Presented at 1992 IEEE International Symposium on Intelligent Control, August 11-13, 1992, Glasgow, Scotland, U.K.

FULL 3-D TRACKING USING THE CONTROLLED ACTIVE VISION PARADIGM

N. P. Papanikolopoulos. B. Nelson, and P. K. Khosla

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

Abstract

This paper presents algorithms for 3-D robotic visual tracking of moving targets whose motion is 3-D and consists of translational and rotational components. The objective is to track selected features of the moving target and to position their projections on the image plane at desired positions. The most important characteristics of the proposed algorithms are the use of a single camera mounted on the end-effector of a robotic device (eye-in-hand configuration) and the fact that these algorithms do not require accurate knowledge of the relative distance of the target from the camera frame. This fact allows for the potential use of these algorithms in poorly calibrated environments. The entire problem is formulated as an application of the *controlled active vision* framework which states that controlled, rather than accidental, motion of the vision sensor can maximize the performance of any active vision algorithm. The camera model introduces a number of parameters that must be estimated on-line. An adaptive control algorithm compensates for the modeling errors, the tracking errors, and the computational delays which are introduced by the time-consuming vision algorithms. Experimental results are presented to verify the potential of the proposed algorithms. The experiments were performed on the TROIKABOT robotic system which consists of three PUMA560's.

1. Introduction

One of the major goals of current robotics research is to create robotic systems which are able to react to sudden changes in the environment and in the tasks the systems perform, Sensors play a critical role in detecting the changes in the environment and in transmitting the relevant information to the robotic devices. This information must be evaluated in conjunction with the models of the robotic system, the sensor and the environment (if such models exist). Evaluation of the sensory information then results in an appropriate reaction by the robotic system. One of the most important factors for successful robotic reactions is the efficient integration of the sensor in the feedback loop of the robotic system.

Some of the most widely used sensors are vision sensors. such as CCD cameras. The primary advantage of vision sensors is their ability to provide information on relatively large regions of the workspace. This same ability, however, presents problems that must be overcome. Noise, time-consuming image processing, and large amounts of visual information make their use challenging. The integration of visual sensors into the feedback loop has attracted much attention in the past five years [1, 2, 3, 4, 5], because efficient visual control can increase the autonomy of industrial robotic devices. Research efforts in visual control can be split into two main categories. In the first category, the vision sensing module is static [1, 4, 5], while in the second category the sensing module is moving [2, 3]. The approach of the second category provides increased sensing abilities at the expense of complex modeling. In addition, the motion of the camera can be used for the recursive estimation of some of the uncertain or unknown parameters in the servoing model. Feddema et al. [3] have addressed the problem of feature selection and feature-based control for the tracking of a moving target by a robot-camera system. Chaumette et al. [2] have introduced the idea of task-based visual control. They have used the knowledge of the target model in combination with the visual control objective to servo around a moving or static target.

This paper deals with the problem of 3-D robotic visual tracking of targets whose motion consists of 3-D translational and rotational components. A camera is mounted on the robot and provides visual measurements based on a pyramidal SSD (Sum-of-Squared Differences) optical flow algorithm. These measurements are fed to an adaptive control algorithm that provides the inputs to a cartesian robot control scheme after each measurement. Numerical issues related to the strong coupling of the rotational and translational degrees of freedom are treated in a way that guarantees tracking of the object. A single camera is used instead of a binocular system because one of our main objectives is to demonstrate that relatively unsophisticated off-the-self hardware can be used to solve the 3-D tracking problem if the proper modeling and control issues are addressed.

The major differences of our algorithms from similar research efforts [1, 2, 3, 4, 6] are the use of a single moving camera. the ability to compensate for inaccurate camera parameters and unknown depth (distance of the target with respect to the camera frame), full 3-D tracking ability. the small number of parameters that are estimated on-line, and the integration of the characteristics of the motion detection algorithm into the mathematical model for tracking. These differences allow the use of the proposed algorithms in poorly calibrated spaces or in spaces that are difficult to calibrate. such as underwater, space, and nuclear sites. This paper extends our previous work [7, 8, 9] in controlled active vision by allowing tracking of full 3-D motion (translations and rotations) and by reducing the number of parameters that should be estimated on line. Experimental results are presented to show the strengths and the weaknesses of the proposed approach. The experiments are performed on the TROIKABOT multi-robotic system which operates under the CHIMERA II real-time operating system. The TROIKABOT system consists of three PUMA560's. One PUMA carries the camera while another holds the *target.*

The organization of this paper is as follows: Section 2 describes the mathematical framework under which our problem is solved. Section 3 gives an outline of the vision techniques (pyramidal optical flow and confidence measures) used for the estimation of the positions of the features' image projections. The control, filtering, and estimation strategies are discussed in Section 4. The experimental results are presented in Section 5. Finally, in Section 6, the paper is summarized.

2. Modeling of the Visual Servoing Problem

This section describes the mathematical modeling of our problem. We assume a pinhole camera model with a frame R, placed at the focal point of the lens as depicted in Figure 1. Consider a static target with a feature located at a point P with coordinates (X_s, Y_s, Z_s) in R. The projection of this point on the image plane in front of the focal point is the point p with image coordinates (x, y) given by:

$$x = \frac{fX_s}{Z_s s_x} \quad \text{and} \quad y = \frac{fY_s}{Z_s s_y} \tag{1}$$

where f is the focal length of the camera and s_x, s_y are the dimensions (mm/pixel) of the camera's pixels. In addition, it is assumed that $Z_s >> f$. This assumption holds because the the maximum focal length of our cameras is 16mm while Z_s is larger than 250mm. If (c_x, c_y) is the origin of the image coordinate system F_s then:

$$x_a = x + c_x \quad \text{and} \quad y_a = y + c_y \tag{2}$$

where x, and y, are the actual image coordinates in F_a. Let us assume

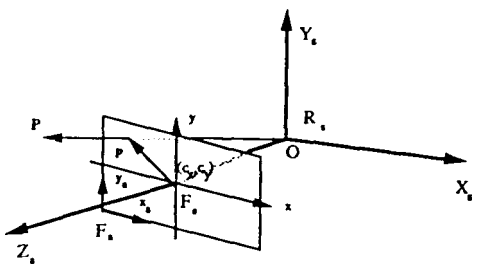

Figure 1: Perspective Projection and Coordinate System R_s.

that the camera moves in a static environment with a translational velocity $\mathbf{T} = (T_x, T_y, T_z)^T$ and with an angular velocity $\mathbf{R} = (R_x, R_y, R_z)^T$ with respect to the camera frame R. The velocity of point P with respect to the R, frame is:

$$\frac{d\mathbf{P}}{dt} = -\mathbf{T} - \mathbf{R} \times \mathbf{P}. \tag{3}$$

By taking the time derivatives of the expressions for x and y and using (1) and (3) we obtain:

$$u = x\frac{T_z}{Z_s} - \frac{fT_x}{Z_s s_x} + \frac{xy s_y}{f} R_x - \left(\frac{f}{s_x} + \frac{x^2 s_x}{f}\right) R_y + \frac{y s_y}{s_x} R_z \tag{4}$$

$$v = y\frac{T_z}{Z_s} - \frac{fT_y}{Z_s s_y} + \left(\frac{f}{s_y} + \frac{y^2 s_y}{f}\right) R_x - \frac{xy s_x}{f} R_y - \frac{x s_x}{s_y} R_z \tag{5}$$

where $u = \dot{x}$ and $v = \dot{y}$. The terms u and v are also known as the optical flow. Consider now a neighborhood S_w of p in the image plane. Assume that the optical flow of the point p at time kT is $(u(kT), v(kT))$ where T is the time between two consecutive frames. It can be shown that at time kT, the optical flow is given by:

$$u(kT) = u_o(kT) + u_c(kT) \tag{6}$$

$$v(kT) = v_o(kT) + v_c(kT) \tag{7}$$

where $u_c(kT)$ and $v_c(kT)$ are the components of the optical flow induced at the time instant kT by the tracking motion of the camera. and $u_o(kT)$ and $v_o(kT)$ are the components of the optical flow induced at the time instant kT by the motion of the object. To keep the notation simple and without any loss of generality, equations (6) and (7) will be used with k instead of kT. Equations (6) and (7) do not include any computational delays associated with the computation and the realization of the tracking motion of the camera. If we include these delays in the model, equations (6) and (7) will be transformed to:

$$u(k) = u_o(k) + q^{-d+1} u_c(k) \tag{8}$$

$$v(k) = v_o(k) + q^{-d+1} v_c(k) \tag{9}$$

where d is the delay factor ($d \in \{1, 2, \ldots\}$) and q^{-1} is the backward shift operator. From the previous analysis, $u_c(k)$ and $v_c(k)$ are given by:

$$u_c(k) = x(k)\frac{T_z(k)}{Z_s(k)} - \frac{fT_x(k)}{s_x Z_s(k)} + \frac{x(k)y(k)s_y}{f} R_x(k) - \left(\frac{f}{s_x} + \frac{x^2(k)s_x}{f}\right) R_y(k) + \frac{y(k)s_y}{s_x} R_z(k) \tag{10}$$

$$v_c(k) = y(k)\frac{T_z(k)}{Z_s(k)} - \frac{fT_y(k)}{Z_s(k)s_y} + \left(\frac{f}{s_y} + \frac{y^2(k)s_y}{f}\right) R_x(k) - \frac{x(k)y(k)s_x}{f} R_y(k) - \frac{x(k)s_x}{s_y} R_z(k) \tag{11}$$

In addition, it is known that:

$$u(k) = \frac{x(k+1) - x(k)}{T} \tag{12}$$

$$v(k) = \frac{y(k+1) - y(k)}{T} \tag{13}$$

If we substitute $u(k)$ and $v(k)$ in (8) and (9) with their equivalent expressions from (12) and (13), then equations (8) and (9) can be written as:

$$x(k+1) = x(k) + Tq^{-d+1} u_c(k) + T u_o(k) \tag{14}$$

$$y(k+1) = y(k) + Tq^{-d+1} v_c(k) + T v_o(k) \tag{15}$$

Further, if we model the inaccuracies of the modei (neglected accelerations, inaccurate robot control) as white noise, (14) and (15) become:

$$x(k+1) = x(k) + Tq^{-d+1} u_c(k) + T u_o(k) + v_x(k) \tag{16}$$

$$y(k+1) = y(k) + Tq^{-d+1} v_c(k) + \mathrm{Tv},(k) + v_y(k) \tag{17}$$

where $v_x(k)$ and $v_y(k)$ are zero-mean, mutually uncorrelated, stationary random variables with variances σ_x^2 and σ_y^2, respectively. The above equations can be written in state-space form as:

$$\mathbf{x}_F(k+1) = \mathbf{A}_F(k)\mathbf{x}_F(k) + \mathrm{B}_F(k-d+1)\mathbf{u}(k-d+1) + \mathbf{E}_F(k)\mathbf{d},(k) + \mathbf{H}_F(k)\mathbf{v}_F(k) \tag{18}$$

where* $\mathbf{A}_F(k) = \mathbf{H}_F(k) = \mathbf{I}_2$, $\mathbf{E}_F(k) = T\mathbf{I}_2$, $\mathbf{x}_F(k) \in R^2$, $\mathbf{d}_F(k) \in R^2$, $\mathbf{u}(k) \in R^6$, and $\mathbf{v}_F(k) \in R^2$. The matrix $\mathbf{B}_F(k) \in R^{2\times 6}$ is:

$$T\begin{bmatrix} \frac{-f}{s_x Z_s(k)} & 0 & \frac{x(k)}{Z_s(k)} & \frac{x(k)y(k)s_x}{f} & \frac{-f^2 - (x(k)s_x)^2}{fs_x} & \frac{y(k)s_x}{s_y} \\ 0 & \frac{-f}{s_y Z_s(k)} & \frac{y(k)}{Z_s(k)} & \frac{f^2 + (y(k)s_y)^2}{fs_y} & \frac{-x(k)y(k)s_x}{f} & \frac{-x(k)s_x}{s_y} \end{bmatrix}$$

The vector $\mathbf{x}_F(k) = (x(k), y(k))'$ is the state vector, $\mathbf{u}(k) = (T_x(k), T_y(k), T_z(k), R_x(k), R_y(k), R_z(k))^T$ is the control input vector. d, $(k) = (u_o(k), v_o(k))^T$ is the exogenous deterministic disturbances vector, and $\mathbf{v}_F(k) = (v_x(k), v_y(k))^T$ is the white noise veaor. The measurement vector $\mathbf{y}_F(k) = (y_x(k), y_y(k))^T$ for this feature is given by:

$$\mathbf{y}_F(k) = \mathbf{C}_F\mathbf{x}_F(k) + \mathbf{w}_F(k) \tag{19}$$

where $\mathbf{w}_F(k) = (w_x(k), w_y(k))^T$ is a white noise vector ($w_F(k) \sim N(0, W)$) and $C_F = \mathrm{I}_2$. The measurement vector is computed using the SSD algorithm which is described in Section 4.

One feature point is not enough to determine the control input vector $\mathbf{u}(k)$. The reason is that the number of system outputs is less than the number of control inputs. Thus, we are obliged to consider more points in our model. In order to solve for the control input that will be sent to the manipulator, it can be shown that at least three non-collinear feature points are needed. The reason for the non-collinearity requirement is investigated in [10].

The state-space model for M ($M \geq 3$) feature points can be written as:

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

where $\mathbf{A}(k) = \mathbf{H}(k) = \mathbf{I}_{2M}$, $\mathbf{E}(k) = T\mathbf{I}_{2M}$, $\mathbf{x}(k) \in R^{2M}$, $\mathbf{d}(k) \in R^{2M}$, and $\mathrm{v}(k) \in R^{2M}$. The matrix $\mathbf{B}(k) \in R^{2M\times 6}$ is:

$$\mathbf{B}(k) = \begin{bmatrix} \mathbf{B}_F^{(1)}(k) \\ \cdots \\ \mathbf{B}_F^{(M)}(k) \end{bmatrix}$$

The superscript (j) denotes each one of the feature points ($(j) \in \{(1), \ldots, (M)\}$). The vector $\mathbf{x}(k) = (x^{(1)}(k), y^{(1)}(k), \ldots, x^{(M)}(k), y^{(M)}(k))^T$ is the new state vector. and $\mathrm{v}(k) = (v_x^{(1)}(k),$

*The symbol I_n denotes the identity matrix of order n

$v_y^{(1)}(k), \ldots, v_x^{(M)}(k), v_y^{(M)}(k))^T$ is the new white noise vector. The new measurement vector $y(k)=(y_x^{(1)}(k), y_y^{(1)}(k), \ldots, y_x^{(M)}(k), y_y^{(M)}(k))^T$ for M ($M \geq 3$) features is given by:

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

where $w(k)=(w_x^{(1)}(k), w_y^{(1)}(k), \ldots, w_x^{(M)}(k), w_y^{(M)}(k))^T$ is the new white noise vector ($w(k) \sim N(0,W)$) and $C = I_{2M}$.

