k) = \frac{y(k+1) - y(k)}{T} \tag{4}$$

the the optical flow is given by:

$$x(k+1) = x(k) + Tu_c(k-d+1) + v_1(k) \tag{5}$$

$$y(k+1) = y(k) + Tv_c(k-d+1) + v_2(k) \tag{6}$$

where $v_1(k)$ and $v_2(k)$ are white noise terms with variances σ_1^2 and σ_2^2, respectively.

The above equations may be written in state-space form as

$$\mathbf{p}(k+1) = \mathbf{A}(k)\mathbf{p}(k) + \mathbf{B}(k-d+1)\mathbf{u}_c(k-d+1) + \mathbf{H}(k)\mathbf{v}(k) \tag{7}$$

where $\mathbf{A}(k) = \mathbf{H}(k) = \mathbf{I}_2$, $\mathbf{p}(k)$ and $\mathbf{v}(k) \in R^2$, and $\mathbf{u}_c(k-d+1) \in R^6$. The vector $\mathbf{p}(k) = (x(k), y(k))^{\mathrm{T}}$ is the state vector, $\mathbf{u}_c(k-d+1) = (T_x(k-d+1), T_y(k-d+1), 0, 0, 0, 0)^{\mathrm{T}}$ is the control input vector (only movement in the x-y plane of the end-effector frame is used), and $\mathbf{v}(k) = (v_1(k), v_2(k))^{\mathrm{T}}$ is the white noise vector. The term, $\mathbf{B}(k-d+1)\mathbf{u}_c(k-d+1) \in R^6$ can be simplified to

$$\mathbf{B}(k-d+1)\mathbf{u}_{\mathrm{c}}(k-d+1) = T\begin{bmatrix} \frac{-f}{Z_W s_x} T_x(k-d+1) \\ \frac{-f}{Z_W s_y} T_y(k-d+1) \end{bmatrix} \tag{8}$$

since Z_{W} does not vary over time and only the $T_x(k-d+1)$ and $T_y(k-d+1)$ control inputs are used. By substitution into equation (7) and simplification, we derive the following:

$$\begin{bmatrix} x(k+1) \\ y(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \end{bmatrix} + T\begin{bmatrix} \frac{-f}{Z_W s_x} T_x(k-d+1) \\ \frac{-f}{Z_W s_y} T_y(k-d+1) \end{bmatrix} + \begin{bmatrix} v_1(k) \\ v_2(k) \end{bmatrix}. \tag{9}$$

The previous equation is used under the assumption that the optical flow induced by the motion of the camera does not change significantly in the time interval *T*. Additionally, the interval *T* should be as small as possible so that the relations that provide $T_x(k-d-1)$ and $T_y(k-d+1)$ are as accurate as possible. The parameter *T* has as its lower bound 16 msec (the sampling rate of standard video equipment). If the upper bound on *T* can be reduced below the sampling rate of the manipulator controller (in our case, 28 msec for the PUMA 560), then the system will no longer be constrained by the speed of the vision hardware.

The objective is to design a specific trajectory (a set of desired $\mathbf{p}^*(k) = (x^*(k), y^*(k))^{\mathrm{T}}$ for successive *k*'s) for every individual feature in order to identify the depth parameter Z_W. Thus, we have to design a control law that forces the eye-in-hand system to track the desired trajectory for every feature. Based on the information from the automatic feature selection procedure, we select a trajectory for the feature. For example, the ideal trajectory

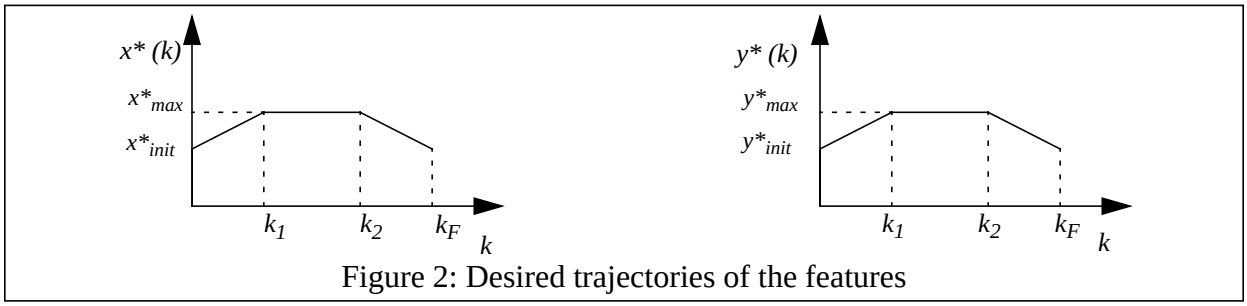


Figure 2: Desired trajectories of the features

for a corner feature is depicted in Figure 2. If the feature belongs to an edge parallel to the y axis, then $y^*(k)$ is held constant at y^*_{init}. Then, the control objective function is selected to be

$$F_o(k+d) = E\{(\mathbf{p}_m(k+d) - \mathbf{p}^*(k+d))^{\mathrm{T}} \mathbf{Q} (\mathbf{p}_m(k+d) - \mathbf{p}^*(k+d)) + \mathbf{u}_{\mathrm{c}}^{\mathrm{T}}(k) \mathbf{L}_{\mathrm{d}} \mathbf{u}_{\mathrm{c}}(k)\} \qquad (10)$$

where $E\{Y\}$ denotes the expected value of the random variable Y, $\mathbf{p}_m(k)$ is the measured state vector, and $\mathbf{Q}$, $\mathbf{L}_{\mathrm{d}}$ are control weighting matrices. Based on the equation (9) and the minimization of the control objective function $F_o(k+d)$ (with respect to the vector $\mathbf{u}_c(k)$), we can derive the following control law:

$$\mathbf{u}_{\mathrm{c}}(k) = -[\hat{\mathbf{B}}^{\mathrm{T}}(k) \mathbf{Q} \hat{\mathbf{B}}(k) + \mathbf{L}_{\mathrm{d}}]^{-1} \hat{\mathbf{B}}^{\mathrm{T}}(k) \mathbf{Q} \left\{ \mathbf{p}_m(k) - \mathbf{p}^*(k+d) + \sum_{i=1}^{i=d-1} \hat{\mathbf{B}}(k-i) \mathbf{u}_{\mathrm{c}}(k-i) \right\} \qquad (11)$$

where $\hat{\mathbf{B}}(k)$ is the estimated value of the matrix $\mathbf{B}(k)$. The matrix $\hat{\mathbf{B}}(k)$ depends on the estimated value of the depth $\hat{Z}_W$. The estimation of the depth parameter is performed by using the procedure described in [30]. In order to use this procedure, we must rewrite equation (9) as follows (for one feature point):

$$\Delta \mathbf{p}_m(\mathrm{k}) = \zeta_W \, \mathbf{u}_{\mathrm{cnew}}(k-d) + \mathbf{n}(k) \qquad (12)$$

where

$$\Delta \mathbf{p}_m(\mathrm{k}) = \mathbf{p}_m(\mathrm{k}) - \mathbf{p}_m(\mathrm{k}-1)$$

$$\zeta_W = f/(s_x Z_W)$$

$$\mathbf{n}(k) \sim N(\mathbf{0}, \mathbf{N}(k))$$

$$\mathbf{u}_{\mathrm{cnew}}(k) = T \begin{bmatrix} -T_x(k) \\ \frac{-s_x}{s_y} T_y(k) \end{bmatrix}.$$

