Applying the Synchronous Approach for Real Time Active Visual Reconstruction

Eric Marchand, Éric Rutten, François Chaumette

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Applying the Synchronous Approach for Real Time Active Visual Reconstruction

Eric Marchand, Eric Rutten , François Chaumette

N° 2383

Octobre 1994

PROGRAMMES 4 et 2

ISSN 0249-6399

Applying the Synchronous Approach for Real Time Active Visual Reconstruction

Eric Marchand*, Eric Rutten **, François Chaumette***

Programmes 4 et 2 — Robotique, image et vision — Calcul symbolique, programmation et génie logiciel
Projet Temis et EP-ATR

Rapport de recherche n ° 2383 — Octobre 1994 — 27 pages

Abstract: In this paper, we apply the synchronous approach to real time active visual reconstruction. It illustrates the adequateness of SIGNAL, a synchronous data flow programming language and environment, for the specification of a system dealing with various domains such as robot control, computer vision and programmation of hierarchical parallel automaton. More precisely, one application consists in the 3D structure estimation of a set of geometrical primitives using an active vision paradigm. At the level of camera motion control, the visual servoing approach (a closed loop with respect to vision data) is specified and implemented in SIGNAL as a function from sensor inputs to control outputs. Furthermore, the 3D reconstruction method is based on the "structure from controlled motion" approach (constraining camera motion for optimal estimation). Its specification is made in parallel to visual servoing, and involves the delay mechanism of SIGNAL for the specification of filters. This reconstruction involves focusing on each object; we thus present a perception strategy for connecting up several estimations, using tasks hierarchies interruption and time intervals in SIGNAL. The integration of these techniques is validated experimentally by their implementation on a robotic cell, from which we present experimental results.

Key-words: Synchronous language, real-time, visual servoing, structure from controlled motion, perception strategy

(Résumé : tsvp)

*IRISA / INRIA e-mail marchand@irisa.fr
**IRISA / INRIA e-mail rutten@irisa.fr
***IRISA / INRIA e-mail chaumett@irisa.fr

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

Application de l'approche synchrone à la reconstruction de scène par vision active

Résumé : Dans cet article, nous utilisons l'approche synchrone pour la spécification et l'implémentation d'une application temps réel de reconstruction de scène par vision active en utilisant une caméra embarquée sur un robot manipulateur. Cette application illustre l'adéquation de SIGNAL, langage synchrone à flots de données muni d'un environnement complet de programmation, pour la spécification de systèmes basés sur des domaines aussi variés que la commande de robot, la vision par ordinateur et la programmation d'automates parallèles hierarchiques. Plus précisément l'application présentée concerne la reconstruction 3D d'un ensemble de primitives géométriques en contrôlant, par vision active, le mouvement de la caméra. Concernant ces aspects de contrôle, nous utilisons les techniques d'asservissement visuel qui sont spécifiées et mises en œuvre en utilisant SIGNAL. On considère l'asservissement comme une fonction dont les entrées sont fournies par le capteur de vision, et dont la sortie est constituée des consignes à envoyer au controleur du robot. Par ailleurs, la méthode de reconstruction 3D que nous avons développée est fondée sur la vision active, ce qui signifie que le mouvement de la caméra est contraint afin d'obtenir une estimation optimale. Sa spécification est faite en parallèle avec le contrôle de mouvement du robot et fait intervenir le mécanisme de délai de SIGNAL, notamment pour la spécification de filtres. Cette méthode de reconstruction, en raison des contraintes introduites, nécessite une focalisation sur chaque objet de la scène. Nous présentons donc une stratégie de perception permettant d'enchaîner plusieurs estimations en utilisant les hiérarchies de préemption de tâches et les intervalles de temps de SIGNAL. L'intégration de ces techniques a été validée expérimentalement par leur mise en œuvre sur une cellule de vision robotique, et nous présentons des résultats expérimentaux d'asservissement visuel et de reconstruction de scènes obtenus sur cette cellule.