We can combine equations (20)-(21) into a MIMO (Multi-Input Multi-Output) model:

$$(1-2q^{-1}+q^{-2})\,y(k) = B(k-d)\,u(k-d) - B(k-d-1)\,u(L-d-1) + n(L) \tag{22}$$

where $n(k)$ is the white noise vector. The new white noise vector $n(k)$ corresponds to the measurement noise, to the modeling errors. and to the noise introduced by inaccurate robot control. If we assume $B(k-d) \approx B(k-d-1)$, then (22) can be rewritten as a MIMO ARX (AutoRegressive with eXternal input) model. This model consists of $2M$ MISO (Multi-Input Single-Output) ARX models. In addition, the new model's equation is:

$$(1-2q^{-1}+q^{-2})\,y(k) = B(k-d)\,\Delta u(k-d) + n(k) \tag{23}$$

where $\Delta u(k-d)$ is defined as:

$$\Delta u(k-d) = u(k-d) - u(k-d-1) \tag{24}$$

In the next d o n , we will examine the way we obtain the position of the features' projections on the image plane.

3. Measurement of Feature Positions

The continuous extraction of the positions of the features' projections on the image plane is based on optical flow techniques. The computation of u and v (optical flow components) has been the focus of much research and many algorithms have been presented [11, 12, 13]. For accuracy reasons, we use a modified version of the matching based technique [11], also known as the Sum-of-Squared Differences (SSD) optical flow. For every point $p_A=(x_A, y_A)^T$ in image A. we want to find the point $p_B=(x_A+u, y_A+v)^T$ to which the point p_A moves in image B (u and v are used as displacements in this section). It is assumed that the intensity values in the neighborhood L of p_A remain almost constant over time, that the point p_B is within an area S of p_A, and that velocities are normalized by the time period T to get the displacements. Thus, for the point p_A the SSD estimator selects the displacement $\Delta x=(u,v)^T$ that minimizes the SSD measure:

$$e(p_A, \Delta x) = \sum_{m,n \in N} [I_A(x_A+m, y_A+n) - I_B(x_A+m+u, y_A+n+v)]^2 \tag{25}$$

where $u, v \in S$, N is an area around the pixel of interest, and I_A and I_B are the intensity functions in images A and B, respectively. Variations of this technique are used in our experiments In the first variation, image A is the first image ($k=0$) acquired by the camera and image B is the current image ($k \neq 0$). This technique is sensitive to large rotations and changes in the lighting. Another variation of the SSD is the one that updates image A every μ images. The most effective variation of the SSD techmque in t a m s of accuracy and computational complexity proved to be the last one.

The computational speed is improved by using a pyramidal approach for the computation of the displacement vector for each feature. In particular, an algorithm similar to one described in [14] is used. An 8x8 window is centered around the feature in the reference image A. A second 64x64 window, centered around the last observed position of the feature is selected in image B. A pyramid of three different levels is implemented. Therefore, the algorithm for each feature in pseudocode is:

```
Optical_Flow_Pyr( )
{
for (i = 2, u = 0, v = 0; i >= 0; i--) {
```

1. *By averaging or subsampling, a window $8 \cdot 2^i \times 8 \cdot 2^i$ around the feature in the image A is transformed to a window 8×8.*
2. *By averaging or subsampling, a window (defied at the initialization or at step 4) $8 \cdot 2^{i+1} \times 8.2^{i+1}$ in the image B is transformed to a window 16×16.*
3. *The SSD measure is calculated in 9^2 locations and the best displacement $(dx, dy)^T$ is found.*
4. *The window in image B is centered around the previously found best match.*
5. *The displacement. vector $(u, v)^T$ is updated as. $u += dx \cdot 2^i$, $v += dy \cdot 2^i$.*

```
I
I
```

The pyramidal approach has some drawbacks, the most important being that it fails to detect features with little context at the coarse resolution. The drawbacks. however. are easily outweighed by the increased speed of the measurement algorithm. The features are selected by using the techniques described in [7]. These techniques are based on the shape of the SSD surface which is created by autocorrelating the initial image.

The next step in our algorithm involves the use of these measurements in the 3-D visual tracking process. These measurements are transformed into control commands for the robotic system, in our case a PUMA560. In the next section, we present the control and estimation techniques for the 3-D visual tracking protlem.

4. Control and Estimation

The control objective is to move the manipulator in a such a way that the projections of the selected features on the image plane move to some desired positions or stay at their desired positions while the target is moving. This d o n examines the control strategies that realize this motion and the estimation scheme used to estimate the unknown parameters of the model. Some implementation issues are also discussed.

Adaptive control techniques can be used for visual servoing around a moving object when the depth of the object is not precisely known. Adaptive control techniques are used for the recovery of the components of the translational and rotational velocity vectors, $T(k)$ and $R(k)$, respectively, and are based on the *estimated* and not the actual values of the system's parameters. This approach is called *certainty equivalence adaptive control* [IS]. 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 this section is devoted to a detailed description of the control and estimation schemes.

4.1. Selection of an Efficient Control Law

The control objective is to track the motion of certain features of the target and place their projections on the image plane at some desired positions. The tracking of the features' projections is realized by an appropriate motion of the robot-camera system. A simple control law can be derived by the minimization of a cost function that includes the feature positional error, the control signal, and the change in the control signal:

$$J(k+d) = [y(k+d) - y^*(k+d)]^T Q\,[y(k+d) - y^*(k+d)] + u^T(k) L\,u(k) + \Delta u^T(k) L_d\,\Delta u(k) \tag{26}$$

The vector $y^*(k)$ represents the desired positions of the projections of the M ($M \geq 3$) features on the image plane. In our experiments, the vector $y^*(k)$ is known *a priori* and is constant over time. By placing weights on the control signal, the change in the control signal, and the error. we can choose how much emphasis the controller is to place on minimizing each of the three quantities. Including the control signal and the change in the control signal in the cost function described by (26)

causes the control input signal to be bounded and feasible. This is in agreement with the structural and operational characteristics of the robotic system and the vision algorithm. A robotic system Cannot track signals that command large changes in the features' image projections during the sampling interval T. In addition, our optical flow algorithm cannot detect displacements larger than 28 pixels per sampling interval T. The term $\Delta u^T(k) L_d \Delta u(k)$ of the cost function (26) introduces an integral term in the control law. This term is desirable since our mathematical model (20) has a deterministic disturbances component. One problem of the introduction of an integral term in the control law is the possible saturation of the control inputs. In order to compensate for this problem, one should turn off the integrator whenever a saturation of the control inputs occurs.

The control law is derived from the minimization of the cost function (26) by taking the derivative of $J(k+d)$ with respect to the vector $u(k)$ and combining the resulting expression with the system model equation (22). The resulting control law is:

$$u(k) = -[B^T(k) Q B(k) + L + L_d]^{-1} [B^T(k) Q \{ \{(d+1) y(k) - y^*(k+d) - d\, y(k-1)) - d\, B(k-d)\, u(k-d) + \sum_{m=1}^{m=d-1} B(k-m)\, u(k-m) \} - L_d\, u(k-1)] \quad (27)$$

Feddema and Lee [3] proposed a similar control law for the robotic visual tracking problem. The main difference of our control law is that, instead of imposing constraints on the optical flow induced by the camera motion (image plane space). we impose constraints on the components of $u(k)$ of the required camera tracking motion (camera frame space). In this way, we directly control the magnitudes of the control signal and the Control signal change. This fact results in a control law that is more robust and feasible than the one proposed in [3]. The design parameters in our control law are the elements of the matrices Q ,L. and L_d. Often, we set L or L_d to zero. In most of the experiments. we set L = 0 and $L_d \neq 0$ in order to achieve a fast and bounded response. If the matrix $B(k)$ is full rank then the matrix $[B^T(k) Q B(k) + L + L_d]$ is invertible. The matrix $B(k)$ is singular when the M feature points are collinear [3, 10]. An extensive study of other conditions which make $B(k)$ singular can be found in [10].

