# Visual Tracking of a Moving Target by a Camera Mounted on a Robot: A Combination of Control and Vision

Nikolaos P. Papanikolopoulos, *Member, IEEE*, Pradeep K. Khosla, *Senior Member, IEEE*, and Takeo Kanade, *Fellow, IEEE*

*Abstract*—**In this paper, we present algorithms for robotic (eye-in-hand configuration) real-time visual tracking of arbitrary 3-D objects traveling at unknown velocities in a 2-D space (depth is given as known). We formulate the problem of visual tracking as a problem of combining control with computer vision. We present a mathematical formulation of a control problem that includes the sensory information of a novel and important feedback sensor (vision sensor). This formulation represents everything with respect to the camera frame and not with respect to the world frame. Due to this fact, we have the ability to quickly and accurately control the camera. We propose using the sum-of-squared differences (SSD) optical flow for the computation of the vector of discrete displacements each instant of time. These displacements can be fed either directly to a PI controller or to a pole assignment controller or to a discrete steady-state Kalman filter. In the latter case, the Kalman filter calculates the estimated values of the system's states and the exogenous disturbances, and a discrete LQG controller computes the desired motion of the robotic system. The outputs of the controllers are sent to a Cartesian robotic controller that drives the robot. The performance of the proposed algorithms has been tested on the CMU Direct-Drive (DD) Arm II, and the results are presented in this paper.**

## I. Introduction

An important component of a robotic system is the acquisition, processing, and interpretation of the available sensory information. At the lowest level, the sensing information is used to derive control signals to drive the robot, and at a higher level this information is used to create models of the system and the environment. The sensory information can be obtained through a variety of sensors such as position, velocity, force, tactile, and vision to cite a few. In this paper, we address the use of vision for dynamically servoing a manipulator for object tracking.

Manuscript received June 3, 1991; revised June 2, 1992. This work was supported in part by the Defense Advanced Research Projects Agency through DARPA Order DAAA-21-89C-0001 and by the Department of Electrical and Computer Engineering and the Robotics Institute of Carnegie Mellon University. Portions of this paper were presented at the IEEE International Conference on Robotics and Automation, Sacramento, CA, 1991.

N. P. Papanikolopoulos is with the Department of Computer Science, University of Minnesota, Minneapolis, MN 55455.

P. K. Khosla is with the Department of Electrical and Computer Engineering, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

T. Kanade is with the Department of Computer Science, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

IEEE Log Number 9203044.

Research in computer vision has traditionally emphasized the paradigm of image understanding. However, some work has been reported toward the use of vision information for tracking [2], [5], [14], [20], [22], [24], [34]. In addition, some research [9], [36], [37] has been conducted on the use of vision information in the dynamic feedback loop. While we address the problem of using vision information in the dynamic feedback loop, our paradigm is slightly different. Specifically, we claim that combining vision with control can result in better tracking. Better tracking and servoing correspond to more effective and more accurate active vision algorithms such as the ones used in the derivation of the structure of environment through motion. It is in this context that we view our current work, which shows that noisy measurements from a vision sensor when combined with an appropriate control law can lead to an acceptable performance of a visual servoing algorithm.

Hunt and Sanderson [20] presented algorithms for visual tracking based on mathematical prediction of the position of the object's centroid. Their algorithm needed the computation of the coordinates of the centroid and could only track slowly moving objects. Weiss *et al.* [36] proposed solutions to the problem of robotic visual tracking under the framework of model reference adaptive control. The proposed framework was verified with a number of simulated examples. Lee and Wohn [22] used image differencing techniques to track the object. Luo *et al.* [24] presented a robot conveyor tracking system that incorporates a combination of visual and acoustic sensing. They used a modified version of the Horn–Schunk [18] algorithm for the calculation of the optical flow at every pixel. This algorithm accomplished 1-D visual tracking and assumed knowledge of the conveyor's speed. Goldenberg *et al.* [14] used the PIPE real-time image processing machine to do visual tracking. Their method used temporal filtering. Corke and Paul [6] used a feature-based method to drive the robotic device. Allen *et al.* [2], [3] presented a method for real-time motion tracking that was based on the idea of spatio-temporal filtering [16]. In addition, Allen *et al.* [3] used the visual information provided by a pair of stationary cameras in order to track a moving object with a robot. Dickmanns and Zapp [7], [8] presented several methods (Kalman filters) for the integration of vision information in the feedback loop of various mechanical systems such as satellites and cars. Tsakiris [34] proposed strategies for visual tracking that are based on

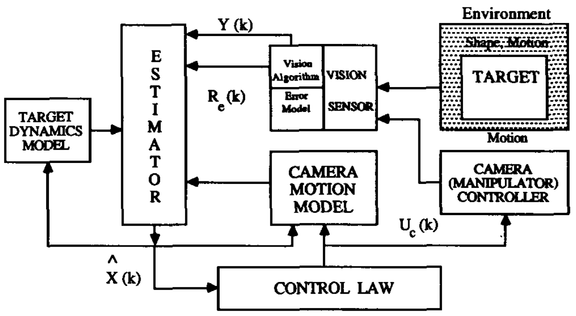


Fig. 1. Architecture of the robotic visual tracking/servoing system.

the computation of the optical flow and the computation of the object's moments. Feddema *et al.* [9]–[11] emphasized the control aspect of the robotic visual tracking problem. Koivo and Houshangi [21] used adaptive control techniques in conjunction with the information provided by a stationary camera in order to control the robotic device.

In this paper, we present algorithms that address the real-time robotic visual tracking (eye-in-hand configuration) of 3-D objects in 2-D space (depth is given as known). In order to achieve our objective, we combine computer vision techniques that detect and measure motion with simple control strategies. The problem has been formulated from the system's theory point of view. Our formulation really concerns a control problem with a novel and important feedback sensor (vision sensor). The architecture of our framework is depicted in Fig. 1. A cross-correlation technique (SSD optical flow) is used for computing the vector of discrete displacements and is combined with an appropriate control scheme to calculate the required motion of the robotic system.

The control schemes that we have used ranged from the simple PI controller to more complex LQG and pole assignment controllers. We introduce algorithms that incorporate sophisticated use of multiple windows and numerically stable confidence measures. In this way, the accuracy of the visual measurements is increased. The selection of the controller is based on the vision technique that is used for the computation of the displacement vector. In particular, multiple windows give accurate measurements, and thus, a simple PI controller is adequate. On the other hand, a small number of windows provides inaccurate measurements and stochastic controllers should be used. The experimental results show that the system performs satisfactorily even with noisy measurements and adapts well to changes in the object's motion. The proposed scheme is modular and thus allows for different techniques to be implemented for the calculation of the vector of discrete displacements. In other words, we can replace the vision algorithm with a new one without changing the basic structure of the tracking system.

The rest of the paper is devoted to the description of our algorithms and is organized as follows: In Section II we review the definition of the optical flow and present methods for computation of the vector of discrete displacements. We also introduce three types of confidence measure for each of the measurements made. The mathematical formulation of the visual tracking problem is described in Section III. The control strategies, the steady-state Kalman filter, and the selection of the appropriate control law with regard to the noise level of the measurements are discussed in Section IV. Section V describes the hardware configuration of our experimental testbed, DD Arm II. Simulation and experimental results are presented in Section VI. Finally, in Section VII, the paper is summarized.

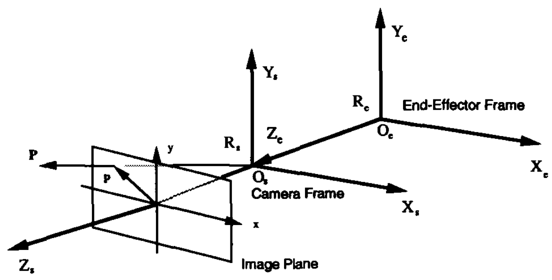


Fig. 2. Camera $R_s$ and end-effector $R_e$ frames.

## II. Optical Flow

An object in an image consists of brightness patterns. As the object moves in 3-D space, the brightness patterns in the image move simultaneously. Horn [19] defines the *optical flow* as "the apparent motion of the brightness patterns." In the case of rigid objects, the optical flow corresponds well to the motion field. We use optical flow as the basis for the computation of the robot's driving signals. In the sequel, we present an outline of our vision techniques in order to illustrate their special characteristics (noise, computational complexity, quantization errors). This outline is essential due to the fact that these characteristics should be taken into consideration in the design of a vision-based controller. In this way, the combination of control and vision techniques will lead to an accurate solution of the robotic visual tracking problem.

We assume a pinhole camera model with a frame $R_s$ attached to it (Fig. 2). We also assume a perspective projection and the focal length to be unity. A point[1] $\boldsymbol{P}$ with coordinates $(X_s, Y_s, Z_s)$ in reference frame $R_s$ projects onto a point $\boldsymbol{p}$ in the image plane with image coordinates $(x, y)$. Let us assume that the camera moves in a static environment with a translational velocity $\boldsymbol{T} = (T_x, T_y, T_z)^T$ and with an angular velocity $\boldsymbol{R} = (R_x, R_y, R_z)^T$ with respect to the camera frame $R_s$. The optical flow equations are [34]

$$u = \left[x\frac{T_z}{Z_s} - \frac{T_x}{Z_s}\right] + [xyR_x - (1 + x^2)R_y + yR_z] \quad (1)$$

$$v = \left[y\frac{T_z}{Z_s} - \frac{T_y}{Z_s}\right] + [(1 + y^2)R_x - xyR_y - xR_z] \quad (2)$$

where $u = \dot{x}$ and $v = \dot{y}$. Now, instead of assuming a static object and a moving camera, assume a static camera and a moving object; then we would obtain the same results as in (1) and (2) except for a sign reversal. The computation of $u$ and $v$ has been the focus of much research and many algorithms have been proposed [1], [16], [18], [32], [33], [35].

[1] Boldface symbols denote vectors or matrices.

For accuracy reasons, we use a matching-based technique [4] also known as the sum-of-squared differences (SSD) optical flow. For a point $\boldsymbol{p}(k-1) = (x(k-1), y(k-1))^T$ in image $(k-1)$ (the symbol $k$ denotes the image $(k)$ which belongs to a sequence of images) we want to find the point $\boldsymbol{p}(k) = (x(k-1)+u, y(k-1)+v))^T$ to which the point $\boldsymbol{p}(k-1)$ moves in image $(k)$. We assume that the intensity values in the neighborhood $N$ of $\boldsymbol{p}(k-1)$ remain almost constant over time, that the point $\boldsymbol{p}(k)$ is within an area $\Omega$ of $\boldsymbol{p}(k-1)$, and that velocities are normalized by time $T$ to get the displacements. Thus, for the point $\boldsymbol{p}(k-1)$ the SSD algorithm selects the displacement $\boldsymbol{d} = (u, v)^T$ that minimizes the SSD measure

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

where $u, v \in \Omega, N$ is an area around the pixel in which we are interested, $m$ and $n$ are the indices for different pixels in the neighborhood $N$, and $I_{k-1}(\cdot,\cdot)$ and $I_k(\cdot,\cdot)$ are the intensity functions in images $(k-1)$ and $(k)$, respectively. The accuracy of this technique can be improved using subpixel fitting and multigrid techniques at the cost of increasing the computational complexity. Its computational speed can be improved by selecting an appropriate small area $N$ and by having velocity fields with few quantization levels. The selection of the window size $(N)$ is important. A small window will not provide an accurate displacement vector in areas where the intensity is almost uniform. On the other hand, an extremely big window also will not provide an accurate displacement vector because it enhances the background of the object. Thus, the algorithm can fail to detect small displacements of the moving object. In addition, the SSD technique fails when the image contains a lot of repeated patterns of same intensity because of multiple matches.

The accuracy of the measurements of the displacement vector can be improved by using multiple windows. The selection of the best measurements is based on the confidence measure of each window. The definition of an efficient and robust confidence measure is critical since images are a noisy source of information and changes in illumination and surface reflectance can deteriorate the performance of any confidence measure. An efficient confidence measure should recognize errors that are due to homogeneous areas and occlusion boundaries. Anandan [4] developed a confidence measure that confronts the majority of these problems. He defined as an *SSD surface* the surface that is created by the different SSD values that correspond to different possible displacements. This surface is used to provide information about the quality of the measurements. An SSD surface that corresponds to a corner point (one of the best features that can be used) presents a sharp minimum at the best match (Fig. 3). A feature point that belongs to an edge (these points provide accurate measurements only in the direction perpendicular to the edge) has an SSD surface that presents multiple local minima in the direction of this edge (Fig. 4). Finally, an SSD surface that corresponds to a feature point that belongs to an homogeneous area (these feature points provide inaccurate data) has multiple small local minima in all the directions (Fig. 5).

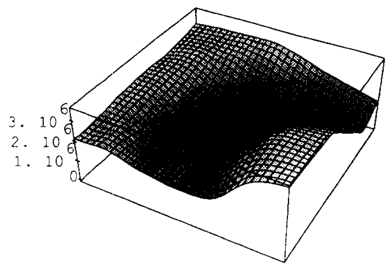


Fig. 3. SSD surface that corresponds to a corner point.

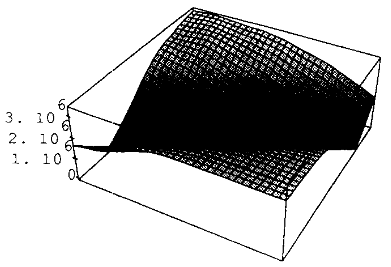


Fig. 4. SSD surface that corresponds to a feature point that belongs to an edge.

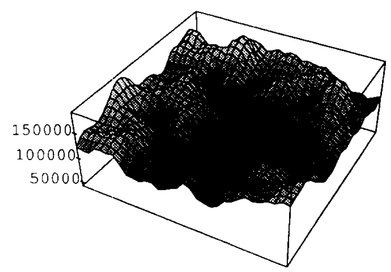


Fig. 5. SSD surface that corresponds to a feature point that belongs to an homogeneous area.

We need a confidence measure that can capture all the topological changes of the SSD surface. It is important to mention that the shape of the SSD surface is maintained even under significant noise corruption [25]. The curvature of the SSD surface seems to be proportional to the quality of the best match [25]. Anandan proposed an algorithm for computing the confidence measures based on the curvatures of the four main axes at the SSD surface minimum. The problem with Anandan's confidence measure is that it is based on the computation of the discrete second order derivatives. This computation is an ill-conditioned problem. Thus, this confidence measure is not robust in the presence of noise. Another problem appears when this confidence measure is applied to a window centered around a point that belongs to an edge. In the direction of the edge, the normalized directional second derivative is close to 1, and thus, the algorithm fails.

