Proceedings of the 1991 IEEE
International Conference on Robotics and Automation
Sacramento, California - April 1991

VISION AND CONTROL TECHNIQUES FOR ROBOTIC VISUAL TRACKING

N. Papanikolopoulos, P. K. Khosla, and T. Kanade

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

Abstract

In this paper, we present algorithms for robotic (hand-eye configuration) real-time visual tracking of arbitrary 3-D objects traveling at unknown velocities in a 2-D space. We formulate the problem of visual tracking as a problem of combining control with computer vision. We present a mathematical formulation that is general enough to be extended to the problem of tracking 3-D objects 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 performance of the proposed algorithms has been tested on a real system, the CMU DDArm II.

1. 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.

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]. In addition, some research [3, 4] 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, 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.

In this paper, we present algorithms that address the real-time robotic visual tracking (hand-eye configuration) of 3-D objects 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 system's theory point of view. This facilitates the extension of the algorithm to 3-D space. 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 combined with an appropriate control scheme to calculate the required motion of the robotic system. The control schemes that we have used range 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, 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

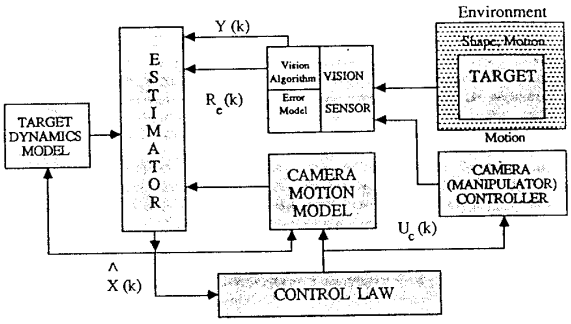

Figure 1: Architecture of the Robotic Visual Tracking/Servoing System

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 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 appropriate control law with regards to the noise level of the measurements are discussed in Section 4. Section 5 describes the hardware configuration of our experimental testbed, DDArm II. Simulation and experimental results are presented in Section 6. Finally, in Section 7, the paper is summarized.

2. Optical Flow

The optical flow will be the basis for the computation of the robot's driving signals. This section will 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 visual 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* $\mathbf{P}$ with coordinates (X_s, Y_s, Z_s) in R_s projects onto a point $\mathbf{p}$ 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 the camera frame R_s. The optical flow equations are [1]:

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

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

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 of much research and many algorithms have been proposed [5, 6, 7, 8]. For accuracy reasons, we use a matching based technique [9] also known as the sum-of-squared differences (SSD) optical flow. For every point $\mathbf{p_A}=(x_A, y_A)$ in image A we want to find the point $\mathbf{p_B}=(x_A+u, y_A+v)$ to which the point $\mathbf{p_A}$ moves in image B. We assume that the intensity in the neighborhood L of $\mathbf{p_A}$ remains almost constant, that the point $\mathbf{p_B}$ is within an area S of $\mathbf{p_A}$, and that velocities are normalized by time T to get the displacements. Thus, for the point $\mathbf{p_A}$ the SSD estimator selects the displacement $\mathbf{d}=(u, v)$ that minimizes the SSD measure:

$$e(\mathbf{p_A}, \mathbf{d})=\sum_{m,n\in N}[I_A(x_A+m, y_A+n) - I_B(x_A+m+u, y_A+n+v)]^2 \tag{3}$$

where $u, v \in S$, and N is an area around the pixel we are interested in. This technique can be improved using sub-pixel fitting and multi-grid techniques at the cost of increasing the computational complexity. The accuracy can also be improved by selecting an appropriate small area N and by having velocity fields with few quantization levels.

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 efficient confidence measure should recognize errors that are due to homogeneous areas and occlusion boundaries. Anandan [9] tried to develop 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. 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. 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 the confidence measure proposed in [9] is that the computation of the discrete second derivatives is ill-conditioned. 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 which 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. [10] have computed the variance in the estimate of one-dimensional displacement. The computation is based on a parabolic fit to the SSD curve. We propose 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 [11]. Thus, we 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 introduce two new

*Bold symbols denote vectors or matrices.

confidence measures which have been used in the experiments [11]. 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 principal 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_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 and computing the arithmetic means for the u and v. The techniques discussed above can be extended from black/white images to color images. In [11], 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

Consider a target 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 S_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 S_w stationary in an area S_o around the origin of the image frame". It is assumed that at initialization of the tracking process, the area S_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 "find the camera rotation (R_x, R_y) with respect to the camera frame that keeps S_w stationary in an area S_o around the origin of the image frame". The second definition of the visual tracking problem does not require the computation of the depth Z_s 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:

$$u((k+1)T) \approx u(kT) + u_c((k-d)T) \tag{9}$$

$$v((k+1)T) \approx v(kT) + v_c((k-d)T) \tag{10}$$

where $u_c((k-d)T), v_c((k-d)T)$ are the components of the optical flow induced by the tracking motion of the camera, and d is the delay factor. For the time being, the delay factor is assumed to be zero. Equations (9) and (10) 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. To keep the notation simple and without any loss of generality, equations (9) and (10) will be used with k and $(k+1)$ instead of kT and $(k+1)T$ respectively.