The objective of the estimation scheme is to estimate the parameter ζ_W for each individual feature. When this computation is completed, it is trivial to compute the parameter $1/Z_W$ that is needed for the construction of the depth maps. It is assumed that the matrix $\mathbf{N}(k)$ is constant ($\mathbf{N}(k) = \mathbf{N}$). We use the estimation scheme described in [25][27] (for one feature point):

$$ {}^{p}\hat{\zeta}_W(k) = {}^{u}\hat{\zeta}_W(k-1) \tag{13} $$

$$ {}^{p}pr(k) = {}^{u}pr(k-1) + s(k-1) \tag{14} $$

$$ {}^{u}pr(k) = \left\{ ({}^{p}pr(k))^{-1} + \mathbf{u}_{\text{cnew}}^{\text{T}}(k-d)\mathbf{N}^{-1}\mathbf{u}_{\text{cnew}}(k-d) \right\}^{-1} \tag{15} $$

$$ \mathbf{g}^{T}(k) = {}^{u}pr(k)\mathbf{u}_{\text{cnew}}^{T}(k-d)\mathbf{N}^{-1} \tag{16} $$

$$ {}^{u}\hat{\zeta}_W(k) = {}^{p}\hat{\zeta}_W(k) + \mathbf{g}^{T}(k)\left\{ \Delta\mathbf{p}_m(k) - {}^{p}\hat{\zeta}_W(k)\mathbf{u}_{\text{cnew}}(k-d) \right\} \tag{17} $$

where the superscript *p* denotes the predicted value of a variable, the superscript *u* denotes the updated value of a variable, and *s (k*) is a covariance scalar. The initial conditions are described in [30]. The term ${}^{p}pr(0)$ can be viewed as a confidence measure for the initial estimate ${}^{p}\hat{\zeta}_W(0)$.

The depth recovery process does not require camera calibration nor precisely known environmental parameters. Camera parameters are taken directly from the manufacturer's documentation for both the CCD array and the lens. Experimental results under this process do not suffer significantly due to these estimates, as shown in the following sections.

6 Experimental Design and Results

6.1 The Minnesota Robotic Visual Tracker

We have implemented the depth recovery and the visual tracking on the Minnesota Robotic Visual Tracker (MRVT) [6][7] system. The MRVT is a multi-architectural system consisting of two parts: the Robot/Control Subsystem (RCS) and the Vision Processing Subsystem (VPS).

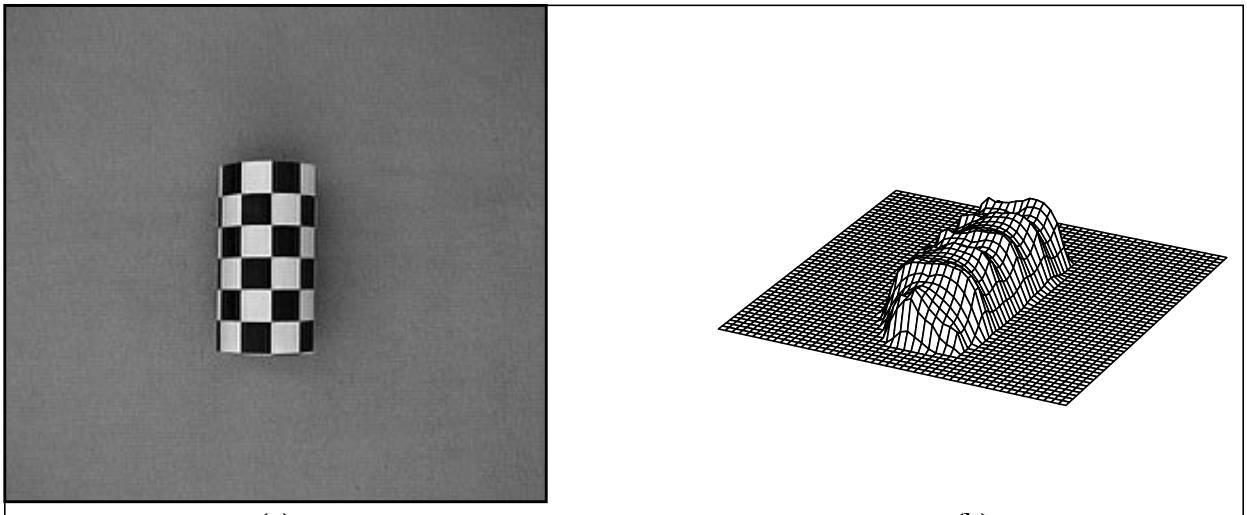
The RCS consists of a PUMA 560 manipulator, its Unimate Computer/Controller, and a VME-based Single Board Computer (SBC). The manipulator's trajectory is controlled via the

Unimate controller's Alter line and requires path control updates once every 28 msec. Those updates are provided by an Ironics 68030 VME SBC running Carnegie Mellon University's CHIMERA [37] real-time robotic environment. A Sun SparcStation 330 serves as the CHIMERA host and shares its VME bus with the Ironics SBC via BIT-3 VME-to-VME bus extenders.

The VPS receives input from a Panasonic GP-KS102 miniature camera that is mounted parallel to the end-effector of the PUMA and provides a video signal to a Datacube system for processing. The camera uses a 7.5 mm focal length lens and the camera scale factors in x and y are 0.01333 mm/pixel and 0.00952 mm/pixel respectively. These parameters are derived from the documentation for the camera and the Datacube system and are not calibrated. The Datacube is the main component of the VPS and consists of a Motorola MVME-147 SBC, a Datacube MaxVideo20 video processor, a Max860 vector processor, and a BIT-3 VME-to-VME bus extender. The VPS performs the optical flow, calculates the desired control input, and supplies the input vector to the Ironics processor for inclusion as an input into the control software.