By selecting L, L_d. and Q. one can place more or less emphasis on the control input. the control input change and the servoing mor. There is no standard procedure for the selection of the elements of these matrices. One technique is the optimization approach [16].

If we want to include the noise of our model and the inaccuracy of the $B(k)$ matrix in our control law, the control objective (26) becomes:

$$J(k+d) = E\{[y(k+d) - y^*(k+d)]^T Q [y(k+d) - y^*(k+d)] + u^T(k) L\, u(k) + \Delta u^T(k) L_d \Delta u(k) | F_k\} \quad (28)$$

where the symbol $E\{X\}$ denotes the expected value of the random variable X and F_k is the sigma algebra generated by the past measurements and the past control inputs up to time k. The new control law is:

$$u(k) = -[\hat{B}^T(k) Q \hat{B}(k) + L + L_d]^{-1} [\hat{B}^T(k) Q \{ \{(d+1) y(k) - y^*(k+d) - d\, y(k-1)) - d\, \hat{B}(k-d)\, u(k-d) + \sum_{m=1}^{m=d-1} \hat{B}(k-m)\, u(k-m) \} - L_d\, u(k-1)] \quad (29)$$

where $\hat{B}(k)$ is the estimated value of the matrix $B(k)$. The matrix $\hat{B}(k)$ is dependent on the estimated values of the features' depth $\hat{Z}_s^{(j)}(k)$ ($(j) \in \{(1), \ldots, (M)\}$) and the coordinates of the features' image projections. In particular, the matrix $\hat{B}(k)$ is defined as follows:

$$\hat{B}(k) = \begin{bmatrix} \hat{B}_F^{(1)}(k) \\ \cdots \\ \hat{B}_F^{(M)}(k) \end{bmatrix}$$

where $\hat{B}_F^{(j)}(k)$ is:

$$T \begin{bmatrix} \frac{-f}{s_x \hat{Z}_s^{(j)}(k)} & 0 & \frac{x^{(j)}(k)}{\hat{Z}_s^{(j)}(k)} & \frac{x^{(j)}(k) y^{(j)}(k) s_y}{f} & \frac{-f^2 - (x^{(j)}(k) s_x)^2}{f s_x} & \frac{y^{(j)}(k) s_y}{s_x} \\ 0 & \frac{f}{s_y \hat{Z}_s^{(j)}(k)} & \frac{y^{(j)}(k)}{\hat{Z}_s^{(j)}(k)} & \frac{f^2 + (y^{(j)}(k) s_y)^2}{f s_y} & \frac{-x^{(j)}(k) y^{(j)}(k) s_x}{f} & \frac{-x^{(j)}(k) s_x}{s_y} \end{bmatrix}$$

In the experiments, the delay factor d is 2, so the control law (29) becomes:

$$u(k) = -[\hat{B}^T(k) Q \hat{B}(k) + L + L_d]^{-1} [\hat{B}^T(k) Q \{ \{3 y(k) - y^*(k+2) - 2 y(k-1)) - 2\hat{B}(k-2) u(k-2) + \hat{B}(k-1) u(k-1)) - L_d\, u(k-1)] \quad (30)$$

4.2. Estimation of the Depth Related Parameters

The estimation of the depth ($Z_s^{(j)}(k)$) related parameters can be done in multiple ways. In this section. we present some of these algorithms. If the inverse of $(s_x Z_s^{(j)}(k)/f)$ is defined as $\zeta_s^{(j)}(k)$, then, equations (18) and (19) can be rewritten as:

$$y_F^{(j)}(k) = 2 y_F^{(j)}(k-1) - y_F^{(j)}(k-2) + \zeta_s^{(j)}(k-d) B_{F_t}^{(j)}(k-d) \Delta T(k-d) + B_F^{(j)}(k-d) \Delta R(k-d) + n_F^{(j)}(k) \quad (31)$$

where the vector $n_F^{(j)}(k)$ is a gaussian noise vector with zero mean and covariance $N^{(j)}(k)$ ($n_F^{(j)}(k) \sim N(0, N^{(j)}(k))$), and $B_{F_t}^{(j)}(k)$, $B_F^{(j)}(k)$ are. given by:

$$B_{F_t}^{(j)}(k) = T \begin{bmatrix} -1 & 0 & \frac{x^{(j)}(k) s_x}{f} \\ 0 & \frac{-s_x}{s_y} & \frac{y^{(j)}(k) s_x}{f} \end{bmatrix},$$

$$B_{F_r}^{(j)}(k) = T \begin{bmatrix} \frac{x^{(j)}(k) y^{(j)}(k) s_y}{f} & \frac{-f^2 - (x^{(j)}(k) s_x)^2}{f s_x} & \frac{y^{(j)}(k) s_y}{s_x} \\ \frac{f^2 + (y^{(j)}(k) s_y)^2}{f s_y} & \frac{-x^{(j)}(k) y^{(j)}(k) s_x}{f} & \frac{-x^{(j)}(k) s_x}{s_y} \end{bmatrix},$$

and

$$\Delta T(k) = T(k) - T(k-1), \quad \Delta R(k) = R(k) - R(k-1).$$

By defining $\Delta u_t^{(j)}(k)$ and $\Delta u_r^{(j)}(k)$ as $\Delta u_t^{(j)}(k) = B_F^{(j)}(k) \Delta T(k)$ and $\Delta u_r^{(j)}(k) = B_F^{(j)}(k) \Delta R(k)$, equation (31) is transformed into:

$$y_F^{(j)}(k) = 2 y_F^{(j)}(k-1) - y_F^{(j)}(k-2) + \zeta_s^{(j)}(k-d) \Delta u_t^{(j)}(k-d) + \Delta u_r^{(j)}(k-d) + n_F^{(j)}(k) \quad (32)$$

The final transformation of equation (32) is done by using the vector $\Delta y_F^{(j)}(k)$ which is defined as:

$$\Delta y_F^{(j)}(k) = y_F^{(j)}(k) - 2 y_F^{(j)}(k-1) + y_F^{(j)}(k-2) - \Delta u_r^{(j)}(k-d)$$

The new form of the equation (32) is:

$$\Delta y_F^{(j)}(k) = \zeta_s^{(j)}(k-d) \Delta u_t^{(j)}(k-d) + n_F^{(j)}(k) \quad (33)$$

The vectors $\Delta y_F^{(j)}(k)$ and $\Delta u_t^{(j)}(k-d)$ are known at every time instant, while the scalar $\zeta_s^{(j)}(k)$ is continuously estimated. It is assumed that an initial estimate $\hat{\zeta}_s^{(j)}(0)$ of $\zeta_s^{(j)}(0)$ is given and $p^{(j)}(0) = E\{[\zeta_s^{(j)}(0) - \hat{\zeta}_s^{(j)}(0)]^2\}$ is a positive scalar p_0. The term $p^{(j)}(0)$ can be interpreted as a measure of the confidence that we have in the initial estimate $\hat{\zeta}_s^{(j)}(0)$. Accurate knowledge of the scalar $\zeta_s^{(j)}(k)$ cor-

responds to a small covariance scalar p_0. In our examples, $\mathbf{N}^{(j)}(k)$ is a constant predefined matrix. To simplify the notation $\mathbf{h}(k)$ is used instead of $\Delta\mathbf{u}_t^{(j)}(k)$.