Mots-clé : Langage synchrone, temps réel, asservissement visuel, reconstruction à partir du mouvement, stratégies de perception

1 Introduction

In this paper we apply the synchronous approach to real time active visual reconstruction. We present the integration of different new techniques for the structure estimation of a robot environment by means of an active vision scheme. Recovering 3D structure from images is one of the main issues in computer vision [1][12][15][16][36][34][37]. The approach we have chosen to get an accurate three-dimensional geometric description of a scene is based on the active vision paradigm and consists in controlling the motion of a moving camera. The idea of using active schemes to address vision issues has been recently introduced [3][5]. Here, the purpose of active vision is to constrain the camera motion in order to improve the quality of the perceptual results. Such constraints are ensured using *the visual servoing approach* [14] which is based on the task-function framework [30] to define the sensor-based control of the robot; in our case, the sensor is a camera mounted on the end effector of a robot arm.

The technique involved for the integration is the *synchronous approach to reactive real time systems* [6]. One way of interpreting the synchrony hypothesis consists in considering that computations produce values that are relevant within a single instant of time. A family of languages is based on this hypothesis [18]. They are provided with environments featuring tools supporting specification, formal verification and generation of executable code, all based on their formal semantics. Among them, SIGNAL is a real-time synchronized data-flow language [24]. Its model of time is based on instants, and its actions are performed within the instants; extensions we propose in this paper provide constructs for the specification of durational tasks.

The synchrony hypothesis clearly applies to the equations defining a sensor-based control law, and benefits to the implementation of the corresponding control loop. Classical asynchronous languages are less adapted to specify and program the algorithms involved in this vision problem because they do not handle properly the simultaneousness of the values involved in equations. Therefore, we propose the use of synchronous languages. This is illustrated by the application of SIGNAL to the specification and implementation of the system, where the adequateness of this language is exploited at the various levels of the application. Such an application allows us to show benefits of using SIGNAL in the following domains involved in robotics and computer vision: robot control, estimation algorithms, and task level programming. Indeed, the active visual reconstruction problem presented in this paper is handled at three levels.

The lowest level concerns the *control of the camera motion*. A new approach to vision-based control was introduced a few years ago [14]. The basic idea consists in considering a vision system as a specific sensor dedicated to a task and included in a control servo loop. At this level, a robot task is seen as a data flow function computing the flow of control values for the actuator from the flow of sensor input data.

The second level concerns the *structure estimation aspect*. Embedded in the same formalism, the "*structure from controlled motion*" paradigm allows us to give an optimal estimation of the parameters of a 3D geometrical primitive [10]. Its specification involves parallelism with the motion control task, as well as a dynamical aspect, in that it is defined in function of past measured values.

The high level deals with *perception strategies*. Since the proposed structure estimation method involves to focus on the considered primitive, it has to be successively performed for each primitive of the scene. Developing perception strategies to get the spatial organization of complex scenes is thus necessary. There, the task-level programming consists in specifying different robot tasks and sequencing them by associating them with modes on which they are enabled [25][29]. For the specification of such hierarchical and parallel transition systems, we extend SIGNAL with the notions of *task* and *time interval* [28].

The remainder of this paper is organized as follows: Section 2 is devoted to image-based control loop description and its specification. In Section 3, structure from motion aspects based on an active vision paradigm are considered. Section 4 is devoted to perception strategies and their specification in

terms of a hierarchy of tasks. Real-time experimental results dealing with the implementation of the three described levels are finally presented in Section 5.

2 Equational Specification of Visual Servoing

Two main approaches are currently used in robot control based on visual data [35]: the *position-based control* which is achieved by computing, from the visual data, the 3D position and orientation of the camera with respect to its environment, and the *image-based visual servoing*, which consists in specifying a task as the regulation in the image of a set of visual features [2][14][17][20][21][22]. In the remainder of this paper, we will only refer to this last approach since it is able to provide robust and stable closed-loop control laws. This section recalls the application of the task function approach to visual servoing and the expression of the resulting control law, before the presentation of its specification in SIGNAL.