6.2 Derivation of Depth

The initial experimental runs were conducted with a soda can wrapped in a checkerboard surface, placed such that the leading edge of the curved surface was 53 cm from the nodal point of the camera (see Figure 3 (a)). The initial depth estimate was set to 1 meter in depth. The camera



(a) (b)

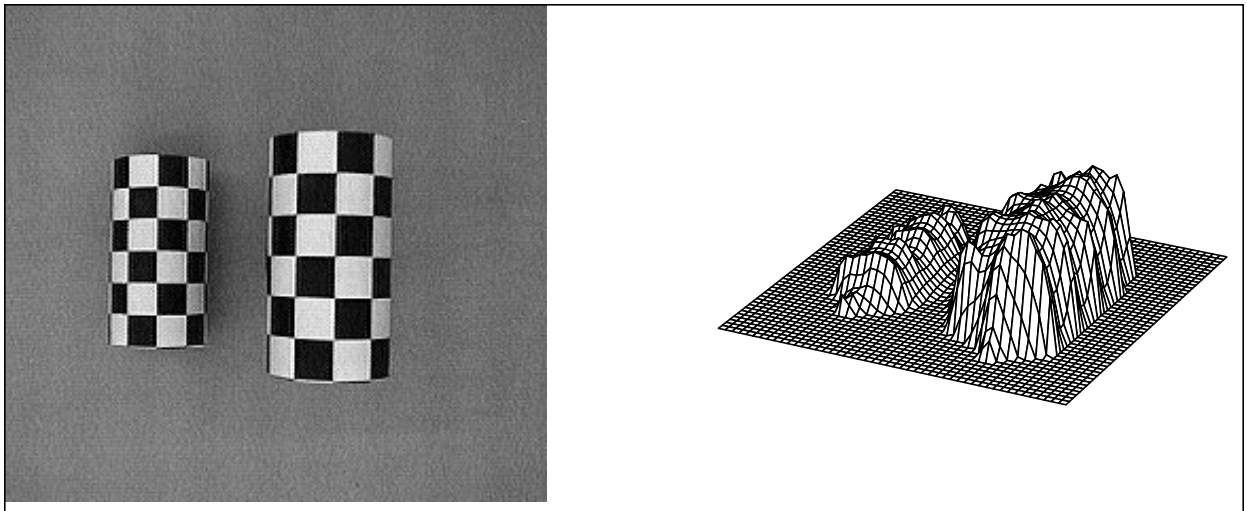
Figure 3: Single can target and surface

scaling factors and focal length were taken directly from the documentation provided by the manufacturer and controller gains were set to match the specifications of the RCS. Neither camera calibration nor gain adjustment was performed for these experiments.

The error in the recovered depth of the majority of feature points was less than the error calculated for a displacement error of one pixel. The reconstructed surface of the single soda appears in Figure 3 (b) and clearly shows that the process has recovered the curvature of the soda can.

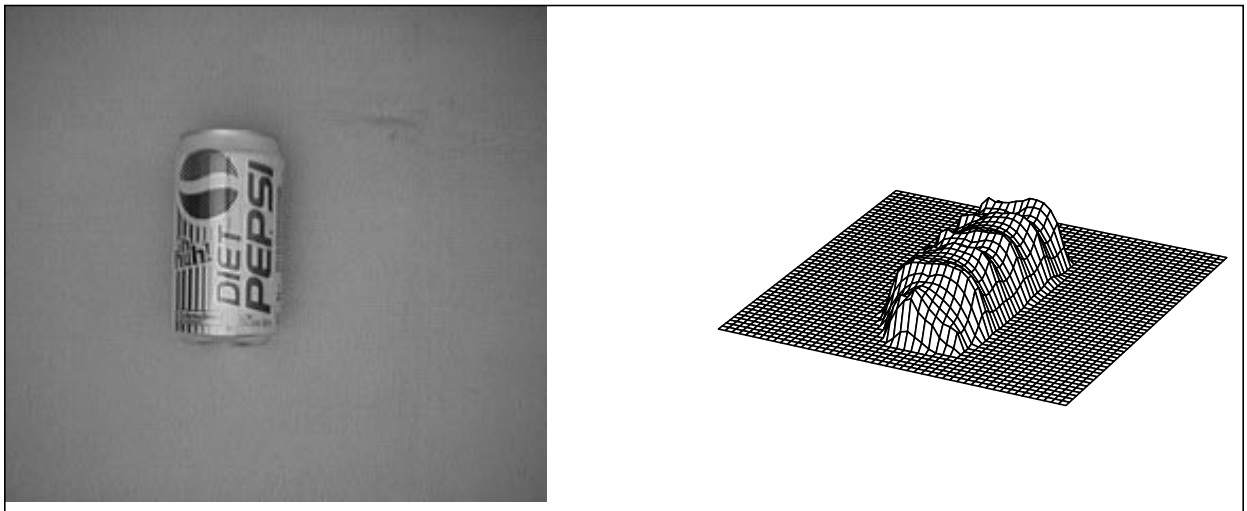
A second set of experiments was conducted using two checkerboard-surfaced soda cans as targets. The leading edges of the cans were placed at 53 cm and 41 cm in depth (see Figure 4 (a)) to demonstrate the ability of the system to find depths over a wide range. Again, the majority of the errors in the calculated depths were sub-pixel. For these runs, the initial depth estimate was an underestimate of 25 cm, in contrast to 1 meter for the previous set of experiments. The reconstructed surfaces of the dual can experiment are shown in Figure 4 (b). Again, the reconstructed surfaces show that the process has captured the curvature of both cans in this experiment.

We duplicated the first two sets of experiments without the checkerboard surfaces. Instead, we used the actual surfaces of the soda cans under normal lighting conditions. This reduced the number of suitable feature points as well as producing a non-uniform distribution of feature points across the surface of the cans. Additionally, the surfaces of the cans exhibited specularities and reflections that added noise to the displacement measures. The target and the result of the single



(a) (b)

Figure 4: Two can target and surface

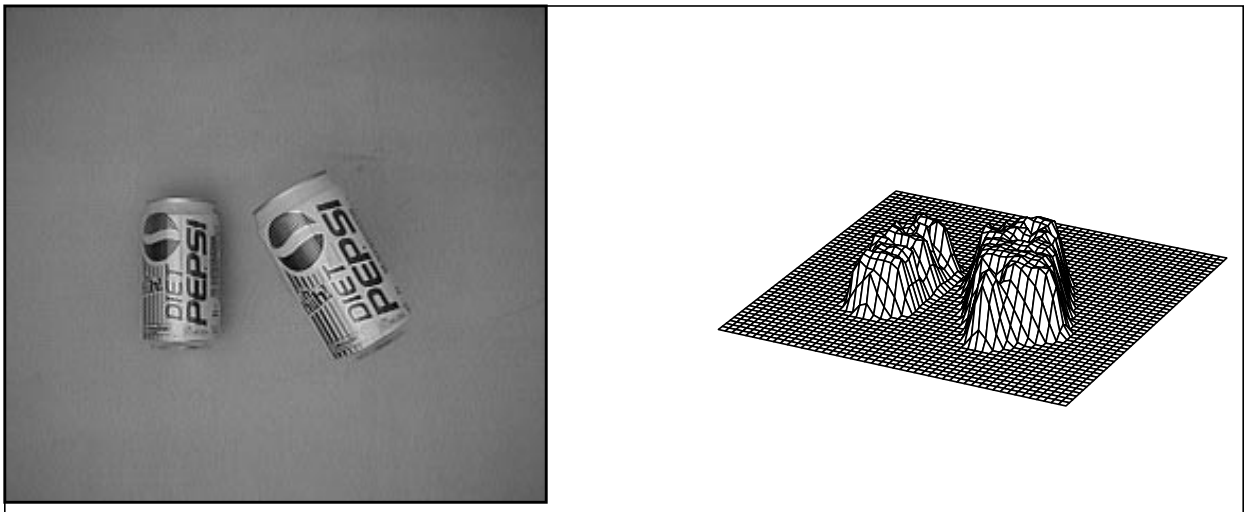


(a) (b)

