Presented at Third NASA Center for Intelligent Robotic Systems for Space Exploration Conf, Troy, NY
November 18-19, 1991

USING CONTROL AND VISION FOR SPACE APPLICATIONS

N. P. Papanikolopoulos and P. K. Khosla

Department of Electrical and Computer Engineering
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

In this paper, we present algorithms that address the real-time robotic visual tracking (eye-in-hand configuration) of satellites that move in 2-D space. To achieve our objective, we combine computer vision techniques, for detection of motion, with simple control strategies. The problem has been formulated from the systems theory point of view. This facilitates the extension of the algorithm to 3-D space. 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 space robotic system. Shared and traded control modes enable the integration of the human operator with the autonomous tracking modules. In this way, the human operator can intervene and correct the tracking motion through a pystick. The performance of the proposed algorithms has been tested on a real system, the CMU DD Arm II, and the results are presented in this paper.

1. Introduction

An important component of a space 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 space 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 name a few. In this paper, we address the use of vision for dynamically servoing a manipulator for satellite tracking.

Research in computer vision has traditionally emphasized the paradigm of image understanding. However, some work has been reported towards the use of vision information for tracking [1, 2, 3, 4]. In addition, some research [5, 6] has been conducted in using vision information in the dynamic feedback loop. While we address the problem of using vision information in the dynamic feedback loop of a space robot, our paradigm is slightly different. Specifically, we claim that combining vision with control can result in better measurements. 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.

The goal of using autonomous robots for applications in space is to remove the human from the task site but not necessarily to remove the human/operator from the task itself. In fact due to safety considerations, it is very important that the human be able to remotely take control of the task and the space robot system. For example, unpredictable events and complex evolving environments require the intervention of the human operator. The integration of the human operator in a space robotic system results in a system that combines man and machine capabilities and such a system is often called a space telerobotic system. We view a space telerobotic system as a general multi-sensor environment where the human operator and the autonomous sensory modules can cooperate and exchange control of the decision-making process. The exchange of control depends on the specific events which occur during the execution of a task.

In this paper, we present algorithms for robotic (eye-in-hand configuration) real-time visual tracking of arbitrary satellites traveling at unknown velocities in a 2-D space. The problem of visual tracking is formulated as a problem of combining control with computer vi-

sion. We present a mathematical formulation that is general enough to be extended to the problem of tracking satellites in 3-D space. We propose the use of 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 of 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 structure of our framework is depicted in Fig. 1. Our architecture provides the means for the intervention of the human operator through a pystick Manual, autonomous, shared, and traded control modes are supported. The performance of the proposed algorithms has been tested on a real system, the CMU DD Arm II, and the results are presented in this paper.

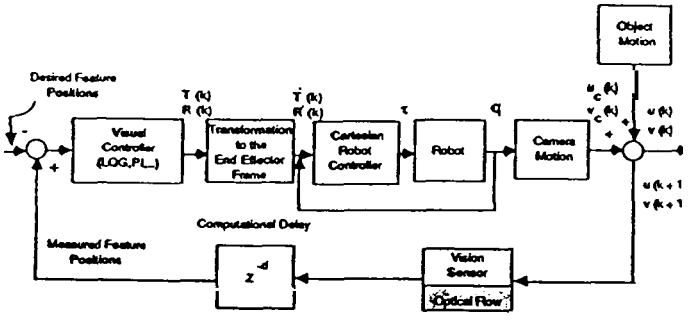

Figure 1: Structure of the Robotic Visual Tracking Scheme.

The rest of the paper is devoted to the description of our algorithms and is organized as follows: In Section 2, 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 3. The control strategies, the steady-state Kalman filter and the selection of the a p propriate control law with regards to the noise level of the measurements are discussed in Section 4. Section 5 describes our strategies for the integration of the human operator with the autonomous visual tracking modules. Cartesian robot control schemes are presented in Section 6. Section 7 describes the hardware configuration of our experimental testbed, DD Arm II. Experimental results are presented in Section 8. Finally, in Section 9, the paper is summarized.

2. 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 [7] defines the optical flow as "the apparent motion of the brightness patterns". For 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. We also assume a perspective projection and the focal length to be unity. A point' P with coordinates (X_s, Y_s, Z_s) in R_s projects onto a point p in the image plane with image coordinates (x, y). Let us assume that the camera moves in a static environment with translational velocity $\mathbf{T} = (T_x, T_y, T_z)^T$ and with angular velocity $\mathbf{R} = (R_x, R_y, R_z)^T$ with respect to thecamera frameR, The optical flow equations are [4]:

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

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

where $u = \dot{x}$ and $v = \dot{y}$. Now, instead of assuming a static object and a moving camera, if we were to 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

Bold symbols denote vectors or matrices

of much research and many algorithms have been proposed [8, 9, 10, 11].

For accuracy reasons, we use a matching based technique [12] also known as the Sum-of-Squared Differences (SSD) optical flow. For a point $p(k-1)=(x(k-1), y(k-1))^T$ in image $(k-1)$ (the symbol k denotes the image (k) which belongs to a sequence of images) we want to find the point $p(k)=(x(k-1)+u, y(k-1)+v))^T$ to which the point $p(k-1)$ moves in image (k). For the point $p(k-1)$, the SSD algorithm selects the displacement $d=(u,v)^T$ that minimizes the SSD measure:

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

where $u, v \in \Omega$, N is an area around the pixel we are interested in, and $I_{k-1}(\cdot,\cdot)$, $I_k(\cdot,\cdot)$ are the intensity functions in images $(k-1)$ and (k), respectively. The accuracy of this technique can be improved using subpixel fitting and multi-grid techniques at the cost of increasing the computational complexity. Its computational speed can be improved by selecting an a p pmpriate 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 not trivial. Images are a noisy source of information and changes in illumination and surface reflectance can deteriorate the performance of any confidence measure. An effiaent confidence measure should recognize errors that are due to homogeneous areas and occlusion boundaries. Anandan [12] developed a confidence measure that confronts the maprity of these problems. He defined as an

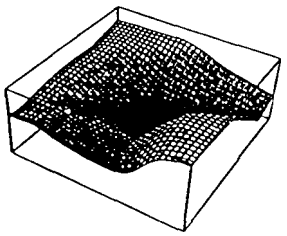

Figure2 SSD surface that corresponds to a corner point.

