Presented at 30th IEEE Conf. on Deicision and Control (CDC), December 11-13, 1991, Brighton, United Kingdom

FEATURE BASED ROBOTIC VISUAL TRACKING OF 3-D TRANSLATIONAL MOTION

N. P. Papanikolopoulos and P. K. Khosla

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

Abstract

This paper addresses the problem of robotic visual backing of a target that moves in 3-D with translational motion. The camera is mounted on the end-effector of the robotic manipulator. We use simple SISO adaptive controllers in conjunction with a cartesian robotic controller for successful tracking. Our scheme does not need explicit computation of the target's relative depth with respect to the camera frame In addition, the relative motion of the target with respect to the camera is measured by using a cross-correlation technique called Sum-of-Squared Differences (SSD) optical flow. Experimental and simulation results are presented to show the efficacy of the proposed approach.

Index Terms: Adaptive Controller, Corrdation, Cartesian Controller, Optical Flow, Visual Tracking, Visual Servoing.

1. Introduction

Operation in a constantly evolving environment requires continuous flow of sensory information that has to be interpreted and transformed into control commands to drive the robotic device Vision sensors enhance the flexibility of a robotic manipulator by making it possible to react to changes in the environment. For example, vision permits on-line correction of a preplanned trajectory, real-time obstacle avoidance and robust visual inspection On the other hand, the robotic visual servoing problem poses some difficulties The first difficulty is in the huge amount of vision information The second difficulty can be attributed to the noisy nature of this information. Vision sensors such as CCD cameras provide a lot of information but this information is corrupted with noise. The noise can be reduced with elaborate vision algorithms that are computationally expensive and thus, inefficient for real-time operation. In addition, if the introduced computational delay is longer than the time in which the environment evolves, the control commands will turn out to be invalid.

In this paper, we choose to deal with a smaller problem in this challenging area. This problem can be defined as: "move the manipulator (the camera is mounted on the end-effector) in such a way that the projection of a moving object is always at the desired location in the image". We propose algorithms that address the real-time robotic visual tracking of objects that move in 3-D with translational motion. To achieve this objective, computer vision techniques for detection of motion are combined with appropriate control strategies to compute the actuating signal for driving the manipulator. The problem is formulated from system theory point of view. An advantage of this approach is that the dynamics of the robotic device can be taken into account without changing the basic structure of the system In order to circumvent the need to explicitly compute the depth map of the target, adaptive control techniques are proposed. Simulation and experimental results are presented to show the strengths and the weaknesses of the proposed approach.

The organization of this paper is as follows: Section 2 gives an outline of the vision techniques (optical flow, SSD surface) used for the computation of the object's motion parameters. The mathematical formulation of the visual tracking problem is described in Section 3. The control strategies are discussed in Section 4. The simulation and the experimental results are presented in Section 5. Finally, in Section 6, the paper is summarized.

2. Measurement of Features' Motion

We assume a pinhole camera model with a frame $\{R_s\}$ attached to it. In addition, it is assumed that the projection is perspective and that the focal length is 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) given by:

$$x = X_s/Z_s \quad and \quad y = Y_s/Z_s. \tag{1}$$

Any displacement of a rigid object can be described by a rotation about an axis through the origin and a translation. If the angle of this rotation is small, the rotation can be characterized by three independent rotations about the X, Y, and Z axes. Let us assume that the camera moves in a static environment with a translational velocity $T = (T_x, T_y, T_z)^T$ and with an angular velocity $R = (R_x, R_y, R_z)^T$ with respect to the camera frame $\{R_s\}$. The optical flow equations are [1, 2]:

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

$$v = [\, y\frac{T_z}{Z_s} - \frac{T_y}{Z_s} \,] + [(1+y^2)R_x - xyR_y - xR_z] \tag{3}$$

where $u = \dot{x}$ and $v = \dot{y}$. u and v are also known as the optical flow measurements. The computation of u and v has been the focus of much research and many algorithms have been proposed [3, 4, 5]. For accuracy reasons, we use a matching based technique [6] 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 corresponding point $p(k) = (x(k-1) + u,\ y(k-1) + v))^T$ to which the point $p(k-1)$

moves in image (k). We assume that the intensity values in a neighborhood N of $p(k-1)$ remain almost constant over time, that the point $p(k)$ is within an area Ω of $p(k-1)$, and that velocities are normalized by time T to get the displacements. Thus, given a point $p(k-1)$, the SSD algorithm selects the displacement $\mathbf{d}=(u,v)^T$ that minimizes the SSD measure:

$$e(p(k-1),\mathbf{d}) = \sum_{m,n \in N} [I_{k-1}(x(k-1)+m, y(k-1)+n) - I_k(x(k-1)+m+u, y(k-1)+n+v)]^2 \quad (4)$$

where $u, v \in \Omega$, N is an area around the pixel we are interested in, and $I_{k-1}(\cdot)$, $I_k(\cdot)$ are the intensity functions in images $(k-1)$ and (k), respectively. Different values of the SSD measure create