Matthies *et al.* [25] computed the variance in the estimate of one-dimensional displacement. The computation is based on a parabolic fit to the SSD curve. The variance has been found to be $2\sigma_I^2/a$ where $\sigma_I^2$ is the variance of the image noise and $a$ is the second-order coefficient of the parabola $e(d_r) = ad_r^2 + bd_r + c$. We have proposed an extension of this technique to two dimensions by fitting parabolas to the directions of the four main axes in the area of the SSD surface minimum [26]. Thus, we can compute the confidence measure for the $i$th window as

$$\text{conf}_i = \min(a_0, a_{45}, a_{90}, a_{135}) \tag{4}$$

where $a_0, a_{45}, a_{90}$, and $a_{135}$ are the second-order coefficients of the parabolas that are fit to the directions of the four main axes. They are computed by using the least squares method. Their computation increases the computational load, but it improves the accuracy of the measurements.

In addition to the previously proposed confidence measure, we have introduced two new confidence measures that have been used in the experiments [26]. Their advantage is that they capture the sharpness and the local properties of the minimum instead of fitting mathematical models of surfaces to the SSD surface. The reasoning behind this is that a second-order approximation of the SSD surface is not always the best representation.

The first confidence measure describes statistically the sharp minimum. Thus, for the $i$th window the confidence measure is

$$\text{conf}_i = \min(s_0, s_{45}, s_{90}, s_{135}) \tag{5}$$

where the $s_k$'s are the sample standard deviations in the directions of the four main axes. Each one of these standard deviations is computed as

$$s_k^2 = \frac{1}{M-1}\left[\sum_{j=1}^{j=M}(e_j^2) - M\bar{e}^2\right] \tag{6}$$

where the minimum of the SSD surface is the median of the samples of size $M$, and $e$ denotes the value of the SSD surface. Each sample consists of the values of the SSD surface adjacent to the minimum in each of the four main axes. The symbol $\bar{e}$ denotes the mean value of each of the samples. Due to the local character of the minimum, the number $M$ should be small. In addition, large $M$ increases the computational load.

The second confidence measure tries, in a heuristic manner, to capture the characteristics of the minimum. In this case, the confidence measure for the $i$th window is

$$\text{conf}_i = \min(s_0, s_{45}, s_{90}, s_{135}) \tag{7}$$

where the $s_k$'s are given by the equation

$$s_k^2 = \frac{1}{M-1}\sum_{j=1}^{j=M}(e_j - e_{\min})^2. \tag{8}$$

The symbol $e_{\min}$ represents the minimum value of the SSD surface and the $s_k$'s are computed along the four main axes.

The selection of the best measurements is based on the values of their confidence measures. If the object moves with two translational degrees of freedom in 2-D space, we need only one feature point. Thus, we have to select the point that has the highest confidence measure. A more complex scheme would involve averaging the measurements that correspond to feature points that have the same depth $Z_s$. This can create large errors due to the fact that we give equal weight to both good and bad measurements. This difficulty can be overcome by assigning different weights to the measurements. We define the $i$th weight $w_i = \text{conf}_i/(\Sigma_j \text{conf}_j)$, where $\text{conf}_i$ is the confidence measure of the $i$th window. The arithmetic means of $u$ and $v$ are

$$\bar{u} = \frac{\sum_j w_j u_j}{\sum_j w_j}, \qquad \bar{v} = \frac{\sum_j w_j v_j}{\sum_j w_j}. \tag{9}$$

In addition, the standard errors of the arithmetic means can be computed as

$$s_{\bar{u}}^2 = \frac{\sum_j w_j(u_j - \bar{u})^2}{(n-1)\sum_j w_j}, \qquad s_{\bar{v}}^2 = \frac{\sum_j w_j(v_j - \bar{v})^2}{(n-1)\sum_j w_j} \tag{10}$$

where $n$ is the number of the feature points that are used. Equations (9) and (10) can be simplified since $\sum_j w_j = 1$.

The techniques discussed above can be extended from black/white images to color images. In [26], a new SSD measure that is efficient for use with color images is proposed. The next step in our algorithm is the use of these measurements in the visual tracking process. Our algorithm transforms these measurements into control commands to the camera system. Thus, a mathematical model for this transformation must be developed. In the next section, we develop the mathematical model for the visual tracking problem. This model combines both control and vision algorithms in order to achieve accurate robotic visual tracking.

## III. Mathematical Formulation of the Visual Tracking Problem

First, we will present the mathematical model for the visual tracking of a feature point. Then, based on this model, we will develop the mathematical model for the 2-D visual tracking of an object. The 2-D visual tracking of an object is realized by visually tracking multiple feature points that belong to the object.

### *A. Visual Tracking of a Single Feature Point*

Consider a target moving on a plane with a feature, located at point $\boldsymbol{P}$, that we want to track. The projection of this point on the image plane is the point $\boldsymbol{p}$. Consider also a neighborhood $\Omega_w$ of $\boldsymbol{p}$ in the image plane. The problem of 2-D visual tracking of a single feature point can be defined as: "find the camera translation $(T_x, T_y)$ with respect to the camera frame that keeps $\Omega_w$ stationary in an area $\Omega_o$ around the origin of the image frame." It is assumed that at initialization of the tracking process, the area $\Omega_w$ is brought to the origin of the image frame and that the plane of motion is perpendicular to the optical axis of the camera. The problem of visual tracking of a single feature point can also be defined as [34]: "find the camera rotation $(R_x, R_y)$ with respect to the camera frame that keeps $\Omega_w$ stationary in an area $\Omega_o$ around the origin of the image frame." Defining the problem in terms of rotations $R_x$ and $R_y$ does not require the determination of depth $Z_s$ of the point $\boldsymbol{P}$.

Assume that the optical flow of point $\boldsymbol{p}$ at instant of time $kT$ is $(u(kT), v(kT))$ where $T$ is the time between two consecutive frames. It can be show that at time $kT$, the optical flow is

$$u(kT) = u_o(kT) + u_c(kT) \tag{11}$$
$$v(kT) = v_o(kT) + v_c(kT) \tag{12}$$

here $u_c(kT)$ and $v_c(kT)$ are the components of the optical flow induced by the tracking motion of the camera, and $u_o(kT)$ and $v_o(kT)$ are the components of the optical flow induced by the motion of the feature. Equations (11) and (12) do not include any computational delays that are associated with the computation and the realization of the tracking motion of the camera. If we include these delays, (11) and (12) are transformed to

$$u(kT) = u_o(kT) + u_c((k-d)T) \tag{13}$$
$$v(kT) = v_o(kT) + v_c((k-d)T) \tag{14}$$

where $d$ is the delay factor. In order to keep the notation simple and without any loss of generality, we assume $d = 0$ and (11) and (12) will therefore be used with index $k$ instead of $kT$.

By using the relations $u(k) = (x(k+1) - x(k)/T)$ and $v(k) = (y(k+1) - y(k)/T)$, (11) and (12) can be transformed to state-space form:

$$\boldsymbol{p}(k+1) = \boldsymbol{A}\boldsymbol{p}(k) + \boldsymbol{B}\boldsymbol{u}_c(k) + \boldsymbol{E}\boldsymbol{d}(k) + \boldsymbol{H}\boldsymbol{v}(k) \tag{15}$$

where[2] $\boldsymbol{A} = \boldsymbol{H} = \boldsymbol{I}_2, \boldsymbol{B} = \boldsymbol{E} = T\boldsymbol{I}_2, \boldsymbol{p}(k) \in R^2, \boldsymbol{u}_c(k) \in R^2, \boldsymbol{d}(k) \in R^2$, and $\boldsymbol{v}(k) \in R^2$. The vector $\boldsymbol{p}(k) = (x(k), y(k))^T$ is the state vector, $\boldsymbol{u}_c(k) = (u_c(k), v_c(k))^T$ is the control input vector, $\boldsymbol{d}(k) = (u_o(k), v_o(k))^T$ is the exogenous disturbances vector, and $\boldsymbol{v}(k) = (v_1(k), v_2(k))^T$ is the white noise vector. Equation (15) will be used under the assumption that the optical flow induced by motion of the feature does not change significantly in the time interval $T$. In addition, $T$ should be as small as possible in order for the relations that provide $u(k)$ and $v(k)$ based on the feature coordinates to be accurate. Unless we use special cameras, $T$ cannot be smaller than 16 ms, which is the intrinsic field rate of the RS-170 signal from the camera. The current total computational time is 100 ms (acquisition of the image is included). The maximum speed that can be tracked is 7 cm/s given an average depth of $Z_s = 680$ mm and a camera with the characteristics described in the experimental section. The measurement vector $\boldsymbol{y}(k) = (y_1(k), y_2(k))^T$ is computed using the SSD algorithm and is given by

$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{p}(k) + \boldsymbol{w}(k) \tag{16}$$

where $\boldsymbol{w}(k) = (w_1(k), w_2(k))^T$ is a white noise vector ($\boldsymbol{w}(k) \sim N(\boldsymbol{0}, \boldsymbol{W})$) and $\boldsymbol{C} = \boldsymbol{I}_2$.

[2]The symbol $\boldsymbol{I}_n$ denotes the identity matrix of order $n$.

If the camera tracks the feature point with translation $T_x(k)$ and $T_y(k)$ with respect to the camera frame, then the optical flow that is generated by the motion of the camera is [34]

$$u_c(k) = -\frac{T_x(k)}{Z_s} \tag{17}$$
$$v_c(k) = -\frac{T_y(k)}{Z_s}. \tag{18}$$

We assume that, for 2-D visual tracking, the depth $Z_s$ remains constant. When the tracking motion of the camera consists of rotations $R_x(k)$ and $R_y(k)$, the optical flow induced by the moving camera is [34]

$$u_c(k) = R_x(k)x(k)y(k) - R_y(k)[x^2(k) + 1] \tag{19}$$
$$v_c(k) = R_x(k)[y^2(k) + 1] - R_y(k)x(k)y(k). \tag{20}$$

The model in (15) and (16) can also be used for keeping the feature point stationary in an area $\Omega_r$ different from the origin. If $(r_x, r_y)$ is the center of this area $\Omega_r$, then by transforming the state variables $x(k)$ and $y(k)$ to a new pair of state variables $x_N(k)$ and $y_N(k) (x_N(k) = x(k) - r_x$ and $y_N(k) = y(k) - r_y)$, we obtain a model for keeping the feature point stationary. The matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{E}$, $\boldsymbol{H}$ remain unchanged under the transformation. The SSD algorithm continuously computes the displacement vector of the feature point from its desired position. Thus, we have the ability to compensate for previous measurement errors that tend to accumulate.

### *B. 2-D Visual Tracking of an Object*

Consider a target that moves on a plane perpendicular to the optical axis of the camera. The projection of the target on the image plane is the area $\Omega_w$ in the image plane. The problem of 2-D visual tracking of a single object can be defined as: "find the camera translation $(T_x, T_y)$ and rotation $(R_z)$ with respect to the camera frame that keeps $\Omega_w$ stationary." It is assumed that the target rotates around an axis $Z$ which at $k = 0$ coincides with the optical axis of the camera. The problem of 2-D visual tracking is fundamentally the same as the problem of tracking several feature points simultaneously. Consequently, a mathematical model for this task can be derived based on the derivation (for a single feature point) as in Section III-A. For completeness sake, the detailed derivation of this model is shown in the Appendix. The state-space representation of this model is

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}_c(k) + \boldsymbol{E}\boldsymbol{d}(k) + \boldsymbol{H}\boldsymbol{v}(k) \tag{21}$$

where $\boldsymbol{A} = \boldsymbol{H} = \boldsymbol{I}_3$, $\boldsymbol{B} = \boldsymbol{E} = T\boldsymbol{I}_3$, $\boldsymbol{x}(k) \in R^3$, $\boldsymbol{u}_c(k) \in R^3$, $\boldsymbol{d}(k) \in R^3$, and $\boldsymbol{v}(k) \in R^3$. The vector $\boldsymbol{x}(k) = (x(k), y(k), \theta(k))^T$ is the state vector, $\boldsymbol{u}_c(k) = (u_c(k), v_c(k), -R_z(k))^T$ is the control input vector, $\boldsymbol{d}(k) = (u_o(k), v_o(k), \omega(k))^T$ is the exogenous disturbances vector, and $\boldsymbol{v}(k) = (v_1(k), v_2(k), v_3(k))^T$ is the white noise vector. $x(k), y(k)$, and $\theta(k)$ are now the $X$, $Y$, and roll components of the tracking error, respectively. The measurement vector $\boldsymbol{y}(k) = (y_1(k), y_2(k), y_3(k))^T$ is

$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{w}(k) \tag{22}$$

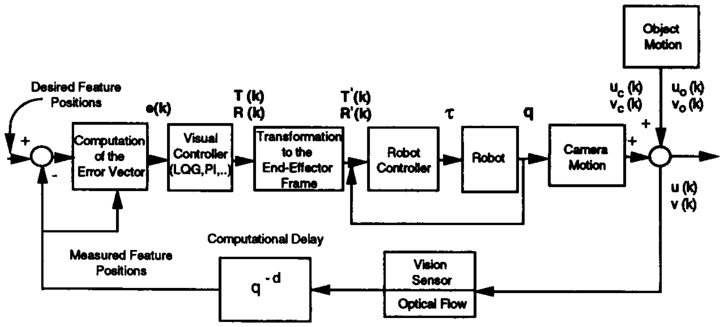


Fig. 6. Structure of the robotic visual tracking scheme.

where $\boldsymbol{w}(k) = (w_1(k), w_2(k), w_3(k))^T$ is a white noise vector ($\boldsymbol{w}(k) \sim N(\boldsymbol{0}, \boldsymbol{W})$) and $\boldsymbol{C} = \boldsymbol{I}_3$. The measurement vector is obtained in a slightly different way than in the case of visual tracking of a single feature point. First, the tracking error of the projections of the two feature points on the image plane is computed by using the SSD algorithm. Then an algebraic system of four equations (two tracking error equations per point) is formulated. The solution of the system is the $X, Y$, and roll components of the tracking error. If the projections of the two feature points on the image plane are not the same, then it is guaranteed that the system of equations has a solution. It is assumed that each one of these features at time $k = 0$ is located at its desired position. The control strategies that keep the target stationary in both cases are discussed in detail in the next section.

## IV. Control Issues in the Visual Tracking Problem

The control techniques that will be presented for the 2-D visual tracking of a moving object can be used easily for the visual tracking of a single moving feature point. The mathematical models for the two cases (feature, object) that were developed in the previous section have the same structure. The only differences are in the order of the systems and in the way that the measurement vector is obtained. The following discussion of various control schemes describes the conditions under which these schemes should be used in visual tracking. Controllers such as PI are attractive due to their simplicity while others (LQG, pole assignment) are more general. Appropriate selection of a control strategy can maximize the improvements that are derived by the combined use of control and vision techniques in visual tracking.

The structure of our scheme is depicted in Fig. 6. This scheme is flexible and can accommodate many different tasks. If the object is static (object motion block is removed from Fig. 6), then the same framework can be used for repositioning the robot-camera system with respect to the static target. Another important application is the repositioning of the robot-camera system with respect to a moving target while the target is being tracked (desired feature positions are different from the initial feature positions on the image plane). Finally, the same scheme in conjunction with estimation algorithms can be used for the recovery of the depth map of a static or moving target. We call our approach *controlled active vision*. This approach states that a controlled and not accidental move of the robot-camera system maximizes the quality and the quantity of the sensory information resulting in improved performance of any active vision algorithm.