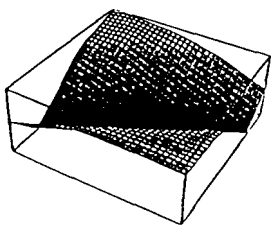

Figure 3: SSD surface that corresponds to a feature point that belongs to an edge.

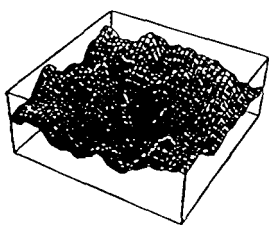

Figure 4: SSD surface that corresponds to a feature point that belongs to an homogeneous area.

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. A SSD surface that corresponds to a corner point (one of the best features which can be used) presents a sharp minimum at the best match (Fig. 2). A feature point that belongs to an edge (these points provide accurate measurements only in the direction perpendicular to the edge) has a SSD surface which presents multiple local minimum in the direction of this edge (Fig. 3). Finally, a SSD surface that corresponds to a feature point that belongs to an homogeneous area (these feature points provide inaccurate data) has multiple small local minimum in all the directions (Fig. 4). 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, commonly found in space applications. The curvature of the SSD surface seems to be proportional to the quality of the best match. Anandan proposed an algorithm for computing the confidence measures based on the principal curvatures and the directions of the principal 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 that is 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. [13] 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 σ_I^2 is the variance of the image noise and a is the second order coefficient of the parabola $e(d_r) = a d_r^2 + b d_r + c$. We have proposed an extension of this technique to two dimensions by fitting parabolas to the directions of the principal axes in the area of the SSD surface minimum [14]. Thus, we can compute the confidence measure for the i-th window as:

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

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 principal 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 which have been used in the experiments [14]. 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

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

where the s_k's are the sample standard deviations in the directions of the four principal 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 prinapal 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

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

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 principal axes.

The selection of the best measurements is based on the values of their confidence measures. If the object moves with two translation 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_c. 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 $w_i = conf_i / (\sum_j conf_j)$, where $conf_i$ is the confidence measure of the i-th window. The arithmetic mean of u and v is

$$\bar{u} = \frac{\sum_J w_j u_j}{\sum_J w_j} \qquad \bar{v} = \frac{\sum_J w_j v_j}{\sum_J w_j} \tag{9}$$

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.

The techniques discussed above can be extended from black/white images to color images. In [14], 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.

3. 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 a satellite. The 2-D visual tracking of a satellite is realized by visually tracking multiple feature points that belong to the satellite.

3.L Visual Tracking of a Single Feature Point

Consider a satellite that moves in a plane with a feature, located at a point P, that we want to track. The projection of this point on the image plane is the point p. Consider also a neighborhood Ω_w of 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 Ω_w stationary in an area Ω_o around the origin of the image frame". It is assumed that at initialization of the tracking process, the area Ω_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 [4]: "find the camera rotation (R_x, R_y) with respect to the camera frame that keeps Ω_w stationary in an area Ω_o around the origin of the image frame".

The second definition of the visual tracking problem does not q u i r e the computation of the depth Z_c of the point P. Both definitions are used in our experiments.

Assume that the optical flow of the point p at the instant of time kT is $(u(kT), v(kT))$ where T is the time between two *consecutive frames. It can be shown that at time $(k+1)T$, the optical flow is:

$$u((k+1)T) \approx u(kT) + u_c(kT) \tag{11}$$

$$v((k+1)T) \approx v(kT) + v_c(kT) \tag{12}$$

where $u_c(kT), v_c(kT)$ are the components of the optical flow induced by the tracking motion of the camera. Equations (11) and (12) are based on the assumption that the optical flow induced by motion of the feature does not change in the time interval T. Therefore, T should be as small as possible. 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, equations (11)and (12) will be transformed to:

$$u((k+1)T) = u(kT) + u_c((k-d)T) \tag{13}$$

$$v((k+1)T) \approx v(kT) + v_c((k-d)T) \tag{14}$$

where d is the delay factor. For the time being, the delay factor is assumed to be zero. To keep the notation simple and without any loss of generality, equations (11) and (12) will be used with k and $(k+1)$ instead of kT and $(k+1)T$ respectively.

By using the relations $u(k) = \frac{x(k) - x(k-1)}{T}$ and $v(k) = \frac{y(k) - y(k-1)}{T}$, equations (11) and (12) can be transformed to state-space form as (the inaccuracies of the model are modeled as white noise):

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

where** $A = H = I_2$, $B = E = T I_2$, $\mathbf{x}(k) \in R^2$, $\mathbf{u}_c(k) \in R^2$, $\mathbf{d}(k) \in R^2$, and $\mathbf{v}(k) \in R^2$. The vector $\mathbf{x}(k) = (x(k), y(k))^T$ is the state vector, $\mathbf{u}_c(k) = (u_c(k), v_c(k))^T$ is the control input vector, $\mathbf{d}(k) = (u(k), v(k))^T$ is the exogenous disturbances vector, and $\mathbf{v}(k) = (v_1(k), v_2(k))^T$ is the white noise vector. The measurement vector $\mathbf{y}(k) = (y_1(k), y_2(k))^T$ is given by

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{w}(k) \tag{16}$$

where $\mathbf{w}(k) = (w_1(k), w_2(k))^T$ is a white noise vector

** The symbol I_n denotes the identity matrix of order n.

$(w(k) \sim N(0,W))$ and $C = I_2$. The measurement vector is computed using the SSD algorithm described in Section 2.

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

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

We assume that for 2-D visual tracking the depth Z, remains constant. When the tracking motion of the camera is rotation with $R_x(k)$ and $R_y(k)$, the optical flow induced by the moving camera is:

$$u_c(k) = R_x(k)x(k)y(k) - R_y(k)[x^2(k)+1] \tag{18}$$

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

The model in equations (15) and (16) can also be used for keeping the feature point stationary in an area Ω_r different from the origin. If (r_x, r_y) is the center of this area Ω_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 A, B, C, E, 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.