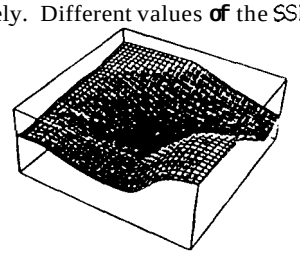

Figure 1: SSD surface that corresponds to a corner point.

a surface called the *SSD surface*. The shape of the SSD surface depends on the position of the feature. The accuracy of the displacement vector measurements can be improved by using multiple windows. The selection of the best windows is based on some confidence measures which capture the special characteristics of the SSD surface. A SSD surface that corresponds to a corner point (one of the best features which can be used) has a sharp minimum at the best match (Fig. 1). 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 has multiple local minimum in the direction of this edge. Finally, a SSD surface that corresponds to a feature point that belongs to an homogeneous area (these feature points provide inaccurate data) has multiple local minimum in all the directions. More details about these Confidence measures can be found in [1].

The next step in our algorithm involves the use of these measurements in the visual tracking process. These measurements should be transformed into control commands to the robotic system Thus, a mathematical model for this transformation must be developed. In the next Section, we present the mathematical model for the visual tracking problem

3. Modeling of the Visual Tracking Problem

Consider a target, with a frame (RJ attached to it, that moves with pure translational 3-D motion The projection of the target on the image plane is .the area Ω_w in the image plane. The problem of 3-D visual tracking of a single object with translational motion can be defined as: "find the camera translation (T_x, T_y, T_z) with respect to the camera frame that keeps Ω_w stationary". It is assumed that at initialization of the tracking process the camera frame $\{R_c\}$ and the target frame $\{R_t\}$ have their Z axes aligned. If M $(M \geq 2)$ feature points of a moving target are selected to be used for visual tracking, it is assumed that the projections on the image plane of at least two of the feature points are not the same. The optical flow of the projection of each one of the M feature points on the image plane is given by:

$$u^{(j)}(k+1) \simeq u^{(j)}(k) + u_c^{(j)}(k) \qquad j=1,\ldots,M \quad (5)$$

$$v^{(j)}(k+1) \simeq v^{(j)}(k) + v_c^{(j)}(k) \qquad j=1,\ldots,M \quad (6)$$

where the index j represents each one of the M feature points, $u^{(j)}(k), v^{(j)}(k)$ are the components of the optical flow for each feature point, and $u_c^{(j)}(k), v_c^{(j)}(k)$ are the components of the optical flow induced by the tracking motion of the camera. It should be mentioned that the depth of each feature point $Z_s^{(j)}$ is now time-varying and thus, we represent its value at instant k by $Z_s^{(j)}(k)$. By using (2) and (3) with $R_x = R_y = R_z = 0$, we obtain:

$$u_c^{(j)}(k) = -\frac{T_x(k) - x^{(j)}(k) T_z(k)}{Z_s^{(j)}(k)} \qquad j=1,\ldots,M \quad (7)$$

$$v_c^{(j)}(k) = -\frac{T_y(k) - y^{(j)}(k) T_z(k)}{Z_s^{(j)}(k)} \qquad j=1,\ldots,M \quad (8)$$

and substitution in (5) and (6) results in:

$$u^{(j)}(k+1) \simeq u^{(j)}(k) + \frac{S_x^{(j)}(k)}{Z_s^{(j)}(k)} \qquad j=1,\ldots,M \quad (9)$$

$$v^{(j)}(k+1) \simeq v^{(j)}(k) + \frac{S_y^{(j)}(k)}{Z_s^{(j)}(k)} \qquad j=1,\ldots,M \quad (10)$$

where $S_x^{(j)}(k)$ and $S_y^{(j)}(k)$ are defined as:

$$S_x^{(j)}(k) = -T_x(k) + x^{(j)}(k) T_z(k) \qquad j=1,\ldots,M \quad (11)$$

$$S_y^{(j)}(k) = -T_y(k) + y^{(j)}(k) T_z(k) \qquad j=1,\ldots,M \quad (12)$$

It is known that:

$$u^{(j)}(k+1) = \frac{x^{(j)}(k+1) - x^{(j)}(k)}{T} \qquad j=1,\ldots,M \quad (13)$$

$$v^{(j)}(k+1) = \frac{y^{(j)}(k+1) - y^{(j)}(k)}{T} \qquad j=1,\ldots,M \quad (14)$$

where T is the time between *two* consecutive images. Then, we substitute in (9) and (10) for $u^{(j)}(k+1)$ and $v^{(j)}(k+1)$ from (13) and (14) to obtain:

$$x^{(j)}(k+1) \simeq x^{(j)}(k) + T\frac{S_x^{(j)}(k)}{Z_s^{(j)}(k)} + T u^{(j)}(k) \qquad j=1,\ldots,M \quad (15)$$

$$y^{(j)}(k+1) \simeq y^{(j)}(k) + T\frac{S_y^{(j)}(k)}{Z_s^{(j)}(k)} + T v^{(j)}(k) \qquad j=1,\ldots,M \quad (16)$$