Figure 5: Single can with real texture and reconstructed surface

can experiment is shown in Figure 5. The surface reconstruction does not exhibit the pronounced curvature found in the earlier experiments on the artificial, high-contrast matte surfaces; however, the errors in the depth measures were typically on the order of a single pixel in nature.

The second set of experiments using the real surfaces of two cans at different depths produced results similar to the single can experiment. In this set of experiments, one can was placed at an angle to the image plane to determine that the method was not sensitive to specific orientations. The targets and the results, shown in Figure 6, demonstrate that the additional noise and the distribution of suitable feature points affected the recovery of the curvature on the two surfaces.



(a) (b)

Figure 6: Dual cans with real texture and reconstructed surface

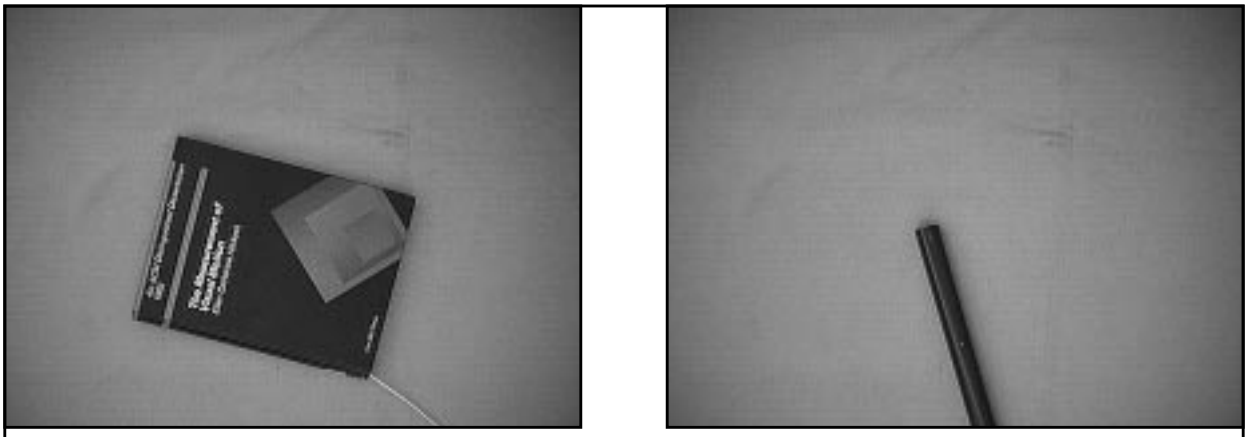


Figure 7: Example targets

6.3 Visual Tracking under X-Y Translation

We conducted multiple experimental runs for the tracking of objects that exhibited unknown two-dimensional, translational motion with a coarse estimate of the depth of the objects. The targets for these runs included books, batons, and a computer-generated target displayed on a large video monitor. During the experiments, we tested the system using targets with linear and curved trajectories.

The first experiments were conducted using a book and a baton (see Figure 7) as targets in order to test the performance of the system both with and without the dynamic pyramiding. These initial experiments served to confirm the feasibility of performing real-time pyramid level switching and to collect data on the performance of the system under the pyramiding and search optimizing schemes. Figure 8 shows the target trajectory and manipulator path from one of these

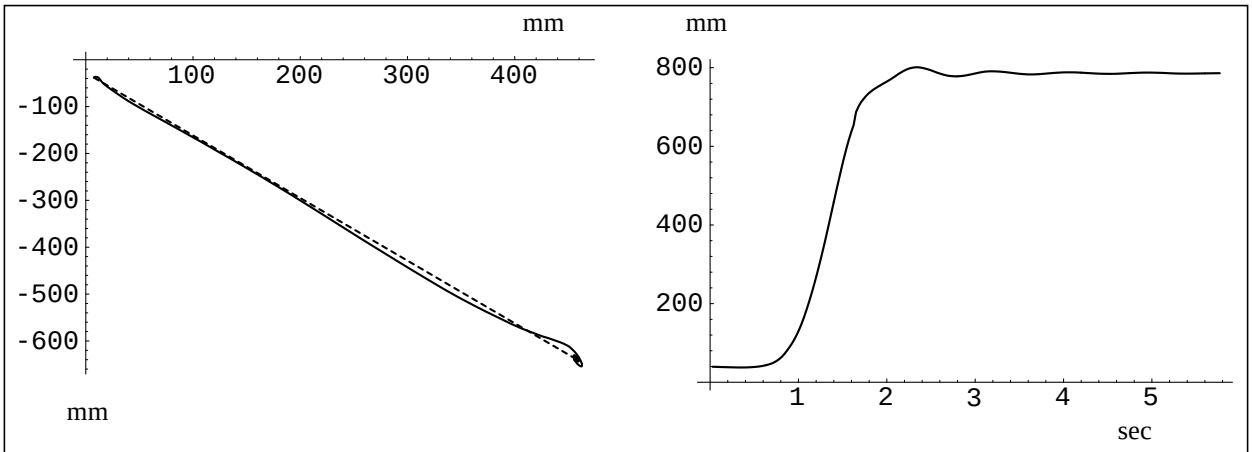


Figure 8: Linear target trajectory and manipulator path

experiments. With the pyramiding and search optimizations, the system was able to track the corner of a book that was moving along a linear trajectory at 80 cm/sec at a depth of approximately 500 cm (see the right plot in Figure 8). In contrast, the maximum speed reported by Papanikolopoulos [30] was 7 cm/sec, representing an order of magnitude increase in tracking speed.

The dashed line in the left plot is the target trajectory and the solid line represents the manipulator trajectory. The start of the experiment is in the upper left corner and the end is where the target and manipulator trajectories come together in the lower right. The oscillation at the beginning is due to the switch to higher pyramiding levels as the system compensates for the speed of the target. The oscillations at the end are produced when the target slows to a stop and the pyramiding drops down to the lowest level and a possible interaction with the pyramiding and an observed controller overshoot in non-pyramided experiments. This results in the manipulator centering above the feature accurately due to the higher resolution available at the lowest pyramid level.