3.2. 2-D Visual Tracking of a Satellite

Consider a satellite that moves in a plane which is perpendicular to the optical axis of the camera. The projection of the satellite on the image plane is the area Ω_w in the image plane, The problem of 2-D visual tracking of a single satellite can be defined as: "find the camera translation (T_x, T_y) and rotation (R_z) with respect to the camera frame that keeps Ω_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 mathematical model of this problem in state-space form is:

$$\mathbf{x}(k+1) = \mathbf{A}\,\mathbf{x}(k) + \mathbf{B}\,\mathbf{u}_c(k) + \mathbf{E}\,\mathbf{d}(k) + \mathbf{H}\,\mathbf{v}(k) \tag{20}$$

where $A = H = I_3$, $B = E = T\,I_3$, $\mathbf{x}(k) \in R^3$, $\mathbf{u}_c(k) \in R^3$, $\mathbf{d}(k) \in R^3$ and $\mathbf{v}(k) \in R^3$. The vector $\mathbf{x}(k) = (x(k), y(k), \theta(k))^T$ is the state vector, $\mathbf{u}_c(k) = (u_c(k), v_c(k), R_z(k))^T$ is the control input vector, $\mathbf{d}(k) = (u(k), v(k), \omega(k))^T$ is the exogenous disturbances vector, and $\mathbf{v}(k) = (v_1(k), v_2(k), v_3(k))^T$ is the white noise vector. $x(k)$, $y(k)$, $\theta(k)$ are now the X, Y and roll component of the tracking error, respectively. The measurement vector $\mathbf{y}(k) = (y_1(k), y_2(k), y_3(k))^T$ is given by

$$\mathbf{y}(k) = \mathbf{c}\,\mathbf{x}(k) + \mathbf{w}(k) \tag{21}$$

where $\mathbf{w}(k) = (w_1(k), w_2(k), w_3(k))^T$ is a white noise vector $(\mathbf{w}(k) \sim N(0,W))$ and $C = I_3$. The measurement vector is obtained in a slightly different way than in the case of the visual tracking of a single feature point. First, the tracking error of the projections of the two different 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 component of the tracking error. If the projections of the two feature points on the image plane are not the same, it is guaranteed that the system of equations has a solution. It is assumed that each one of these features at time $t = 0$ is located at its desired position. The control strategies that keep the projection of the satellite stationary are discussed in detail in the next Section.

4. Control Issues in the Visual Tracking Problem

The control techniques that will be presented for the 2-D visual tracking of a moving satellite can be used easily for the visual tracking of a single moving feature point. The mathematical models for the two cases (feature, satellite) 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 tries to indicate the conditions under which these schemes should be used in the visual tracking problem. Some of these controllers (PI) are attractive due to their simplicity and others (LQG, Pole Assignment) are more general. Appropriate selection can maximize the improvements that are derived by the combined use of control and vision techniques in the visual tracking problem.

4.1. PI Controller for Visual Tracking

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

$$u_c(k) = [K_p e(k) + K_I T \sum_{i=1}^{k} e(i)] T^{-1} \quad (22)$$

where K_p and K_I are constant proportional and integral gain matrices, respectively. There are several techniques for the calculation of the K_p and K_I matrices. One obvious effect of the integral control is that it increases the type of the system by one. Thus, the steadystate 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 K_p and K_I are not properly selected.

4.2. DARMA Model and Pole Assignment Controller

A more general technique for the design of the controller of the 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:

$$A_D(q^{-1}) y(k) = B_D(q^{-1}) u_c(k) \quad k \geq 0 \quad (23)$$

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 equations (20) and (21) can be transformed to the model described by the equation (23). These assumptions are: a) the disturbances are deterministic and constant, and b) the noise tenns are neglected. Even though these assumptions oversimplify the problem, proper selection of the closed-loop poles *can* lead to a robust and effiaent controller. The DARMA model for the 2-D visual tracking problem is

$$A_D(q^{-1}) = (1 - 2q^{-1} + q^{-2}) I, \quad (24)$$

$$B_D(q^{-1}) = T q^{-1} (1 - q^{-1}) I_3 \quad (25)$$

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 the equation

$$L(q^{-1}) u_c(k) = -P(q^{-1}) y(k) \quad (26)$$

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:

$$L'(q^{-1})(1 - q^{-1})^2 I_3 + P(q^{-1}) T q^{-1} I_3 = A^*(q^{-1}) \quad (27)$$

where $A^*(q^{-1})$ is a matrix whose diagonal elements are the desired closed-loop polynomials. Equation (27) 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 $T q^{-1} I$, are relatively prime. We observe that the MIMO (Multi-Input Multi-Output) model is decoupled, so we can work with three SISO (Single-Input Single-Output) models. This simplifies the analysis. Thus, equation (27) can be decomposed into three polynomial equations:

$$L_i'(q^{-1})(1 - q^{-1})^2 + P_i(q^{-1}) T q^{-1} = A_i^*(q^{-1}) \quad i = 1,2,3 \quad (28)$$

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 reasons, 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.