The estimation equations are [17]:

$$^{-}\hat{\zeta}_s^{(j)}(k) = {}^{+}\hat{\zeta}_s^{(j)}(k-1) \tag{34}$$

$$^{-}p^{(j)}(k) = {}^{+}p^{(j)}(k-1) + s^{(j)}(k-1) \tag{35}$$

$$^{+}p^{(j)}(k) = [\{^{-}p^{(j)}(k)\}^{-1} + \mathbf{h}^T(k-d)\ \{\mathbf{N}^{(j)}(k)\}^{-1}\ \mathbf{h}(k-d)]^{-1} \tag{36}$$

$$\mathbf{k}^T(k) = {}^{+}p^{(j)}(k)\mathbf{h}^T(k-d)\ \{\mathbf{N}^{(j)}(k)\}^{-1} \tag{37}$$

$$^{+}\hat{\zeta}_s^{(j)}(k) = {}^{-}\hat{\zeta}_s^{(j)}(k) + \mathbf{k}^T(k)[\Delta\mathbf{y}_F^{(j)}(k) - {}^{-}\hat{\zeta}_s^{(j)}(k)\,\mathbf{h}(k-d)] \tag{38}$$

where $s^{(j)}(k)$ is a covariance scalar which corresponds to the white noise that characterizes the transition between the states, the superscript (–) denotes the predicted value of a variable, and the superscript (+) denotes its updated value. The depth related parameter $\zeta_s^{(j)}(k)$ is a time-varying variable since the target moves in 3-D and the camera translates along its optical axis and rotates along the X and Y axis. The estimation scheme of equations (34)-(38) can compensate for the time-varying nature of $\zeta_s^{(j)}(k)$ because it is designed under the assumption that the estimated variable undergoes a random change. One problem is to keep the covariance scalar $p^{(j)}(k)$ finite. Solutions for this can be found in [15]. In addition, we have implemented other estimation techniques which deal with time-varying parameters. The first technique we have implemented is called *exponential data weighting* [15]. In this case, we assume that the most recent data contains more information than past data and, therefore, old data is exponentially discarded. A second useful technique is *covariance resetting*. In this case, the covariance scalar $p^{(j)}(k)$ is reset when the estimated variable is drastically changed. In addition to the previous techniques, we propose the use of a more accurate form for the state update of $\zeta_s^{(j)}(k)$. This form is based on the equation (computational delays are included):

$$Z_s^{(j)}(k+1) \approx Z_s^{(j)}(k) + \Delta Z_{os}^{(j)}(k) + q^{-d+1}\Delta Z_{ms}^{(j)}(k) \tag{39}$$

where $\Delta Z_{ms}^{(j)}(k)$ is defined as:

$$\Delta Z_{ms}^{(j)}(k) = -\{T_z(k) + [R_x(k)\,y^{(j)}(k)\,s_y - R_y(k)\,x^{(j)}(k)\,s_x]\frac{Z_s^{(j)}(k)}{f}\}T$$

and $\Delta Z_{os}^{(j)}(k)$ is the change in depth induced by the motion of the target. It is assumed that $\Delta Z_{os}^{(j)}(k)$ does not change significantly between two time instances. The term $\Delta Z_{ms}^{(j)}(k)$ is created by the motion of the *camera* and is derived by substituting the terms $X_s^{(j)}(k)$ and $Y_s^{(j)}(k)$ in (3) with their equivalent expressions from (1). Equation (39) provides an approximation of the change in the feature's depth $Z_s^{(j)}(k)$ between two time instances given the feature's image coordinates and the camera motion. This equation can be rewritten as:

$$Z_s^{(j)}(k) \approx 2Z_s^{(j)}(k-1) - Z_s^{(j)}(k-2) + \Delta Z_{ms}^{(j)}(k-d) - \Delta Z_{ms}^{(j)}(k-d-1) \tag{40}$$

By inverting the terms of the previous equation (40), the following equation is derived:

$$\zeta_s^{(j)}(k) = \zeta_s^{(j)}(k-1)/\{2 - \frac{\zeta_s^{(j)}(k-1)}{\zeta_s^{(j)}(k-2)} + \zeta_s^{(j)}(k-1)\frac{s_x}{f}[\Delta' Z_{ms}^{(j)}(k-d) - \Delta' Z_{ms}^{(j)}(k-d-1)]\} \tag{41}$$

where

$$\Delta' Z_{ms}^{(j)}(k) = -\{T_z(k) + [R_x(k)\,y^{(j)}(k)\,s_y - R_y(k)\,x^{(j)}(k)\,s_x]\frac{1}{\zeta_s^{(j)}(k)}\}T$$

If we substitute. the values of $\zeta_s^{(j)}(k)$ with their estimates, (41) will be transformed into:

$$^{-}\zeta_s^{(j)}(k) = {}^{+}\zeta_s^{(j)}(k-1)/\{2 - \frac{{}^{+}\zeta_s^{(j)}(k-1)}{{}^{\bullet}\zeta_s^{(j)}(k-2)} + {}^{+}\zeta_s^{(j)}(k-1)\frac{s_x}{f}[{}^{+}\Delta' Z_{ms}^{(j)}(k-d) - {}^{+}\Delta' Z_{ms}^{(j)}(k-d-1)]\} \tag{42}$$

The term ${}^{+}\Delta' Z_{ms}^{(j)}(k)$ is derived from $\Delta' Z_{ms}^{(j)}(k)$ by substituting $\zeta_s^{(j)}(k)$ with ${}^{+}\zeta_s^{(j)}(k)$. In addition. equation (35) should be modified to incorporate the new equation for the updates of states. These estimation schemes require. the estimation of one parameter per feature-point and. therefore, the real-time implementation of the estimation scheme is feasible. In addition, we have implemented an estimation scheme that computes two parameters per feature point. This scheme is a variation of the previous estimation scheme and separately estimates the depth related parameters $(f/(s_x Z_s^{(j)}(k)))$ and $(f/(s_y Z_s^{(j)}(k)))$ in the X and Y directions on the image plane. In theory. this formulation can estimate the depth related parameters more accurately.

The matrices $\mathbf{B}_{F_1}^{(j)}(k)$ and $\mathbf{B}_{F_2}^{(j)}(k)$ are transformed and decomposed as follows:

$$\mathbf{B}_{F_{11}}^{(j)}(k) = T\ [\ -1 \quad 0 \quad \frac{x^{(j)}(k)s_x}{f}\ ]$$

$$\mathbf{B}_{F_{12}}^{(j)}(k) = T\ [\ 0 \quad \frac{-s_x}{y} \quad \frac{Y^{(j)}(k)s_y}{f}\ ]$$

$$\mathbf{B}_{F_{21}}^{(j)}(k) = T\ [\ \frac{x^{(j)}(k)y^{(j)}(k)s_y}{f} \quad \frac{-f^2-(x^{(j)}(k)s_x)^2}{fs_x} \quad \frac{y^{(j)}(k)s_y}{s_x}\ ]$$

$$\mathbf{B}_{F_{22}}^{(j)}(k) = T\ [\ \frac{f^2+(y^{(j)}(k)s_y)^2}{fs_y} \quad \frac{-x^{(j)}(k)y^{(j)}(k)s_x}{f} \quad \frac{-x^{(j)}(k)s_x}{s}\ ]$$