### A. PI Controller for Visual Tracking

The control objective is to minimize at each instant of time the error vector $\boldsymbol{e}(k) = (x(k) - 0, y(k) - 0, \theta(k) - 0)^T$ by choosing an appropriate control input vector $\boldsymbol{u}_c(k)$. One simple technique for the elimination of the disturbances is the proportional-integral control technique (PI regulator). This linear control law is given by

$$\boldsymbol{u}_c(k) = \left[\boldsymbol{G}_p \boldsymbol{e}(k) + \boldsymbol{G}_I T \sum_{i=1}^{k} \boldsymbol{e}(i)\right] T^{-1} \qquad (23)$$

where $\boldsymbol{G}_p$ and $\boldsymbol{G}_I$ are constant proportional and integral gain matrices, respectively. There are several techniques for the calculation of the $\boldsymbol{G}_p$ and $\boldsymbol{G}_I$ matrices. One obvious effect of integral control is that it increases the type of the system by one. Thus, the steady-state error is reduced and the disturbances are suppressed. On the other hand, the new system can be less stable than the original or even become unstable if the matrices $\boldsymbol{G}_p$ and $\boldsymbol{G}_I$ are not properly selected.

### B. DARMA Model and Pole Assignment Controller

A more general technique for the design of controller for a visual tracking system is the closed-loop pole assignment technique. This technique is superior to the simple PI regulation technique because it permits flexible tuning of the controller's performance by simply changing the positions of the closed-loop poles. The PI regulator can also be tuned by changing the values of the elements of the gain matrices but the locations of the closed-loop poles are not apparent. Let the discrete-time description of the system be

$$\boldsymbol{A}_D(q^{-1})\boldsymbol{y}(k) = \boldsymbol{B}_D(q^{-1})\boldsymbol{u}_c(k), \qquad k \geq 0 \qquad (24)$$

where the $q^{-1}$ is the backward shift operator. The above model is called the deterministic autoregressive moving average (DARMA) model. Under certain assumptions (described in the sequel), the mathematical model of (21) and (22) can be transformed to the model described by (24). These assumptions are: 1) the disturbances are deterministic and constant, and 2) the noise terms are neglected. Even though these assumptions oversimplify the problem, proper selection of the closed-loop poles can lead to a robust and efficient controller as will be shown later in our experiments. The DARMA model for the 2-D visual tracking problem is

$$A_D(q^{-1}) = (1 - 2q^{-1} + q^{-2})I_3 \tag{25}$$
$$B_D(q^{-1}) = Tq^{-1}(1 - q^{-1})I_3. \tag{26}$$

The derivation of the model is based on a procedure described in [15]. If the desired output sequence is zero, the feedback control law is given by

$$L(q^{-1})u_c(k) = -P(q^{-1})y(k) \tag{27}$$

where $L(q^{-1}) = L'(q^{-1})(1 - q^{-1})I_3$. The matrices $L'(q^{-1})$ and $P(q^{-1})$ are computed by solving the closed-loop pole assignment equation

$$L'(q^{-1})(1 - q^{-1})^2 I_3 + P(q^{-1})Tq^{-1}I_3 = A^*(q^{-1}) \tag{28}$$

where $A^*(q^{-1})$ is a matrix whose diagonal elements are the desired closed-loop polynomials. Equation (28) is always solvable for $L'(q^{-1})$ and $P(q^{-1})$ since the polynomials that constitute the diagonal elements of the matrices $(1 - q^{-1})^2 I_3$ and $Tq^{-1}I_3$ are relatively prime. We observe that the multi-input multi-output (MIMO) model is decoupled, so we can work with three single-input single-output (SISO) models. This simplifies the analysis. Thus, (28) can be decomposed into three polynomial equations:

$$L_i'(q^{-1})(1 - q^{-1})^2 + P_i(q^{-1})Tq^{-1} = A_i^*(q^{-1}), \quad i = 1, 2, 3. \tag{29}$$

Each one of the polynomials $A_i^*(q^{-1})$ must be of order 3 and the polynomials $L_i'(q^{-1})$ and $P_i(q^{-1})$ must be of order 1. For stability, the closed-loop poles should be chosen inside the unit circle. If we choose to locate all the poles at the origin, the controller becomes a one-step ahead controller. In the presence of noisy measurements, a controller of this type exhibits large oscillations. Thus, we prefer to place the poles at locations that guarantee stability and a good transient response.

### C. ARMAX Model and Pole Assignment Controller

The DARMA model does not incorporate the existing stochastic disturbances. A more general model that includes both the stochastic and the deterministic disturbances is the autoregressive moving average model with auxiliary input (ARMAX):

$$\bar{A}_D(q^{-1})D(q^{-1})y(k) = \bar{B}_D(q^{-1})D(q^{-1})u_c(k) + \bar{C}_D(q^{-1})D(q^{-1})w(k), \quad k \geq 0 \tag{30}$$

where $w(k)$ is the white noise vector that corrupts the measurement vector $y(k)$ and $D(q^{-1}) = (1 - q^{-1})I_3$. The use of the ARMAX model in visual servoing has also been proposed by Koivo and Houshangi [21]. If we assume constant deterministic disturbances, we obtain that $\bar{A}_D(q^{-1}) = \bar{C}_D(q^{-1}) = (1 - q^{-1})I_3$ and $\bar{B}_D(q^{-1}) = Tq^{-1}I_3$. If the set point is described by the relationship $S(q^{-1})y^*(k) = 0(y^*(\mathbf{k})$ is the desired output sequence), then a robust control law is given by the equation [17]

$$L(q^{-1})S(q^{-1})D(q^{-1})u_c(k) = P(q^{-1})[y^*(k) - y(k)]. \tag{31}$$

Each one of the diagonal elements of the matrices $L(q^{-1})$ and $P(q^{-1})$ should satisfy the equation

$$L_i(q^{-1})S_i(q^{-1})(1 - q^{-1})^2 + P_i(q^{-1})Tq^{-1} = C_i^s(q^{-1})A_i^*(q^{-1}), \quad i = 1, 2, 3 \tag{32}$$

where

$$\bar{C}_{Di}(q^{-1})D_i(q^{-1}) = C_i^s(q^{-1}) + C_i^\varepsilon(q^{-1}), \quad i = 1, 2, 3. \tag{33}$$

The polynomials $C_i^s(q^{-1})$ should be asymptotically stable and $C_i^\varepsilon(q^{-1})$ are residuals that can be made as small as desired. Further, it can be shown [17] that the tracking error converges to

$$y_i(k) - y_i^*(k) = \frac{L_i(q^{-1})S_i(q^{-1})}{A_i^*(q^{-1})}\left[1 + \frac{C_i^\varepsilon(q^{-1})}{C_i^s(q^{-1})}\right]w_i(k), \quad i = 1, 2, 3. \tag{34}$$

The performance of the controller can be made as close to optimal as desired by choosing $C_i^\varepsilon(q^{-1})$ to be sufficiently small. At the same time, the selection of $C_i^\varepsilon(q^{-1})$ should guarantee that the roots of $C_i^s(q^{-1})$ are located in the unit circle ($C_i^s(q^{-1})$ asymptotically stable).

### D. LQG Controller for Visual Tracking

A more complex control method is the linear quadratic Gaussian (LQG) control technique. This technique permits us to model the deterministic disturbances as time-varying while the previously mentioned control algorithms do not cover this case. The LQG controller can efficiently confront a larger set of object trajectories. Neglecting for the time being the white noise terms of our system, we will consider the more general problem of determining the matrices $G$ and $G_d$ in the linear control law

$$u_c(k) = -Gx(k) - G_d d(k). \tag{35}$$

The reason it is necessary to separate the exogenous variables from the process state vector $x(k)$, rather than deal directly with the metastate vector $x_M^T(k) = (x^T(k), d^T(k))$, is that, in developing the theory for the design of the gain matrix, we assume that the underlying process is controllable. If we try to create a new system with metastate vector $x_M(k) = (x(k), y(k), \theta(k), u_o(k), v_o(k), \omega(k))^T$, we can show, by using the controllability matrix, that the new system is uncontrollable. Thus, it is necessary to work with the form of the system (see (21) and (22)) developed in Section III. Since the matrices $E$ and $B$ are equal, (21) and (22) can be rewritten as

$$x(k+1) = Ax(k) + Bu_n(k) + Hv(k) \tag{36}$$

and

$$y(k) = Cx(k) + w(k) \tag{37}$$

where $\boldsymbol{u}_n(k) = \boldsymbol{u}_c(k) + \boldsymbol{d}(k)$. A performance criterion that can be minimized for the selection of the optimum gain matrix $\boldsymbol{G}$ is

$$J = \sum_{k=0}^{\infty} [\boldsymbol{x}^T(k)\boldsymbol{Q}\boldsymbol{x}(k) + \boldsymbol{u}_n^T(k)\boldsymbol{R}\boldsymbol{u}_n(k)] \tag{38}$$

where $\boldsymbol{Q} = \boldsymbol{Q}^T \geq \boldsymbol{0}$ and $\boldsymbol{R} = \boldsymbol{R}^T > \boldsymbol{0}$. The performance criterion contains a quadratic form in the state vector $\boldsymbol{x}(k)$ plus a second quadratic form in the vector $\boldsymbol{u}_n(k)$. Physically, the first quadratic form represents a penalty for the tracking error and the second corresponds to a modified cost of control. The performance criterion is minimized by selecting an appropriate gain matrix $\boldsymbol{G}$. Taking into consideration the white noise terms of our system, the controller becomes a LQG controller and the control law is

$$\boldsymbol{u}_n(k) = -\boldsymbol{G}\hat{\boldsymbol{x}}(k) \tag{39}$$

where $\hat{\boldsymbol{x}}(k)$ is the estimated value of the state vector $\boldsymbol{x}(k)$. Thus, $\boldsymbol{u}_c(k)$ is given by

$$\boldsymbol{u}_c(k) = -\boldsymbol{G}\hat{\boldsymbol{x}}(k) - \hat{\boldsymbol{d}}(k) \tag{40}$$

where $\hat{\boldsymbol{d}}(k)$ is the estimated value of the disturbance vector $\boldsymbol{d}(k)$. The performance criterion now is the expected value of $J$

$$\bar{J} = E\left\{\sum_{k=0}^{\infty} [\boldsymbol{x}^T(k)\boldsymbol{Q}\boldsymbol{x}(k) + \boldsymbol{u}_n^T(k)\boldsymbol{R}\boldsymbol{u}_n(k)]\right\}. \tag{41}$$

The optimal control gain matrix $\boldsymbol{G}$ is $\boldsymbol{G} = (\boldsymbol{B}^T\boldsymbol{P}\boldsymbol{B} + \boldsymbol{R})^{-1}\boldsymbol{B}^T\boldsymbol{P}\boldsymbol{A}$ with $\boldsymbol{P}$ being the unique symmetric positive definite solution of the matrix algebraic Riccati equation

$$\boldsymbol{A}^T[\boldsymbol{P} - \boldsymbol{P}\boldsymbol{B}(\boldsymbol{B}^T\boldsymbol{P}\boldsymbol{B} + \boldsymbol{R})^{-1}\boldsymbol{B}^T\boldsymbol{P}]\boldsymbol{A} + \boldsymbol{Q} = \boldsymbol{P}. \tag{42}$$

The design parameters are the elements of the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$. By selecting these matrices, one can obtain the desired gain matrix $\boldsymbol{G}$. There is no standard procedure for the selection of the elements of these matrices. One technique is the optimization approach [23]. Assume that $x_{i\,\max}$ is the maximal $x(k)$ state deviation, $y_{i\,\max}$ is the maximal $y(k)$ state deviation, $\theta_{i\,\max}$ is the maximal $\theta(k)$ state deviation, $u_{ni\,\max}$ is the maximal $u_n(k)$ control amplitude, $v_{ni\,\max}$ is the maximal $v_n(k)$ control amplitude, and $R_{ni\,\max}$ is the maximal $R_n(k)$ control amplitude. The terms $v_n(k), u_n(k)$, and $R_n(k)$ are the components of the vector $\boldsymbol{u}_n(k)$ that was defined above. Then, the control designer chooses $\boldsymbol{Q} = \text{diag}\{x_{i\,\max}^{-2}, y_{i\,\max}^{-2}, \theta_{i\,\max}^{-2}\}$ and $\boldsymbol{R} = \text{diag}\{u_{ni\,\max}^{-2}, v_{ni\,\max}^{-2}, R_{ni\,\max}^{-2}\}$. In this way, the constraints that the robotic device imposes on the maximal control amplitudes are included in the control law. It is not possible to have infinite maximal control amplitudes. The same is true for the states. Since the states are measured directly by the SSD algorithm, only a certain range of values can be measured. The next step in our algorithm is the computation of the vectors $\hat{\boldsymbol{x}}(k)$ and $\hat{\boldsymbol{d}}(k)$. We design an observer for the estimation of the metastate vector $\boldsymbol{x}_M(k)$. The state-space model of (21) and (22) can be rewritten as (white noise terms are assumed to accompany only the terms $u_o(k), v_o(k)$, and $\omega(k)$)

$$\boldsymbol{x}_M(k+1) = \boldsymbol{A}_M\boldsymbol{x}_M(k) + \boldsymbol{B}_M\boldsymbol{u}_c(k) + \boldsymbol{H}_M\boldsymbol{v}_M(k) \tag{43}$$
$$\boldsymbol{y}(k) = \boldsymbol{C}_M\boldsymbol{x}_M(k) + \boldsymbol{w}(k) \tag{44}$$

where

$$\boldsymbol{A}_M = \begin{bmatrix} 1 & 0 & 0 & T & 0 & 0 \\ 0 & 1 & 0 & 0 & T & 0 \\ 0 & 0 & 1 & 0 & 0 & T \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{B}_M = \begin{bmatrix} T & 0 & 0 \\ 0 & T & 0 \\ 0 & 0 & T \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

and

$$\boldsymbol{C}_M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{H}_M^T = \begin{bmatrix} 0 & 0 & 0 & T & 0 & 0 \\ 0 & 0 & 0 & 0 & T & 0 \\ 0 & 0 & 0 & 0 & 0 & T \end{bmatrix}.$$

A more accurate form for the matrix $\boldsymbol{H}(k)$ is

$$\boldsymbol{H}_M^T = \begin{bmatrix} T^2/2 & 0 & 0 & T & 0 & 0 \\ 0 & T^2/2 & 0 & 0 & T & 0 \\ 0 & 0 & T^2/2 & 0 & 0 & T \end{bmatrix}.$$