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 with $T_x(k)$ and $T_y(k)$ is:

$$u_c(k) = -\frac{T_x(k)}{Z_s} \quad , \quad v_c(k) = -\frac{T_y(k)}{Z_s} \tag{11}$$

We assume that for 2-D visual tracking the depth Z_s 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:

$$u_c(k) = R_x(k)\,x(k)\,y(k) - R_y(k)\,[x^2(k)+1] \tag{12}$$

$$v_c(k) = R_x(k)\,[y^2(k)+1] - R_y(k)\,x(k)\,y(k) \tag{13}$$

Equations (9) and (10) can be transformed after some simple calculations to state-space form as (the inaccuracies of the model are modeled as white noise):

$$\mathbf{x}(k+1) = \mathbf{A}\,\mathbf{x}(k) + \mathbf{B}\,\mathbf{u}_c(k) + \mathbf{E}\,\mathbf{d}(k) + \mathbf{H}\,\mathbf{v}(k) \tag{14}$$

where** $\mathbf{A} = \mathbf{H} = \mathbf{I}_2$, $\mathbf{B} = \mathbf{E} = T\,\mathbf{I}_2$, $\mathbf{x}(k) \in R^2$, $\mathbf{u}_c(k) \in R^2$,

**The symbol $\mathbf{I}_n$ denotes the identity matrix of order n.

$\mathbf{d}(k) \in R^2$ 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 $\mathbf{z}(k)=(z_1(k), z_2(k))^T$ is given by:

$$\mathbf{z}(k)=\mathbf{C}\,\mathbf{x}(k)+\mathbf{w}(k) \tag{15}$$

where $\mathbf{w}(k)$ is a white noise vector, and $\mathbf{C}=\mathbf{I}_2$. The measurement vector is computed using the SSD algorithm described in Section 2.

The same model can be used for keeping the feature point stationary in an area S_r different from the origin. Assume (r_x, r_y) is the center of this area S_r. A model similar to the one previously mentioned can be derived by transforming the state variables $x(k)$ and $y(k)$ to a new pair of state variables, $x_N(k)$ and $y_N(k)$. The new state variables are $x_N(k)=x(k)-r_x$ and $y_N(k)=y(k)-r_y$. The matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{E}, \mathbf{H}$ 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.

Consider a target that moves in a plane which is perpendicular to the optical axis of the camera. The projection of the target on the image plane is the area S_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 S_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 (a formal derivation is given in [11]):

$$\mathbf{x}(k+1)=\mathbf{A}\,\mathbf{x}(k)+\mathbf{B}\,\mathbf{u}_c(k)+\mathbf{E}\,\mathbf{d}(k)+\mathbf{H}\,\mathbf{v}(k) \tag{16}$$

where $\mathbf{A}=\mathbf{H}=\mathbf{I}_3$, $\mathbf{B}=\mathbf{E}=T\,\mathbf{I}_3$, $\mathbf{x}(k) \in R^3$, $\mathbf{u}_c(k) \in R^3$, $\mathbf{d}(k) \in R^3$ 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 $\mathbf{z}(k)=(z_1(k), z_2(k), z_3(k))^T$ is given by:

$$\mathbf{z}(k)=\mathbf{C}\,\mathbf{x}(k)+\mathbf{w}(k) \tag{17}$$

where $\mathbf{w}(k)$ is a white noise vector and $\mathbf{C}=\mathbf{I}_3$. 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 target stationary in both cases 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 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 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.

A simple technique for the elimination of the disturbances and of the noise of the system described previously, is the proportional-integral control technique (PI regulator). An obvious effect of the 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 is less stable than the original or can even become unstable if the gain matrices are not properly selected.

