Journal of Intelligent and Robotic Systems **12**: 1–21, 1995.

Visual Servoing in the Task-Function Framework: a Contour Following Task

EVE COSTE-MANIÈRE*, PHILIPPE COUVIGNOU**, and PRADEEP K. KHOSLA
The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A.

(Received: 16 December 1992; in final form: 11 August 1993)

Abstract. We describe an approach to contour following unknown objects using a hand-eye robotic system. Relevant and sufficient feature points providing optical flow data are extracted from the edges of the target object. The desired motion of the end-effector is computed with the objective of keeping the visual features always at the same target location in the image plane. A cartesian PD controller is used to perform the desired motion by the robot's end-effector. To address the *control issues,* we take advantage of the unifying robot control theory stated in the literature as the *task-function approach* [21]. To validate our approach, we restricted our experiments to motionless objects positioned in a plane parallel to the image plane: three degrees of freedom (two of translation, one of rotation) are thus controlled.

Key words. Robotic visual servoing, task-function approach, control, contour following.

1. Introduction

The spectrum of applications for contour following tasks includes various force/motion control problems such as surface inspection, tracking of seams in arc welding, telerobotics, contour deburring to name a few. In this paper, we describe an approach to *automatic contour following* of unknown objects based on *visual servoing.* Our approach provides information that can be integrated in a force control framework in order to enhance the execution of the various industrial applications already mentioned. For example, in seam tracking systems our visual servoing approach will tolerate variations in the seam by altering the robot trajectory as it tracks different seams. Mereover, it can be used as an alternative to probing tasks usually carried before executing the hybrid force/motion tasks [12]. It can also be applied to mobile robot guidance.

Vision sensors and image-processing systems have reached the level of performance where they can be used in real-time to control the position, orientation and speed of a robot end-effector. They allow the manipulators to carry out

* Present affiliation: INRIA Sophia Antipolis France, e-mail: eve@sophia.inria.fr.
** Renault Direction Recherche France

hybrid tasks (particular case of *redundant tasks*) combining the regulation of vision sensor outputs with any other objectives, such as positioning and trajectory tracking. Therefore, accurate positioning and inspection of objects features (edges, corners, etc.) can be directly specified in terms of the regulation of a set of chosen visual features extracted from the image.

The *visual servoing* approach closes the loop between vision and robot control. Usually the specification of the global task leads to the derivation of visual-based control laws. It allows, for example, the selected features (e.g. the edges) to remain invariant with respect to a chosen working window, and leads to the computation of the end-effector displacements, which compensate the measurement errors and satisfy the tracking objective. Research in this area include [8, 11, 14–16, 19, 20, 25].

In [11], the authors propose a general methodology for the design of tasks that use a visual sensor inside the control loop. It is based on the *task-function* approach to the robot control problem proposed in [21]. This approach is particularly attractive because it demonstrates robustness properties. Furthermore, although vision is recognized for its powerful sensory capabilities, proximity, touch and force sensing also play a significant role in the inprovement of robot performance. As shown in [21–23], the task-function approach can be used successfully to describe any sensor-based actions. Therefore, this approach offers a large number of advantages because of its generality: it allows homogeneous study of the task in vision-based control as described here and of force-based control as described in [5, 23].

In this paper, we take advantage of this approach to implement the control of a complex hand-eye visual servoing system: the goal consists of following the contour of a dark object on a white background. We restrict the scope of our experiments to motion of the camera parallel to the image plane, at given depth and given velocity. We will only consider motionless objects with differentiable boundaries, or with edges that are piecewise continuous. Figure 1 shows the system configuration.

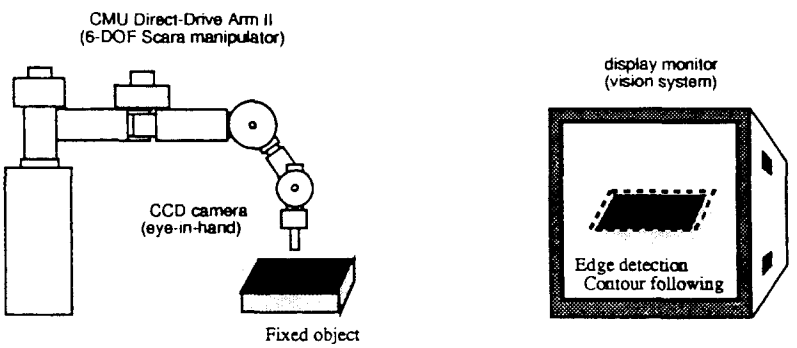

Fig. 1. Eye-in-hand configuration for robotics visual contouring.

The paper is organized as follows: our work is based on image motion processing which is described in Section 2. Relevant information to the estimation of the desired end-effector motion is computed using optical flow equations derived form the perspective transformation of the camera. The task-function approach presented in [21] is recalled in Section 3. We formulate our contribution within the framework of vision-based robotic servoing as described in [11] and briefly recalled in Section 4. We describe the selection of the visual feature points in Section 5. The robotic configuration and image processing hardware are then presented in Section 6. We shose a ruler and an unpolished metal part as two examples of objects to be tracked in 3-DOF motions parallel to the image plane of the camera. The experiments presented in Section 7 validate our theoretical approach.

2. Camera and Robot Modeling

2.1. CAMERA PROJECTION MODEL

Our approach is restricted to eye-in-hand configurations. The camera coordinate system and the end-effector coordinate system are identical, the optical center of the camera being the origin of the coordinate system attached to the end-effector. We assume our CCD camera model consists of a perspective projection with focal length f (mm) followed by a linear scaling with factor γ (pixel/mm). With no loss of generality, it is assumed the same scaling factor $\gamma = \gamma_x = \gamma_y$ for both horizontal and vertical directions on the image. Any point M_i with position vector $P_i = (X_i, Y_i, Z_i)^T$ relative to the camera frame is projected on

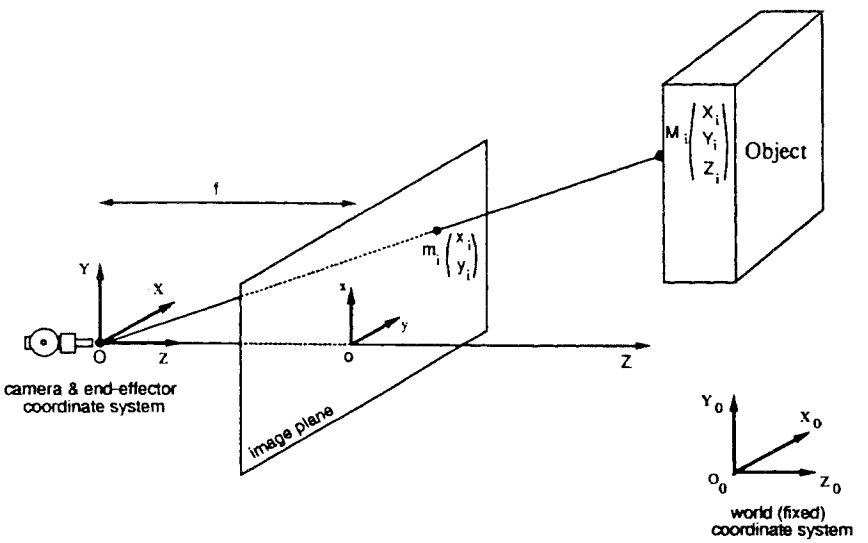

Fig. 2. Camera coordinate system and perspective projection model.