Due to the fact that $T$ is pretty small, the term $T^2/2$ can be set to zero. As it was mentioned before, the measurement vector consists of the measured translational components of the tracking error $x(k), y(k)$, and of the roll component of it, $\theta(k)$. A steady-state Kalman filter [13] can be designed for the estimation of the metastate vector $\boldsymbol{x}_M(k)$. The assumptions are that $\boldsymbol{Q}_e = E\{\boldsymbol{v}_M(k)\boldsymbol{v}_M^T(k)\}, \boldsymbol{R}_e = E\{\boldsymbol{w}(k)\boldsymbol{w}^T(k)\}, \boldsymbol{x}_{Mo} = E\{\boldsymbol{x}_M(0)\}$, and $E\{\boldsymbol{v}_M(k)\boldsymbol{w}^T(j)\} = \boldsymbol{0}$ for all $k$ and $j$. The state update equation is

$$\hat{\boldsymbol{x}}_M(k+1) = \boldsymbol{A}_M\hat{\boldsymbol{x}}_M(k) + \boldsymbol{B}_M\boldsymbol{u}_c(k) + \boldsymbol{K}_e(\boldsymbol{y}(k) - \boldsymbol{C}_M\hat{\boldsymbol{x}}(k)) \tag{45}$$

where

$$\boldsymbol{K}_e = \boldsymbol{A}_M\boldsymbol{P}_e\boldsymbol{C}_M^T[\boldsymbol{C}_M\boldsymbol{P}_e\boldsymbol{C}_M^T + \boldsymbol{R}_e]^{-1} \tag{46}$$

and $\boldsymbol{P}_e$ satisfies the matrix algebraic Riccati equation

$$\boldsymbol{A}_M[\boldsymbol{I} - \boldsymbol{P}_e\boldsymbol{C}_M^T(\boldsymbol{C}_M\boldsymbol{P}_e\boldsymbol{C}_M^T + \boldsymbol{R}_e)^{-1}\boldsymbol{C}_M]\boldsymbol{P}_e\boldsymbol{A}_M^T + \boldsymbol{H}_M\boldsymbol{Q}_e\boldsymbol{H}_M^T = \boldsymbol{P}_e. \tag{47}$$

The time-invariant steady-state Kalman filter can be implemented easily and does not require a large number of calculations. In addition to the steady-state Kalman filter, we use the time-varying discrete Kalman filter, which constantly updates the Kalman gain matrix $\boldsymbol{K}_e$ [30]. This improves the performance of our observer, but it is computationally more expensive than the time-invariant Kalman filter. The equations for this observer are

$$\hat{\boldsymbol{x}}_M(k+1) = \boldsymbol{A}_M\hat{\boldsymbol{x}}_M(k) + \boldsymbol{B}_M\boldsymbol{u}_c(k) + \boldsymbol{K}_e(k)(\boldsymbol{y}(k) - \boldsymbol{C}_M\hat{\boldsymbol{x}}(k)) \tag{48}$$

$$\boldsymbol{K}_e(k) = \boldsymbol{A}_M\boldsymbol{P}_e(k)\boldsymbol{C}_M^T[\boldsymbol{C}_M\boldsymbol{P}_e(k)\boldsymbol{C}_M^T + \boldsymbol{R}_e]^{-1} \tag{49}$$

$$\boldsymbol{P}_e(k+1) = [\boldsymbol{A}_M - \boldsymbol{K}_e(k)\boldsymbol{C}_M]\boldsymbol{P}_e(k)\boldsymbol{A}_M^T + \boldsymbol{H}_M\boldsymbol{Q}_e\boldsymbol{H}_M^T \tag{50}$$

where $\hat{\boldsymbol{x}}_{Mo} = E\{\boldsymbol{x}_M(0)\}$ and $\boldsymbol{P}_e(0) = E\{(\boldsymbol{x}_M(0) - \hat{\boldsymbol{x}}_{Mo})(\boldsymbol{x}_M(0) - \hat{\boldsymbol{x}}_{Mo})^T\}$. The performance of the observer depends on the selection of the $\boldsymbol{Q}_e$ and $\boldsymbol{R}_e$ matrices. We should mention that the white noise model is only an approximation to the actual noise model of the camera. Thus, the selection of the $\boldsymbol{Q}_e$ and $\boldsymbol{R}_e$ matrices is done empirically and a search for the best set of noise variances is conducted. The initialization of the vectors $\hat{\boldsymbol{x}}(k)$ and $\hat{\boldsymbol{d}}(k)$ is

$$\hat{\boldsymbol{x}}(0) = \boldsymbol{y}(0) \tag{51}$$

$$\hat{\boldsymbol{d}}(0) = T^{-1}(\boldsymbol{y}(0) - \mathbf{o}). \tag{52}$$

### *E. Computation of the Rotational and the Translational Velocity Vectors*

The next step of our algorithm is the calculation of the pair $(T_x(k), T_y(k))$ or of the pair $(R_x(k), R_y(k))$ or of the triple $(T_x(k), T_y(k), R_z(k))$. As was previously mentioned, the first two cases correspond to the visual tracking of a point while the last case represents the visual tracking of an object. In the first case, $T_x(k)$ and $T_y(k)$ are computed by (17) and (18), which require knowledge of the depth $Z_s$. In the second case, we have to solve the system of equations (19) and (20) and calculate $R_x(k)$ and $R_y(k)$. The determinant of the system is nonzero in an area around the origin, and thus the system has a unique solution. In the third case, the calculation of the $T_x(k)$ and $T_y(k)$ is done in the same way as in the first case, and $-R_z(k)$ is given directly as the computed control signal.

The knowledge of the depth $Z_s$ can be acquired in two ways. The first way is direct computation by a range sensor or by stereo techniques [25]. The use of stereo for the recovery of the depth is a difficult procedure because it requires the solution of the correspondence problem. A more effective strategy that requires the use of only one vision sensor is to use adaptive control techniques. The control law is based on the estimated on-line values of the model's parameters that depend on the depth as described in [27]–[29].

### *F. Robot Control Schemes*

After the computation of the translational $\boldsymbol{T}(k)$ and rotational $\boldsymbol{R}(k)$ velocity vectors with respect to the camera frame $R_s$, we transform them to the end-effector frame $R_e$ with the use of the transformation ${}^eT_s$. The transformed signals are fed to the robot controller. We experimented with two Cartesian robot control schemes: a Cartesian **computed torque** scheme and a Cartesian ***PD*** scheme with gravity compensation. The selection of the appropriate robot control method is essential to the success of our algorithms because small oscillations can create blurring in the acquired images. Blurring reduces the accuracy of the visual measurements, and, as a result, the system cannot accurately track the moving object.

The mathematical model of the robot's dynamics is

$$\boldsymbol{D}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{53}$$

where $\boldsymbol{q}$ is the vector of the joint variables of the robotic arm, $\boldsymbol{D}$ is the inertial acceleration related matrix, $\boldsymbol{c}$ is the nonlinear Coriolis and centrifugal torque vector, $\boldsymbol{g}$ is the gravitational torque vector, and $\boldsymbol{\tau}$ is the generalized torque vector. The model is nonlinear and coupled. The control law for the Cartesian computed torque scheme requires the inversion of the manipulator's Jacobian $\boldsymbol{J}(\boldsymbol{q})$ and is given by

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\boldsymbol{q})[\ddot{\boldsymbol{x}}_u - \dot{\boldsymbol{J}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}] \tag{54}$$

$$\ddot{\boldsymbol{x}}_u = \boldsymbol{K}_{\mathbf{p}}\Delta\boldsymbol{x} + \boldsymbol{K}_{\mathbf{v}}\Delta\dot{\boldsymbol{x}} + \ddot{\boldsymbol{x}}_{\mathbf{d}} \tag{55}$$

$$\Delta\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}_{\mathbf{d}} - \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{56}$$

where $\Delta\boldsymbol{x}^T = (\Delta\boldsymbol{x}_{\mathbf{p}}^T, \Delta\boldsymbol{x}_{\mathbf{o}}^T)$ is the position and orientation error vector, while $\boldsymbol{K}_{\mathbf{p}}$ and $\boldsymbol{K}_{\mathbf{v}}$ are diagonal gain matrices. The subscript $d$ denotes the desired quantities. The actuator's torque vector $\boldsymbol{\tau}$ is derived from (53) after the computation of $\ddot{q}$. In our experiments, $\ddot{\boldsymbol{x}}_{\mathbf{d}}$ is selected to be zero. This scheme is accurate but the inversion of the Jacobian can drive the manipulator unstable. For out testbed, the CMU DD Arm II, the Jacobian becomes singular when any two of the six joints are aligned.

The second control scheme assumes that all velocity dependent terms in (53) can be neglected. This implies that $\dot{J} = c(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \mathbf{o}$. Thus, the actuators' torque vector $\boldsymbol{\tau}$ is given by

$$\boldsymbol{\tau} = \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{F} + \boldsymbol{g}(\boldsymbol{q}) \tag{57}$$

$$\boldsymbol{F} = \boldsymbol{K}_{\mathbf{p}}\Delta\boldsymbol{x} + \boldsymbol{K}_{\mathbf{v}}\Delta\dot{\boldsymbol{x}} + \ddot{\boldsymbol{x}}_{\mathbf{d}} \tag{58}$$

where $\boldsymbol{F}$ is the generalized force vector. This approach is more robust than the full dynamics approach because it does not require the inversion of the Jacobian. On the other hand, it is less accurate. Due to its robustness, the second approach is used extensively in our experiments.

The next section describes the hardware configuration of our experimental testbed (CMU DD Arm II robotic system).

## V. Hardware Configuration

The vision module of the CMU DD Arm II system is a part of a bigger hardware environment, which is depicted in Fig. 7. This hardware environment runs under the CHIMERA II [31] real-time programming environment and consists of the following devices:

- A Sun 3/260 host system on a VME bus.
- Multiple Ironics M68020 boards.
- A Mercury 32000 Floating Point Unit.
- An IDAS/150 image processing system.

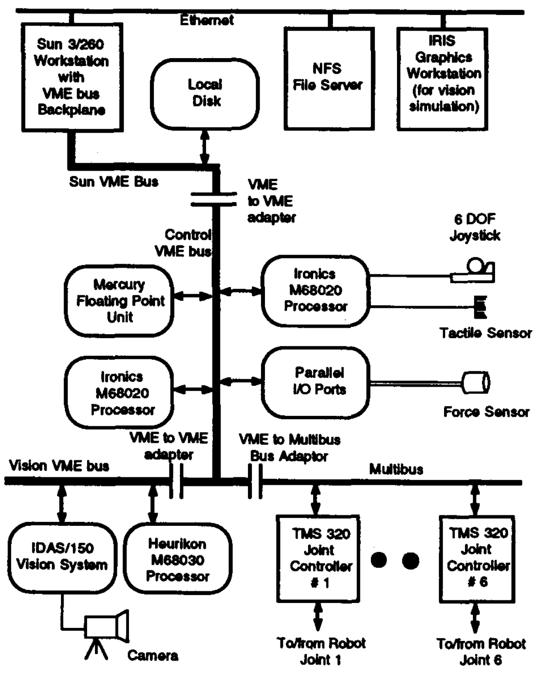
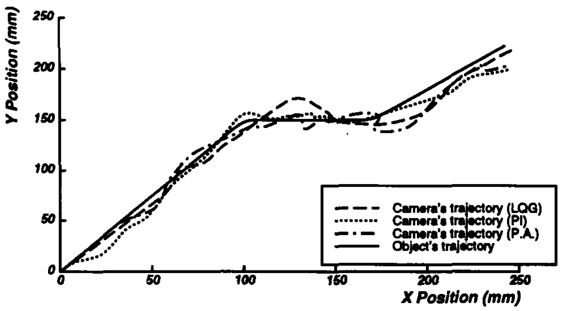


Fig. 7. DD Arm II hardware configuration (figure taken from [31]).



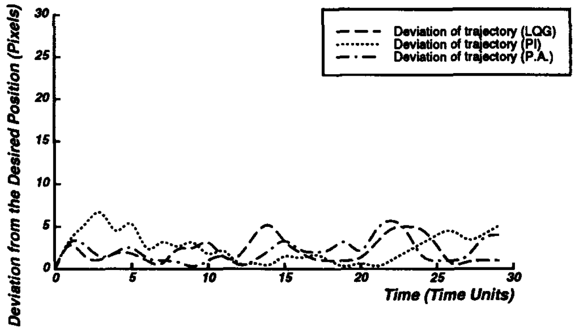


Fig. 8. Example with measurements from one big window (simulation).

- A Panasonic industrial CCD color camera, Model GP-CD1H.
- Six Texas Instrument TMS320 DSP processors, each controlling one joint of the CMU DD Arm II system.

The IDAS/150 image processing system carries out all the computational load of the image processing calculations, while the Mercury Floating Point Unit does all the control calculations. The IDAS/150 contains a Heurikon 68030 board as the controller of the vision module and two floating point boards, each one with computational power of 20 Mflops. The system can be expanded to contain as many as eight of these boards. The software is organized around three processes:

- *Vision process:* This process does all the image processing calculations and has a period of 100 ms. The system is designed in way that the acquisition of images is synchronized with the robot motion. This fact guarantees that the delay factor in (13) and (14) is zero or one.
- *Interpolation process:* This process reads the data from the vision process, interpolates the data, and sends the reference signals to the robot Cartesian controller. The interpolation process continuously feeds data to the robot controller in order to provide a smooth robot trajectory. Since the vision process is slow, the interpolation process is needed in order to guarantee a continuous motion of the arm. In addition, it checks for large values of the reference signals that can saturate the motors. It has a period of 7 ms.
- *Robot controller process:* This process drives the robot and has a period of 3.33 ms.

In the next section, we describe the simulation and experimental results.

## VI. Simulation and Experimental Results

### A. Simulation

A number of examples, simulated on a Silicon Graphics Inc. IRIS graphics workstation, have been used for determining the efficiency of the developed algorithms. In order to accomplish our objective, we created synthetic images that emulate real-world images. The objects used in the tracking examples are geometrical 3-D solid objects such as cubes, cylinders, and prisms. These objects have different colors and move in front of a highly textured background. The synthetic images are 767 x 767 and are quantized to 256 gray levels.