The subscript i denotes the X or Y direction. The estimation equations for each feature point are $(i = 1, 2)$:

$$^{-}\hat{\zeta}_{si}^{(j)}(k) = {}^{+}\hat{\zeta}_{si}^{(j)}(k-1) \tag{43}$$

$$^{-}p_i^{(j)}(k) = {}^{+}p_i^{(j)}(k-1) + s_i^{(j)}(k-1) \tag{44}$$

$$^{+}p_i^{(j)}(k) = [\{^{-}p_i^{(j)}(k)\}^{-1} + h_i(k-d)\{n_i^{(j)}(k)\}^{-1}h_i(k-d)]^{-1} \tag{45}$$

$$\kappa_i(k) = {}^{+}p_i^{(j)}(k)\,h_i(k-d)\{n_i^{(j)}(k)\}^{-1} \tag{46}$$

$$^{+}\hat{\zeta}_{si}^{(j)}(k) = {}^{-}\hat{\zeta}_{si}^{(j)}(k) + \kappa_i(k)[\Delta y_{Fi}^{(j)}(k) - {}^{-}\hat{\zeta}_{si}^{(j)}(k)h_i(k-d)] \tag{47}$$

where $\Delta y_{Fi}^{(j)}(k)$ and $h_i(k)$ denote the X or Y components of the vectors $\Delta\mathbf{y}_F^{(j)}(k)$ and $\mathbf{h}(k)$, respectively, and $\hat{\zeta}_{si}^{(j)}(k)$ is the estimated value of eitherthe term $(f/(s_x Z_s^{(j)}(k)))$ or the team $(f/(s_y Z_s^{(j)}(k)))$. In practice. the experimental results from the implementation of this estimation scheme prove to be comparable with the results of the first estimation scheme. Some researchers [3) propose the use of an adaptive scheme that estimates all the elements of the block matrix $\mathbf{B}(k)$ on-line. This approach is computationally expensive and not necessary.

4.3. Implementation Issues and Robot Controllers

In the experiments, we are forced to bound the input signals in order to avoid saturation of the actuators. Thus. after the computation of the translational $\mathbf{T}(k) = (T_x(k), T_y(k), T_z(k))^T$ and rotational velocity 'vectors $\mathbf{R}(k) = (R_x(k), R_y(k), R_z(k))^T$, we limit the input signals by performing the following steps. Let us assume that the maximum translational speed in 3-D which can be achieved by the robot is D_{max} and the Euclidean norm of the vector $\mathbf{T}(k)$ is $\|\mathbf{T}(k)\|$. In addition, the maximum permissible rotational *speed* of the end-effector in X. Y, and Z (camera frame) is R_{max}. Then, the vectors $\mathbf{T}(k)$ and $\mathbf{R}(k)$ are transformed to $\mathbf{T}'(k)$ and R' (k), respectively. by the following equations:

$$T'_x(k) = T_x(k)\frac{D_{max}}{\|T(k)\|} \quad (48)$$

$$T'_y(k) = T_y(k)\frac{D_{max}}{\|T(k)\|} \quad (49)$$

$$T'_z(k) = T_z(k)\frac{D_{max}}{\|T(k)\|} \quad (50)$$

$$if \quad |R_x(k)| > R_{max} \quad then\ R'_x(k) = R_{max}\,sign(R_x(k)) \ else\ R'_x(k) = R_x(k) \quad (51)$$

$$if \quad |R_y(k)| > R_{max} \quad then\ R'_y(k) = R_{max}\,sign(R_y(k)) \ else\ R'_y(k) = R_y(k) \quad (52)$$

$$if \quad |R_z(k)| > R_{max} \quad then\ R'_z(k) = R_{max}\,sign(R_z(k)) \ else\ R'_z(k) = R_z(k) \quad (53)$$

This modification is necessary since the manipulator cannot track high speeds successfully. In order to guarantee the properties of the parameter estimator. we must use the modified components of the translational and rotational velocity vectors for the computation of the past input signals $\Delta\mathbf{u}_t^{(j)}(k)$ and $\Delta\mathbf{u}_r^{(j)}(k)$. Thus, instead of using the signals $\Delta\mathbf{u}_t^{(j)}(k)$ and $\Delta\mathbf{u}_r^{(j)}(k)$ in the estimation process. we use the signals $\Delta\underline{\mathbf{u}}_t^{(j)}(k)$ and $\Delta\underline{\mathbf{u}}_r^{(j)}(k)$ which are given by:

$$\Delta\underline{\mathbf{u}}_t^{(j)}(k) = \mathbf{B}_F^{(j)}(k)\,\Delta\mathbf{T}'(k), \quad \Delta\underline{\mathbf{u}}_r^{(j)}(k) = \mathbf{B}_F^{(j)}(k)\,\Delta\mathbf{R}'(k) \quad (54)$$

where $\Delta\mathbf{T}'(k)$ and $\Delta\mathbf{R}'(k)$ are defined as:

$$\Delta\mathbf{T}'(k) = \mathbf{T}'(k) - \mathbf{T}'(k-1), \quad \Delta\mathbf{R}'(k) = \mathbf{R}'(k) - \mathbf{R}'(k-1).$$

After computing the translational velocity vector $\mathbf{T}'(k)$ and the rotational velocity vector $\mathbf{R}'(k)$ with respect to the *camera* frame R_c, we transform it to the end-effector frame R_e with the use of the transformation ${}^eT_c$. The transformed signals are fed to the robot controller of the PUMA which acts as the tracker. We use the Unimation controllers which are interfaced to our system through multiple Ironics IV-3230 CPU boards. The *Alter* line is used and the desired trajectory in Cartesian space is updated every 28ms. We are currently in the process of substituting the Unimation controllers with Trident boards which can be programmed in C. Finally, the whole system runs under the CHIMERA II real-time operating system [18]. The hardware configuration of the TROIKABOT system is shown in Figure 2. The next section describes the experimental results of our algorithms on the TROIKABOT multi-robotic system.

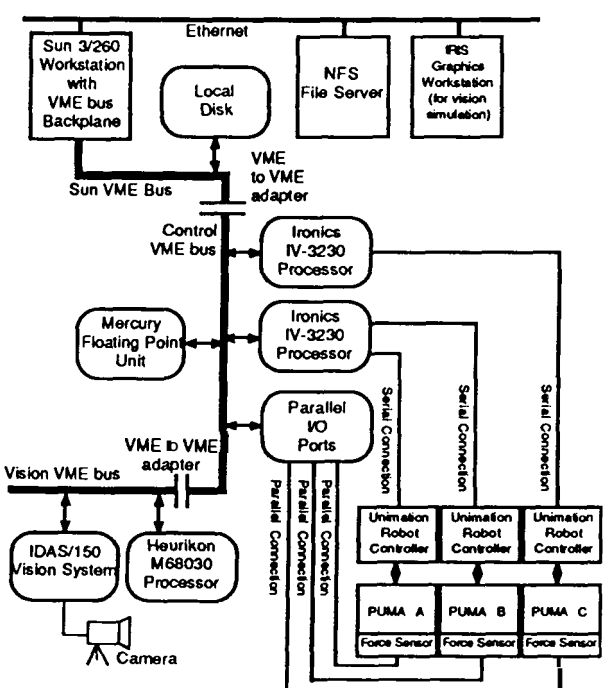

Figure 2: Hardware configuration of the TROIKABOT system.