4.3. ARMAX Model and Pole Assignment Controller

The DARMA model does not incorporate the existing stochastic disturbances. Thus, we should use a more general model that includes both the stochastic and the deterministic disturbances. An efficient mathematical model for this type of problem is the *AutoRegressive Mooing-Amrage model with eXternal input* (ARMAX)

$$\bar{A}_D(q^{-1}) D(q^{-1}) y(k) = \bar{B}_D(q^{-1}) D(q^{-1}) u_c(k) + \\ + \bar{C}_D(q^{-1}) D(q^{-1}) w(k) \quad k \geq 0 \quad (29)$$

where $w(k)$ is the white noise vector that corrupts the measurement vector $y(k)$ and $D(q^{-1}) = (1 - q^{-1}) I_3$. 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 $\bar{B}_D(q^{-1}) = T q^{-1} I_3$. If the set point is described by the relationship $S(q^{-1}) y^*(k) = 0$, then a robust control law is given by the equation (16):

$$L(q^{-1}) S(q^{-1}) D(q^{-1}) u_c(k) = P(q^{-1}) [y^*(k) - y(k)] \quad (30)$$

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

$$L_i(q^{-1}) S_i(q^{-1}) (1 - q^{-1})^2 + P_i(q^{-1}) T q^{-1} = \\ = C_i^*(q^{-1}) A_i^*(q^{-1}) \quad i = 1.23 \quad (31)$$

where

$$\bar{C}_{D\,i}(q^{-1})D_i(q^{-1}) = C_i^s(q^{-1}) + C_i^c(q^{-1}) \quad i = 1,2,3 \tag{32}$$

The polynomials $C_i^s(q^{-1})$ should be asymptotically stable and $C_i^c(q^{-1})$ are residuals that can be made as small as desired. Further, it can be shown [16] that the tracking error converges to

$$y_i(k) - y_i^*(k) = \frac{L_i(q^{-1})S_i(q^{-1})}{A_i^*(q^{-1})}\left[1 + \frac{C_i^c(q^{-1})}{C_i^s(q^{-1})}\right] w_i(k) \quad i = 1,2,3 \tag{33}$$

The performance of the controller can be made as close to optimal as desired by choosing $C_i^c(q^{-1})$ sufficiently small and $A_i^*(q^{-1})$ sufficiently big. The robustness of the controller can also be enhanced by keeping the roots of $C_i^c(q^{-1})$ a reasonable distance from the unit circle. Big $A_i^*(q^{-1})$ results in large rise time (or delay) which it is not desirable in a visual tracking system. The vision algorithms that we propose cannot detect image plane displacements greater than 18 pixels per sampling period T. Thus, a compromise between these contradicting requirements should be made.

4.4. LQG Controller for Visual Tracking

A more complex control method is the LQG (Linear Quadratic Gaussian) control technique. This technique permits us to model the deterministic disturbances as time-varying while the previous mentioned control algorithms do not cover this case. The LQG controller can efficiently confront a larger set of 2-D satellite's trajectories. The control input vector $\mathbf{u}_c(k)$ is given by

$$\mathbf{u}_c(k) = -\mathbf{G}\hat{\mathbf{x}}(k) - \hat{\mathbf{d}}(k) \tag{34}$$

where $\hat{\mathbf{x}}(k)$ is the estimated value of the state vector $\mathbf{x}(k)$, and $\hat{\mathbf{d}}(k)$ is the estimated value of the disturbance vector $\mathbf{d}(k)$. More details about the design of the LQG controller can be found in [17].

4.5. 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 mentioned above, the first two cases correspond to the visual tracking of a point while the last case represents the visual tracking of a satellite. In the first case, $T_x(k)$ and $T_y(k)$ are computed by the equations (17). As we have mentioned before, these equations require the knowledge of the depth Z_s. In the second case, we have to solve the system of equations (18) and (19) 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. $R_z(k)$ 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 [13]. 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 visual 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. More details about our adaptive control schemes can be found in [18, 19].

5. Shared and Traded Visual Tracking

In shared control mode, some degrees of freedom (DOF) are commanded by the human operator and some by the autonomous modules. In traded control mode, there is a transition from the autonomous modules to the human operator and vice versa. There are a lot of different ways of looking at the design of modules that support these modes. In [20], *mixing matrices* and *weights* ... are used to decide when and how the autonomous modules will mix with the human operator. On the other hand, in [21, 22], the human operator and the autonomous modules are not allowed to command the same degree of freedom (DOF). The reason is to avoid the "fork in the road" problem which may occur when the human operator and the autonomous modules suggest opposite directions of motion along a specific degree of freedom (DOF).

We choose to follow the latter approach. The control input signal $\mathbf{Dx}_c(k)$ is defined as $\mathbf{Dx}_c^T(k) = (\mathbf{T}^T(k), \mathbf{R}^T(k))$ and is expressed with respect to the camera frame R. Let us assume that the human operator commands a speed signal ${}^j\mathbf{Dx}_c(k)$ with respect to the pystick frame. The speed signal ${}^j\mathbf{Dx}_c(k)$ can be transformed through the transformation ${}^cT_j$ to the camera frame signal ${}^c\mathbf{Dx}_c(k)$. We define the matrices Ψ, and Ω $(\Psi, \Omega \in R^{6\times6})$:

$$\Psi(\alpha,\beta) = \begin{cases} 1 & \text{if } \alpha = \beta \;\&\; \text{dof}(\alpha) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{35}$$

$$\Omega(\alpha,\beta)=\begin{cases}1-\Psi(\alpha,\beta) & \text{if } \alpha=\beta\\ 0 & \text{otherwise}\end{cases} \quad (36)$$

where $dof(\alpha)=1$ $(\alpha \in \{1,\ldots,6\})$ implies that the human operator controls the a degree of freedom (DOF). The new rate reference signal ${}^{c}Dx_c(k)$ is expressed in camera frame coordinates as:

$${}^{c}Dx_c(k)=\Omega\, Dx_c(k)+\Psi\, {}^{m}Dx_c(k) \quad (37)$$

One of the possible problems of shared control mode is the possible coupling between the degrees of freedom (DOF) that the human operator commands and the degrees of freedom commanded by the autonomous modules. For example, in our system, there is a strong coupling between $T_x(k)$ and $R_y(k)$, and between $T_y(k)$ and $R_x(k)$. Therefore, a modification of one of them makes necessary a modification of the other's value. The implementation of this scheme depends on the knowledge of the human operator about the type of coupling. The selection of the degrees of freedom (DOF) which are manually commanded is not an easy task. The way that the CCD camera is mounted on the end-effector and the 2-D nature of the visual information force the human operator to select $R_z(k)$, or T_x and $T_y(k)$, as possible teleoperated degrees of freedom (DOF). This observation is verified by the experimental results which are presented in Section 8.