Equations (15)-(16) can be augmented by introducing white noise terms to compensate for inaccuracies in modeling. In addition, we can subtract the coordinates of the desired position of the image features from both sides of the equations (15)-(16) without changing their structure. The control strategies that keep the target stationary are discussed in detail in the next Section.

4. Adaptive Visual Control

Adaptive control techniques can be used for the visual tracking of either a feature or an object when the depth information is not directly available. Adaptive control techniques are used only for recovering $T_z(k)$, $T_y(k)$, and $T_z(k)$. The block diagram of our scheme is shown in Fig. 2. Based on the *certainty equivalence* approach [7], the adaptive control techniques use the estimated and not the actual values of the system's parameters. A large number of algorithms can be generated, depending on which parameter estimation scheme is used and which control law is chosen. The rest of the Section will be devoted to the detailed

description of the system model, the estimation scheme, and finally, the computation of the control signal. Equations (15)-(16)

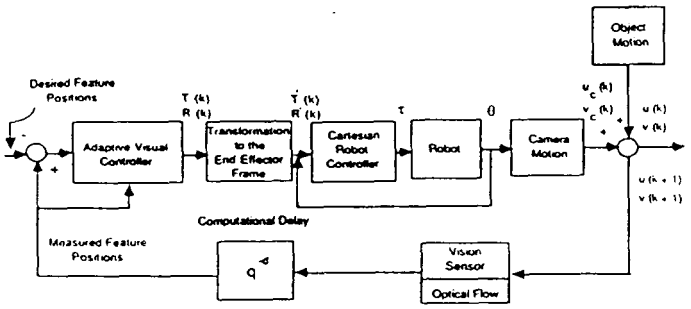

Figure 2: Structure of the adaptive robotic visual tracking scheme.

(index j is neglected for simplicity) can be transformed with the addition of noise terms ($n_{wx}(k)$, $n_{wy}(k)$ are white noise terms) into the following equations for a single feature point:

$$x(k+1)=x(k)+b_x S_x(k)+Tu(k) \tag{17}$$

$$y(k+1)=y(k)+b_y S_y(k)+Tv(k) \tag{18}$$

$$u(k+1)=u(k)+T n_{wx}(k) \tag{19}$$

$$v(k+1)=v(k)+T n_{wy}(k) \tag{20}$$

where coefficients b_x and b_y depend on the values of the depth Z_s and of the sampling interval T. For M feature points, we have M sets of the equations (17)-(20).

The next step in our modeling is to remove the terms $u(k)$ and $v(k)$ from our formulation. In order to achieve that, we derive two time-varying SISO ARMAX models by subtractions and additions of the expressions for $x(k)$, $x(k-1)$ and $x(k-2)$, and $y(k)$, $y(k-1)$ and $y(k-2)$, respectively. For each feature, the corresponding two time-varying SISO ARMAX models are:

$$A_i(q^{-1})y_i(k)=q^{-d}B_i(q^{-1})u_{ci}(k)+C_i(q^{-1})w_i(k) \quad k\geq 0 \quad i=1,2 \tag{21}$$

where

$$A_i(q^{-1})=1+a_{i1}q^{-1}+a_{i2}q^{-2} \quad i=1,2$$
$$B_i(q^{-1})=b_{i0}+b_{i1}q^{-1} \quad i=1,2$$
$$C_i(q^{-1})=1+c_{i1}q^{-1}+c_{i2}q^{-2} \quad i=1,2$$

The values of the coefficients $a_{i1}, a_{i2}, b_{i0}, b_{i1}, c_{i1}, c_{i2}$ depend on the values of T, b_x, and b_y. The noise sequences $w_i(k)$ are assumed to satisfy the assumptions:

$$E\{w_i(k)\mid F_{k-1}^i\}=0 \quad E\{w_i^2(k)\mid F_{k-1}^i\}=\sigma_i^2 \quad i=1,2$$

where the symbol $E\{X\}$ denotes the expected value of the random variable X and F_{k-1}^i is the sigma algebra generated by the past measurements and the past control inputs up to time $k-1$. The subscript i corresponds to the two SISO ARMAX models, the scalar input $u_{ci}(k)$ now represents either $S_x(k)$ or $S_y(k)$, and the scalar $y_i(k)$ corresponds to the measured deviation of the feature point from its desired position in one of the X or Y directions. It can be shown that $b_{10}=-b_{11}=b_x$, and $b_{20}=-b_{21}=b_y$. For the time being, it is assumed that $d=1$.