Each pixel's intensity is corrupted with white noise. This means that at each intensity value, we add a random number that is within a certain range. This range is determined by the percentage of the white noise. Our algorithms perform satisfactorily with white noise in the range between 10% and 30%. With higher noise, the results degenerate. Even though the assumptifon of white noise does not hold entirely for real images, 30% white noise is excessive. The examples shown in Figs. 8–10 are implemented with 10% white noise. The object's centroid trajectories are shown by the solid lines. The focal length of the camera is 7.5 mm (same as the one of the actual camera). The simulated camera pixel's dimensions are $s_x = s_y = 0.03$ mm/pixel. Distortion and effects of camera sampling are neglected. The depth, when needed, is assumed to be known ($Z_s = 1$ m). As mentioned earlier, when the tracking

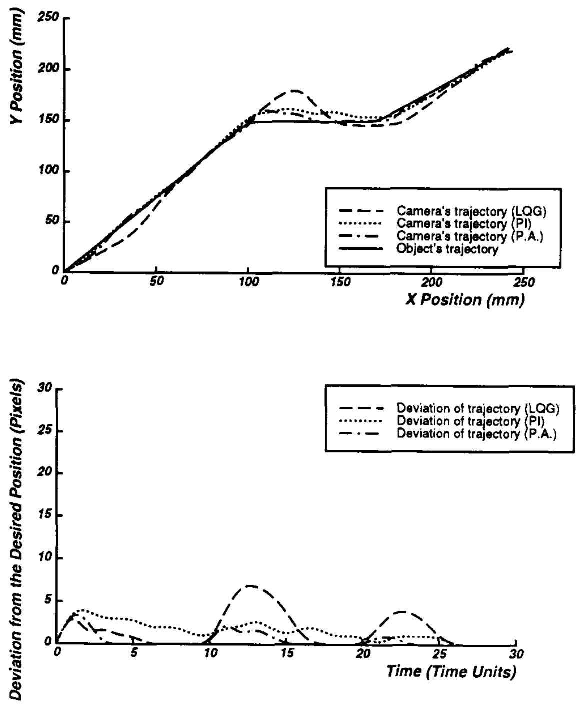


Fig. 9. Example with measurements from multiple windows (simulation).

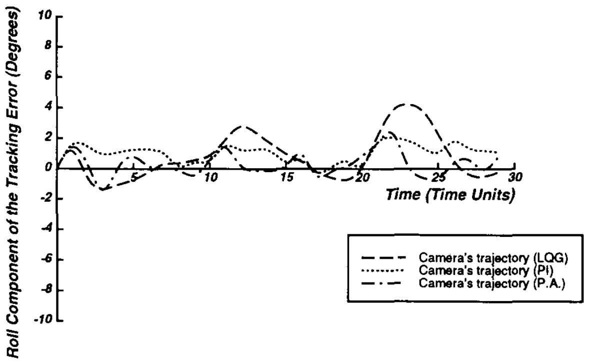


Fig. 10. Roll component of the tracking error for the previous example. The measurements are derived from multiple windows (simulation).

is done with rotation $R_x(k)$ and $R_y(k)$, the knowledge of depth is not required. In addition, when depth $Z_s$ is required, it can be obtained by a range sensor or by stereo computations or by estimation techniques. Without loss of generality, we assume $T = 1$.

The tuning of the PI controller is done with the help of the MATLAB software package. The matrices $\boldsymbol{G}_p$ and $\boldsymbol{G}_I$ are chosen in a way that keeps the closed-loop system stable [15]. The simulations are done with the matrices $\boldsymbol{G}_p = -\boldsymbol{I}_3$ and $\boldsymbol{G}_I = -0.1\boldsymbol{I}_3$. We placed the closed-loop poles of the pole assignment controller (PA) not only inside the unit circle but also within a small distance from the origin of the $z$ plane. The reason is that a one-step-ahead controller performs very well without the presence of noise, but in noisy conditions large oscillations appear around the desired trajectory. We

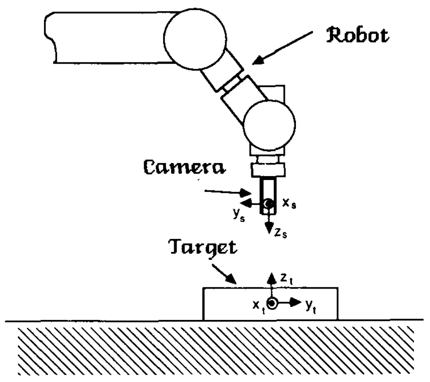


Fig. 11. Experimental setup.

avoid placing the closed-loop poles very close to the unit circle because the rise time increases drastically. As a result, the controller fails to track fast changes of the trajectory. Finally, we placed all the closed-loop poles at $z_i = 0.2$. The performance of the pole assignment controller is shown by the dotdashed trajectories in Figs. 8–10.

The gains of the LQG regulator are computed with the help of MATLAB also. By selecting $\boldsymbol{Q} = \boldsymbol{I}_3$ and $\boldsymbol{R} = 0.01\boldsymbol{I}_3$, we obtain $\boldsymbol{G} = 0.9902\boldsymbol{I}_3$. The best $\boldsymbol{G}$ has eigenvalues close to 1. This is to be expected, because $\boldsymbol{A} = \boldsymbol{B} = \boldsymbol{I}_3$ and the eigenvalues of the matrix $\boldsymbol{R}$ are pretty small. Physically, this means that the control law is trying to minimize the tracking error almost in one step.

By using MATLAB, we obtain different $\boldsymbol{K}_e$ matrices for different $\boldsymbol{Q}_e$ and $\boldsymbol{R}_e$ for a steady-state Kalman filter. For a Kalman filter to yield optimal performance, it is necessary to provide the correct $\boldsymbol{A}_M, \boldsymbol{C}_M, \boldsymbol{H}_M, \boldsymbol{Q}_e$, and $\boldsymbol{R}_e$. The innovation property [13] is used as a criterion to test for optimality. The experimentally measured steady-state correlation function $E\{\boldsymbol{w}(k)\boldsymbol{w}^T(k-j)\}$ is processed to identify unknown $\boldsymbol{Q}_e$ and $\boldsymbol{R}_e$ for known $\boldsymbol{A}_M, \boldsymbol{C}_M$, and $\boldsymbol{H}_M$. For $\boldsymbol{Q}_e = 0.1\boldsymbol{I}_3$ and $\boldsymbol{R}_e = \boldsymbol{I}_3$, we obtain $\boldsymbol{K}_e(1,1) = \boldsymbol{K}_e(2,2) = \boldsymbol{K}_e(3,3) = 0.5531$ and $\boldsymbol{K}_e(4,1) = \boldsymbol{K}_e(5,2) = \boldsymbol{K}_e(6,3) = 0.2114$. All the other elements of the matrix $\boldsymbol{K}_e$ are zero. We also implement a time-varying discrete Kalman filter with reinitialization each time the error is greater than a constant $\varepsilon$. The improvement is not significant, and the time-varying Kalman filter increases the computational load. As an alternative solution, an adaptive Kalman filter is used [13], but the results are similar to the results of the time-varying Kalman filter. Thus, all the examples shown in Figs. 8–10 are done by using a steady-state discrete Kalman filter. In Figs. 8–10, the LQG controlled camera's trajectories correspond to the dashed trajectories.

As mentioned earlier, the SSD optical flow technique is used for the computation of the measurement vector. The simulations are done as follows: In Fig. 8, one big window is used and the object's speed has no roll component. The object's speed is 11.8 mm/Time_Unit during the first section of the trajectory, 7 mm/Time_Unit during the second section, and 11.31 mm/Time_Unit during the last section. In Figs. 9 and 10, multiple small windows are used. The object has the same translational trajectory as before. In addition, it rotates with a

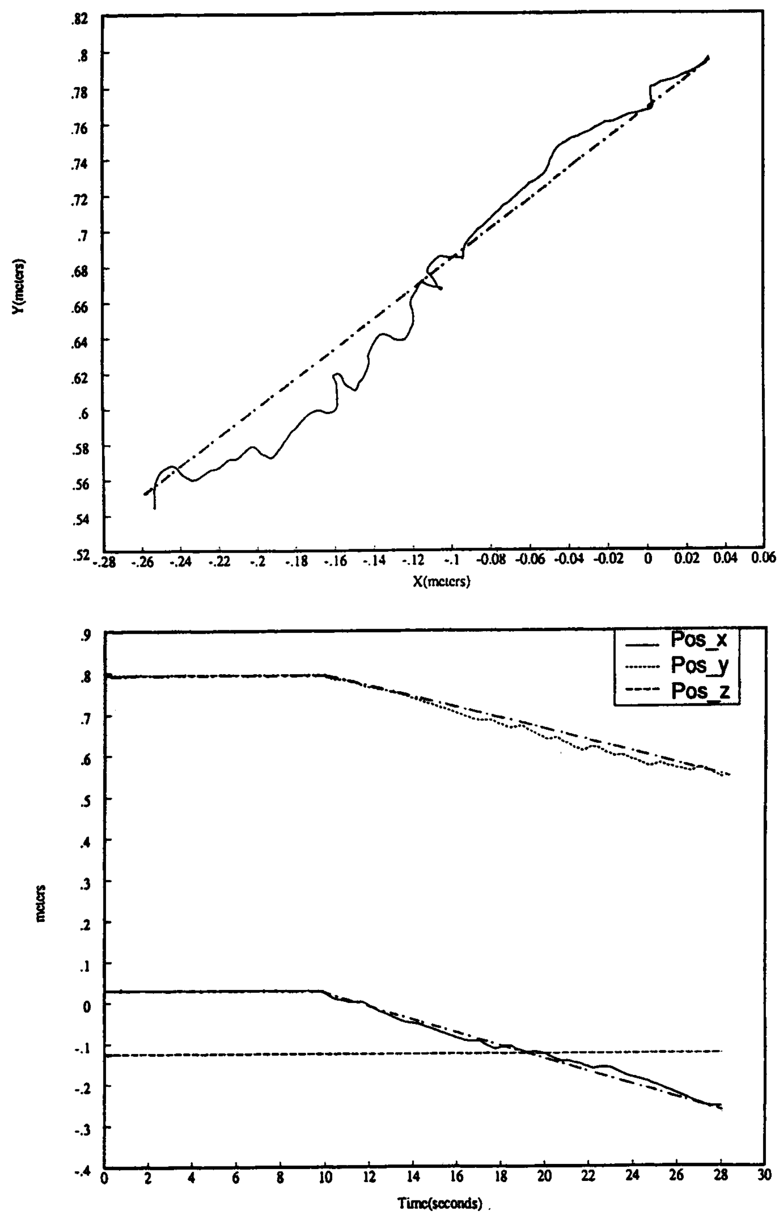


Fig. 12. PI controller in conjunction with a Cartesian PD robot controller with gravity compensation (experimental).

speed of 2°/Time_Unit during the first section of the trajectory, 3°/Time_Unit during the second section, and 4°/Time_Unit during the last section. In the first case, the size of window $N$ is 70. The speed of the algorithm is improved by employing the undersampling technique that reduces the computations without significant loss of information. In the examples, we use 14 × 14 pixels, and the largest displacement that can be detected is 18 pixels. The largest detected displacement can be increased by quantization of the space of possible displacements. As a result, a larger tracking error occurs, which can be compensated by proper tuning of the Kalman filter. The larger the accelerations, the larger the deviation from the desired position. As depicted in the figures of the examples, the system compensates for these changes successfully. Each trajectory consists of 30 points. The objects move at different speeds and abrupt changes happen to their trajectories.

Fig. 8 shows that, when abrupt changes occur, there is an increase in the error. The largest error is around 10 pixels. In Fig. 8, the PI controller seems to have a better performance than the LQG regulator and the pole assignment controller in the abrupt changes of the trajectories. This is due to the fact that, at these instants of time, the estimated values of the disturbances present a large error. LQG regulator outperforms PI at the beginning of each trajectory. PI regulators perform a large

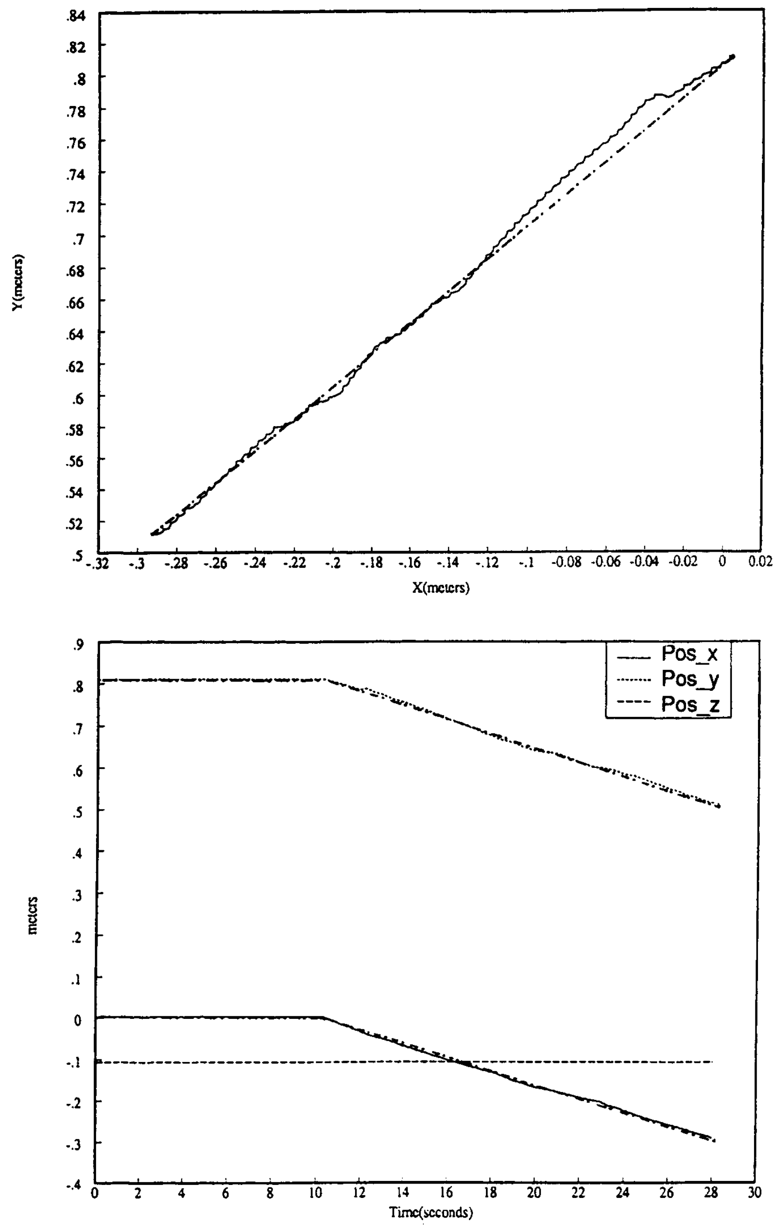


Fig. 13. PI controller in conjunction with a Cartesian computed torque robot controller (experimental).

number of fluctuations around the desired trajectory because the measurements are not very precise. The pole assignment controller has the worst performance. The reason is that we try to compensate fast for changes that are not measured with accuracy. We also tried a one-step-ahead controller, and the results present errors with double the magnitude of the errors of other controllers.