the image plane as a point m_i with position vector $p_i = (x_i, y_i)^T$ according to (see Figure 2):

$$x_i = \gamma f \frac{X_i}{Z_i} \quad \text{and} \quad y_i = \gamma f \frac{Y_i}{Z_i}. \tag{1}$$

2.2. MOTION MAPPING

The camera velocity is denoted by the 6-dimensional vector $T_c = (t_{cX}, t_{cY}, t_{cZ}, w_{cX}, w_{cY}, w_{cZ})^T$. The motion of the feature point M_i relative to the camera frame follows the well-known law of rigid motions:

$$\dot{P}_{i/\text{camera}} = (t_M - t_c) + (w_M - w_c) \times P_i, \tag{2}$$

where $\dot{P}_{i/\text{camera}} = (\dot{X}_{i/\text{camera}}, \dot{Y}_{i/\text{camera}}, \dot{Z}_{i/\text{camera}})^T$ is the 3D velocity vector of the feature point M_i; $t_M = (t_{MX}, t_{MY}, t_{MZ})^T$ and $w_M = (w_{MX}, w_{MY}, w_{MZ})^T$ are respectively the translational and rotational parts of the generalized velocity vector $T_M = (t_{MX}, t_{MY}, t_{MZ}, w_{MX}, w_{MY}, w_{MZ})^T$ of the object with respect to the fixed coordinate system $[O_0, X_0, Y_0, Z_0]$; $t_c = (t_{cX}, t_{cY}, t_{cZ})^T$ and $w_c = (w_{cX}, w_{cY}, w_{cZ})^T$ are respectively the translational and rotational parts of the generalized velocity vector $T_c = (t_{cX}, t_{cY}, t_{cZ}, w_{cX}, w_{cY}, w_{cZ})^T$ of the camera with respect to the fixed coordinate system $[O_0, X_0, Y_0, Z_0]$. $T_M - T_c$ represents the relative six-dimensional velocity vector of the rigid object with respect to the camera. Taking the time-derivative of equation (1) with respect to the camera frame yields the velocity vector of the image point m_i :

$$\dot{p}_{i/\text{camera}} = \begin{pmatrix} \dot{x}_{i/\text{camera}} \\ \dot{y}_{i/\text{camera}} \end{pmatrix} = \frac{\gamma f}{Z_i^2} \begin{pmatrix} \dot{X}_{i/\text{camera}} Z_i & - & X_i \dot{Z}_{i/\text{camera}} \\ \dot{Y}_{i/\text{camera}} Z_i & - & Y_i \dot{Z}_{i/\text{camera}} \end{pmatrix}. \tag{3}$$

In the sequel, we note $\dot{p}_i = \dot{p}_{i/\text{camera}}$. After elimination of $\dot{X}_{i/\text{camera}}, \dot{Y}_{i/\text{camera}}, \dot{Z}_{i/\text{camera}}$ between equations (2) and (3), we obtain:

$$\dot{p}_i = L_i(T_M - T_c), \tag{4}$$

where the 2×6 matrix L_i has elements depending on x_i, y_i and Z_i, and is given by:

$$L_i = \begin{pmatrix} \gamma f/Z_i & 0 & -x_i/Z_i & -x_i y_i/\gamma f & \gamma f + x_i^2/\gamma f & -y_i \\ 0 & \gamma f/Z_i & -y_i/Z_i & -\gamma f - y_i^2/\gamma f & x_i y_i/\gamma f & x_i \end{pmatrix}.$$

In equation (4), the image velocity vector $\dot{p}_i$ is a linear form of the relative velocity vector $T_M - T_c$. This equation is called the *optical-flow equation* associated to the i-th feature point M_i. We define the $2N$-dimensional vector of

image positions p and velocities $\dot{p}$ by combining the N position vectors p_i, and the N velocity vectors $\dot{p}_i$, $i = 1, \ldots, N$, as follows:

$$p = \left(p_1^T, p_2^T, \ldots, p_N^T\right)^T, \tag{5}$$

$$\dot{p} = \left(\dot{p}_1^T, \dot{p}_2^T, \ldots, \dot{p}_N^T\right)^T. \tag{6}$$

Similarly, the matrices L_i are combined to form the $2N \times 6$ optical-flow matrix $L = (L_1^T | L_2^T | \ldots | L_N^T)^T$, so that the N equations (4), $i = 1, \ldots, N$, can be written:

$$\dot{p} = L(T_M - T_c). \tag{7}$$

Equation (7) shows the linear mapping between the optical flow vector $\dot{p}$ and the relative velocity vector of the object with respect to the camera. We also define $p - p^*$ as the *position error vector* on the image plane, p^* being time-invariant $2N$-dimensional vector of *desired image positions* of the feature points. The vector p^* is constant because we aim at positioning the feature points at the same, invariant location on the image. The desired image velocity vector $\dot{p}^*$ is null, therefore we can rewrite equation (7) as:

$$\frac{\mathrm{d}}{\mathrm{d}t}(p - p^*) = -L(T_c - T_M). \tag{8}$$

2.3. DYNAMIC MODEL OF THE ROBOT

We consider a rigid robotic manipulator with n joints, whose positions are given by the vector q. In the sequel, we assume $n = 6$ angular joints to comply with our experimental 6-DOF CMU-Direct-Drive-Arm-II manipulator; here, $q = (\theta_1, \theta_2, \ldots, \theta_6)^T$is the 6-dimensional vector of angular positions. The dynamic equation of the robot is:

$$\Gamma = M(q)\ddot{q} + N(q, \dot{q}, t), \tag{9}$$

where Γ is the vector of actuator torques, $M(q)$ is the inertia matrix, $N(q, \dot{q}, t)$ includes centrifugal, Coriolis, friction and gravity forces. Also, the jacobian matrix, $J = J(q)$, of the manipulator is defined as $J = \partial x / \partial q$ and verifies $\dot{x} = J\dot{q}$, where x is the 6-dimensional vector that denotes the position and orientation of the end-effector in cartesian coordinates with respect to a world coordinate system and $\dot{x}$ is the time-derivative of x.

3. Vision-based Control in the Task-Function Framework

3.1. THE TASK-FUNCTION APPROACH

We now give the main ideas of the task-function approach developed by Samson and Espiau in [21] and [22], for robot control and sensor-based control. The

approach is based on the specification of a desired objective to be reached by the robot by defining an output n-dimensional second-order continuously differentiable vector function, $e(q,t)$, called *task function,* to be regulated to zero within a finite horizon. Staightforward examples of task-functions include:

- $e(q,t) = q - q_d(t)$, where $q_d(t)$ is a desired trajectory in joint coordinates,
- $e(q,t) = x - x_d(t)$, where $x_d(t)$ is a desired trajectory in cartesian coordinates.