A more general technique for the design of the controller of the visual tracking system is the closed-loop pole assignment technique. Let the discrete-time description of the system be:

$$\overline{\mathbf{A}}_D(q^{-1})\,\mathbf{z}(k)=\overline{\mathbf{B}}_D(q^{-1})\,\mathbf{u}_c(k) \quad + \quad \overline{\mathbf{C}}_D(q^{-1})\,\mathbf{w}(k) \quad k \geq 0 \tag{18}$$

where $\mathbf{w}(k)$ is the white noise vector that corrupts the measurement vector $\mathbf{z}(k)$, and q^{-1} is the backward shift operator. The above model is called the *autoregressive moving-average model with auxiliary input* (ARMAX). The mathematical model of equations (14) and (15) can be transformed to an ARMAX model. We want to design a control law that places the closed-loop poles of the system at some desired positions. 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 presents large oscillations. Thus, we prefer to put the poles at locations that guarantee stability and a good transient response. Hersh [12] has described a robust control law that is based on the principles of the pole assignment method. If we neglect the stochastic term of the ARMAX model, then we have a *deterministic autoregressive moving average* (DARMA) model. Under certain assumptions, we can work with a simple DARMA model. These assumptions are: a) the disturbances are deterministic and constant, and b) the noise terms are neglected. Even though these assumptions are based on over-simplification of the problem, proper selection of the closed-loop poles can lead to a robust and efficient controller.

A more complex method for the control of the visual tracking system is a modified version of the LQG (Linear Quadratic Gaussian) control technique. The control law is given by

$$\mathbf{u}_c(k) = -\mathbf{G}\hat{\mathbf{x}}(k) - \hat{\mathbf{d}}(k) \quad (19)$$

where $\hat{\mathbf{d}}(k)$ is the estimated value of the disturbance vector $\mathbf{d}(k)$ and $\hat{\mathbf{x}}(k)$ is the estimated value of the state vector $\mathbf{x}(k)$. $\mathbf{G}$ is the unique symmetric positive definite solution of a matrix algebraic Ricatti equation. The vectors $\hat{\mathbf{x}}(k)$ and $\hat{\mathbf{d}}(k)$ can be computed by a steady-state or time-varying Kalman filter [13]. In particular, we design an observer of the disturbances and of the states. The time-invariant steady state Kalman filter can be implemented easily and does not require a large number of calculations. Instead of the steady state Kalman filter, one can use the time-varying discrete Kalman filter which constantly updates the Kalman gain matrix [14]. This improves the performance of our observer but it is computationally more expensive than the time-invariant Kalman filter. 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 an object. In the first case, $T_x(k)$ and $T_y(k)$ are computed by the equations (11). In the second case, we have to solve the system of equations (12) and (13) and calculate $R_x(k)$ and $R_y(k)$. 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 [10]. 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 [15].

After the computation of $u_{ci}(k)$ signals 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 next Section describes the hardware configuration of our experimental testbed (CMU DDArm II robotic system).

5. Hardware Configuration

The vision module of the CMU DDArm II system is a part of a bigger hardware environment. This hardware environment runs under the CHIMERA II [16] real-time programming environment. 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 3 processes: a) **Vision process** which does all the image processing calculations and has a period of 150 ms, b) **Interpolation process** which reads the data from the vision system, interpolates the data and sends the reference signals to the robot cartesian controller, and c) **Robot controller process** which drives the robot and has a period of 3.33 ms. In the next Section, we describe the simulation and experimental results.

6. Simulation and Experimental Results

6.1. 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. 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 767x767 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 which 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. The examples shown in Fig. 2-4 are implemented with 10% white noise. The object's centroid trajectories are shown by the solid lines. The focal length of the camera is unity. Distortion and effects of camera sampling are neglected. The depth when needed, is assumed to be known. Without loss of generality, we assume $T = 1$.

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 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 Fig. 2-4. We implemented a time-varying discrete Kalman filter with reinitialization each time the error is greater than a constant ε. 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 Fig. 2-4 are done by using a steady-state discrete Kalman filter. In Fig. 2-4, 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. 2, one big window is used and the object's speed has no roll component. In Fig. 3-4, multiple small

windows are used. In the first case, the size of the 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 14x14 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. Because of this, 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 would be. 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. 2 shows that when abrupt changes happen, there is an increase in the error. The largest error is around 10 pixels. In Fig. 2, 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 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. Each one of these windows is now 10x10 and are spread out in the image plane. For each window, we compute a confidence measure that reflects the accuracy of the measurements that the window provides. We tried three different confidence measures that were proposed in Section 2. The results of our implementation can be seen in Fig. 3-4. 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 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 DDArm II system.