These measurements can be improved by using a larger number of windows. We increased the windows to $10 \times 10$ and spread them out in the image plane. For each window, we computed a confidence measure that reflects the accuracy of the measurements that the window provides. We tried three different confidence measures proposed in Section II. The results of our implementation can be seen in Figs. 9 and 10. We observe some interesting points from these results. The pole assignment controller has the best performance and the LQG has the worst due to the fact that the measurements are quite accurate. The LQG controllers have a large error in the abrupt changes of the trajectory because they need a finite amount of time to adapt to the new values of the disturbances. In conclusion, the selection of the control scheme is dependent on the vision algorithms that we use. Vision

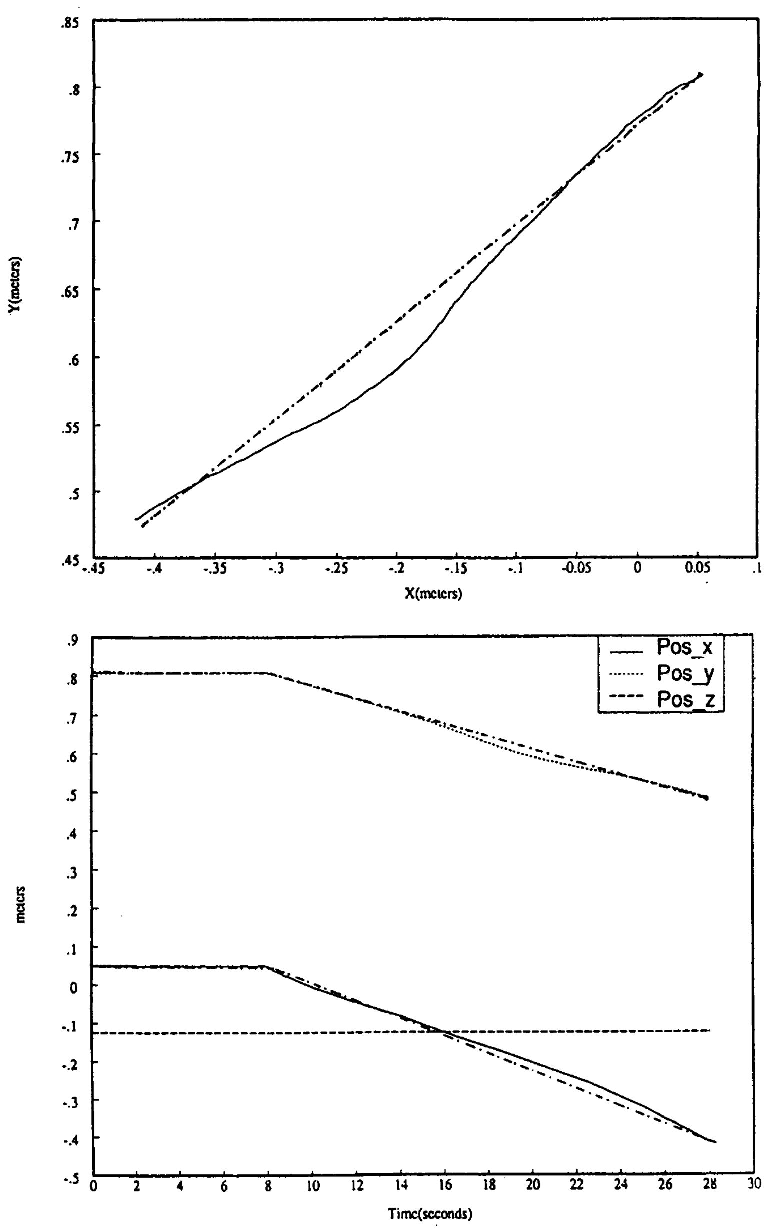


Fig. 14. Pole assignment (poles at 0.2) controller in conjunction with a Cartesian PD robot controller with gravity compensation (experimental). The camera is focused.

algorithms that produce extremely noisy measurements need controllers that can deal satisfactorily with noise. On the other hand, more accurate vision algorithms are computationally expensive. Large computational delays can make a tracking system to fail. The next step of our work involved testing the algorithms on the CMU DD Arm II system.

### B. Experiments

The experiments are performed on the CMU DD Arm II system. The camera is mounted on the end-effector and has a focal length of 7.5 mm while the objects are moving on a plane (average depth $Z_s = 680$ mm). The experimental setup is shown in Fig. 11. In the first set of experiments, the center of mass of each one of these objects moves across the line $Y = 0.74X + 0.77$ ($Y$ and $X$ in meters). The objects move with an average speed between 2.5 and 3 cm/s. In order to determine the real trajectories of the targets, the targets are moved by a PUMA manipulator or by a small mobile robot. The maximum speed that can be tracked is 7 cm/s. For the tracking of faster moving objects, additional hardware should be used in order to reduce the computational time needed for the detection of the relative

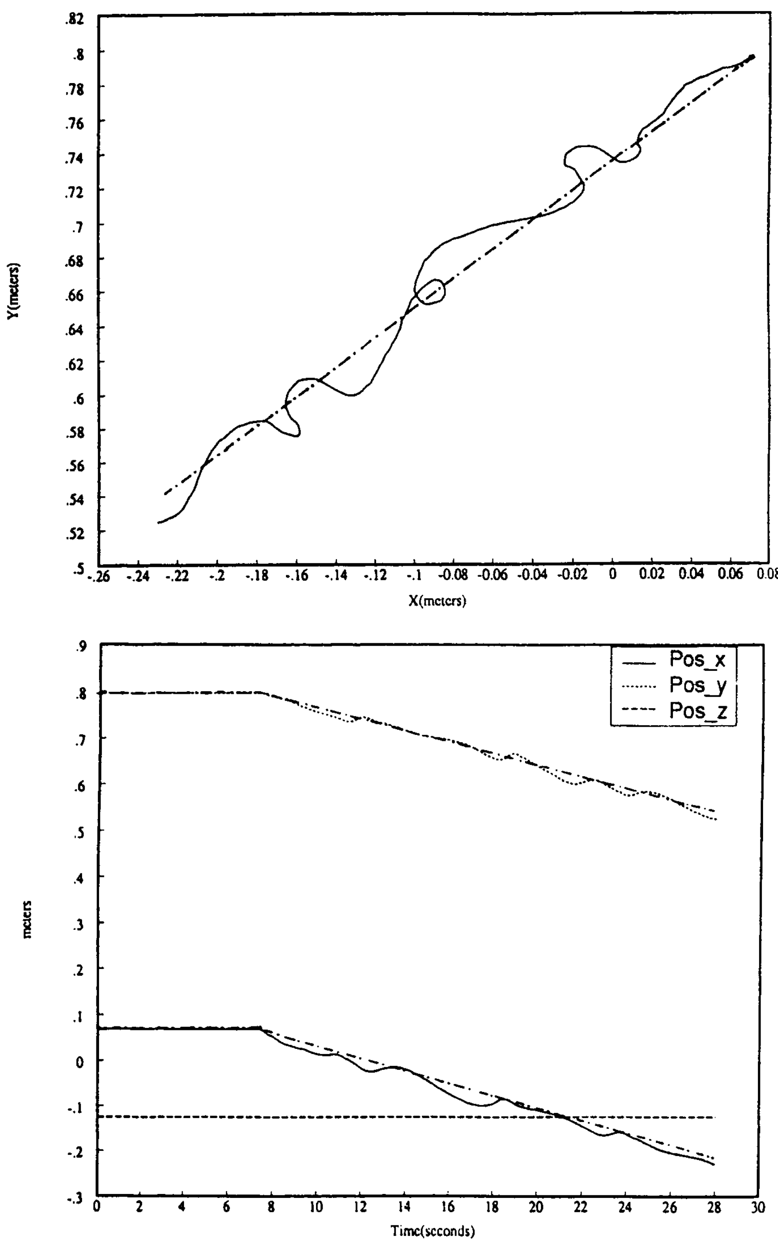


Fig. 15. Pole assignment (poles at 0.2) controller in conjunction with a Cartesian PD robot controller with gravity compensation (experimental). The camera is defocused and the number of feature points is decreased.

motion. The camera's pixel dimensions are $s_x = 0.01278$ mm/pixel and $s_y = 0.00986$ mm/pixel. The real images are $492 \times 510$ and are quantized to 256 gray levels. The objects used in the tracking examples are books, pencils, and generally items with distinct features. The features to be tracked are chosen at initialization by the user. Then the system evaluates on-line the quality of the measurements, based on the confidence measures described in Section II. Currently, four features are used, and the size of the attached windows is $10 \times 10$.

The gains for the controllers are the same as the ones used in the simulation. The experimental results are plotted in Figs. 12–20 where the dotdashed trajectories correspond to the trajectories of the center of mass of the moving objects. The variables Pos_x, Pos_y, and Pos_z represent the $X$, $Y$, and $Z$ components of the position vector of the end-effector in the world frame. The experimental results lead to some interesting observations. The Cartesian computed torque scheme provides better performance than the simple PD with gravity compensation scheme. This is apparent from Figs. 12

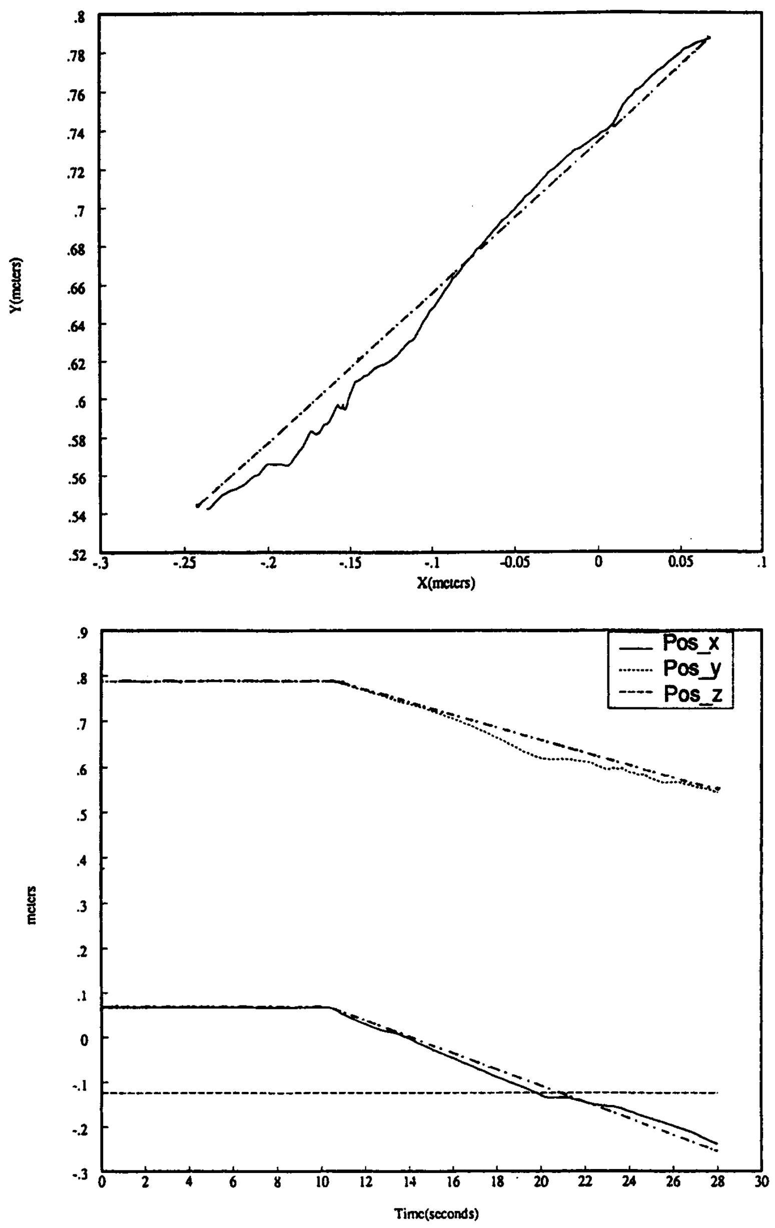


Fig. 16. Pole assignment (poles at 0.8) controller in conjunction with a Cartesian PD robot controller with gravity compensation (experimental). The camera is defocused and the number of feature points is decreased.

and 13. Both schemes require high values for the $K_p$ and $K_v$ matrices. The simple PD produces oscillations around the desired trajectory. The computed torque scheme is more accurate because it takes into consideration the full dynamics of the robot. The problem with the computed torque scheme is that it requires the inversion of the Jacobian. Thus, the DD Arm II can easily become unstable (whenever two of the joints are aligned).

Another interesting observation is that the selection of the controller depends on the image noise and the number and quality of the used feature points. A decrease in the number of the used features and a slightly defocused camera cause the controllers with the poles near zero to fail as in Fig. 15. In this example, a controller with the poles at 0.8 works better (Fig. 16). On the other hand, when we increase the number of the used features and focus the camera correctly, a controller with the poles near zero has a better performance (Fig. 14). These experiments are done with the same object.

The previous observation is not apparent in the simulation results. The reason is that the simulation results are produced in an artificial environment where it is difficult to reproduce the noise that is present in actual images. The

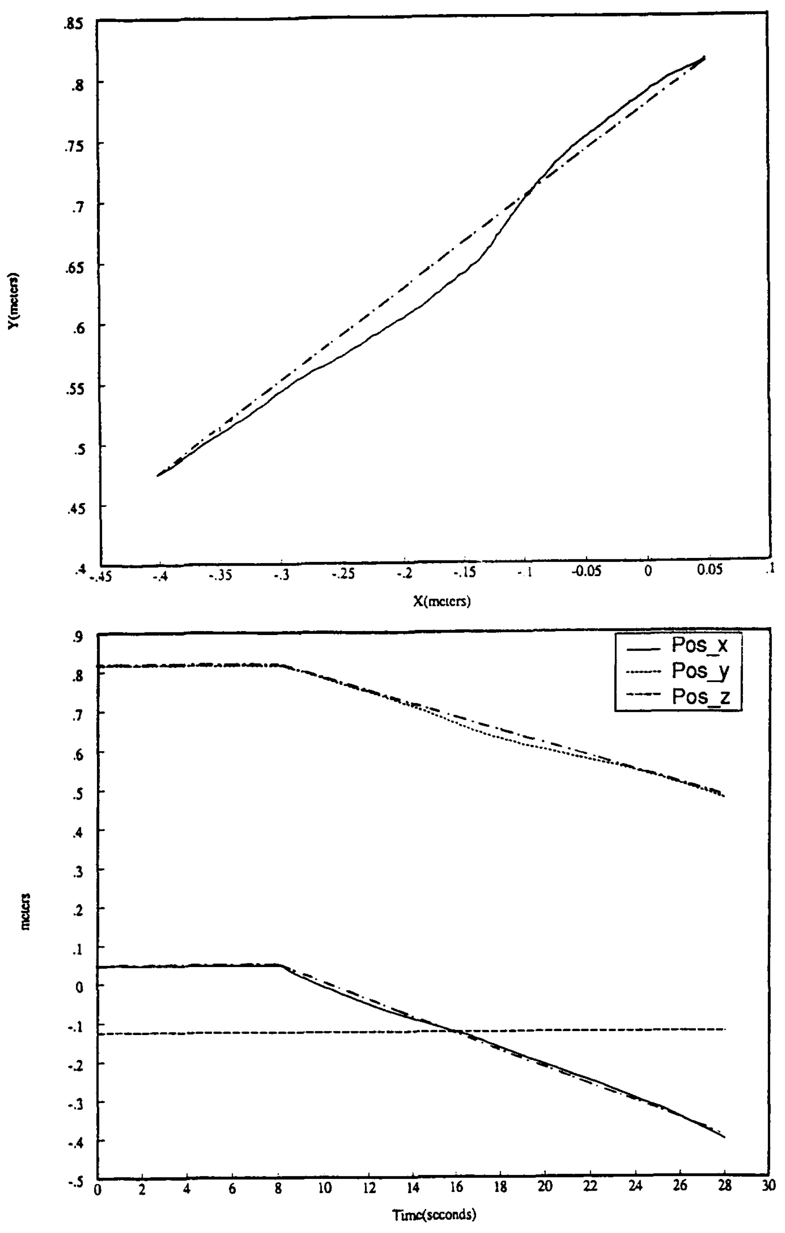


Fig. 17. LQG controller in conjunction with a Cartesian PD robot controller with gravity compensation (experimental).