2.1 Visual Sensing - the Interaction Matrix

We first examine what data can be extracted from an image and incorporated in a vision-based control scheme. In fact, it has been shown [14] that such an ability relies on the explicit knowledge of the spatio-temporal evolution of a visual feature with respect to camera motion (in the following, we represent this evolution by *the interaction matrix* related to the considered feature).

Let us model a camera by a perspective projection (see Fig. 1). Without loss of generality, the camera focal length is assumed to be equal to 1, so that any point with coordinates $\underline{x} = (x, y, z)^T$ is projected on the image plane as a point with coordinates $\underline{X} = (X, Y, 1)^T$ with:

$$\underline{X} = \frac{1}{z}\,\underline{x} \tag{1}$$

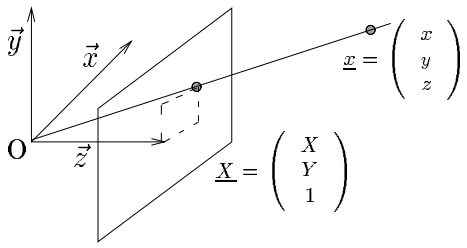

Figure 1: A simple camera model

Let us consider a geometrical primitive $\mathcal{P}_s$ of the scene; its configuration is specified by an equation of the type:

$$h(\underline{x}, \underline{p}) = 0 \ , \forall \underline{x} \in \mathcal{P}_s \tag{2}$$

where h defines the kind of the primitive and the value of parameter vector $\underline{p}$ stands for its corresponding configuration.

Using the perspective projection equation (1), we can define from (2) the two following functions [14]:

$$\begin{cases} g(\underline{X}, \underline{P}) = 0 \ , \forall \underline{X} \in \mathcal{P}_i \\ 1/z = \mu(\underline{X}, \underline{p}_0) \end{cases} \tag{3}$$

where:

- $\mathcal{P}_i$ denotes the projection in the image plane of $\mathcal{P}_s$
- g defines the kind of the image primitive and the value of parameter vector $\underline{P}$ its configuration.
- function μ gives, for any point of $\mathcal{P}_i$ with coordinates $\underline{X}$, the depth of the point of $\mathcal{P}_s$ the projection of which results in point $\underline{X}$.
- parameters $\underline{p}_0$ describe the configuration of μ and are function of parameters $\underline{p}$.

More precisely, for planar primitives (a circle for example), the function μ represents the plane in which the primitive lies. For volumetric primitives (sphere, cylinder, torus,...), function g represents the projection in the image of the primitive limbs and function μ defines the 3D surface in which the limbs lie (see Fig. 2). Function μ will be therefore called the limb surface.

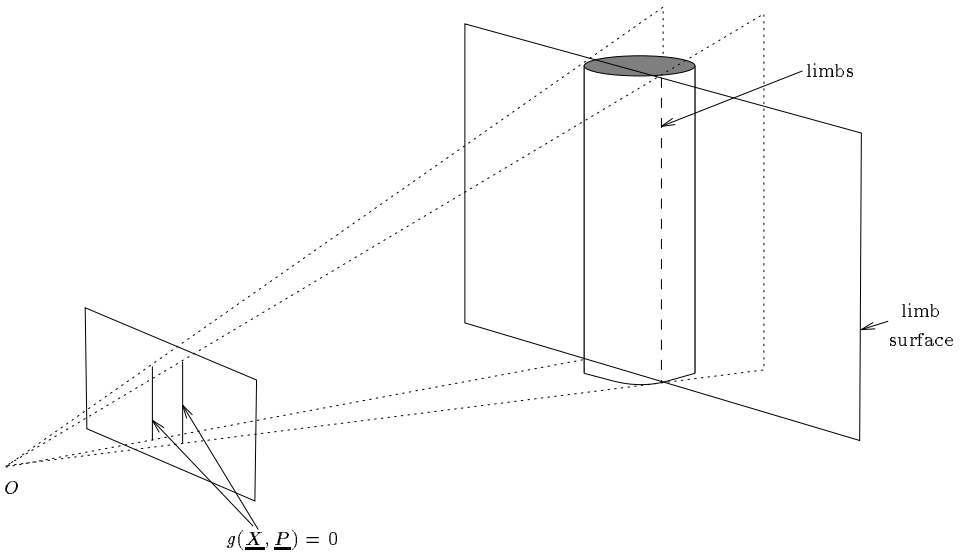

Figure 2: Projection of the primitive in the image (g) and limb surface (μ)