The next step in our modeling is the formulation of an efficient model for estimation. The optimal one-stepahead predictor [7] which gives an optimal prediction of the output based on past measurements and inputs, has the form ($d=1$):

$$C_i(q^{-1})y_i^*(k+1\mid k)=A_i^*(q^{-1})y_i(k)+B_i^*(q^{-1})u_{ci}(k) \quad k\geq 0 \quad i=1,2 \tag{22}$$

where

$$B_i^*(q^{-1})=B_i(q^{-1}) \quad i=1,2$$
$$A_i^*(q^{-1})=q[C_i(q^{-1})-A_i(q^{-1})] \quad i=1,2$$
$$y_i^*(k+1\mid k)=E\{y_i(k+1)\mid F_k\}=y_i(k+1)-w_i(k+1) \quad i=1,2$$

$y_i^*(k\mid(k-1))$ denotes the optimal prediction of the output while the $\hat{y}_i(k)$ denotes the output of the adaptive predictor. Since the $y_i^*((k-1)\mid(k-2))$, $y_i^*((k-2)\mid(k-3))$,... are not directly available, the previously estimated values $\hat{y}_i(k-1)$, $\hat{y}_i(k-2)$... can be used instead. The $\hat{y}_i(k)$ are given by:

$$\hat{y}_i(k)=f_i^T(k-1)\hat{q}_i(k-1) \quad i=1,2 \tag{23}$$

where

$$f_i^T(k-1)=[y_i(k-1), y_i(k-2), \Delta u_{ci}(k-1), -\hat{y}_i(k-1), -\hat{y}_i(k-2)] \quad i=1,2$$

$$\hat{q}_i^T(k-1)=[\hat{\alpha}_{i1}(k-1), \hat{\alpha}_{i2}(k-1), \hat{b}_{i0}(k-1), \hat{c}_{i1}(k-1), \hat{c}_{i2}(k-1)] \quad i=1,2$$

The symbols $\{\hat{\alpha}_{i1}(k-1), \hat{\alpha}_{i2}(k-1)\}$, $\{\hat{b}_{i0}(k-1)\}$, and $\{\hat{c}_{i1}(k-1), \hat{c}_{i2}(k-1)\}$ denote the estimated values of the coefficients of the polynomials $A_i^*(q^{-1})$, $B_i^*(q^{-1})$ and $C_i(q^{-1})$, respectively. The $\Delta u_{ci}(k)$ can be computed from the equation $\Delta u_{ci}(k)=u_{ci}(k)-u_{ci}(k-1)$. The next steps are the on-line estimation of these parameters and the appropriate selection of the control law. A detailed description can be found in [2]. Experimental results [2] show that the best control law can be derived by a cost function that includes the control signal change:

$$J_i(k+d)=E[(y_i(k+d)-y_i^*(k+d))^2+\rho_i\Delta u_{ci}^2(k)\mid F_k^i] \quad i=1,2 \tag{24}$$

This control law (STRWDU) introduces an integrator in the system by weighting the control signal change. This is in agreement with the structural and operational characteristics of a robotic system A robotic system cannot track objects that have large changes in their image projections during the sampling interval T. In addition, there are upper limits in the robotic tracking ability. Based on the relations between the coefficients of $B_i^*(q^{-1})$, the modified one-step-ahead predictor [7] is given by

$$C_i(q^{-1})\frac{\omega}{b_{i0}^2+P_i}[y_i^*(k+1\mid k)-y_i^*(k+1)+\frac{\rho_i}{b_{i0}}\Delta u_{ci}(k)]=f_i'^T(k)q_i'(k)+\Delta u_{ci}(k) \quad i=1,2 \tag{25}$$

where $q_i'(k)$ is the vector that contains the coefficients of the polynomials:

$$\frac{b_{i0}}{b_{i0}^2+\rho_i}\{A_i^*(q^{-1}), \frac{\rho_i}{b_{i0}}[C_i(q^{-1})-1]q, -C_i(q^{-1})\} \quad i=1,2$$

and the $f_i'(k)$ is given by:

$$f_i'^T(k)=[y_i(k), y_i(k-1), \Delta u_{ci}(k-1), \Delta u_{ci}(k-2), y_i^*(k+1), y_i^*(k), y_i^*(k-1)] \quad i=1,2$$