Tracking and following the contour of an unknown object by using a vision sensor requires the definition of a *hybrid tracking/following task.* Redundancy in the task-function approach allows a rigorous definition of *hybrid tasks.* An adequate choise of the task-function e allows the design of redundant tasks and the derivation of complex sensor-based schemes. The task-function vector is derived from a first control objective called *primary task error function* e_1, and from a *secondary cost function* h_s to be minimized under the constraint $e_1 = 0$, with gradient $\partial h_s/\partial x$ with respect to the cartesian position vector x of the end-effector. The general form of the *global task error function* e is:

$$e = W^{\dagger} e_1 + \alpha\left(I_6 - W^{\dagger}W\right) \frac{\partial h_s}{\partial x}, \tag{10}$$

where W is a $m \times 6$ rectangular matrix ($m \leqslant 6$) and α is a positive scalar. $I_6 - W^{\dagger}W$ is a projection (square) matrix of $\mathbb{R}^6$ which projects the 6-DOF end-effector's displacement vectors onto the null space of the matrix W (because $W(I_6 - W^{\dagger}W) = 0$). Therefore, the subspaces of $\mathbb{R}^6$ spanned by the columns of the matrices $W^{\dagger}$ and $I_6 - W^{\dagger}W$ have no vector in common but the null vector, which guarantees the separation of the degrees-of-freedom being controlled by the error vector e_1 and by the minimization of h_s in (10).

3.2. HYBRID TASK-FUNCTION FOR VISION-BASED CONTOUR FOLLOWING

Here, the first control objective corresponds to the positioning of the camera in end-effector's coordinates:

$$e = x - x_d. \tag{11}$$

The primary task error function e_1 integrates the displacements computed from the position error vector $p - p^*$ in the image. The scalar function h_s represents the objective of following the contour of the object at a desired velocity.

The $2N$-dimensional vector $p - p^*$ of the feature points' position errors in the image carries the visual measurements that link the end-effector camera to the target object. It has been shown in Section 2 that the time-derivative of p can be expressed as a function of the relative velocity vector $T_M - T_c$ of the object with respect to the camera. Equation (7) shows that the relation between $\dot{p}$ and

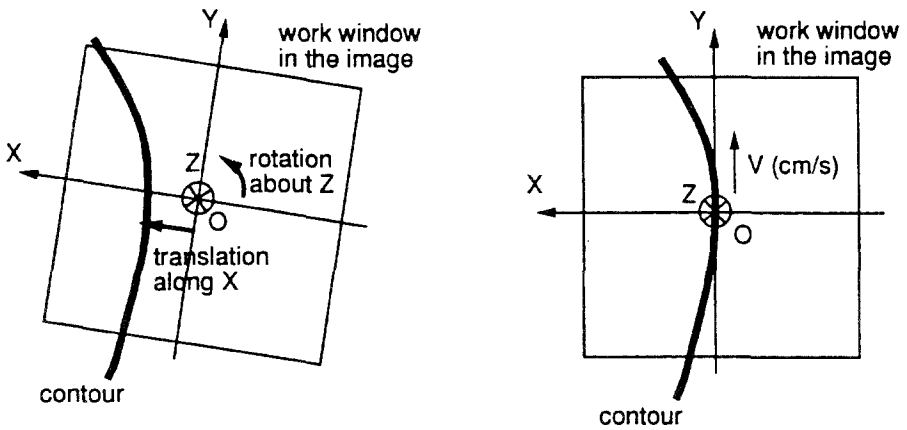

Fig. 3. *Left:* Two-DOF cartesian displacements of the end-effector necessary to position the camera on the contour. *Right:* Contouring velocity V.

$T_M - T_c$ is a linear mapping of matrix L. In the general framework of sensor-based control developed in [21], L is called the *interaction matrix* and is used to model any control involving sensor information that depend on the robot's position (radar, ultrasonic, vision, etc.). In our case, L is a $2N \times 6$ rectangular matrix.

Given a contour measured in the image, we need to find the 2-DOF cartesian displacement on the end-effector that compensates for the contour error: the two degrees-of-freedom consist of translational motions along the X-axis and rotational motions around the Z-axis of the end-effector's coordinate system as shown in Figure 3.

Moreover, it is assumed that the error vector $p - p^*$ remains small during the servoing task. Thus, we can use the integral form of equation (8) to estimate the 6-DOF relative camera displacement that would yield the same displacement $p - p^*$ of the feature points in the image. The optimal camera displacement (in the least squares sense) is then the 6-dimensional vector:

$$e_1 = -L^\dagger(p - p^*) \tag{12}$$

The vector $-e_1 = L^\dagger(p - p^*)$ represents the 6-DOF end-effector displacement that would optimally compensate for the image error $p - p^*$ if it were directly performed by the manipulator. The matrix $W^\dagger$ projects the 6-DOF vector e_1 on the subspace spanned by the 2-DOF displacements to be controlled, which are the translation along the X-axis and the rotation around the Z-axis of the end-effector:

$$W = \mathrm{diag}\,(1,0,0,0,0,1), \quad W^\dagger = W, \quad I_6 - W^\dagger W = \mathrm{diag}\,(0,1,1,1,1,0).$$

The remaining four degrees-of-freedom are controlled by minimizing the secondary cost function h_s. The corresponding projection matrix if the diagonal

matrix $I_6 - W^{\dagger}W$. Here, the objective is to follow the contour of the object at a constant desired velocity V. Under the constraint $e_1 = 0$, the contour of the object is parallel to the Y-axis of the camera's (or end-effector's) coordinate system. Therefore, the cost function to be minimized is $h_s = \frac{1}{2}(Y - Y(0) - Vt)^2$, and the gradient vector $\partial h_s/\partial x = (0, Y - Y(0) - Vt, 0, 0, 0, 0)^T$ has only one non-zero component that corresponds to a translation along the Y-axis. We also let $\alpha = 1$ in (10). The global task-function error $e = x - x_d$ is then computed from e_1 and $\partial h_s/\partial x$ by applying equation (10); then the task-function vector e becomes input to the robot control law (13) as explained later.

3.3. REMARKS ABOUT THE INTERACTION MATRIX L

The choice of a perspective projection model and the associated *interaction matrix* L are detailed in Section 2. In the experiments, we assume that the initial depth Z_i is perfectly known and is kept constant for every feature point M_i of the object. With this simplifying assumption, no identification scheme is required to estimate the values of Z_i with appear in the matrix L. More sophisticated algorithms make use of adaptive estimation and control techniques [14, 19, 20] to overcome this simplifying restriction. Recursive indentification algorithms can be incorporated in our framework. Our assumption about the knowledge of every depth Z_i does not impact the generality of our scheme.