Another potential problem is the delay in the manual rate signal. If d_m is a delay factor associated with the transmission of the manual rate signal, then equation (37) becomes:

$${}^{c}Dx_c(k)=\Omega\, Dx_c(k)+\Psi\, {}^{m}Dx_c(k-d_m) \quad (38)$$

Large transmission delays which are often present in space applications can significantly influence the tracking performance and the stability of our algorithms. The estimation schemes are influenced too, due to the fact that we apply different inputs to the system than the ones that the estimation schemes consider as current. Therefore, the transmission delays must be taken into consideration during the design phase of the whole system.

The traded control mode is useful during three phases: a) Initialization, b) Emergency events, and c) Fiial stage. During these phases, there is an exchange of control between the human operator and the autonomous modules. In particular, during Initialization, the human operator grossly positions the manipulator with respect to the satellite and selects a number of candidate features for tracking. Then, he/she passes control to the autonomous modules. During Emergency events (e.g. loss of one feature due to occlusion by an unknown object, singular configuration of the manipulator), the human operator takes control and tries to solve the accumulated problems. If the problems are solved, then he/she again passes control to the autonomous modules. Finally, during the Final stage, the human operator takes control of the system and places the manipulator in its home position.

In the next Section, we discuss the Cartesian robot control schemes we used in our experiments on traded and shared control.

6. Robot Control Schemes

After the computation of the translational ${}^{c}T(k)$ and rotational ${}^{c}R(k)$ velocity vectors with respect to the camera frame R, (${}^{c}T(k)$ and ${}^{c}R(k)$ are the components of the vector ${}^{c}Dx_c(k)$), we transform them to the end-effector frame R_e with the use of the transformation ${}^{e}T_c$. The transformed signals are fed to the robot controller. We experimented with two artesian robot control schemes: a artesian 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 satellite. The next Section describes the hardware configuration of our experimental testbed (CMU DD Arm II robotic system).

7. Hardware Configuration

The vision and pystick module of the CMU DD Arm II (Direct Drive Arm II) system are parts of a bigger hardware environment Fig. 5) which runs under the CHIMERA II [23] real-time operating system and consists of the following devices:

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

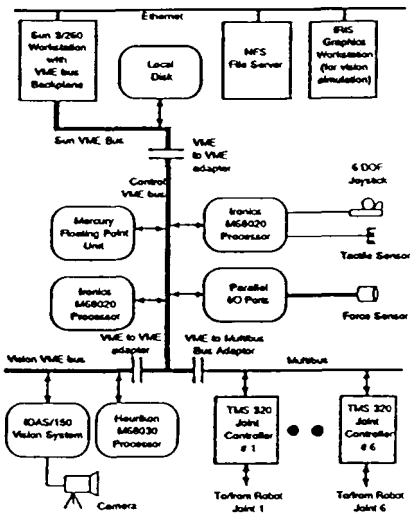

Figure 5: DD Arm II Hardware Configuration (Fig. taken from [23]).

- A Panasonic industrial CCD color camera, Model GP-CD1H.
- Six Texas Instrument TMS320 DSP processors, each controlling one pint of the CMU DD Arm II system.
- Sensors such as a tactile Sensor and a force sensor.
- A six degrees of freedom joystick.

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. An Ironics board is responsible for the realization of the shared control planner. 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 *six* degrees of freedom pystick is a Dimension 6 TrackBall [24, 21]. The applied forces and torques to the trackball are transformed to *six* reference signals which are computed at once. The reference signals correspond *to* either velocities or forces. The human operator *can* select the type and the reference frame of the reference signals on-line. In our experiments, the user commands motions with respect to the pystick frame, and then these motions are transformed in Camera frame coordinates. The camera frame is parallel to the end-effector frame.

The software is organized around 5 processes:

- Vision process. This process does all the image processing calculations and has a period of 150 ms.
- Interpolation process. This program reads the data from the vision system, interpolates the data and sends the reference signals to the robot cartesian controller. It has a period of 5ms.
- Robot controller process. This process drives the robot and has a period of 333 ms.
- Joystick process. This process reads the data from the joystick and docs all the necessary data transformations. It has a period of 30 ms.
- Joystick reference process. This process interpolates the data from the joystick process in order to guarantee a smooth robot trajectory. Its period is 9 ms.

The next Section describes the experimental results.

8. Experimental Results

A number of experiments were performed on the CMU DD Arm II system. We performed two sets of experiments. In the first *set,* the autonomous tracking modules were used. In the second *set,* the human operator commanded motion along some degrees of freedom (DOF) in conjunction with the autonomous modules. During the experiments, the camera is mounted on the end-effector and has a focal length of 7.5mm while the objects are moving on a plane (average depth Z_c= 680mm). The experimental setup is shown in Fig. 6. In the first set of experiments, the first trajectory is the line $Y=0.74X+0.77$ (Y and X in meters). The camera's pixel dimensions are: s_x=0.01278mm and s_y=0.00986mm. The real images are 510x492 and are quantized to 256 gray levels. The user, by moving the mouse around, proposes to the system some of the object's features that he is interested in. Then, the system evaluates on-line the quality of the measurements, based on the confidence measures described in Section 2. Currently, four features are used and the size of the attached windows

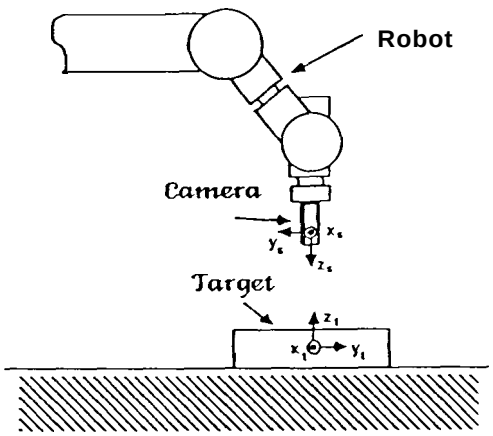

Figure 6: Experimental Setup.

is 10x10. The gains for the controllers can be found in [25]. The experimental results are plotted in Fig. 7-14 where the dotdashed trajectories correspond to the trajectories of the center of mass of the moving objects. The vector $MezP$ represents the position of the end-effector in the world frame The experimental results lead to some interesting observations. The computed torque artesian scheme provides better performance than the simple PD with gravity compensation scheme. This is apparent from Fig. 7 and 8. The simple PD produces oscillations around the desired *trajectory.* The computed toque scheme is more accurate because it takes into consideration the full dynamics of the robot The problem with the computed toque 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). Due to this fact, in all other examples. the Cartesian PD with gravity compensation robotic controller is used.