LQG (Fig. 17) and the time-varying LQG (Fig. 18) have similar performances. The experimental results show that the use of the more complex time-varying LQG is not justified. The same conclusion is derived from the simulation results. The observed oscillations are due to the fact that the robotic controller (PD with gravity compensation) does not take into consideration the robot dynamics. The performance of the LQG controller in a nonlinear trajectory is shown in Figs. 19 and 20. In this example, along with the translational motion, the object performs a rotational motion around an axis that passes through the center of mass of the object. The target rotates with a constant speed of 9.41°/s while its translational speed has a constant magnitude of 4.1 cm/s. Due to noisy measurements, the LQG controller seems to have the best performance. This becomes more obvious when one reduces the number of the windows used. The PI is simple, but it does not compensate for the noise that is apparent in the actual measurements. The pole assignment has comparable performance with the LQG, but the selection of the closed-loop poles should be done carefully. Selecting the closed-loop poles without taking into consideration the special characteristics (noise, camera model) of the vision algorithm

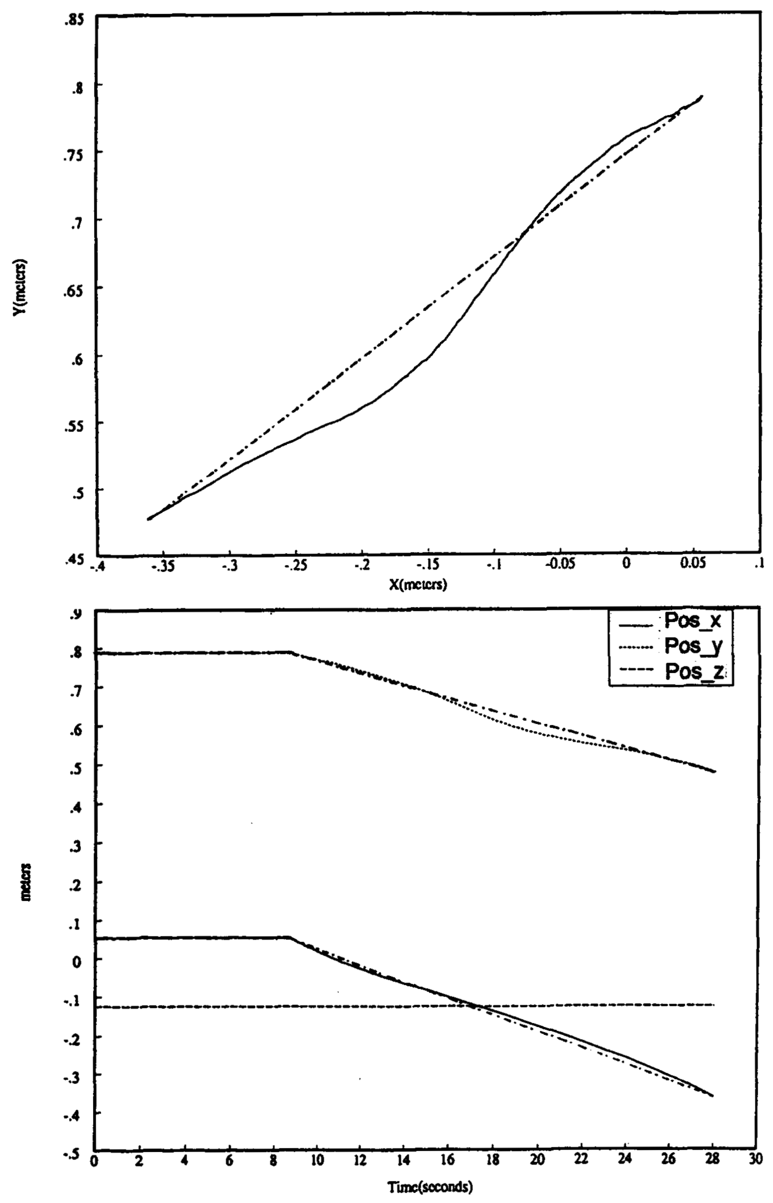


Fig. 18. Time-varying LQG controller in conjunction with a Cartesian PD robot controller with gravity compensation (experimental).

can lead to inaccurate tracking. In conclusion, accurate vision algorithms require simple controllers (PI) for successful visual tracking. However, accurate vision algorithms are computationally expensive and computational delays are introduced. As an alternative, stochastic controllers (LQG) can be used to compensate for inaccurate measurements without significantly increasing the complexity of the whole system.

## VII. Conclusions

In this paper, we considered the robotic visual tracking problem. Specifically, we addressed the robotic visual tracking of arbitrary 3-D objects traveling at unknown velocities in a 2-D space (depth is given as known). We first presented a mathematical model of the problem. This model is a major contribution of our work and is based on measurements of the vector of discrete displacements, which are obtained by the sum-of-squared differences (SSD) optical flow. This technique seems to be accurate enough even though it is time consuming. Other contributions of this work are a sophisticated multiple windows measurement scheme and new efficient confidence measures that improve the accuracy of the visual measurements. The speed has also been improved by coarse-to-fine techniques and subpixel fitting.

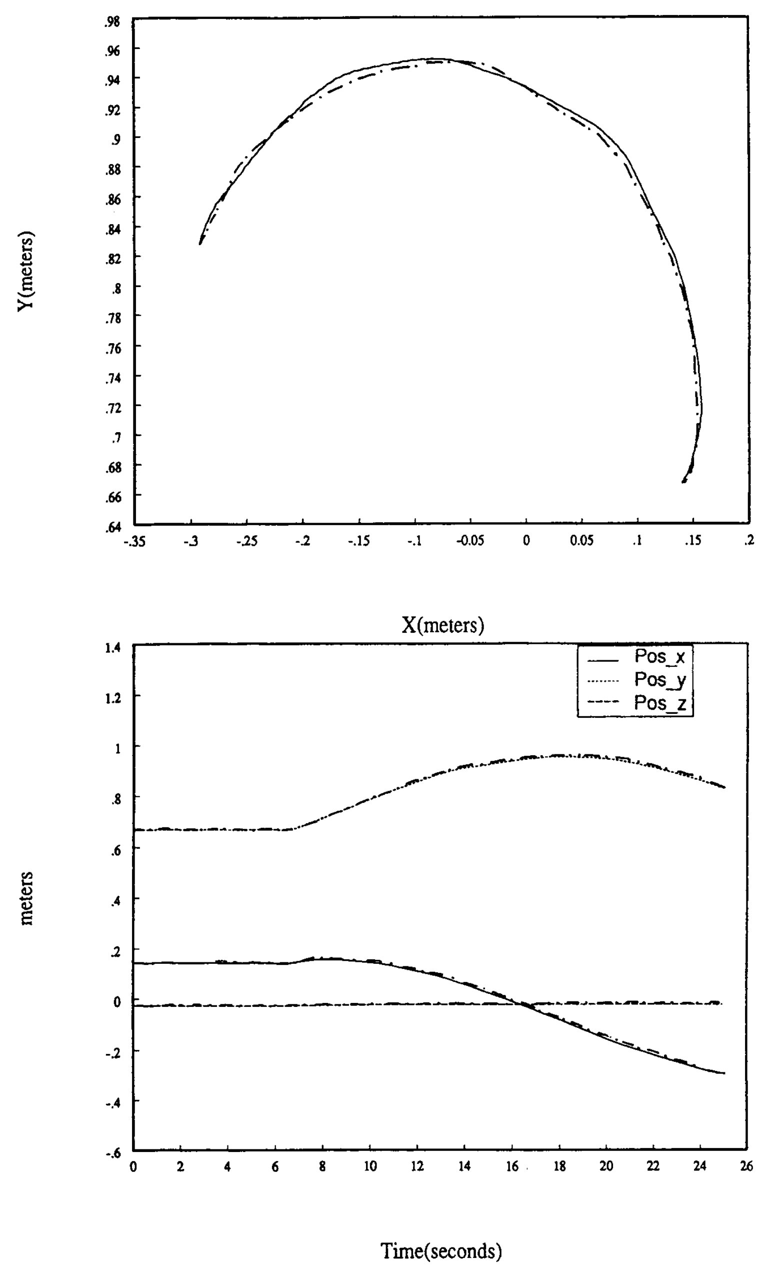


Fig. 19. LQG controller in conjunction with a Cartesian PD robot controller with gravity compensation (experimental). The trajectory in $X$–$Y$ is nonlinear, and the object performs a rotational motion along with its translational motion.

The next step was to show the effectiveness of the introduced idea of the combination of control with vision for an efficient solution to the tracking problem. LQG or PI or pole assignment regulators in conjunction with Cartesian robotic controllers were studied as possible controllers. The selection of the most efficient is based on the vision technique that is used for the calculation of the displacement vector. Stochastic control techniques are effective in the case of noisy visual measurements while one-step-ahead controllers are appropriate for quite accurate visual observations. Generally, these controllers are tuned easily and deal satisfactorily with the tracking constraints and with the noisy type of the measurements. The algorithms were tested on a real robotic environment, the CMU DD Arm II. Experimental results show that these algorithms are promising. Careful implementation of the appropriate algorithm may lead to robust and fast tracking. Speed is dependent on the characteristics of the camera and the computational speed of the available hardware. It should

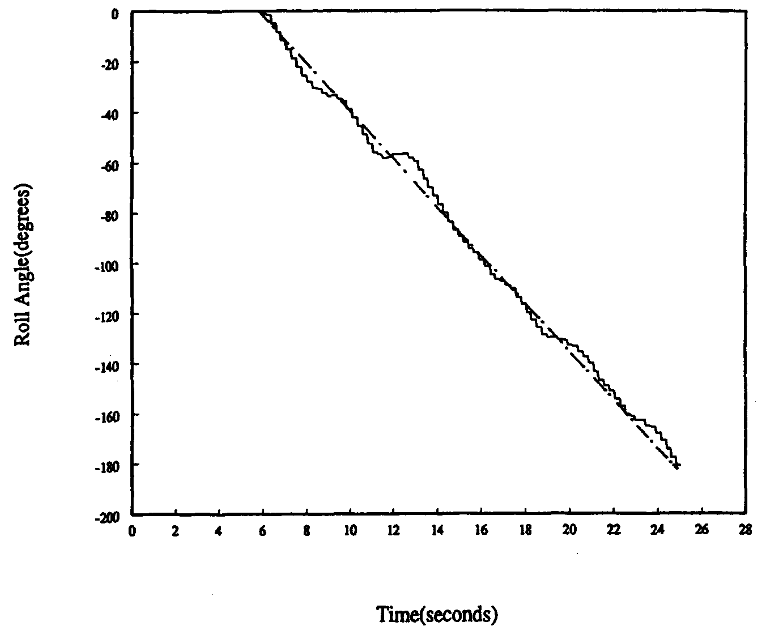


Fig. 20. Roll trajectory of the robot end-effector and the object in the previous example (experimental).

be mentioned that the extension to full 3-D tracking is an open research problem. The reason is that the visual tracking problem becomes much more complex and nonlinear in the 3-D case. Adaptive control techniques [28], [29] give some promising results and may lead to an efficient solution of this difficult problem.

The extension of the method to the problem of 3-D visual tracking, the use of elaborate adaptive control schemes, the integration of the robot dynamics in the state model, the investigation for more efficient motion detection algorithms, and the interaction of the control schemes with the noisy vision algorithms are currently being addressed.

## APPENDIX
## MODELING THE 2-D VISUAL TRACKING OF AN OBJECT

Two feature points of a moving target are selected to be used for the 2-D visual tracking of it. It is assumed that the two points have the same depth $Z_s$ and that their projections on the image plane are not the same. The index $j$ represents each one of them. The optical flow of their projections on the image plane is given by:

$$u^{(j)}(k) = u_o^{(j)}(k) + u_c^{(j)}(k), \qquad j = 1,2 \qquad \text{(A1)}$$
$$v^{(j)}(k) = v_o^{(j)}(k) + v_c^{(j)}(k), \qquad j = 1,2 \qquad \text{(A2)}$$

where $u_c^{(j)}(k)$ and $v_c^{(j)}(k)$ are the components of the optical flow induced by the tracking motion of the camera, and $u_o^{(j)}(k)$ and $v_o^{(j)}(k)$ are the components of the optical flow induced by the motion of the target. Equations (59) and (60) are derived in the same way as (11) and (12). By using (1) and (2) we obtain

$$u_c^{(j)}(k) = -\frac{T_x(k)}{Z_s} + y^{(j)}(k)R_z(k)$$
$$= u_{ct}^{(j)}(k) + u_{cr}^{(j)}(k), \qquad j = 1,2 \qquad \text{(A3)}$$
$$v_c^{(j)}(k) = -\frac{T_y(k)}{Z_s} - x^{(j)}(k)R_z(k)$$
$$= v_{ct}^{(j)}(k) + v_{cr}^{(j)}(k), \qquad j = 1,2 \qquad \text{(A4)}$$

where $u_{ct}^{(j)}(k)$ and $v_{ct}^{(j)}(k)$ are the translational components of the camera induced optical flow, and $u_{cr}^{(j)}(k)$ and $v_{cr}^{(j)}(k)$ are its rotational components. In addition, each one of the $u^{(j)}(k), v^{(j)}(k), u_o^{(j)}(k)$, and $v_o^{(j)}(k)$ can be decomposed to a translational component and a rotational one. Thus, $u^{(j)}(k), v^{(j)}(k), u_o^{(j)}(k)$, and $v_o^{(j)}(k)$ are given by

$$u^{(j)}(k) = u_t^{(j)}(k) + u_r^{(j)}(k), \qquad j = 1,2 \qquad \text{(A5)}$$
$$v^{(j)}(k) = v_t^{(j)}(k) + v_r^{(j)}(k), \qquad j = 1,2 \qquad \text{(A6)}$$
$$u_o^{(j)}(k) = u_{ot}^{(j)}(k) + u_{or}^{(j)}(k), \qquad j = 1,2 \qquad \text{(A7)}$$
$$v_o^{(j)}(k) = v_{ot}^{(j)}(k) + v_{or}^{(j)}(k), \qquad j = 1,2 \qquad \text{(A8)}$$

where $u_t^{(j)}(k), v_t^{(j)}(k), u_{ot}^{(j)}(k)$, and $v_{ot}^{(j)}(k)$ are the translational components and $u_r^{(j)}(k), v_r^{(j)}(k), u_{or}^{(j)}(k)$, and $v_{or}^{(j)}(k)$ are the rotational components. Under the assumption that the two points have the same depth, it can be shown that $u_t^{(1)}(k) = u_t^{(2)}(k), v_t^{(1)}(k) = v_t^{(2)}(k), u_{ot}^{(1)}(k) = u_{ot}^{(2)}(k)$, and $v_{ot}^{(1)}(k) = v_{ot}^{(2)}(k)$. By using (A1)–(A8) the following equations are derived:

$$u_t^{(j)}(k) = u_{ot}^{(j)}(k) + u_{ct}^{(j)}(k), \qquad j = 1,2 \qquad \text{(A9)}$$
$$v_t^{(j)}(k) = v_{ot}^{(j)}(k) + v_{ct}^{(j)}(k), \qquad j = 1,2 \qquad \text{(A10)}$$
$$u_r^{(j)}(k) = u_{or}^{(j)}(k) + u_{cr}^{(j)}(k), \qquad j = 1,2 \qquad \text{(A11)}$$

$$v_r^{(j)}(k) = v_{or}^{(j)}(k) + v_{cr}^{(j)}(k), \qquad j = 1, 2. \qquad (A12)$$