Once system performance met our minimal requirements, testing proceeded to a computer-generated target where the trajectory and speed could be accurately controlled. For these experiments, the setup of the PUMA and the video monitor to display the targets is shown in Figure 9.

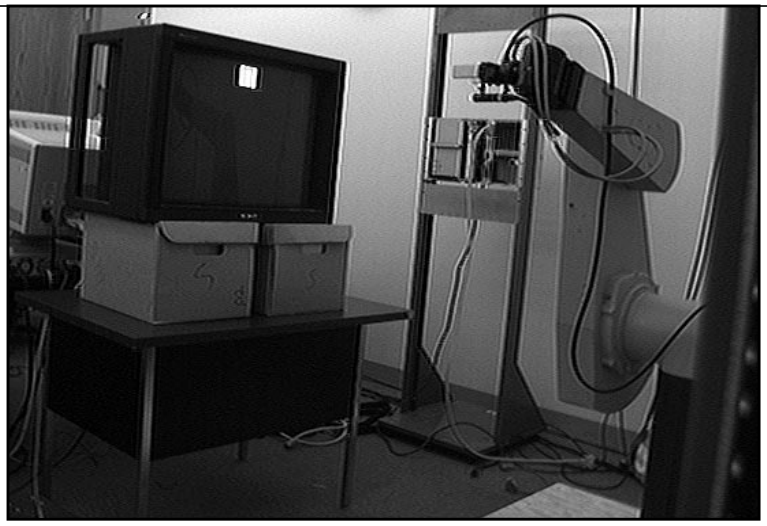


Figure 9: Experimental setup

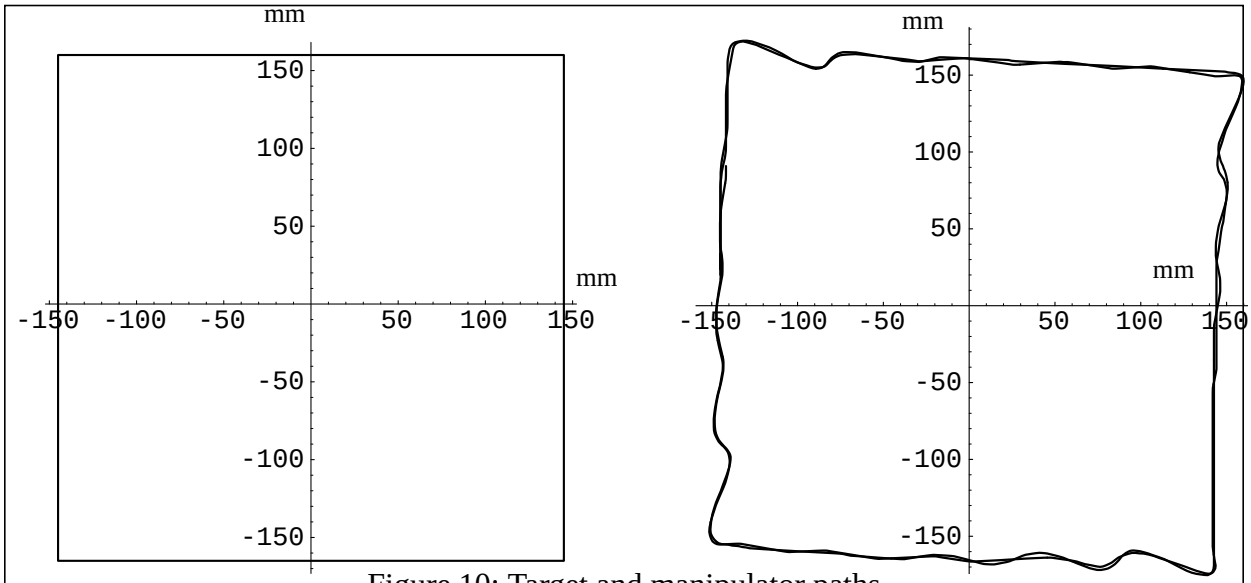


Figure 10: Target and manipulator paths

The target (the white box) traced either a square or a circle on the video monitor while the PUMA tracked the target.

The first experiments traced a square and demonstrated robust tracking in spite of the target exhibiting infinite accelerations (at initial start-up) and discontinuities in the velocity curves (at the corners of the square). The results in Figure 10 clearly show the oscillatory nature of the manipulator trajectory at the points where the velocity curve of the target is discontinuous. The speed of the target in these experiments approached 15 cm/sec.

The next set of experiments was performed using a target that traced a circular path on the video monitor. This path produces a smoothly varying velocity curve while retaining the infinite acceleration of the target at start-up. We allowed the experiments to continue for many circuits of the target in order to determine the response of the system over time. The results demonstrated an initial oscillation due to the acceleration at start-up followed by smooth tracking once the controller compensated for the extreme initial disturbances. The results of the circular tracking are shown in Figure 11. The multiple manipulator paths demonstrate the repeatability achieved using the controller, algorithms, and optimizations. The target in this experiment moved with a speed of 35 cm/sec. The oval-shaped manipulator trajectory visible in the figure is caused by start-up condi-

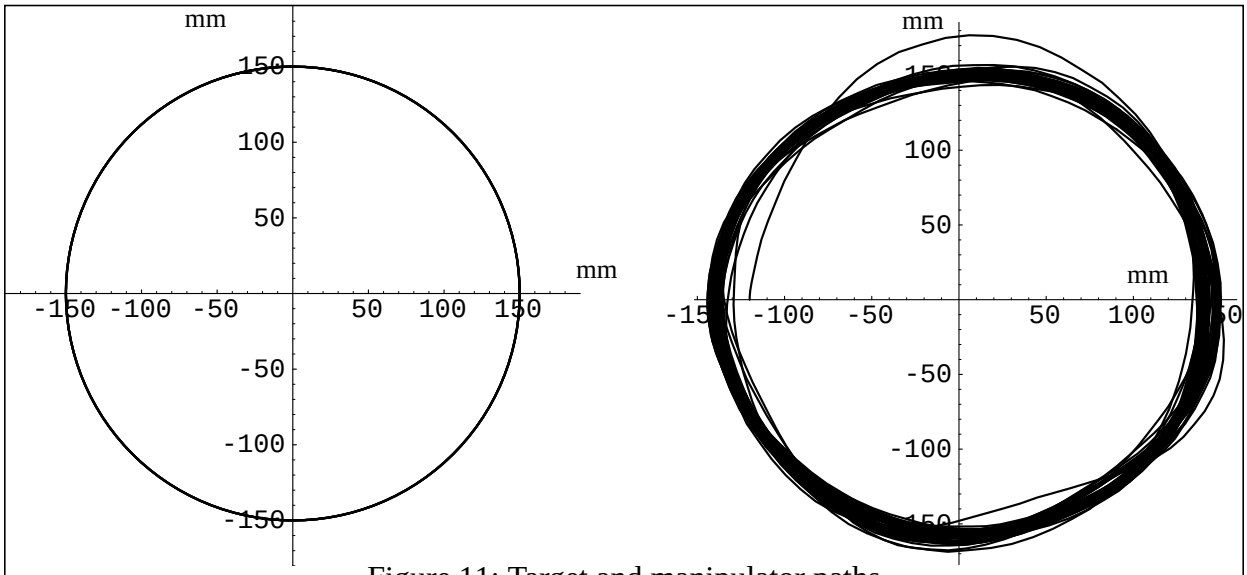


Figure 11: Target and manipulator paths

tions in the system. The last three loops of the manipulator plotted in Figure 12 show the nearly circular trajectory achieved once the system has stabilized.