Let us point out that uncertainties about the target's depth Z and sensor's characteristics (γ, f) alter the knowledge of the elements of L. During task execution we approximate the elements of L by their initial value computed at the goal image $p = p^*$. Thus, the opticalflow matrix L and its pseudo-inverse $L^{\dagger}$ are approximated by constant matrices throughout the tracking process. Doing so, significant computation time is saved at each iteration. The approximation is valid under the assumption of good tracking-performance, or equivalently when the image error vector $p - p^*$ remains small. In practice, only the first and sixth rows of $L^{\dagger}$ need to be computed since the position error term is $W^{\dagger}e_1 = -W^{\dagger}L^{\dagger}(p - p^*) = (LW)^{\dagger}(p - p^*)$. More precisely, the pseudo-inversion is better performed on the $2N \times 6$ matrix LW which has all columns zero except the first and sixth columns. The computation of $(LW)^{\dagger}$ is done after a singular value decomposition of the two non-zero columns of the rectangular matrix LW. In addition, the N feature points must be carefully selected on the object so that the two relevant columns of the corresponding matrix LW are linearly independent [7]. A low condition number for LW is also desirable for the sake of numerical stability.

4. Robot Control

The overall organization of the visual servoing system is shown in Figure 4.

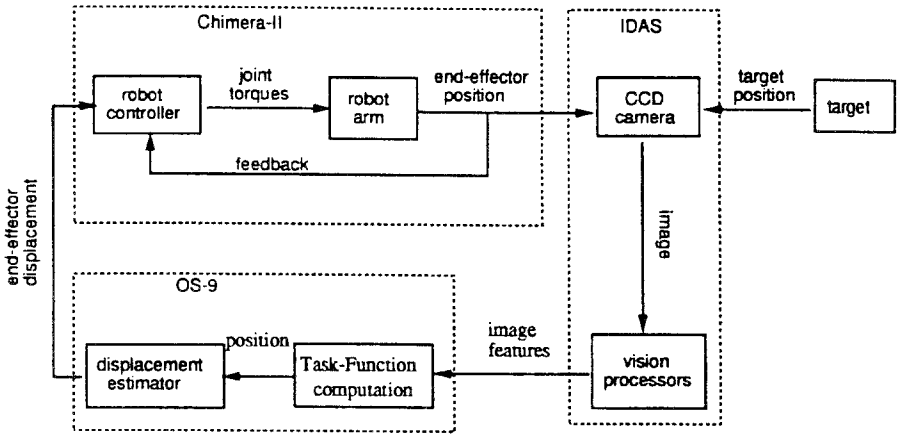

Fig. 4. Schematic organization of the visual servoing system.

4.1. GENERAL CONTROL SCHEME

Once computed, the task-error vector $e = x - x_d$ is used as input to the robot controller. It is shown in [21] that a control law that ideally decouples and ensures a linear behavior of the task error vector e follows the general expression:

$$\Gamma = M\left(\frac{\partial e}{\partial q}\right)^{-1}\left[-kG(\mu De + \dot{e})\right] + N - M\left(\frac{\partial e}{\partial q}\right)^{-1} f, \tag{13}$$

where Γ is the vector of applied actuator torques introduced in equation (9), f comes from terms in the second-order time-derivative of e, $\dot{e} = (\partial e/\partial q)\dot{q} + (\partial e/\partial t) = J\dot{q} + (\partial e/\partial t)$, G and D are positive matrices, and k and μ are positive scalars. In practice, the control in (13) is implemented by using model-based approximations or recursive estimates of the different parameters $M(\partial e/\partial q)^{-1}, N, f$. Equation (13) covers most of existing schemes: computed torque, resolved motion rate or acceleration control, indirect adaptive control and so forth. The theory also provides sufficient conditions of the parameters to ensure the stability of the control.

4.2. CARTESIAN PD CONTROLLER

We experimented with a cartesian PD robot control scheme with gravity compensation, the mathematical model of which is now recalled. Given a desired cartesian position $x_d(t)$, velocity $\dot{x}_d(t)$ and acceleration $\ddot{x}_d(t)$ for the end-effector, the cartesian PD controller computes a torque vector Γ and sends it to the joint actuators. The torque vector Γ is computed as follows [17]:

$$\Gamma = J^T F + g \qquad \text{with} \tag{14}$$

$$F = K_p(x_d - x) + K_d(\dot{x}_d - \dot{x}) + \ddot{x}_d, \tag{15}$$

where g accounts for gravitation forces, F is a 6-dimensional vector of virtual generalized force expressed in cartesian coordinates, K_p and K_d are diagonal gain matrices.

Let us identify the parameters $M(\partial e/\partial q)^{-1}$, N, D, G, f, k, μ, which appear in the general control law (13) with the parameters K_p, K_d, g, actually used for implementing the cartesian PD controller with gravity compensation as described by equations (14)–(15). In our experiments, the control inputs are restricted to end-effector position errors $e = x - x_d$ computed from visual measurements and to joint angular velocities $\dot{q}$. Therefore, we must let $x_d - x = -e$, $\dot{x}_d = 0$, $\ddot{x}_d = 0$, and $\dot{x} = J\dot{q}$ in equations (14)–(15) which give:

$$\Gamma = -J^T K_p e - J^T K_d J\dot{q} + g. \tag{16}$$

From the expression for the gradient $\partial h_s/\partial x$ of the chosen secondary cost function h_s, we obtain $|\partial e/\partial t| = \alpha|V| = |V|$. The approximation $\partial e/\partial t = 0$ necessary to simplify $\dot{e} = J\dot{q} + \partial e/\partial t$ in (13) is valid under the assumption that the contouring velocity V is small, which is the case in our experiments $|V| < 2$cm/s). Thus, the different parameters in the general control equation (13) are easily identified as follows *: $\widehat{M(\partial e/\partial q)^{-1}} = J^T$ with an obvious breach of notation. This is justified since the matrices product $(\widehat{\partial e/\partial q}) \cdot (\widehat{\partial e/\partial q})^{-1} = J\widehat{M}^{-1}J^T$ is strictly positive, and therefore the stability of the control is ensured. Also, $\widehat{N} = g, \hat{f} = 0, k = \mu = 1, G = K_d$ and $D = K_d^{-1}K_p$.

5. Feature Points Selection

We now discuss the extraction of the characteristic parameters to be used in the control. Frequent approarches encountered in the literature consist of using contour primitives (like points, corners, segments, rectangles, general ellipses) [3, 9, 10, 13] or region primitives (like areas, centroids and central moments, etc) [11, 25]. Here, *feature points* are chosen. We do not assume any *a priori* knowledge of the geometric structure of the 3D scene to analyze. The approach presented above in general enough to perform robust contour following on lines, curves, burrs and so on.

5.1. PRE-POSITIONING

Before carrying out thee contour-following task itself, we wish to pre-position the end-effector along the edge of our target object. The user characterizes the edge by setting, in the image plane, a working window 'on' the edge of the object. N feature points are automatically extracted on the object's edge, as local maxima

* We use carets to denote the model estimates of the true terms.