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. When we decrease the number of the used features and slightly defocus the camera, controllers with the poles near zero fail as in Fig. 11. In this example, a controller with the poles at 0.8 works better (Fig. 9). On the other hand, when we increase the number of the used features and focus correctly the camera, a controller with the poles near zero has a better performance (Fig. 10). These experiments are done with the same object.

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 Fig. 13 and 14. 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. Due to noisy measurements, LQG controller seems to have the best performance. This becomes more obvious, when one reduces the number of the windows which are used. PI is simple but 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 technique 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 arc introduced. As an alternative, stochastic controllers (LQG) can be used to compensate for inaccurate measurements without significantly increasing the complexity of the whole sys tem.

In the second set of experiments, the operator by using the joystick commands motions in the end-effector frame which *is* parallel to the camera frame. In these experiments, the objective is the tracking of a moving target (2-D translational and rotational motion) by combining the human operator with the autonomous LQG visual tracking modules. The goal is to show the performance of the human operator in tracking. The results are shown in Fig. 15. We observe that the performance of the human operator in this *case* is very satisfactory. $MezP[0]$ and $MezP[1]$ are commanded by autonomous LQG visual *tracking* modules, The trajectory in the X-Y plane is nonlinear which results in oscillatory tracking performance. It is assumed that the depth Z_c is known and that at time $k=0$ the optical axis of the camera is aligned with the axis of rotation of the target. Rnally, it is assumed that the two feature points which are selected for tracking have similar depth with respect to

the camera frame. The current and the desired positions of the features are continuously highlighted in order to help the human operator accomplish his/her task. The system provides messages about the current tracking errors and the timing of the task These graphical aids are realized on a graphical overlay of the video display which has a frame rate of 30 frames/sec.

9. Conclusions

In this paper, we considered the robotic visual tracking problem of satellites. Specifically, we addressed the autonomous or shared robotic visual tracking of arbitrary satellites traveling at unknown velocities in a 2-D space. 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 coarseto-fine techniques and sub-pixel fitting.

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 of satellites. 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, thew controllers are tuned easily and deal satisfactorily with the tracking constraints and with the noisy type of the measurements. Our flexible architecture provides the means for effective integration of the human operator with the autonomous tracking modules. Autonomous, manual, traded, and shared control modes are currently supported. The algorithms were tested on a real robotic environment, the CMU DD Arm II. Experimental results show that the methods are quite accurate, robust and promising. Their most important characteristic is that they can be implemented in real-time.

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

10. Acknowledgements

This research was supported by the Defense Advanced Research Projects Agency, through ARPA Order Number DAAA-21-89C-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies. We also thank Innovision Inc. for providing us with the image processing equipment.

References

1. P.K. Allen, "Real-time motion tracking using spatio-temporal filters", *Proc. DARPA Image Understanding Workshop*. 1989, pp. 695-701.

2. R. Goldenberg, W.C Lau, A. She, and AM. Waxman, "Progress on the prototype PIPE", *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1987, pp. 1267-1274.

3. R.C. Luo, R E Mullen Jr., and D.E Wessel, "An adaptive robotic tracking system using optical flow", *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1988, pp. 568-573.

4. D. Tsakiris, "Visual tracking strategies", Master's thesis, Department of Electrical Engineering. University of Maryland, 1988.

5. J.T. Feddema, and C.S.G. Lee, "Adaptive Image Feature Prediction and Control for Visual Tracking with a Hand-Eye Coordinated Camera", *IEEE Trans. Systems, Man and Cybernetics*, Vol. 20, No. 5, 1990, pp. 1172-1183.

6. L.E. Weiss, A.C. Sanderson, and C.P. Neuman, "Dynamic sensor-based control of robots with visual feedback", *IEEE Journal 4 Robotics end Automation*, Vol. RA-3, No. 5, October 1987, pp. 404-417.

7. B.K.P. Horn. *Robot vision*, MIT Press, Cambridge, 1986.

8. E.H. Adelson and J.R. Bergen, 'Spatiotemporal energy models far the perception of motion'', Journal of *the Optical Society* of *America*, Vol. 2, No. 2, February 1985, pp. 284-298.

9. D.J. Heeger, "Depth and flow from motion *energy*", *Science*, No. 86, 1986, pp. 657-663.

10. B.K.P Horn and B.C. Schunck, "Determining optical flow", *Artificial Intelligence*, Vol. 17, 1981, pp. 185-204.

11. R.Y. Tsai and TS. Huang, "Estimating 3-D motion parameters of a rigid planar patch", *IEEE Trans. on Acoustics, Speech and* Signal *Processing*, Vol. 29, 1981, pp. 1147-1152.

12. P. Anandan, "Measuring visual motion from image sequences", Tech. report COINS-TR-87-21, COINS Department, University of Massachusetts, 1987.

13. L. Matthies, R. Szeliski, and T. Kanade, "Kalman filter-based algorithms for estimating depth from image sequences", Tech. report 88-1, Carnegie Mellon University, The Robotics Institute, 1988.

14. N. Papanikolopoulos and P.K. Khosla, "Optical flow and confidence measures", Tech. report, Carnegie Mellon University, The Robotics Institute, 1990.

15. G.C. Goodwin and K.S. Sin, *Adaptive filtering, prediction and control*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, Information and Systems Science Series, Vol. 1, 1984.

16. M.A. Hersh and M.B. Zarrop, "Stochastic adaptive control of nonminimum phase systems", Tech. report, University of Manchester Institute of Science and Technology, 1981.