The control law is:

$$\Delta u_{ci}(k)=-f_i'^T(k)\hat{q}_i'(k) \quad i=1,2 \tag{26}$$

The estimation scheme consists of the equations [7]:

$$\hat{q}_i'(k)=\hat{q}_i'(k-1)+\frac{1}{b_{i0}^{*}+\frac{Qi}{b_{i0}^{*}}}\ P_i(k-1)f_i'(k-1)e_i(k)\quad i=1,2 \tag{27}$$

$$P_i(k-1)=\frac{1}{\lambda_i(k-1)}\ [P_i(k-2)- -\frac{P_i(k-2)f_i'(k-1)f_i'^{T}(k-1)P_i(k-2)}{\lambda_i(k-1)+f_i'^{T}(k-1)P_i(k-2)f_i'(k-1)}\ ]\quad i=1,2 \tag{28}$$

$$e_i(k)=\frac{\rho_i}{b_{i0}^{*}}\ \Delta u_{ci}(k-1)+y_i(k)-y_i^{*}(k)\quad i=1,2 \tag{29}$$

The symbol b_{i0}^{*} denotes the initial estimate of the coefficient b_{i0} and the scalar $\lambda_i(k)=\overline{\lambda}_i$ is the forgetting factor which is selected to be 0.95. It is assumed that the $\hat{q}_i(0)$ is given and $P_i(-1)$ is any positive definite matrix P_{i0}. P_{i0} can be Seen as a measure of the confidence that we have in the initial estimate $\hat{q}_i(0)$. Accurate knowledge of the vector $\hat{q}_i(0)$ corresponds to a small covariance matrix P_{i0}. The above algorithm minimizes the cost function (24) with a modified ρ_i that we call $\rho_i'=\rho_i b_{i0}/b_{i0}^{*}$. Due to the fact that $y_i^{*}(k)=0\ \forall\ L>0$, the three last coefficients of the $q_i'(k)$ need not be computed. In total, eight parameters per feature point should be estimated on-line. This control scheme seems the most appropriate for the specific control problem that we have to solve The disadvantage of this type of controllers is the *integral* wind-up effect. This effect is more obvious in the presence of saturation in the input or output. *Our* algorithm takes care of this by switching off the integrator.

The next step of our algorithm is the computation of the $T_x(k)$, $T_y(k)$, and $T_z(k)$. From the adaptive controllers of each feature point, the $u_{ci}(k)$'s are computed each instant k. The $u_{ci}(k)$ signal is either the $S_x(k)$ signal or the $S_y(k)$ signal, defined by (11) or (12), respectively. Thus, we create a set of equations for M feature points which are compactly represented as:

$$\mathbf{H}(k)\mathbf{T}(k)=\mathbf{S}(k) \tag{30}$$

where the matrix $H(k)\in R^{2M\times 3}$ is

$$\mathbf{H}(k)=\begin{vmatrix} -1 & 0 & x^{(1)}(k) \\ 0 & -1 & y^{(1)}(k) \\ \cdots & & \cdots \\ \cdots & & \cdots \\ -1 & 0 & x^{(M)}(k) \\ 0 & -1 & y^{(M)}(k) \end{vmatrix}$$

the vector $T(k)=(T_x(k),T_y(k),T_z(k))^T$ is the translational velocity vector, and the vector $S(k)=(S_x^{(1)}(k),S_y^{(1)}(k),\ \ldots\ S_x^{(M)}(k),S_y^{(M)}(k))^T$ is composed of the signals computed by the adaptive controllers attached to each feature point. The translational velocity vector T can be computed as follows (least squares formulation):

$$T(k)=(H^T(k)H(k))^{-1}H^T(k)S(k) \tag{31}$$

The matrix $H^T(k)H(k)$ is invertible if at least two of the M feature points have different projections on the image plane (a formal proof is given in Appendix I).

Assume that the maximum tmnslational speed in 3-D which can be achieved by the robot is D_{max} and the Euclidean norm of the vector $T(k)$ is $\|T(k)\|$. Then, the vector $T(k)$ is transformed to the normalized vector $T'(k)$ with the following components:

$$T_x'(k)=T_x(k)\frac{D_{max}}{\|T(k)\|},\quad T_y'(k)=T_y(k)\frac{D_{max}}{\|T(k)\|},\quad T_z'(k)=T_z(k)\frac{D_{max}}{\|T(k)\|} \tag{32}$$

This normalization is necessary due to the fact that the manipulator cannot track high speeds successfully. In addition, this normalization affects the estimation scheme that we are using. To guarantee a good performance of the estimator, we must use normalized components of the translational velocity vector for the computation of the past input signals $\Delta u_{ci}(k)$. Thus, instead of using $\Delta u_{ci}(k)$ signals in the estimation process, we use signals $\Delta u_{ci}'(k)$ which are given by:

$$\Delta u_{c1}'(k)=-(T_x'(k)-T_x'(k-1))+(x(k)T_z'(k)-x(k-1)T_z'(k-1)) \tag{33}$$

$$\Delta u_{c2}'(k)=-(T_y'(k)-T_y'(k-1))+(y(k)T_z'(k)-y(k-1)T_z'(k-1)) \tag{34}$$

After the computation of the translational velocity vector $T'(k)$ with respect to the camera frame $\{R_c\}$, we transform it 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 use a cartesian PD robot control scheme with gravity compensation.