6.2. Experiments

A number of experiments were performed on the CMU DDArm II robotic system. 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 $Z_s = 680$mm). 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 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 objects used in the tracking examples are books, pencils, and generally, items with distinct features. 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 a previous Section. Currently, four features are used and the size of the attached windows is 10x10.

The gains for the controllers are the same as the ones used in the simulation. The experimental results are plotted in Fig. 5-10 where the dotdashed trajectories correspond to the trajectories of the center of mass of the moving objects. The vector Mez_P represents the position of the end-effector in the world frame. The experimental results lead to some interesting observations. The computed torque cartesian scheme provides better performance than the simple PD with gravity compensation scheme. This is apparent from Fig. 5 and 6. Both schemes require high values for the gain 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 DDArm 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 relation of the quantization error to the displacements. When the displacement is comparable with the quantization error of the vision algorithm, controllers with the poles near to zero fail as in Fig. 8. On the other hand, when the displacement is larger than the quantization error, a controller with the poles near to zero has a better performance (Fig. 7). These experiments are done with the same object (book) and the same features. In the simulation results, the previous observation is not apparent. 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 LQG (Fig. 9) and the time-varying LQG (Fig. 10) 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 by 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. Due to the noisy measurements, the LQG 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 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.

7. 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. 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 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. The selection of the most efficient controller 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. Experimental results show that the methods are quite accurate, robust and promising.

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.

8. 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.

References

1. D. Tsakiris, "Visual tracking strategies", Master's thesis, Department of Electrical Engineering, University of Maryland, 1988.
2. R.C. Luo, R.E. Mullen Jr., and D.E. Wessel, "An adaptive robotic tracking system using optical flow", *Proc. of the IEEE Intern. Conf. on Robotics and Automation*, 1988, pp. 568-573.
3. J.T. Feddema, C.S.G. Lee, and O.R. Mitchell, "Automatic selection of image features for visual servoing of a robot manipulator", *Proc. of the IEEE Intern. Conf. on Robotics and Automation*, May 1989, pp. 832-837.
4. L.E. Weiss, A.C. Sanderson, and C.P. Neuman, "Dynamic sensor-based control of robots with visual feedback", *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 5, October 1987, pp. 404-417.
5. B.K.P Horn and B.G. Schunck, "Determining optical flow", *Artificial Intelligence*, Vol. 17, 1981, pp. 185-204.
6. R.Y. Tsai and T.S. 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.
7. S. Ullman, *The interpretation of visual motion*, MIT Press, 1979.
8. D.J. Heeger, "Depth and flow from motion energy", *Science*, Vol. , No. 86, 1986, pp. 657-663.
9. P. Anandan, "Measuring visual motion from image sequences", Tech. report COINS-TR-87-21, COINS Department, University of Massachusetts, 1987.
10. 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.
11. N. Papanikolopoulos, P. Khosla, and T. Kanade, "Robotic visual tracking: Theory and experiments", Tech. report , Carnegie Mellon University, The Robotics Institute, 1990.
12. 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.
13. A. Gelb, *Applied optimal estimation*, MIT Press, Cambridge, 1974.
14. A.P. Sage, *Optimum systems control*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1968.
15. N. Papanikolopoulos, P. Khosla, and T. Kanade, *Adaptive robotic visual tracking*, Accepted to the American Control Conference, , 1991.
16. D.B. Stewart, D.E. Schmitz, and P.K. Khosla , "Implementing real-time robotic systems using CHIMERA II", *Proc. of 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, May 1990, pp. 598-603.

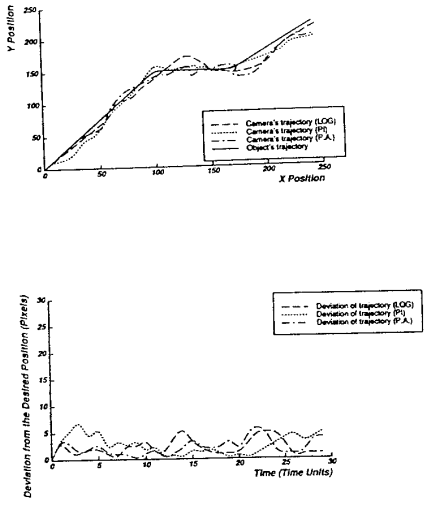

Figure 2: Example with measurements from one big window (Simulation).