17. N. Papanikolopoulos and P.K. Khosla, *Real-time LQG robotic visual tracking*, The European Robotics and Intelligent Systems Conference, Corfu, Greece, June 1991.

18. N. Papanikolopoulos, P.K. Khosla, and T. Kanade, "Adaptive robotic visual tracking", *Proc. of the American Control Conference*, June 1991, pp. 962-967.

19. N. P. Papanikolopoulos and P. K. Khosla, "Feature based robotic visual tracking of 3 D translational motion", *Proc.* of the *30th IEEE CDC, Brighton, UK*, December 1991.

20. S. Hayati and S.T. Venkataraman, "Design and implementation of robot control system with traded and shared control capability", *Proc.* of *the IEEE Int. Conf. on Robotics and Automation*, 1989, pp. 1310-1315.

21. H. Schneiderman, "Issues in a telerobotic system: control trading, control sharing, and safety during manual operation", Master's thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1990.

22. H. Schneiderman and P.K. Khasla, "Implementation of traded and shared control strategies for exploration using a tactile sensor", *Proc. of the* Fourth *ANS Topical Meeting on Robotics and Remote Systems*, February 25-27 1991, pp. 217-226.

23. D.B. Stewart, D.E. Schmitz, and P.K. Khosla , "Implementing real-time robotic systems using CHIMERA II", *Proc. 4 1990 IEEE Int. Conf. on Robotics and Automation*, Cincinnatti, Ohio, May 1990, pp. 598-603.

24. G. Hirzinger, "The space and telerobotic concepts of DFVLR ROTEX", *Proc. of the IEEE Int. Conf. on Robotics and* Automation, March 31-April 31987, pp. 443-449.

25. N. Papanikolopoulos, P.K. Khosla, and T. Kanade, "Vision and control techniques for robotic visual tracking", *Proc.* of *the IEEE Int. Conf. on Robotics and Automation*, 1991, pp. 857-864.

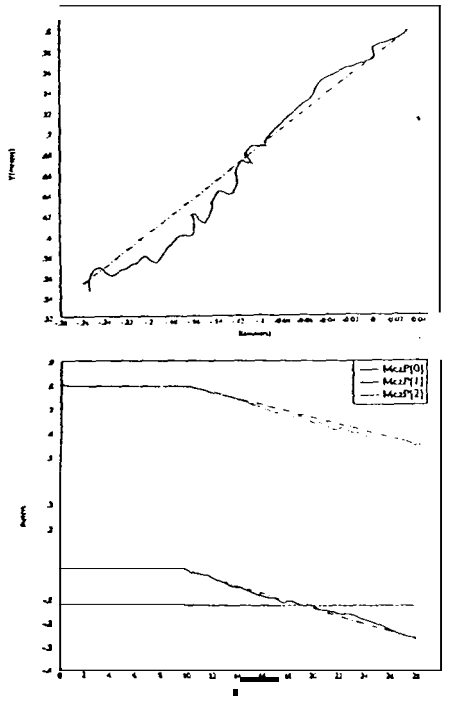

Figure 7: PI controller in conjunction with a Cartesian PD robotic controller with gravity compensation.

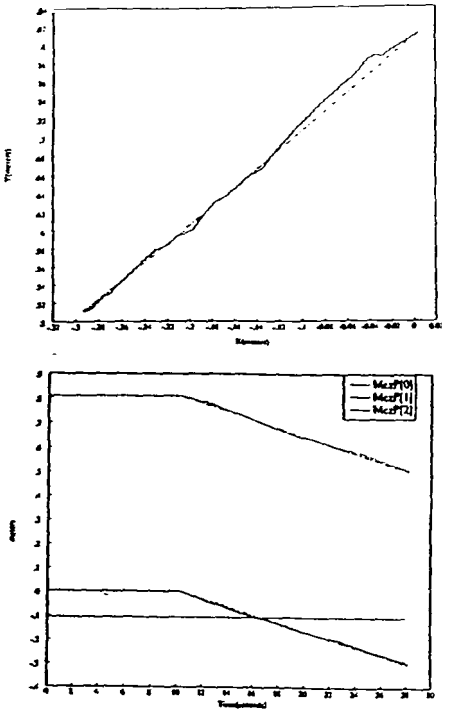

Figure 8: PI controller in conjunction with a Cartesian computed torque robotic controller.

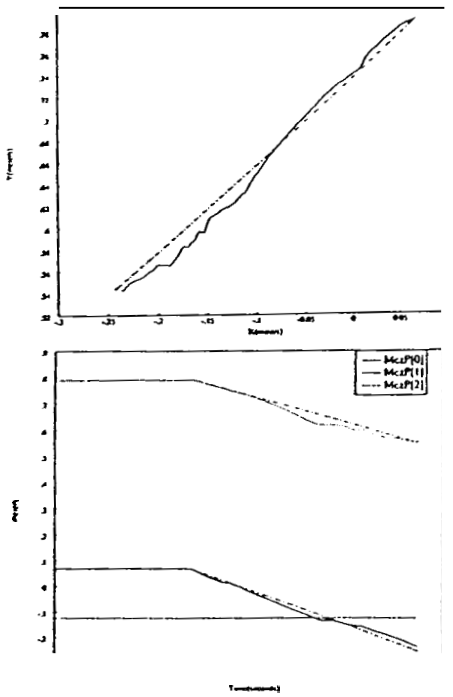

Figure 9: Pole assignment (poles at 0.8) controller in conjunction with a artesian PD robotic controller with gravity cornpensation.