6.4 Visual Tracking under X-Y Translation and Z Rotation

Further tracking experiments were conducted using the video target and the experimental arrangement previously described (see Figure 9). These experiments introduced a Z-axis rotation of the target while the manipulator is guided using an appropriate system model and controller to tracking the rotational motion [30].

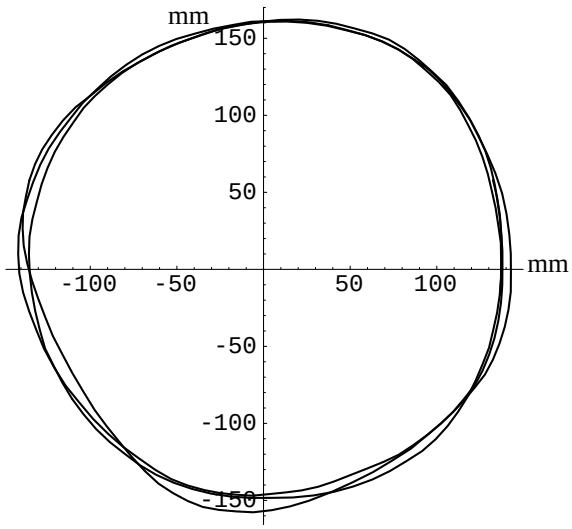


Figure 12: Last three loops of circular tracking

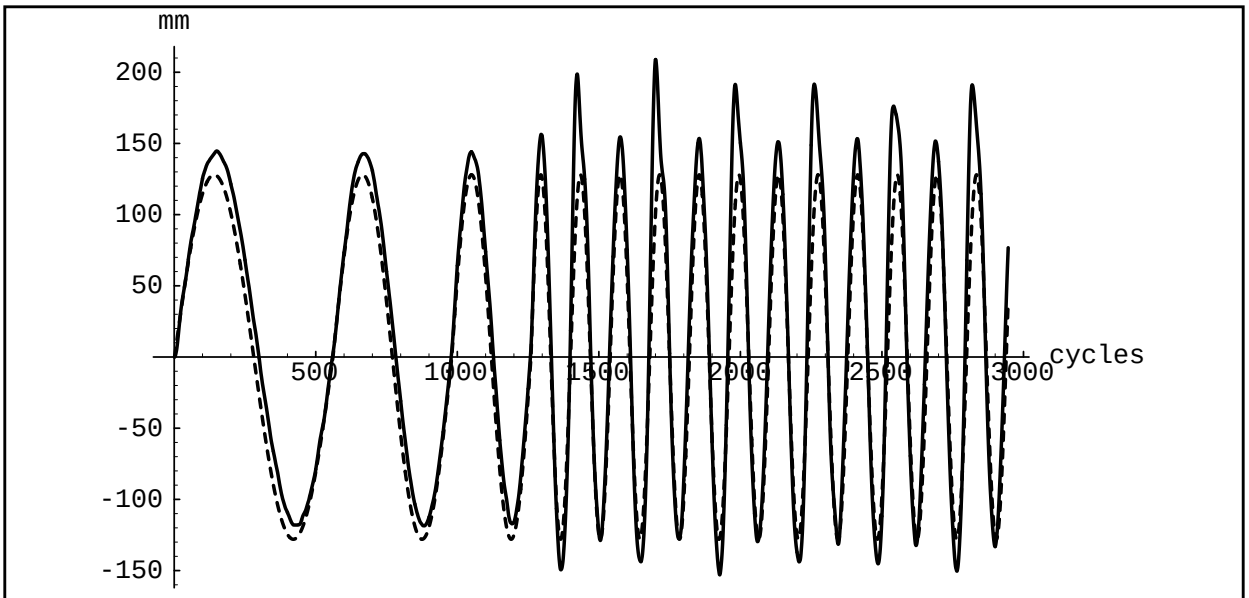


Figure 13: X-component of motion

The translational speed of the target was set at approximately 20 cm/sec as the target traced a roughly circular trajectory (see Figure 13 and Figure 14). The rotational speed of the target during these experiments was set to approximately 11 deg/sec (see Figure 15). To avoid the problems with large discontinuities in accelerations during start-up previously encountered with this experimental configuration, the target completes three circuits of the path at one fourth, one third, and