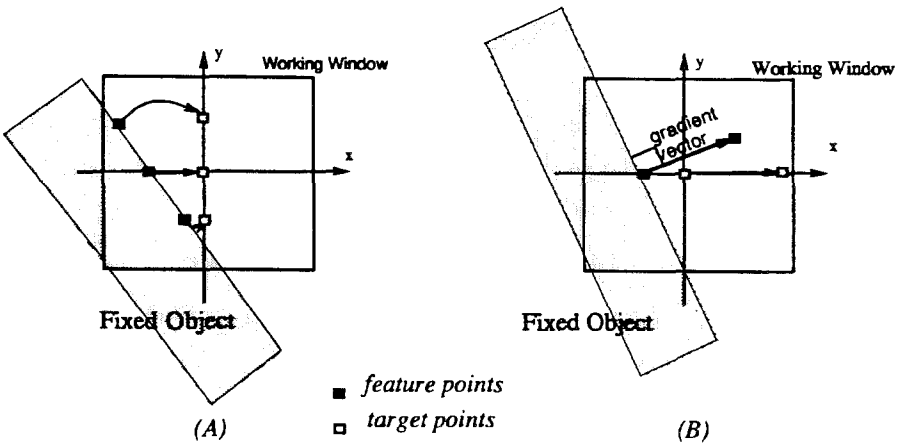

Fig. 5. (A): Line detection; (B): Gradient computation.

of the gradient magnitude in the gradient direction on N equidistant horizontal lines. The $2N$-dimensional vector of image positions p is thus defined. As illlustrated in Figure 5-(A), the N target points are defined on the y axis of the window's frame by rotation and translation of the N features points. The $2N$-dimensional vector p^* of desired image positions (x_i^*, y_i^*), $i = 1, \ldots, N$, is thus introduced in the control scheme. The cartesian displacement of the end-effector can then be computed to compensate for the position error vector on the image plane $p - p^*$, as detailed in Section 3.

5.2. CONTOUR FOLLOWING

The local maxima of the gradient magnitude in the gradient direction is now computed along the local x axis of the window frame defining a feature point on the edge of the object. The corresponding target point is the center of the window. A sub-window of fixed size is automatically generated and centered around the feature point. The average image gradient vector is computed in this area. This vector is normalized, and its end-point defines a feature point. The corresponding target point is positioned on the local x axis, and define a unit vector. The end-effector displacement is then generated (cf. Section 3).

5.3. REMARKS

- Instead of using two points to represent the gradient vector, that is to say four redundant scalars while three are enough, we could have used the relevant components for the straight lines as mentioned in [11]. Choosing points ease the understanding of the equations.

- We chose the tracked edge to be vertical in the image. The problem of locating the edge is then reduced to one dimension (along the x axis of the window frame). This simplification reduces the processing time required by the vision process. It also improves the robustness and the speed of the image point detection process (disturbances in the image not parallel to the edge can be avoided). This constraint is restrictive since if may lead to joint limits on the end-effector. It must be understood as a first step to contour following task in the force control framework.
- Because of noise in the measurement of the image error vector $p - p^*$, the desired end-effector's motion must be filtered before they are fed to the robot controller. The two non-zero displacements of the primary task-error vector We_1 (translation along X and rotation about Z) are filtered in parallel by two *independent* LQG regulators (see [1] and [8] for details). More precisely each displacement measurement passes through a recursive Kalman filter with a constant linear feedback on the optimal state estimate. The displacements may as well be controlled in parallel by PID regulators. However, we obtained the best results using the steady-state solution of the LQG problem.

6. Hardware Implementation

Our experimental hardware consists of the CMU-DiretDrive-Arm-II manipulator and a vision processing system, both integrated under the Chimera-II [24] real-time operating system. The CMU-DDARM-II is a six degrees-of-freedom Scara-like manipulator, whose joints are actuated by direct-drive DC-brushless

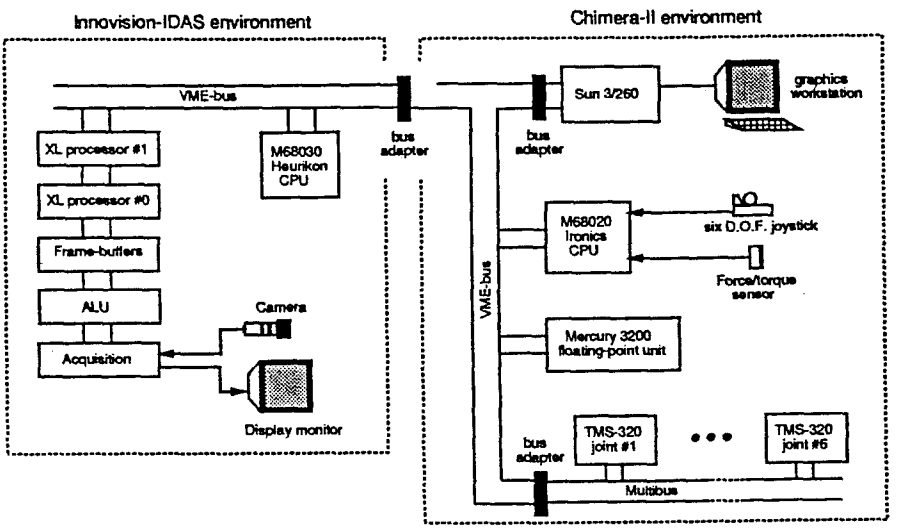

Fig. 6. Robotic and vision systems architecture.

motors [17, 18]. The controller architecture shown in Figure 6 includes the following units:

- six Texas-Instruments TMS-320 processors each controlling one joint of the robot.
- one Mercury-3200 floating-point processor which supports the computational requirements of advanced model-based control schemes like computed torque.
- one M68020 Ironics unit for executing and sequencing robotic tasks. Integrated sensing devices include a force-torque sensor and a six degree-of-freedom joystick.
- one Sun 3/260 workstation which serves as a programming environment to develop, down-load and monitor programs executing on the above processing units.

All units are connected through a VME-bus and run the Chimera-II real-time multiprocessor operating system. A VME-to-VME bus adapter connects the Chimera-II environment to the vision system. Figure 6 also shows the vision system architecture. We mounted a CCD camera (Panasonic CCD color camera model GP-CD1H) of focal length $f = 7.5$ mm in the end-effector of the robot. Full 492×512-pixel images can be acquired at video rate (30 Hz), each pixel is quantized into 256 gray levels. Image processors, frame-buffers, ALU and acquisition units are organized into a pipeline architecture (Innovision IDAS/ITEX-150 vision system). Control and sequencing are perfomed by an M68030 Heurikon processor running the OS-9 operating system (Microware Systems OS-9 operating system). All code for control and image processing has been written in C under Chimera-II and OS-9/IDAS programming environments.

The experiments were performed in conjunction with a cartesian PD controller with gravity compensation at a frequency of 500 Hz. The dynamic equations and control algorithms are recalled in Section 2. The desired and-effector motions are updated every 100–200 ms (5–10 Hz). Because the CMU-DDARM-II can move very fast, the reference inputs must be extrapolated by a linear ramp between two consecutive vision measurement inputs. Extrapolated data are then sent to the cartesian PD controller every 2 ms (500 Hz). As a result, the trajectories of the robot are significantly smoothed.