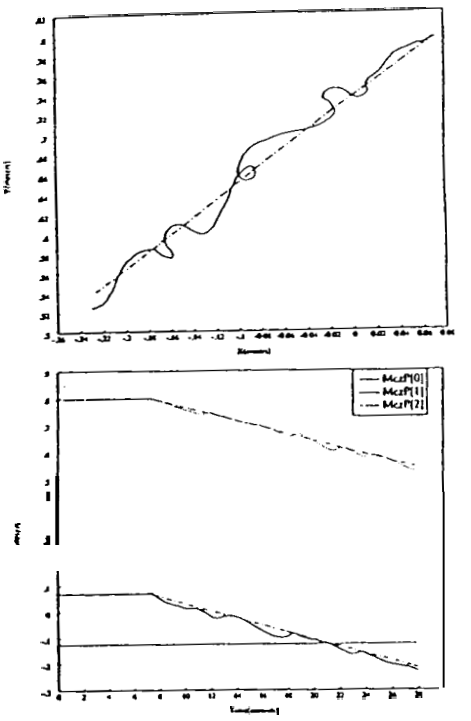

Figure 11: Pole assignment (poles at 0.2) controller in conjunction with a artesian PD robotic controller with gravity compensation.

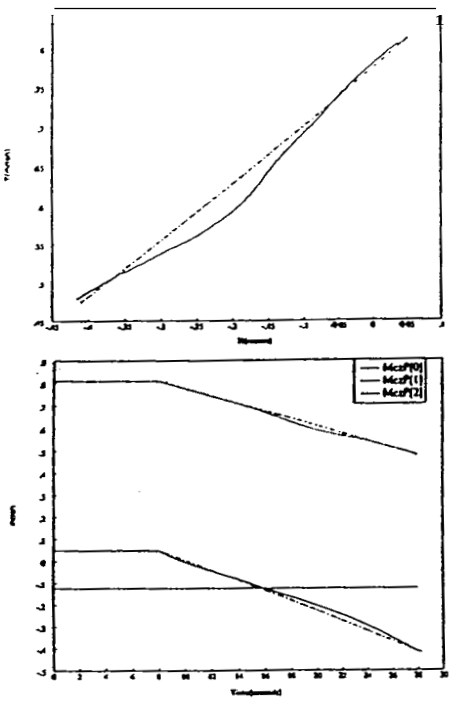

Figure 10: Pole assignment (poles at 0.2) controller in conjunction with a **cartesian** PD robotic controller **with gravity** compensation.

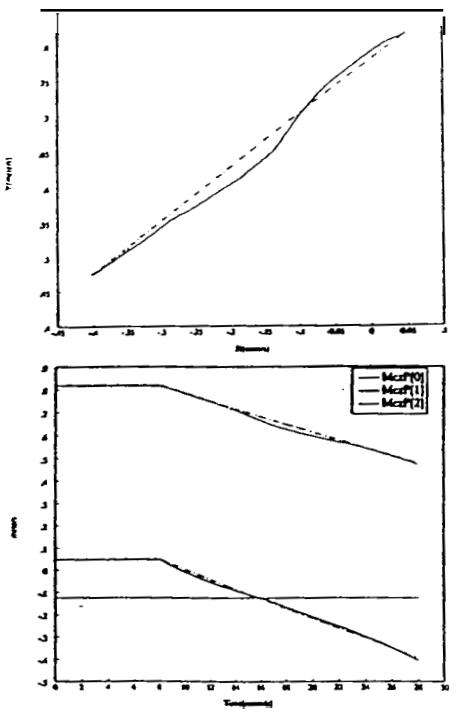

Figure 12: LQG controller in conjunction with a cartesian PD robotic controller with gravity compensation.

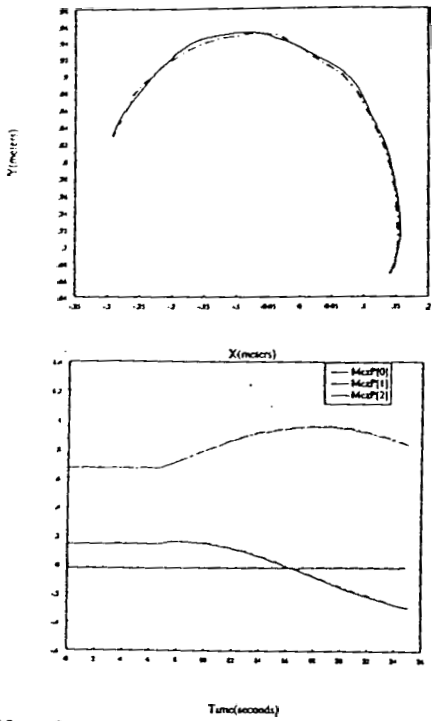

Figure 13: LQG controller in conjunction with a cartesian PD robotic controller with gravity compensation.

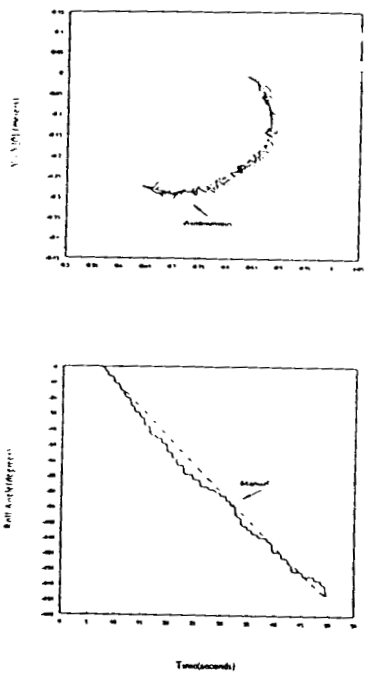

Figure 15: LQG control and the human operator (2-D). *MezP*[0] and *MezP*[1] are commanded by autonomous visual servoing modules while the roll angle is commanded by manual control (shared control).

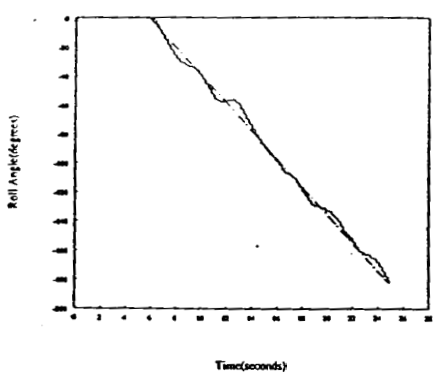

Figure 14: Roll trajectory of the robot end-effector and the object in the previous example.