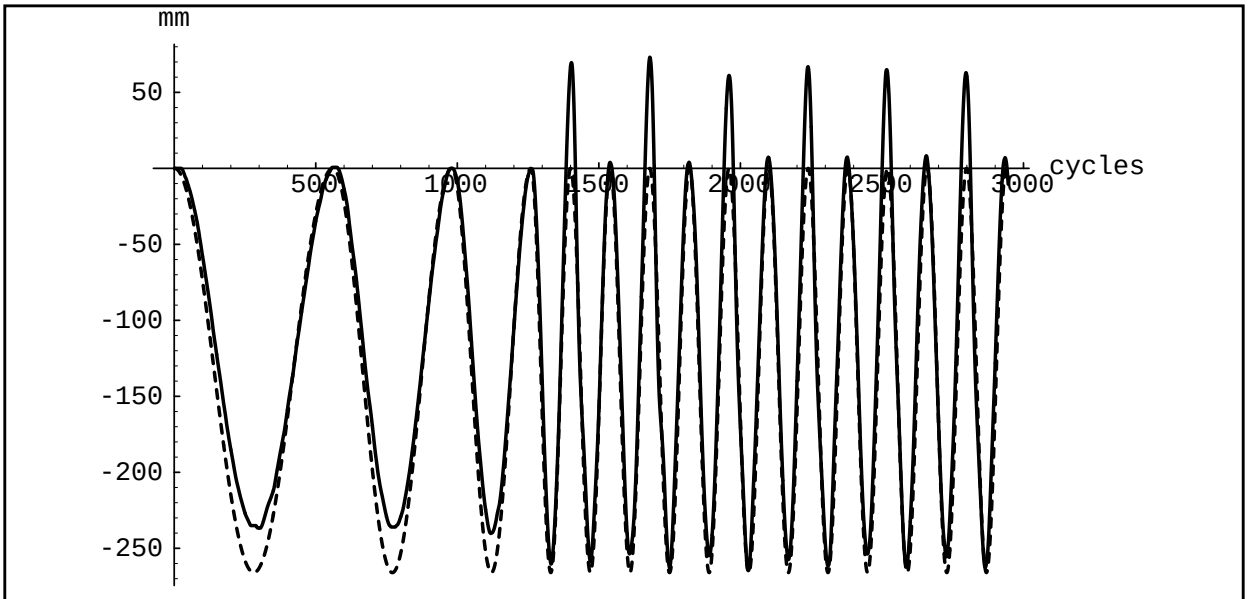


Figure 14: Y-component of motion

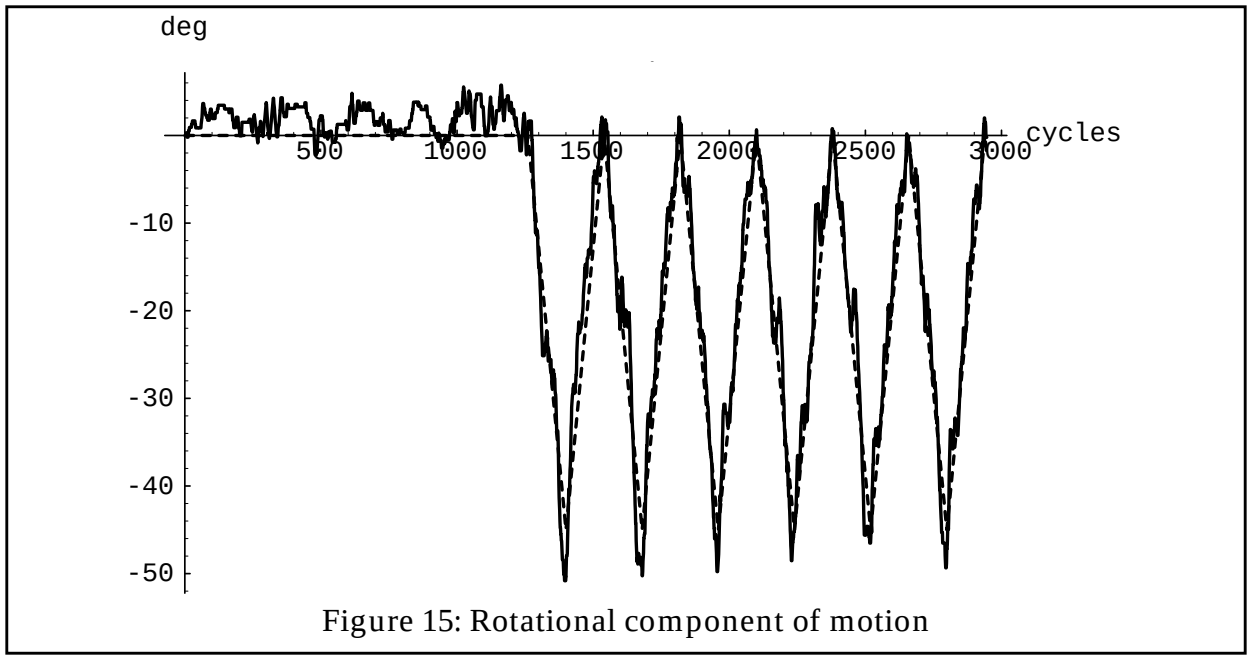


Figure 15: Rotational component of motion

one half of the final speed of the target, respectively. During this velocity "wind-up" the rotational component of the target is held constant at zero (see Figure 15).

In all of the data plots the horizontal axis has a unit of cycles that corresponds to the period of the Puma controller. Each cycle is 28 msec in length. Therefore, the experiment from which this data is taken lasted approximately 82 seconds. The dashed line in the plots is the target trajectory and the solid line represents the manipulator trajectory.

The errors observed in the tracking are due to the latency of the vision system and the coupling of the rotation and translation. This effect is clearly demonstrated by the degradation of the translational tracking accuracy at the point when the target rotation is introduced.

7 Conclusion

This paper presents robust techniques for the operation of robotic agents in uncalibrated environments. The techniques provide ways of recovering unknown workspace parameters using the Controlled Active Vision framework [30]. In particular, this paper discusses novel techniques for computing depth maps and for visual tracking in uncalibrated environments.

For the computation of depth maps, we propose a new scheme that is based on the automatic selection of features and the design of specific trajectories on the image plane for each individual

feature. Unlike similar approaches [20][25][26][33][38], this approach aids the design of trajectories that provide maximum identifiability of the depth parameter. During the execution of the specific trajectory, the depth parameter is computed with a simple estimation scheme that takes into consideration the previous movements of the camera and the computational delays. We have tested the approach and have presented several experimental results.

For the problem of visual tracking, we propose a technique based upon earlier work in visual servoing [30] that achieves superior speed and accuracy through the introduction of several performance-enhancing techniques. Specifically, the dynamic pyramiding technique provides a satisfactory compromise to the speed/accuracy trade-off inherent in static pyramiding techniques. The method-specific optimizations presented also enhance overall system performance without affecting worst-case execution times. These optimizations apply to various region-based vision processing applications (such as blind search and gradient-based descent), and in our application they provide the speedup required to increase the effectiveness of the real-time vision system.

The results we present in this paper demonstrate the effectiveness of the new enhancements to the SSD optical flow calculations and the feasibility of performing real-time visual pyramiding. These enhancements are used in both robotic visual tracking and a new method for the derivation of depth based upon purposeful camera movement and adaptive control. In addition, the techniques presented are easily adapted to other applications, as demonstrated by the preliminary results of two transportation related applications presented in [35]. All of the work presented in this paper has been implemented on the MRVT developed at the University of Minnesota, demonstrating the flexibility of the algorithms that have been presented.

8 Acknowledgments

This work has been supported by the Department of Energy (Sandia National Laboratories) through Contracts #AC-3752D and #AL-3021, the National Science Foundation through Contracts #IRI-9410003 and #IRI-9502245, the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAH04-95-2-0003/contract number DAAH04-95-C-0008 (the content of which

does not necessarily reflect the position of the policy of the government, and no official endorsement should be inferred), the University of Minnesota Doctoral Dissertation Fellowship Program, and McKnight Land-Grant Professorship Program at the University of Minnesota.

9 References

[1] P.K. Allen, "Real-time motion tracking using spatio-temporal filters," *Proceedings of the DARPA Image Understanding Workshop*, 659-701, 1989.

[2] P.K. Allen, B. Yoshimi, and A. Timcenko, "Real-time visual servoing," *Proceedings of the IEEE International Conference on Robotics and Automation*, 851-856, April, 1991.

[3] P. Anandan, "Measuring visual motion from image sequences," Technical Report COINS-TR-87-21, COINS Department, University of Massachusetts, 1987.

[4] M. Athans, *Systems, Networks, and Computation: Multivariable Methods*, McGraw-Hill, New York, NY, 1974.