5. Experimental Results

The algorithms have been verified by performing a number of experiments on the TROIKABOT robotic system [19]. A camera is mounted on the end-effector of one of the PUMAs which acts as the tracker. The other PUMA holds a target and moves it accordingly. The real images are 492x510 and are quantized to 256 gray levels. The camera's pixel dimensions are: s_x=0.011mm/pixel and s_y=0.013mm/pixel. The focal length of the camera is 16mm and the objects move with full 3-D motion. The initial depth of the objects' center of mass with respect to the camera frame Z_c is 290mm. The maximum permissible translational velocity of the end-effector of the tracking robot is 10cm/sec and each of the components of the endeffector's rotational velocity (roll, pitch, yaw) is not allowed to exceed 0.3rad/sec. The objective is to move the manipulator so that the image projections of certain features of the moving object move to some desired image positions or stay at their initial positions. The objects used in the servoing examples are books. pencils. or any item with distinct features. The user uses the mouse to select features of the object to be used in tracking. Then, the system evaluates on-line the quality of the features. based on the confidence measures described in [7]. The same operation *can* be done automatically by a computer process that runs once and needs 2 or 3 minutes, depending on the size of the interest operators which are used. The four best features are selected and used for the robotic visual servoing task. The positions of the four features on the image are shown in Figure 3. The size of windows is 8x8 while the search area is 64x64. The maximum displacement per sampling period T that can be detected is 28 pixels. The SSD algorithm has been implemented by using the pyramidal structure described in Section 3. An interesting solution to the automatic detection and selection of point features has been proposed by Tomasi and Kanade [20]. We are currently investigating the potential of this approach as an alternative to our algorithms for the selection of the best feature points.

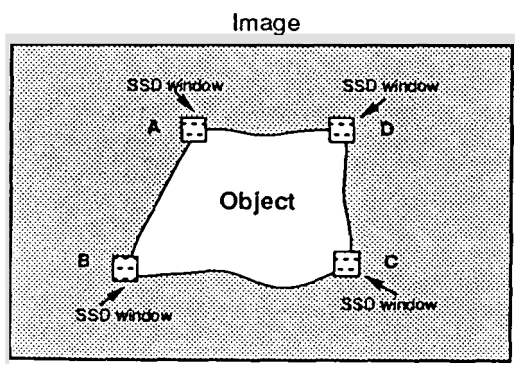

Figure 3: The positions of the four selected features in the image used in the experiments.

Experimental results are presented in Figures 4 through 9. The gains for the controllers are $Q = 0.9I_8$, $L = 0$, and $L_d = diag\{0.04, 0.04, 1.0, 5x10^5, 5x10^5, 5x10^5\}$. The diagonal elements of the Q, L, and L, can vary by a factor of between 2 and 3 and the system will continue to track successfully. The delay factor d is 2. The vector $\mathbf{y}^*(k)$ is given every instant of time k by the relation $\mathbf{y}^*(k) = \mathbf{y}(0)$. This implies that the objective of our scheme is to keep the features at their initial positions during the motion of the target.

The computation of the $[\hat{\mathbf{B}}^T(k)\,Q\,\hat{\mathbf{B}}(k) + L + \mathbf{L_d}]^{-1}$ matrix is done on a

Heurikon 68030 board. The technique used is the same as the one described in [10]. The total computation time (image processing and control calculations) of $T'(k)$ and $R'(k)$ is approximately 220 ms. The knowledge of the depth Z_t is assumed to be inaccurate. For all the features. $\hat{\zeta}_s^{(i)}(0)$ is initialized to 3.63 and $p^{(i)}(0)$ is 0.1.

In the example depicted in Figures 4 through 9, the performance of the control and estimation algorithms is illustrated. The target's trajectory is ploaed with respect to the frame R, which is attached to the target at the time instant k=0. At the same instant, the Z axis of the R_t frame is aligned with the optical axis of the camera. The estimation scheme which is used estimates one parameter per feature point, thus. four parameters are estimated in total. The forgetting factor is 0.99. The measured deriations of the features from their desired positions appear noisy. The fact that the errors on the image plane are bounded guarantees that the errors are within the search range of the SSD algorithm, thus, the SSD algorithm can accurately measure the features' positions. Tnc errors reach a maximum value when the target changes its trajectory sharply. The control and estimation algorithms compensate quickly and after 10 seconds the errors are reduced. The error in the Z direction is large. The reason is that the noisy measurements, the camera geometry, and the experimental setup make the accurate computation of the tracking motion in the Z direction (along the optical axis of the camera) difficult. Another interesting observation is that there is a small error in yaw even though there is no yaw component in the target's motion. This phenomenon occurs since there is a strong coupling between the yaw component and the Y translational component of the tracking motion. The same is true for the pitch component and the X translational component of the tracking motion. In other words, the tracking system mes to track X translational or Y translational motion of the targe: with the rotational degrees of freedom, R_y or R_x, respectively. Numerically, this implies that the condition number c ($c = \sigma_{max}/\sigma_{min}$, a ratio of singular values) of the matrix $\hat{B}(k)$ is large. Appropriate selection of the feature points and the relative position of the camera with respect to the target can minimize the condition number. If the relative distance of the camera (assuming the same focal length for the camera) from the target is more than 2 meters, the condition number becomes too large and tracking is impossible. In addition, full tracking is impossible when the four feature points are close to each other, or if they are very close to the piercing point.

6. Conclusions

In this paper, we have examined the problem of robotic visual tracking of 3-D motion by a monocular robotic tracker. A camera is mounted on the end-effector of the robotic device and provides visual information about the motion of the target. The detection of motion is based on an optical flow technique called Sum-of-Squared Differences (SSD) optical flow. This algorithm, which has been implemented in a pyramidal scheme for computational efficiency. provides the displacement vector of certain selected features of the target. Under the general guidelines of the *controlled active vision* framework which was introduced in [8], we combine these measurements with appropriate control and estimation techniques. Adaptive control techniques are introduced to compensate for uncertainties in the model, unknown depth related parameters. and computational delays. The computational burden is reduced by estimating only one or two parameters per feature point. Our algorithms do not require accurate calibration of the workspace, and, thus, can be efficiently used in assembly lines in order to track moving items. In addition, these algorithms make possible autonomous satellite docking and recovery. The algorithms were extensively tested in several experiments which were performed on the TROIKABOT multi-robotic system. The real-time experiments show the feasibility and efficiency of our algorithms. In general, these algorithms show that monocular vision in conjunction with efficient motion of the vision sensor and adaptive control algorithms can be a viable alternative to standard stereo vision techniques.

Some of the areas for future research which we are currently considering include the use of more elaborate MIMO adaptive control techniques than those that have been implemented, the computational improvement of our algorithms, and the introduction of algorithms for using edges as the source of motion information. We are currently pursuing the use of "snakes" for contour servoing. the application of adaptive algorithms to model-based visual tracking and servoing, and the derivation of depth maps through appropriate motion of the robot-camera system in conjunction with simple adaptive filtering techniques.

7. Acknowledgements

This research was supported by the Defense Advanced Research Projects Agency through ARPA Order Number DAAA-21-89C-0001, and by the U.S. Amy Research Office through grant N'umber DAAL03-91-G-0272. 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. P.K. Allen, "Real-time motion tracking using spatio-temporal filters", *Proc. DARPA Image Understanding Workshop*, 1989. pp. 695-701.

2. F. Chaumette, P. Rives. and B. Espiau, "Positioning of a robot with respect to an object. tracking it and estimating its velocity by visual servoing", *Proc. of the IEEE Int. Conf. on Robotics and Automation*, April 1991. pp. 2248-2253.

3. J.T. Feddema and C.S.G. Lee, "Adaptive image feature prediction and control f a visual backing with a hand-eye coordinated camera", *IEEE Trans. on Systems. Man and Cybernetics*, Vol. 20, No. 5, 1990, pp. 1172-1 183.