Let $T_c = (V, \Omega)^T$ be the camera kinematic screw where $V = (V_x, V_y, V_z)$ and $\Omega = (\Omega_x, \Omega_y, \Omega_z)$ represent its translational and rotational components. The time variation of $\underline{P}$, which links the motion of the primitive in the image to the camera motion T_c, can be explicitly derived [14] and we get:

$$\dot{\underline{P}} = L_{\underline{P}}^T(\underline{P}, \underline{p}_0)\, T_c \tag{4}$$

where $L_{\underline{P}}^T(\underline{P}, \underline{p}_0)$, called the interaction matrix related to $\underline{P}$, fully characterizes the interaction between the camera and the considered primitive. involved in

In [35] and [19], an experimental learning approach is proposed to compute the interaction matrix related to points. But it is also possible to derive it in an explicit way [17]. Indeed, in that case of a point where $\underline{P} = (X, Y)^T$, the related interaction matrix is given by:

$$L_{\underline{P}}^T = \begin{pmatrix} -1/z & 0 & X/z & XY & -(1+X^2) & Y \\ 0 & -1/z & Y/z & 1+Y^2 & -XY & -X \end{pmatrix}. \tag{5}$$

More generally, in [14], a systematic method for computing the interaction matrix of any set of visual features corresponding to geometrical primitives (lines, spheres, cylinders,...) is proposed.

We may thus choose as visual features in a visual servoing framework the parameters $\underline{P}$ which describe the configuration of one or several primitives observed in the image (such as the coordinates of a point, the orientation and distance to origin of a line, the inertial moments of an ellipse, etc) or,

more generally, any differentiable expression obtained from $\underline{P}$ (such as the distance between a point and a line, the orientation between two lines, etc).

The design of a vision-based task now consists in selecting the visual features $\underline{P}$, able to realize the specified task, and their desired value $\underline{P}_d$ to be reached in the image. As shown in the next section, the control law able to perform such a task is essentially based on the interaction matrix related to $\underline{P}$ (we will see in Section 3 that the interaction matrix is also involved in our 3D structure estimation method).

2.2 Expression of the Task Function and Control

Embedding visual servoing in the task function approach [30] allows us to take advantage of general results helpful for analysis and synthesis of efficient closed loop control schemes. We only recall the obtained results, all the developments being fully described in [30] and, in the particular case of vision-based control, in [14]. We define a vision-based task, $\underline{e}_1$:

$$\underline{e}_1 = C\,(\underline{P} - \underline{P}_d) \tag{6}$$

where:

- $\underline{P}_d$ is the desired value of the selected visual features;
- $\underline{P}$ is their current value, measured from the image at each iteration of the control law;
- C is called combination matrix and can be defined as:
  - $C = W L_{\underline{P}}^{T+}(\underline{P}, \widehat{\underline{p}_0})$ if the 3D parameters $\underline{p}_0$, involved in the interaction matrix, can be estimated on-line (using for example the 3D structure estimation method we present in Section 3). In that case, W is defined as a full rank matrix such that Ker W = Ker $L_{\underline{P}}^T$.
  - $C = W L_{\underline{P}}^{T+}(\underline{P}_d, \underline{p}_{0_d})$ if the value of the interaction matrix can not be updated at each iteration of the control law. Assumptions on the shape and on the geometry of the considered primitives in the scene have thus generally to be done in order to compute the desired values $\underline{p}_{0_d}$. Such a choice allows us to avoid the on-line estimation of parameters $\underline{p}_0$. In that case, we set W as a full rank matrix such that Ker W = Ker $L_{\underline{P}}^T(\underline{P}_d, \underline{p}_{0_d})$.