7. Experimental Results

In this section, we present the results of two experiments performed in real-time with the robotic equipment described in Section 6. Two dark objects on a white background, in a plane parallel to the image plane, offered a maximum contrast and were observed under regular lighting condition:

- a dark ruler provides straigh line reference trajectories necessary to check the good following of linear contours,

- an unpolished industrial metallic object (a brake for heavy truck vehicle) is used to verify the performance when following curves.

In all experiments, the working window is first selected manually. With a mouse device, the user selects a window which includes a portion of the boundaries of the object. Then the N feature points are automatically generated as explained in Section 5; they provide the reference position vector p^* in the image. The user must specify the desired number N of feature points as well as the desired contouring velocity V. It is clear that N must be greater than or equal to 2 in order to allow the 2-DOF motion of the object (translation along X and rotation about Z) to be detectable and computable. The targets are observed at a known distance of 70 cm from the camera. The maximum contouring velocity at this distance is about $V = 2$ cm/sec. The experimental setting and the unpolished brake for heavy trucks vehicle are shown on Figure 7.

The vision process computes the desired motion of the end-effector (two translational DOF along X and Y, and a rotational DOF around Z) with a period of $\Delta t = 70$ ms (≈ 14 Hz) when the N-points are extracted, and a period of $\Delta t = 170$ ms (≈ 6 Hz) when the image gradient is computed.

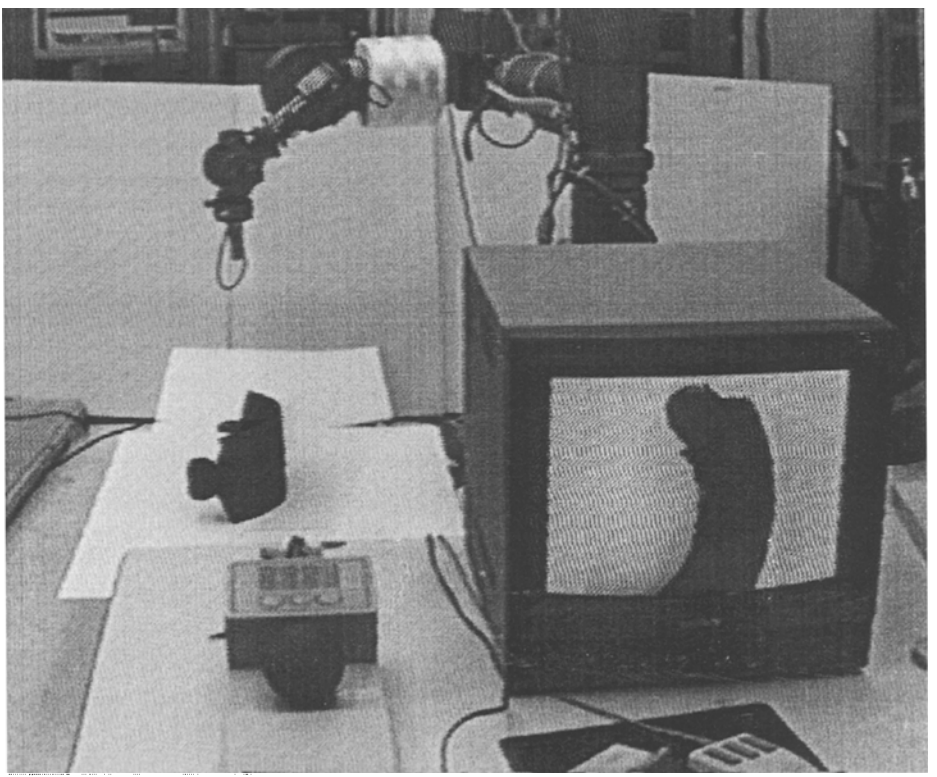

Fig. 7. Experimental setting and the unpolished brake for heavy trucks vehicle.

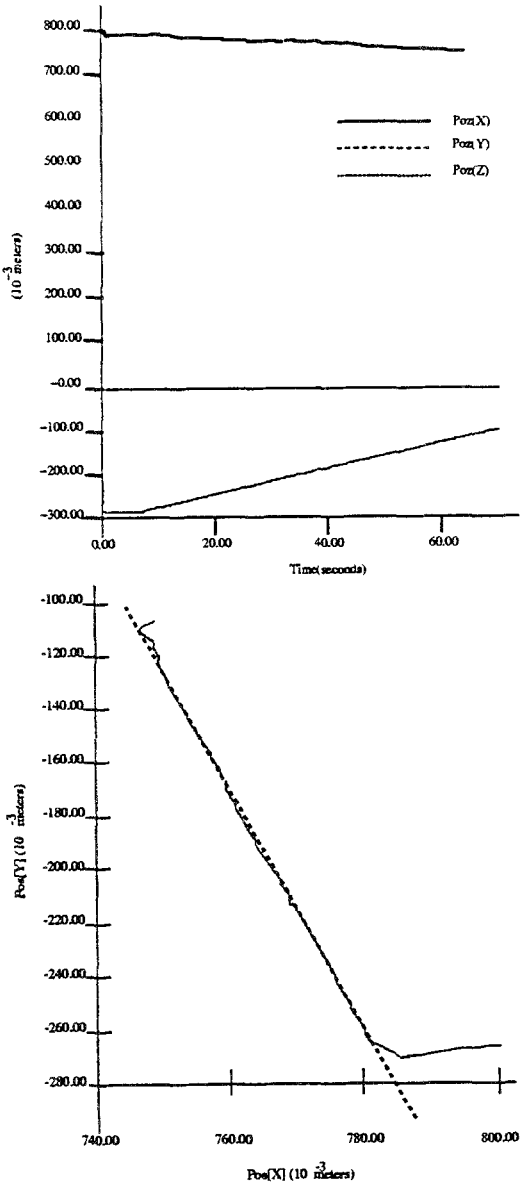

Fig. 8. *Top:* End-effector position *vs.* time as the robot is following a line (cartesian PD controller). *Botton:* End-effector trajectory in the X–Y plane.

The plots in Figures 8 and 9 show the measured end-effector positions in world-frame coordinates at a desired speed of 1.5 cm/s (the desired trajectories are drawn with dot-dashed lines). The first trajectory follows a portion of the line $aX+bY-1=0$, from point $(X_s, Y_s =)(0.77$ m,-0.26 m) to point $X_e, Y_e =(0.85$ m,$-0.10)$. The second trajectory shows a portion of the circle $(X-X_0)^2+(Y-$