The next Section describes the simulation and the experimental results.

5. Simulation and Experiments

Our work involves both simulations and experiments. A number of simulations have been used for determining the efficiency of the developed algorithms. The objects used in the simulated tracking examples are 3-D solid objects, such as cubes, cylinders, and prisms. These objects move in front of a highly textured background. Each pixel's intensity is corrupted with white noise The focal length of the simulated camera is 7.5mm and without loss of generality, we assume $T=1$.

The example shown in Fig. 3 has been implemented with 10% white noise. The objects' center of mass trajectories are given with respect to the target frame $\{R_t\}$ whose Z axis at $k=0$ is aligned with the Z axis of the camera frame $\{R_c\}$. As mentioned earlier, the SSD optical flow technique is used for the computation of the measurement vector. The measurements are taken from multiple windows. Each one of these windows is 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. The example shown in Fig. 3 is used to test the efficiency of the adaptive control techniques. The average initial depth of the center of mass of the targets is 600mm. Assuming that the depth Z_c is unknown, the ELS algorithm (version with exponential data weighting) is used to estimate the values of the coefficients of the two SISO ARMAX models per feature point. In all the cases, $P_{i0}=I_4$ and the initial value of the vector $\hat{q}_i^T(k)$ is $\hat{q}_i^T(0)=[-1.0,1.0,1.0,1.0]$. The value of the scalar ρ_i is 0.2. Larger ρ_i means less emphasis on the minimization of the tracking error and more emphasis on the minimization of the control signal change. The disadvantage of the proposed scheme ($\rho_i\neq 0$) is that it exhibits errors of large magnitude when abrupt changes of the trajectories happen. The reason for this is that the change of the control signal is included in the cost function. Thus, the regulator cannot compensate adequately for abrupt changes. In addition, a larger tracking error appears in the Z direction. The reason is that the computation of

the Z component of the translational velocity vector $T(k)$ is sensitive to image noise. It should be mentioned that for the specific camera model and the specific positions of the features in the image plane, tracking error smaller than 5mm in the Z direction cannot be detected as a pixel displacement. Besides these disadvantages, it seems to be the most effiient adaptive control algorithm for robotic visual tracking. The next step of our work was to test the algorithms on a real system

A number of experiments were performed on the CMU DD Arm II (Direct Drive Arm II) system The hardware configuration of the CMU DD Arm II is described in [1]. The camera is mounted on the end-effcctor. The focal length of the camera is 7.5mm while the objects are moving on a ramp (initial depth of the object's center of mass with respect to the the camera frame $Z_s = 600mm$). The maximum permissible velocity of the end-effector is 10 cm/sec. The objective is to keep the projection of the target on the image plane stationary. The objects used in the tracking examples are books, pencils, and generally, items with distinct features. The user at initialization selects the features that must be tracked. Then, the system evaluates on-line the quality of the measurements, based on the confidence measures described in [1]. The same operation can be done automatically by a computer process that runs once and needs between 2 and 3 secs, depending on the size of the interest operators which are used. Currently, four features are used and the size of the attached windows is 10x10.

The gains of the controllers and the initial values of the parameters are the same as the ones used in the simulation. The experimental results are plotted in Fig. 4-5 where the dotdashed trajectories correspond to the trajectories of the center of mass of the moving objects. The *MezP* vector represents the position of the end-effector with respect to the world frame. In particular, in Fig. 4 and 5, the three dotdashed trajectories correspond to the trajectories of the center of mass of the object in 3-D. In Fig. 5, the vector $(X[0], Y[0], Z[0])^T$ denotes the initial position of the object or the manipulator's end-effcctor with respect to the world frame, while the vector $(X, Y, Z)^T$ denotes the trajectories in 3-D of the object or the manipulator's end-effector.

The experimental results lead to some interesting observations. The observed oscillations are due to the fact that the robotic controller (PD with gravity compensation) does not take into consideration the robot dynamics. In addition, the computation of the translational velocity vector $T(k)$ is extremely sensitive to image noise. Image noise can seriously affect the motion of the manipulator in the Z direction. In addition, tracking errors smaller than 5mm in the Z direction cannot be detected as a pixel displacement in the image plane. This is due to the specific camera model and the specific positions of the features in the image plane. The experimental results for this case are worse than the results provided by the simulations. The reason is that the large variations in the control signal create blurring in the images, and thus, errors in the visual measurements.