[5] B. Bhanu, P. Symosek, J. Ming, W. Burger, H. Nasr, and J. Kim, "Qualitative target motion detection and tracking," *Proceedings of the DARPA Image Understanding Workshop*, 370-389, 1989.

[6] S.A. Brandt, "Enhanced robotic visual tracking under the controlled active vision framework," Master's Thesis, Department of Computer Science, University of Minnesota, 1993.

[7] S.A. Brandt, C.E. Smith, and N.P. Papanikolopoulos, "The Minnesota Robotic Visual Tracker: a flexible testbed for vision-guided robotic research," To appear, *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 1994.

[8] F. Chaumette and P. Rives, "Vision-based-control for robotic tasks," *Proceedings of the IEEE International Workshop on Intelligent Motion Control*, 395-400, August 1990.

[9] J.J. Craig, *Introduction to robotics: mechanics and control*, Addison-Wesley, Reading, MA, 1985.

[10] E.D. Dickmanns and A. Zapp, "Autonomous high speed road vehicle guidance by computer vision," *Proceedings of the 10th IFAC World Congress*, July, 1987.

[11] J.T. Feddema and C.S.G. Lee, "Adaptive image feature prediction and control for visual tracking with a hand-eye coordinated camera," *IEEE Transactions on Systems, Man, and Cybernetics*, 20(5):1172-1183, 1990.

[12] J.T. Feddema, C.S.G. Lee, and O.R. Mitchell, "Weighted selection of image features for resolved rate visual feedback control," *IEEE Transactions on Robotics and Automation*, 7(1):31-47, 1991.

[13] J.T. Feddema and O.R. Mitchell, "Vision guided servoing with feature-based trajectory gen-

eration," *IEEE Transactions on Robotics and Automation*, 5(5):691-699, 1989.

[14] D.B. Gennery, "Tracking known three-dimensional objects," *Proceedings of the AAAI 2nd National Conference on AI*, 13-17, 1982.

[15] H. Hashimoto, T. Kubota, M. Kudou, and F. Harashima, "Self-organizing visual servo system based on neural networks," *IEEE Control Systems Magazine*, 12(2):31-36, 1992.

[16] J. Heel, "Dynamic motion vision," *Robotic and Autonomous Systems*, 6(3):297-314, 1990.

[17] A.E. Hunt and A.C. Sanderson, "Vision-based predictive tracking of a moving target," Technical Report CMU-RI-TR-82-15, The Robotics Institute, Carnegie Mellon University, January, 1982.

[18] C.P. Jerian and R. Jain, "Structure from motion—a critical analysis of methods," *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):572-588, 1991.

[19] A.J Koivo and N. Houshangi, "Real-time vision feedback for servoing of a robotic manipulator with self-tuning controller," *IEEE Transactions on Systems, Man and Cybernetics*, 21(1):134-142, 1991.

[20] K.N. Kutulakos and C.R. Dyer, "Recovering shape by purposive viewpoint adjustment," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 16-22, 1992.

[21] D. Lowe, "Fitting parameterized 3-D models to images," Technical Report 89-26, University of British Columbia, 1989.

[22] R.C. Luo, R.E. Mullen Jr., and D.E. Wessel, "An adaptive robotic tracking system using optical flow," *Proceedings of the IEEE International Conference on Robotics and Automation*, 568-573, 1988.

[23] D. Marr, *Vision*, W.H. Freeman and Company, San Francisco, CA, 1982.

[24] D. Marr and T. Poggio, "Cooperative computation of stereo disparity," *Science*, 194(4262):283-287, 1976.

[25] L. Matthies, T. Kanade, and R. Szeliski, "Kalman filter-based algorithms for estimating depth from image sequences," *International Journal of Computer Vision*, 3(3):209-238, 1989.

[26] L. Matthies, "Passive stereo range imaging for semi-autonomous land navigation," *Journal of Robotic Systems*, 9(6):787-816, 1992.

[27] P.S. Maybeck, *Stochastic models, estimation, and control*, Vol. 1, Academic Press, Orlando, FL, 1979.

[28] W.T. Miller, "Real-time applications of neural-networks for sensor-based control of robots with vision," *IEEE Transactions on Systems, Man, and Cybernetics*, 19(4):825-831, 1989.

[29] M. Okutomi and T. Kanade, "A locally adaptive window for signal matching," *International Journal of Computer Vision*, 7(2):143-162, 1992.

[30] N. Papanikolopoulos, "Controlled active vision," Ph.D. Thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1992.

[31] N. Papanikolopoulos, P.K. Khosla, and T. Kanade, "Visual tracking of a moving target by a camera mounted on a robot: a combination of control and vision," *IEEE Transactions on Robotics and Automation*, 9(1):14-35, 1993.

[32] J.W. Roach and J.K. Aggarwal, "Computer tracking of objects moving in space," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(6):583-588, 1979.

[33] C. Shekhar and R. Chellappa, "Passive ranging using a moving camera," *Journal of Robotic Systems*, 9(6):729-752, 1992.

[34] C.E. Smith, S.A. Brandt, and N.P. Papanikolopoulos, "Controlled active exploration of uncalibrated environments," *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 792-795, 1994.

[35] C.E. Smith, S.A. Brandt, and N.P. Papanikolopoulos, "Vision sensing for intelligent vehicle and highway systems," To appear, *Proceedings of the IEEE International Conference on Multi-sensor Fusion and Integration*, 1994.

[36] C.E. Smith and N.P. Papanikolopoulos, "Computation of shape through controlled active exploration," *Proceedings of the IEEE International Conference on Robotics and Automation*, 2516-2521, 1994.

[37] D.B. Stewart, D.E.Schmitz, and P.K. Khosla, "CHIMERA II: A real-time multiprocessing environment for sensor-based robot control," *Proceedings of the Fourth International Symposium on Intelligent Control*, 265-271, September, 1989.

[38] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *International Journal of Computer Vision*, 9(2):137-154, 1992.

[39] M.M. Trivedi, C. Chen, and S.B. Marapane, "A vision system for robotic inspection and manipulation," *Computer*, 22(6):91-97, June, 1989.

[40] D. Tsakiris, "Visual tracking strategies," Master's Thesis, Department of Electrical Engineering, University of Maryland, 1988.

[41] 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):404-417, October, 1987.

[42] D. Weinshall, "Direct computation of qualitative 3-D shape and motion invariants," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(12):1236-1240, 1991.

[43] P. H. Winston, *Artificial Intelligence*, Addison-Wesley, Reading, MA, 1992.

[44] P.R. Wolf, *Elements of Photogrammetry, with Air Photo Interpretation and Remote Sensing*, McGraw-Hill Book Co., New York, 1983.

[45] B.H. Yoshimi and P. K. Allen, “Active, uncalibrated visual servoing,” *Proceedings of the IEEE International Conference on Robotics and Automation*, 156-161, 1994.