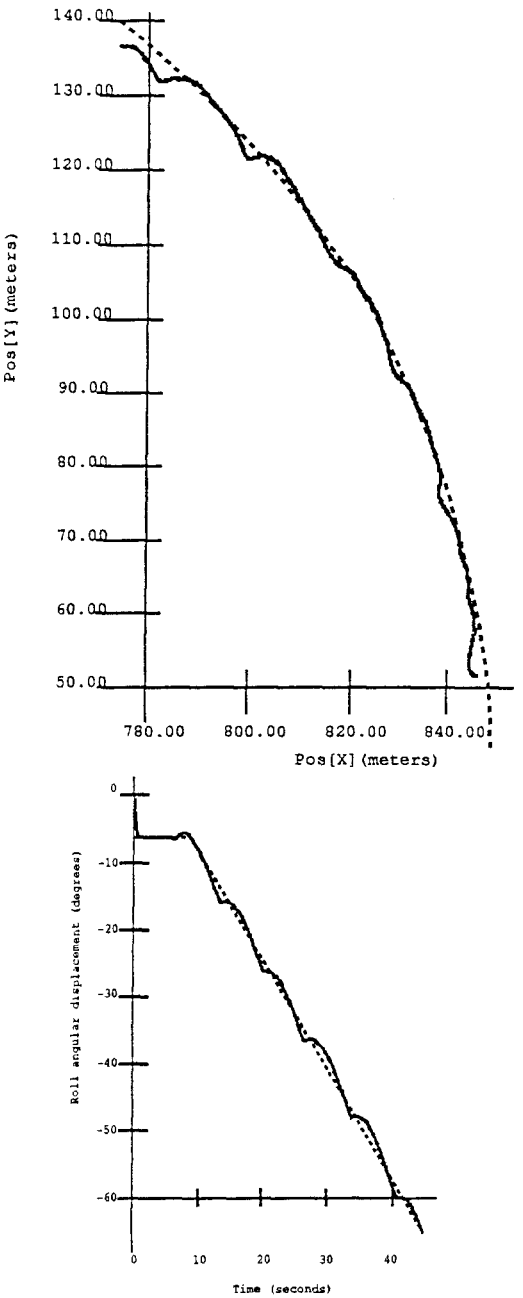

Fig. 9. *Top:* End-effector position in the X–Y plane as the robot is following a circular contour (cartesian PD controller). The oscillations are due to uncompensated friction in the robot's sixth joint. *Bottom:* Roll angular displacement of the end-effector *vs.* time.

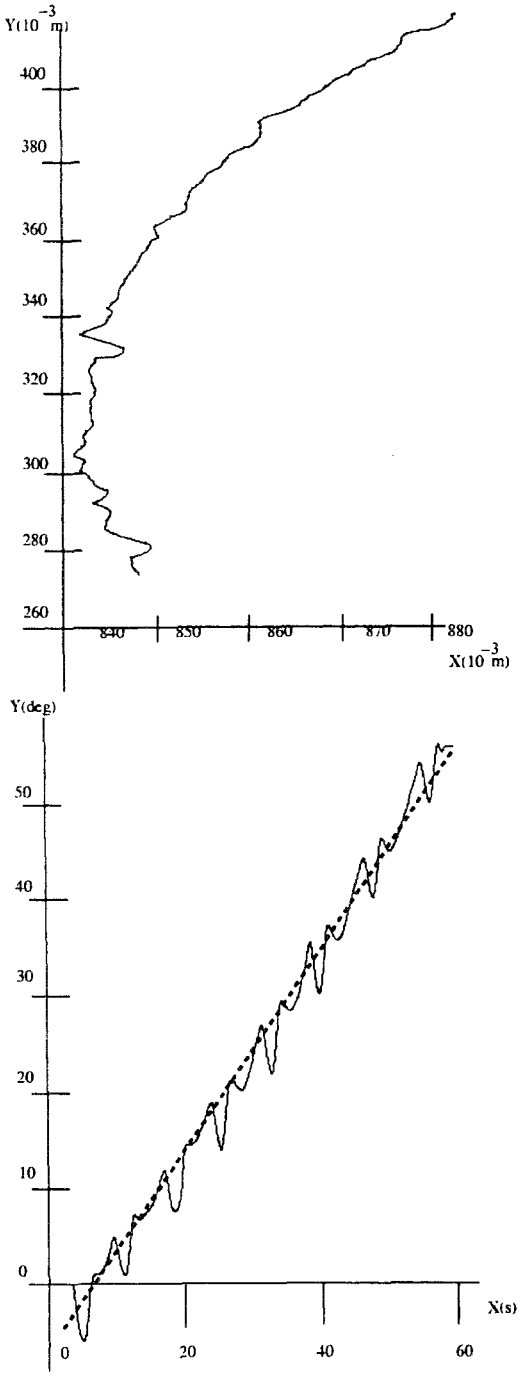

Fig. 10. *Top:* End-effector position *vs.* time as the robot is following a burrs on the outside curve of a brake on a *patterned background. Bottom:* Roll angular displacement of the end-effector *vs.* time.

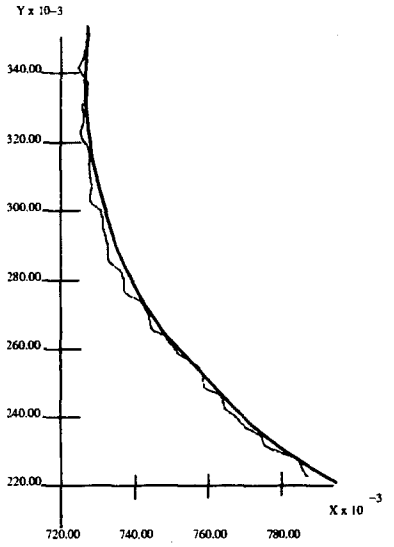

Fig. 11. End-effector position in the X–Y plane as the robot is following the burrs on a cirsular contour.

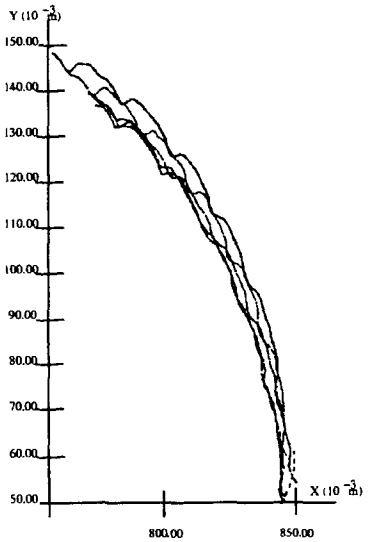

Fig. 12. End-effector position in the X–Y plane as the robot is repeatidly following the same cirsular contour.

$Y_0)^2 - R^2 = 0$, with $(X_0, Y_0) \approx$(0.5 m, 0.01 m), and $R = 200$ mm. The target is positioned in the horizontal plane at $Z = -70$ cm, so that the desired trajectory of the end-effector remains in the plane $Z = 0$. The tracking error remains within ± 3 mm or ± 10 pixels. The oscillations are partially due to the low accuracy

of the cartesian PD control scheme, which does not take into account the full dynamics of the robot arm. Further experiments could be carried out with a computed-torque control scheme (running at 350 Hz) to outline the difference in performance. In addition, friction was not totally compensated on the sixth joint of the manipulator.

During the experiments, the proposed method was found to be very robust and accurate. Significant perturbations done by moving the object or experimenting on a patterned background were successufully rejected (cf. Figure 10). Following the contours of small details (like burrs on the molded object, see [6] for a detailed study) has been carried out successfully. An example of burrs following is fiven Figure 11. The repeatability offers good results for industrial applications, given the accuracy of the DDARM (cf. Figure 12).