4. A.J. Koivo and N. Houshangi. "Real-time vision feedback for servoing of a robotic manipulator with self-tuning controller", *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 21, No. 1, 1991, pp. 134-142.

5. 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.

6. E.D. Dickmanns. B. Mysliwetz, and T. Christians, *"An* integrated spatio-temporal approach *to* automatic visual guidance *of* autonomous vehicles", *IEEE Trans. on Systems, Man. and Cybernetics*, Vol. 20, No. 6,1990, pp. 1273-1284.

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

8. N. Papanikolopoulos, P.K. Khosla. and T. Kanade, "Adaptive robotic visual tracking", *Proc. of the 1991 American Control Conference*, Junc 1991, pp. 962-%7.

9. N.P. Papanikolopoulos and P.K. Khosla, "Feature baud robotic visual tracking of 3-D translational motion". *Proc. of the 30th IEEE CDC. Brighton, UK,*. December 1991. pp. 1877-1882.

10. N.P. Papanikolopoulos and P.K. Khosla, "Robotic visual servoing around a static target: an cxamplc of controlled active vision", Proc. of *the 1992 American Control Conference*, Junc 1992.

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

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

13. S. Ullman, *The interpretation* of visual *motion*, MIT Press, 1979.

14. D.H. Ballard and C.M. Brown, *Computer* vision, Prentice-Hall, Inc.. Englewood Cliffs, NJ 07632, 1982.

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. F.L. Lewis. *Optimal control*, John Wilcy & Sons, New York, 1986.

17. P.S. Maybeck, *Stochastic models, estimation, and control*, Academic Press, London. 1979.

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

19. FE. Acker, I. Ince and B.D. Ottinger, "TROIKABOT - A multi-armed assembly robot", *Proc. of the Robots 9 Conference, Detroit, MI*, June 3-6 1985.

20. C. Tomasi and T. Kanade, "Detection and tracking of point features", Tech. report CMU-CS-91-132, Carnegie Mellon University, School of Computer Science, 1991.

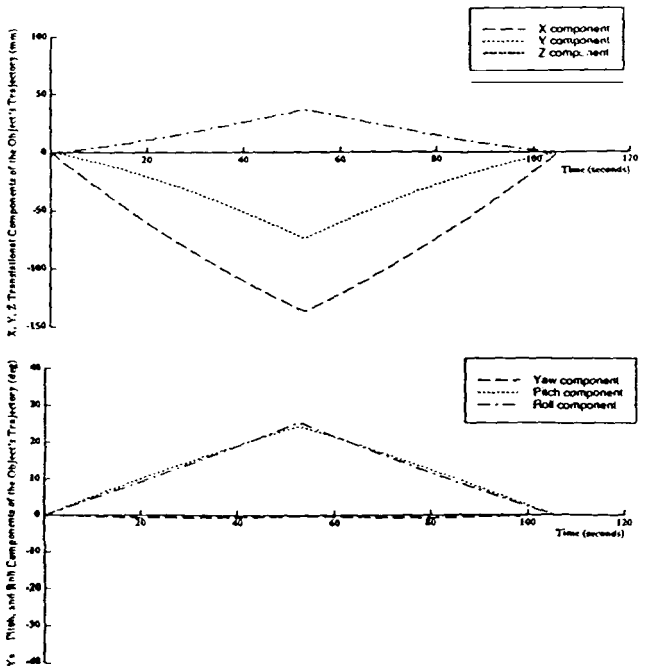

Figure 4: Translational and rotational trajectories (Example A) of the moving object with respect to its initial pose (Experimental).

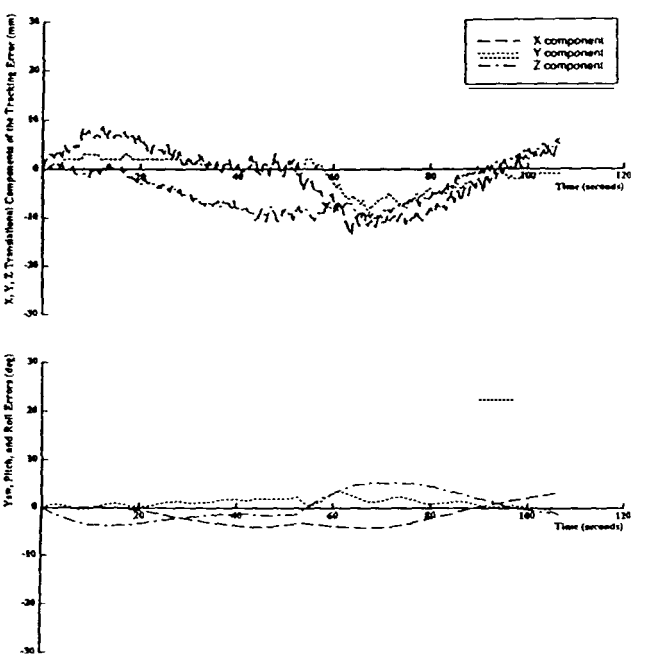

Figure 5: Translational and rotational tracking errors in the previous example (Experimental).

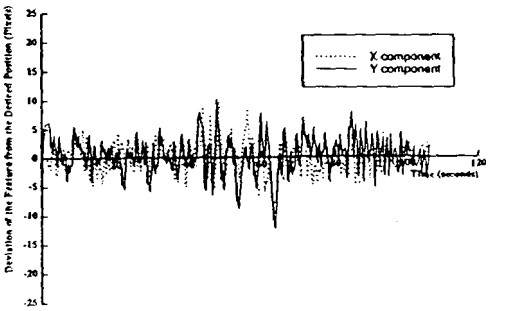

Figure 6: Deviation of feature A from its desired position in the previous example (Experimental)

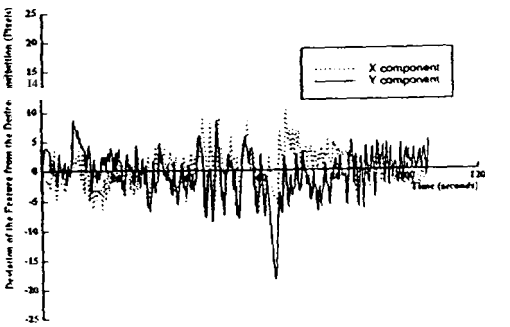

Figure 7: Deviation of feature B from its desired position in the previous example (Experimental).

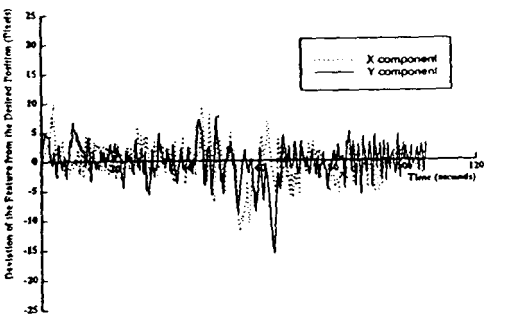

Figure 8: Deviation of feature C from its desired position in the previous example (Experimental).

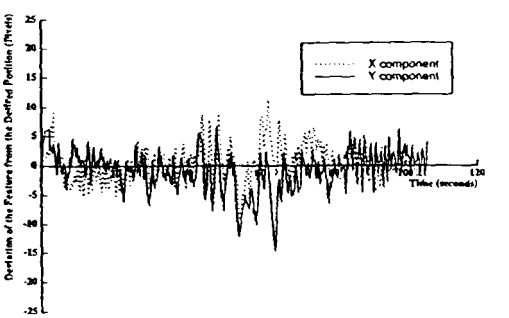

Figure 9: Deviation of feature D from its desired position in the previous example (Experimental).