Equations (A9) and (A10) can be combined together because all the projections of the feature points on the image plane have the same translational components of the optical flow. In addition, (A11) and (A12) can be combined to a single equation because $u_{or}^{(j)}(k) = -y^{(j)}(k)\omega(k)$ and $v_{or}^{(j)}(k) = x^{(j)}(k)\omega(k)$ (these relations hold since we assume that the optical axis and the axis of rotation coincide at the initialization stage). The term $\omega(k)$ is the roll component of the angular velocity of the target. Thus, we can summarize the model to the following equations:

$$x(k+1) = x(k) + Tu_c(k) + Tu_o(k) + v_1(k) \quad (A13)$$
$$y(k+1) = y(k) + Tv_c(k) + Tv_o(k) + v_2(k) \quad (A14)$$
$$\theta(k+1) = \theta(k) - TR_z(k) + T\omega(k) + v_3(k) \quad (A15)$$

where $x(k), y(k)$, and $\theta(k)$ are now the $X$, $Y$, and roll components of the tracking error, respectively. $u_c(k)$ and $v_c(k)$ represent the translational components of the camera induced optical flow, and $v_1(k), v_2(k)$, and $v_3(k)$ are the white noise terms. The above equations can be written in state-space form as

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}_c(k) + \boldsymbol{E}\boldsymbol{d}(k) + \boldsymbol{H}\boldsymbol{v}(k) \qquad (A16)$$

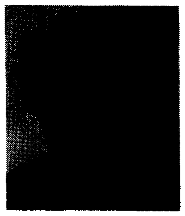
where $\boldsymbol{A} = \boldsymbol{H} = \boldsymbol{I}_3, \boldsymbol{B} = \boldsymbol{E} = T\boldsymbol{I}_3$, $\boldsymbol{x}(k) \in R^3$, $\boldsymbol{u}_c(k) \in R^3, \boldsymbol{d}(k) \in R^3$, and $\boldsymbol{v}(k) \in R^3$. The vector $\boldsymbol{x}(k) = (x(k), y(k), \theta(k))^T$ is the state vector, $\boldsymbol{u}_c(k) = (u_c(k), v_c(k), -R_z(k))^T$ is the control input vector, $\boldsymbol{d}(k) = (u_o(k), v_o(k), \omega(k))^T$ is the exogenous disturbances vector, and $\boldsymbol{v}(k) = (v_1(k), v_2(k), v_3(k))^T$ is the white noise vector.

## Acknowledgment

We thank the anonymous reviewers for their valuable comments. We would also like to thank Innovision Inc. for providing us with the image-processing equipment.

## References

[1] E. H. Adelson and J. R. Bergen, "Spatiotemporal energy models for the perception of motion," *J. Opt. Soc. Amer.*, vol. 2, no. 2, pp. 284–298, Feb. 1985.
[2] P. K. Allen, "Real-time motion tracking using spatio-temporal filters," in *Proc. DARPA Image Understanding Workshop*, 1989, pp. 695–701.
[3] P. K. Allen, B. Yoshimi, and A. Timcenko, "Real-time visual servoing," in *Proc. IEEE Int. Conf. Robotics Automat.*, Apr. 1991, pp. 851-856.
[4] P. Anandan, "Measuring visual motion from image sequences," COINS Dept. Univ. of Massachusetts, Tech. Rep. COINS-TR-87-21, 1987.
[5] B. Bhanu *et al.*, "Qualitative target motion detection and tracking," in *Proc. DARPA Image Understanding Workshop*, 1989, pp. 370–398.
[6] P. I. Corke and R. P. Paul, "Video-rate visual servoing for robots," Grasp Lab, Dept. of Comput. and Info. Sci., Univ. of Pennsylvania, Tech. Rep. MS-CIS-89-18, Feb. 1989.
[7] E. D. Dickmanns and A. Zapp, "Autonomous high speed road vehicle guidance by computer vision," in *Proc. 10th IFAC World Congr.*, July 1987.
[8] E. D. Dickmanns, "Computer vision for flight vehicles," *Z. Flugwiss. Weltraumforsch*, vol. 12, pp. 71–79, 1988.
[9] 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. Syst. Man Cybern.*, vol. 20, no. 5, pp. 1172–1183, 1990.
[10] J. T. Feddema, C. S. G. Lee, and O. R. Mitchell, "Weighted selection of image features for resolved rate visual feedback control," *IEEE Trans. Robotics Automat.*, vol. 7, no. 1, pp. 31–47, 1991.
[11] J. T. Feddema and O. R. Mitchell, "Vision guided servoing with feature-based trajectory generation," *IEEE Trans. Robotics Automat.*, vol. 5, no. 5, pp. 691–699, 1989.
[12] B. Friedland, *Control System Design: An Introduction to State-Space Methods.* New York: McGraw-Hill, 1986.
[13] A. Gelb, *Applied Optimal Estimation.* Cambridge, MA: MIT Press, 1974.
[14] R. Goldenberg, W. C. Lau, A. She, and A. M. Waxman, "Progress on the prototype PIPE," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1987, pp. 1267–1274.
[15] G. C. Goodwin and K. S. Sin, *Information and Systems Science Series*, vol. 1, *Adaptive Filtering, Prediction and Control.* Englewood Cliffs, NJ: Prentice-Hall, 1984.
[16] D.J. Heeger, "Depth and flow from motion energy," *Science*, vol. 86, pp. 657–663, 1986.
[17] M. A. Hersh and M. B. Zarrop, "Stochastic adaptive control of nonminimum phase systems," Univ. of Manchester Inst. of Sci. and Technol., Tech. Rep., 1981.
[18] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intell.*, vol. 17, pp. 185–204, 1981.
[19] B. K. P. Horn, *Robot Vision.* Cambridge, MA: MIT Press, 1986.
[20] A. E. Hunt and A. C. Sanderson, "Vision-based predictive tracking of a moving target," Carnegie Mellon Univ., The Robotics Inst., Tech. Rep. CMU-RI-TR-82-15, Jan. 1982.
[21] A. J. Koivo and N. Houshangi, "Real-time vision feedback for servoing of a robotic manipulator with self-tuning controller," *IEEE Trans. Syst. Man Cybern.*, vol. 21, no. 1, pp. 134–142, 1991.
[22] S. W. Lee and K. Wohn, "Tracking moving objects by a mobile camera," Dept. of Comput. and Info. Sci., Univ. of Pennsylvania, Tech. Rep. MS-CIS-88-97, Nov. 1988.
[23] F. L. Lewis, *Optimal Control.* New York: Wiley, 1986.
[24] R. C. Luo, R. E. Mullen Jr., and D. E. Wessel, "An adaptive robotic tracking system using optical flow," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1988, pp. 568–573.
[25] L. Matthies, T. Kanade, and R. Szeliski, "Kalman filter-based algorithms for estimating depth from image sequences," *Int. J. Comput. Vision*, vol. 3, pp. 209–236, 1989.
[26] N. P. Papanikolopoulos, "Controlled active vision," PhD dissertation, Dept. of Electrical and Comput. Eng., Carnegie Mellon Univ., Pittsburgh, PA, Aug. 1992.
[27] N. Papanikolopoulos, P. K. Khosla, and T. Kanade, "Adaptive robotic visual tracking," in *Proc. Amer. Control Conf.*, June 1991, pp. 962–967.
[28] N. P. Papanikolopoulos and P. K. Khosla, "Adaptive robotic visual tracking: theory and experiments," *IEEE Trans. Automat. Contr.*, to be published, Mar. 1993.
[29] ——, "Feature based robotic visual tracking of 3-D translational motion," in *Proc. 30th IEEE CDC* (Brighton, U.K.), Dec. 1991, pp. 1877–1882.
[30] A. P. Sage, *Optimum Systems Control.* Englewood Cliffs, NJ: Prentice-Hall, 1968.
[31] D. B. Stewart, D. E. Schmitz, and P. K. Khosla, "Implementing real-time robotic systems using CHIMERA II," in *Proc. IEEE Int. Conf. Robotics Automat.* (Cincinnati, OH), May 1990, pp. 598–603.
[32] R. Y. Tsai and T. S. Huang, "Estimating 3-D motion parameters of a rigid planar patch," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. 29, pp. 1147–1152, 1981.
[33] ——, "Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 13–27, 1984.
[34] D. Tsakiris, "Visual tracking strategies," Master's thesis, Dept. of Electrical Eng., Univ. of Maryland, College Park, 1988.
[35] S. Ullman, *The Interpretation of Visual Motion.* Cambridge, MA: MIT Press, 1979.
[36] L. E. Weiss, A. C. Sanderson, and C. P. Neuman, "Dynamic sensor-based control of robots with visual feedback," *IEEE J. Robotics Automat.*, vol. RA-3, no. 5, pp. 404–417, Oct. 1987.
[37] D. B. Westmore and W. J. Wilson, "Direct dynamic control of a robot using an end-point mounted camera and Kalman filter position estimation," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1991, pp. 2376–2384.



**Nikolaos P. Papanikolopoulos** (S'88–M'93) was born in Piraeus, Greece, in 1964. He received the Diploma degree in electrical and computer engineering from the National Technical University of Athens, Athens, Greece, in 1987. He received the M.S.E.E. degree in electrical engineering and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University (CMU), Pittsburgh, PA, in 1988 and 1992, respectively.

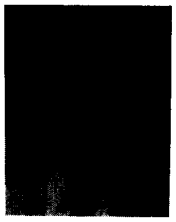
From 1987 to 1988, he was a Research Assistant in the Engineering Design Research Center at CMU. From 1988 to 1992, he was a Research Assistant in the Robotics Institute at CMU. Currently, he is an Assistant Professor in the Department of Computer Science at the University of Minnesota, Minneapolis. His research interests include robotics, control, computer vision, and CIM. He has authored or coauthored more than 25 journal and conference papers in these areas.

Dr. Papanikolopoulos has been a member of the National Technical Chamber of Greece since 1987. He is also a member of the IEEE Robotics and Automation Society, the IEEE Control Systems Society, the IEEE Systems, Man, and Cybernetics Society, and the IEEE Computer Society. He was a finalist for the Anton Philips Award for Best Student Paper in the 1991 IEEE Robotics and Automation Conference. He was a recipient of the Kritski Fellowship in 1986 and 1987.

**Pradeep K. Khosla** (S'83–M'86–SM'91) received the M.S. and Ph.D. degrees from Carnegie Mellon University (CMU), Pittsburgh, PA.

He is currently an Associate Professor in the Department of Electrical and Computer Engineering at CMU. He is also a member of the Robotics Institute and Director of the Advanced Manipulators Laboratory. Prior to joining CMU, he worked with Tata Consulting Engineers and Siemens in the area of real-time control. His research interests are in the area of real-time sensor-based manipulation, architectures for real-time control, integrated design-assembly systems, and robotic applications in space, field, and manufacturing environments. He is involved in electrical and computer engineering and robotics education both at the graduate and undergraduate levels. He was a member of the committee that formulated a curriculum for the Ph.D. program in robotics at CMU. He was also a member of the Wipe the Slate Clean Committee that created a new four-year undergraduate ECE degree curriculum at CMU. His research has resulted in more than 100 journal articles, conference papers, and book contributions. He also serves as a consultant to several industries in the United States.

Prof. Khosla is the Chairman of the Education Committee of the IEEE Robotics and Automation Society. He served as Program Vice-Chairman in 1989 and General Chairman in 1990 of the IEEE International Conference on Systems Engineering. He has also served on the Program Committees of several other international conferences. He is currently the Director of the Robotics and Expert Systems Division of the Instrument Society of America, a member of the IEEE Systems, Man, and Cybernetics Society AdCom, an IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION Technical Editor, and PACE Chairman of the IEEE Robotics and Automation Society. He is a recipient of the Inlaks Fellowship (United Kingdom) and received the Carnegie Institute of Technology Ladd award for excellence in research in 1989. He is a member of the Technical Advisory Board of the Next Generation Controller Project (WPAFB, U.S. Air Force, and Martin Marietta) and the Sample Acquisition Analysis and Preservation Project (Jet Propulsion Laboratory, NASA). He has been an invited participant to the Department of Commerce workshops on the Intelligent Manufacturing Systems program and the U.S.A.–Japan R&D consortia and collaboration.



**Takeo Kanade** (M'80–SM'88–F'92) received the Doctoral degree in electrical engineering from Kyoto University, Kyoto, Japan, in 1974.

After holding a faculty position at the Department of Information Science, Kyoto University, he joined Carnegie Mellon University (CMU), Pittsburgh, PA, in 1980, where he is currently Professor of Computer Science and Co-Director of the Robotics Institute. He established the robotics Ph.D. program at CMU and is currently chairman of the program. He has made technical contributions in multiple areas of robotics: vision, manipulators, autonomous mobile robots, and sensors. His contributions in vision include shape recovery from line drawings (known as Origami World theory and skew symmetry), stereo, color, and face recognition. He is the codeveloper of the concept of direct-drive manipulators and the world's first prototype (the CMU DD Arm I). In the area of autonomous mobile robots, he has been the coprincipal investigator of CMU's NavLab—DARPA's autonomous land vehicle project—and CMU's Ambler—NASA's planetary exploration robot project.

Dr. Kanade is a Founding Fellow of the American Association of Artificial Intelligence, the founding editor of the *International Journal of Computer Vision*, and a member of the IEEE Robotics and Automation Society AdCom. He has received several awards including the Marr Prize Paper Award in 1990; his paper was selected as one of the most influential papers that appeared in the journal *Artificial Intelligence*. He has served for many government, industry, and university advisory panels, including the NASA Advanced Technology Advisory Committee and the Canadian Institute for Advanced Research.