6. Conclusions

We have proposed an adaptive control scheme as a solution to the robotic visual tracking (eye-in-hand configuration) of an object that moves with 3-D translational motion. We first presented a mathematical formulation of the problem. The derived model is a major contribution of this 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 model is extended to 3-D by including a larger number of feature points. In addition, it can incorporate the robot dynamics. The next step was to introduce an adaptive control scheme for the case of inaccurate knowledge of some of the system's parameters (relative depth, noise model). This adaptive control scheme was implemented on our experimental testbed, the DD Arm II system Experimental real-time results show that the method is quite accurate, robust and promising. The extension of the algorithm to tracking of rotational motions, the explicit use of the robot dynamics, the integration of other sensory measurements in the scheme, and the use of the 3-D model of the target, arc issues currently being addressed.

7. Acknowledgements

This research was supported in part, by the Defense Advanced Research Projects Agency, through ARPA Order Number DAAA-21-89C-0001, and by Innovision Inc.

L Conditions for the nonsingularity of the matrix $(H^T(k)H(k))$

The matrix $(H^T(k)H(k))$ is given by:

$$H^T(k)H(k) = \begin{bmatrix} M & 0 & \Sigma_{13}(k) \\ 0 & M & \Sigma_{23}(k) \\ \Sigma_{31}(k) & \Sigma_{32}(k) & \Sigma_{33}(k) \end{bmatrix}.$$

where $\Sigma_{13}(k)$, $\Sigma_{31}(k)$, $\Sigma_{23}(k)$, $\Sigma_{32}(k)$, and $\Sigma_{33}(k)$ are defined as:

$$\Sigma_{13}(k) = \Sigma_{31}(k) = \sum_{l=1}^{l=M} -x^{(l)}(k)$$

$$\Sigma_{23}(k) = \Sigma_{32}(k) = \sum_{l=1}^{l=M} -y^{(l)}(k)$$

$$\Sigma_{33}(k) = \sum_{l=1}^{l=M} ((x^{(l)}(k))^2 + (y^{(l)}(k))^2)$$

The determinant of the matrix $(H^T(k)H(k))$ is:

$$|H^T(k)H(k)| = 2M \sum_{l=1}^{l=M} (x^{(l)}(k)x^{(l)}(k) + y^{(l)}(k)y^{(l)}(k)) - M(M-1) \sum_{l=1}^{l=M} \sum_{m=1, m\neq l}^{l=M} (x^{(l)}(k)x^{(m)}(k) + y^{(l)}(k)y^{(m)}(k)) \quad (35)$$

By rearranging the terms of the determinant, we can rewrite it as:

$$|H^T(k)H(k)| = M \sum_{l=1}^{l=M} \sum_{m=1, m\neq l}^{l=M} [(x^{(l)}(k) - x^{(m)}(k))^2 + (y^{(l)}(k) - y^{(m)}(k))^2] \quad (36)$$

We can conclude from the form of (36) that the determinant is not zero when at least two of the feature points do not have the same projection on the image plane ($x^{(l)}(k) \neq x^{(m)}(k)$ or $y^{(l)}(k) \neq y^{(m)}(k)$ for at least one pair of m, l).

References

1. N. Papanikolopoulas. 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.

2. N. Papanikolopoulos, P. K. Khosla, and T. Kanadr, "Adaptive robotic visual tracking", Proc. *of the American* Control Conference, June 1991, pp. %2-967.

3. E.H. Adelson and J.R. Bergen, "Spatiotemporal energy models for

the perception of motion", *Journal of the Optical Society of America*, Vol. 2, No. 2, February 1985, pp. 284-298.