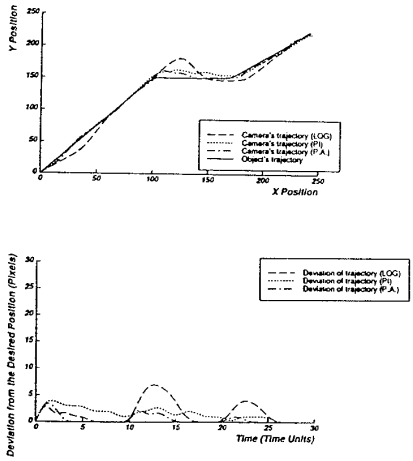

Figure 3: Example with measurements from multiple windows (Simulation).

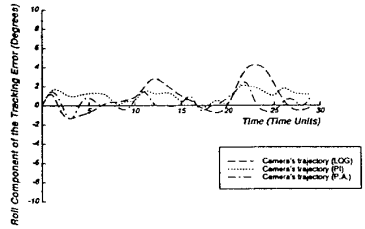

Figure 4: Roll component of the tracking error for the previous example. The measurements are derived from multiple windows (Simulation).

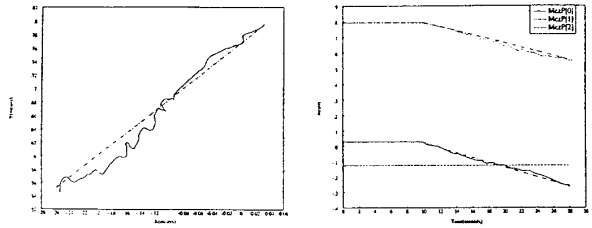

Figure 5: PI controller in conjunction with a cartesian PD robotic controller with gravity compensation (Experimental).

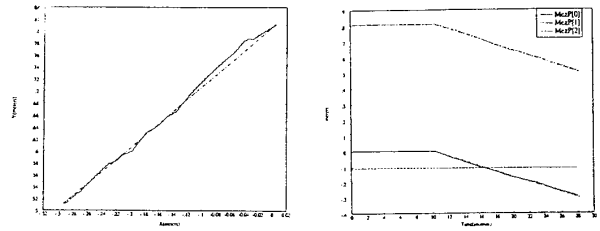

Figure 6: PI controller in conjunction with a cartesian computed torque robotic controller (Experimental).

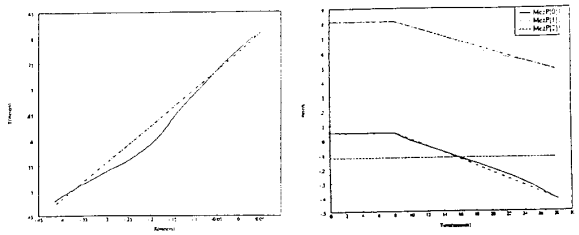

Figure 7: Pole Assignment (poles at 0.2) controller in conjunction with a cartesian PD robotic controller with gravity compensation (Experimental).

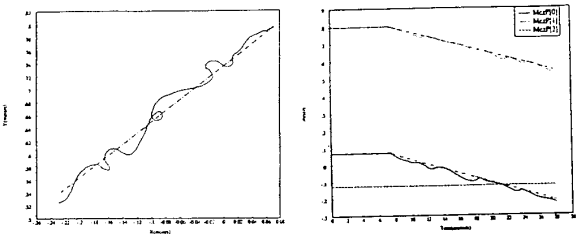

Figure 8: Pole Assignment (poles at 0.2) controller in conjunction with a cartesian PD robotic controller with gravity compensation (Experimental).

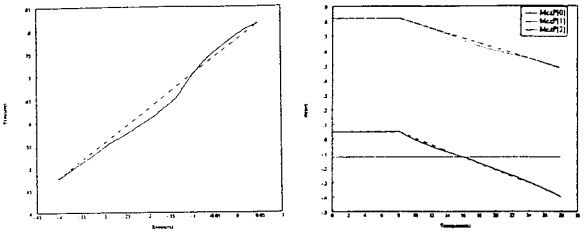

Figure 9: LQG controller in conjunction with a cartesian PD robotic controller with gravity compensation(Experimental).

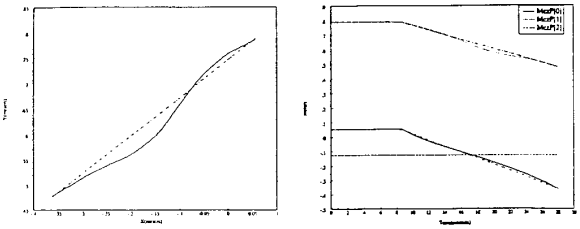

Figure 10: Time varying LQG controller in conjunction with a cartesian PD robotic controller with gravity compensation (Experimental).