8. Summary

We have presented theoretical aspects and experimental results of a contour following scheme in the framework of robotics visual servoing with a hand-eye camera. The contribution of this work is new and original because our method is presented using the generality of the *task-function* approach with all its theoretical advantages. The experimental results achieved on a real-time robotic environment demonstrate the feasibility, performance and robustness of the proposed approach to perform contour following of unknown objects.

Future research issues include the implementation of adaptive control techniques in order to evaluate and control the distance between the camera and the object. Also, we restriced our study to the formal specification in continuous time of the control law. However, the discrete-time, event-driven behavior of the sensor-based task, associated with the logical structure of its execution, could be defined using synchronous programming languages like ESTEREL [2] as advocated in [4, 5]. It would facilitate the handling of exceptions and conditions like joint limitations, feature points occlusion or obstacle avoidance. As mentioned in the Introduction, the contour following task carried out with a hand-eye system can be used as a preliminary step before a hybrid force/position exploration task of the same object. Dynamic reconfiguration of the computational modules associated to the necessary control algorithms allows both tasks to be performed in a row. Particular attention must be paid to handle corners: it is necessary to detect corners whenever the object has sharp contours. Corners (or discontinuities) can be handled using the same proposed approach, provided that the desired motion is smaller than the maximum rotation allowed by the end-effector. An alternate solution would be to define a 'spy'window ahead of the work window, where an accurate corner detection algorithm [10] is implemented. Once detected, it is possible to pass through the discontinuity at a given speed.

Acknowledgments

We acknowledge the helpful suggestions and comments provided by François Chaumette, Mark Delouis, Bernard Espiau, Gérard Giraudon, Jean-Pierre Merlet, Brand Nelson, Daniel Simon, Nick Papanikolopolos throughout this work.

Part of this research was done under a PostDoctoral fellowship for Eve Coste-Manière with a support from the INRIA: *Institut National de Recherche en Informatique et en Automatique* (France). Partial support for Philippe Couvignou was provided by *Renault Automobiles, Direction de la Recherche,* under an exchange program VSNA (*Volontaires Service National Actif*). This work also supported, in part, by Sandia National Laboratory under contract AC-3752-D. The views and conclusions expressed in this document are those of the authors, and do not necessarily reflect the official policies, either expressed or implied, of the funding agencies.

References

1. Anderson, B. D. and Moore, J. B.: *Optimal Control-Linear Quadratic Methods*, Prentice-Hall, 1990.
2. Boussinot, F. and de Simone, R.: The ESTEREL language, *Proc. IEEE* Special Issue, **79**(9) (Sep. 1991).
3. Canny, J. F.: A computational approach to edge detection, *IEEE/PAMI*, **8**(6) (Nov. 1986), 679–698
4. Coste-Manière, E., Espiau, B., and Rutten, E.: Task-level programming combining object-oriented design and synchronous approach, in *IEEE Internat. Conf. on Robotics and Automation*, Nice, May 1992, pp. 2751–2756.
5. Coste-Manière, E., Espiau, B. and Simon, D.: Reactive objects in a task-level open controller, in *IEEE Internat. Conf. on Robotics and Automation*, Nice, May 1992, pp. 2732–2737.
6. Coste-Manière, E. and Gatenholm, M.: A contribution of visual servoing techniques to robotics deburring, in *SPIE: Intelligent Robots and Computer Vision XII: Algorithms and Techniques*, Boston, September 7–10, 1993, pp. 344–355.
7. Couvignou, P. and Khosla, P. K.: Singularities in the recovery of rigid body motions from aptical flow data, Technical Report to appear, Carnegie Mellon University, The Robotics Institut, 1993.
8. Couvignou, P., Papanikolopoulos, N., and Khosla, P. K.: Hand-eye robotic visual servoing around moving objects using active deformable models, in *IEEE/RSJ Internat. Conf. on Intelligent Robots and Systems (IROS'92)*, Vol. 3, Raleigh, July 1992.
9. Deriche, R.: Using canny's criteria to derive a recursively implemented optimal edge detector, *Int. J. Computer Vision* (1987), 167–187.
10. Deriche, R. and Giraudon, G.: Accurate corner detection: an analytical study, Technical Report 1420, INRIA Research Report, April 1991.
11. Espiau, B., Chaumette, F., and Rives, P.: A new approach to visual servoing in robotics, *IEEE Trans. on Robotics and Automation* **8** (1992), 313–326.
12. Espiau, B., Merlet, J. P., and Samson, C.: Force-feedback control and non-contact sensing: a unified approach, in *Romansy-90*, Krakow, Poland, July 1990.
13. Espiau, B. and Rives, P.: Closed-loop recursive estimation of 3d features for a mobile vision system, in *IEEE Internat. Conf. on Robotics and Automation*, Raleigh, April 1987, pp. 1436–1443.

14. Feddema, J. T. and Lee, C. S. G.: Adaptive image feature prediction and control for visual tracking with a hand-eye coordinated camera, *IEEE Trans. on Robotics and Automation* **20**(5) (1990), 1172–1183.
15. Feddema, J. T., Lee, C. S. G., and Mitchell, O. R.: Automatic selection of image features for visual servoing of a robot manipulator, in *IEEE Internat. Conf. on Robotics and Automation*, Scottsdale, May 1989, pp. 832–837.
16. Jang, W. and Bien, Z.: Feature-based visual servoing of an eye-in-hand robot with improved tracking and performance, in: *IEEE Internat. Conf. on Robotics and Automation*, Sacramento, April 1991, pp. 2254–2260.
17. Khosla, P. K. and Kanade, T.: Experimental evaluation of non-linear feedback and feedforward control schemes for manipulators, *Int. J. Robotics Res.* **7**(1) (Feb. 1988).
18. Khosla, P. K. and Kanade, T.: Real-time implementation and evaluation of the computed torque scheme, *IEEE Trans. Robotics Automation* **5**(2) (April 1989), 245–253.
19. Sanderson, A. C. and Neuman, C. P.: Dynamic sensor-based control of robots with visual feedback, *IEEE J. Robotics Automation* **3**(5) (1987), 404–417.
20. Papanikolopoulos, N., Khosla, P. K. and Kanade, T.: Vision and control techniques for robotic visual tracking, in *IEEE Internat. Conf. on Robotics and Automation, April 1991, pp. 857–864.*
21. Samson, C., Le Borgne, M. and Espiau, B.: *Robot Control: the Task-Function Approach*, Clarendon Press, Oxford 1991.
22. Samson, C. and Espiau, B.: Application of the task-function approach to sensor-based control of robot manipulators, in *IFAC*, Tallinn, USSR, August 1990.
23. Simon, D. and Joubert, A.: Orccad: Towards an open robot controller computer aided design system, Technical Report 1396, INRIA Research Report, February 1991.
24. Stewart, D. B., Schmitz, D. E., and Khosla, P. K.: Implementing real-time robotic systems using chimera-II, in *Proc. IEEE Internat. Conf. on Robotics and Automation*, September 1990, pp. 598–603.
25. Weiss, L. E.: Dynamic visual servo control of robots. An adaptative image based approach, Technical Report CMU-RI-TR-84-16, The Robotics Institute, Carnegie-Mellon University, 1984.