4. D.J. Heeger, 'Depth and flow from motion energy", *Science*, No. 86.1986. pp. 657-663.

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

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

7. C.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.

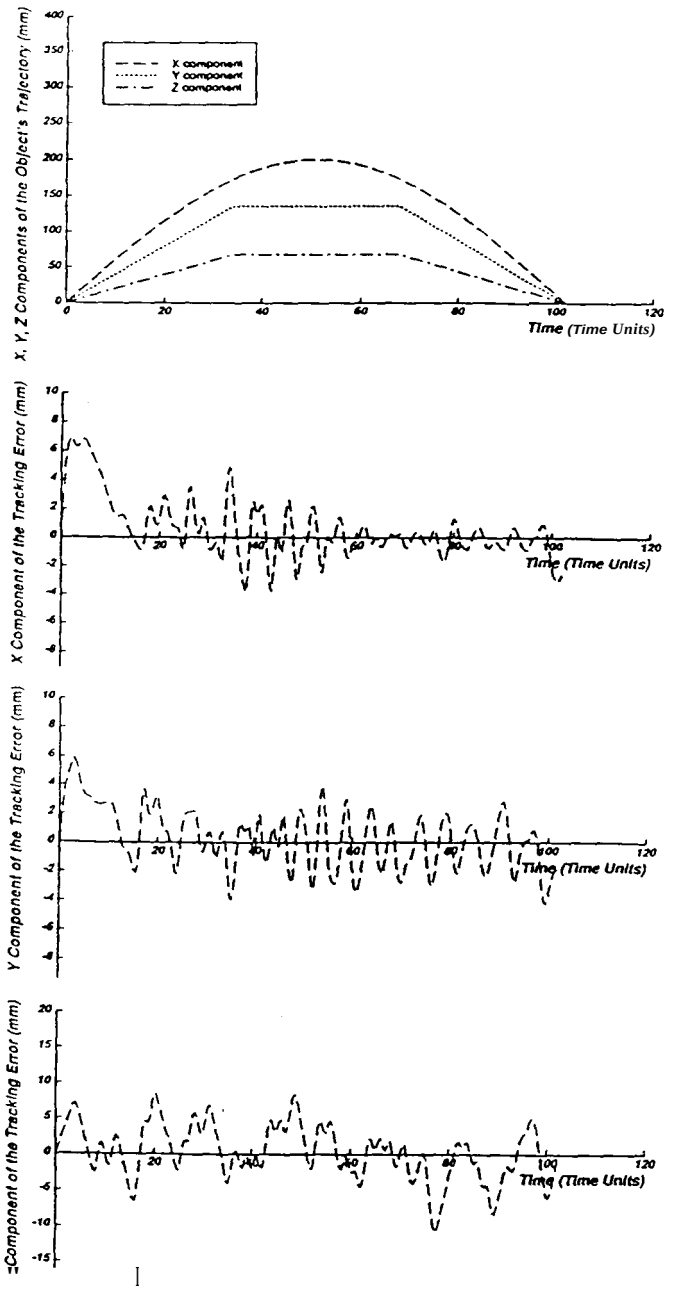

Figure 3: Example of 3-D robotic visual tracking with measurements from multiple windows (Simulation).

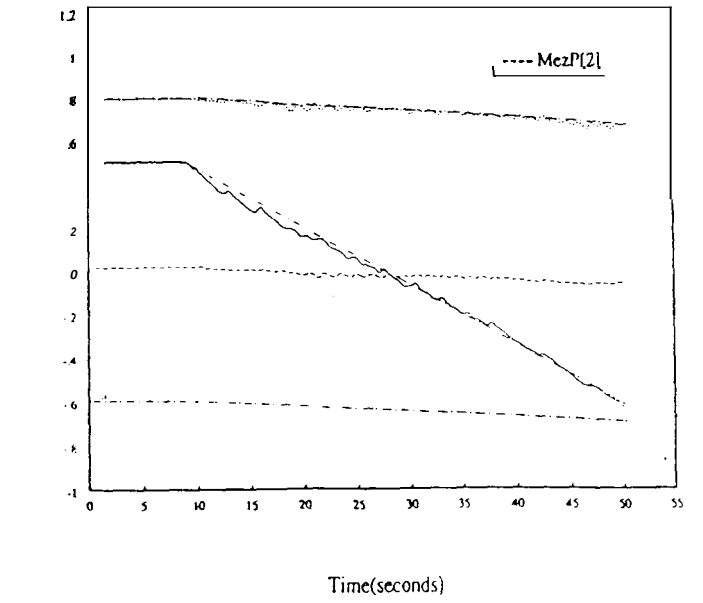

Figure 4: STRWDU SISO adaptive controllers in conjunction with a cartesian PD robotic controller with gravity compensation (Experimental). *This* Figure shows the trajectories of the moving object and the manipulator's end-effector with respect to the world frame.

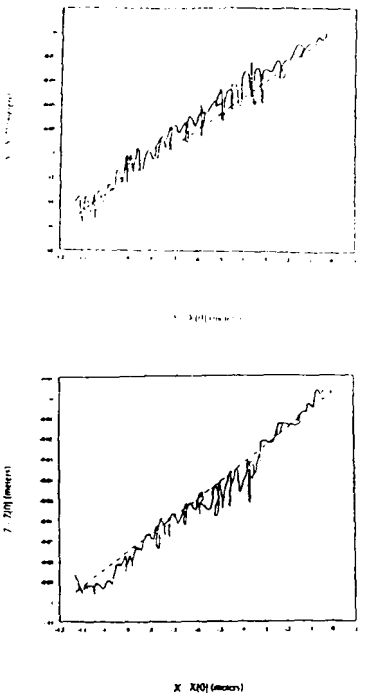

Figure 5: STRWDU SISO adaptive controllers (previous example) in conjunction with a Cartesian PD robotic controller with gravity compensation (Experimental). This Figure shows the relative trajectories of the object and the manipulator's end-effector in Y-X and Z-X.