When the vision-based task does not constrain all the six camera degrees of freedom, a secondary task, such as a trajectory tracking, can be combined with $\underline{e}_1$. It can be expressed as the minimization of a cost function h_s, with gradient function $\underline{g}_s$. The task function $\underline{e}$, minimizing h_s under the constraint $\underline{e}_1 = 0$, takes the form:

$$\underline{e} = \; W^+ \underline{e}_1 + (\mathbb{I}_6 - W^+ W)\, \underline{g}_s^T \tag{7}$$

where W^+ and $\mathbb{I}_6 - W^+W$ are two projection operators which guarantee that the camera motion due to the secondary task is compatible with the regulation of $\underline{P}$ to $\underline{P}_d$.

A general control scheme aimed at minimizing the task function $\underline{e}$ is described in [30]. We here only present the simplified control scheme that we have used to perform the experimentations described in the final section of this paper. Similar control approaches can be found in [20] and [26].

For making $\underline{e}$ exponentially decrease and then behave like a first order decoupled system, we have [14]:

$$T_c = -\lambda \underline{e} - \widehat{\frac{\partial \underline{e}}{\partial t}} \tag{8}$$

where:

- T_c is the desired camera velocity given as input to the robot controller;

- λ is the proportional coefficient involved in the exponential convergence of $\underline{e}$;
- $\widehat{\frac{\partial \underline{e}}{\partial t}}$ can be written under the form:

$$\widehat{\frac{\partial \underline{e}}{\partial t}} = W^{+} \widehat{\frac{\partial \underline{e}_1}{\partial t}} + (\mathbb{I}_6 - W^{+}W) \frac{\partial \underline{g}_s^T}{\partial t} \tag{9}$$

The choice of the secondary cost function generally allows us to know $\frac{\partial \underline{g}_s^T}{\partial t}$. On the other hand, vector $\widehat{\frac{\partial \underline{e}_1}{\partial t}}$ represents an estimation of a possible autonomous target motion. If the target moves, this estimation has to be introduced in the control law in order to suppress tracking errors. It can be obtained using classical filtering techniques such as Kalman filter [22] [11] or $\alpha-\beta-\gamma$ filter [2]. In our case, since we are interested in the 3D reconstruction of static scenes, we will assume that $\frac{\partial \underline{e}_1}{\partial t} = 0$.

In the next section, we give a concrete example dealing with a positionning task with respect to a cylinder combined with a trajectory tracking.

2.3 Positionning with Respect to a Cylinder

We want to position the camera with respect to a static cylinder in order that the cylinder appears centered and vertical in the image. When this position is reached, a camera trajectory around the cylinder is performed. At a desired position, the cylinder equation is given by:

$$h(\underline{x}, \underline{p}) = x^2 + (z - z_d)^2 - r^2 = 0 \tag{10}$$

where r is the radius of the cylinder and z_d is the desired distance between the camera and the cylinder. The image of the cylinder is characterized by two straight lines with equations:

$$\begin{cases} D_1: & X - \rho_d = 0 \\ D_2: & X + \rho_d = 0 \end{cases} \quad \text{with } \rho_d = -r/\sqrt{{z_d}^2 - r^2} \tag{11}$$

By choosing for $\underline{P}$ the parameters ρ_1, θ_1, ρ_2 and θ_2 (such that $X\cos\theta_i + Y\sin\theta_i - \rho_i = 0, \forall (X,Y) \in D_i$), we obtain $\underline{P}_d = (\rho_d, 0, \rho_d, \pi)$. Since the cylinder radius is here assumed to be known, we will choose C as $C = W L^T_{\underline{P}|\underline{P}=\underline{P}_d}$. The interaction matrix related to $\underline{P}_d$ can easily be derived and is given by [14]:

$$L^T_{\underline{P}|\underline{P}=\underline{P}_d} = \begin{pmatrix} \lambda_\rho & 0 & -\lambda_\rho\rho_d & 0 & -(1+{\rho_d}^2) & 0 \\ 0 & 0 & 0 & -\rho_d & 0 & -1 \\ -\lambda_\rho & 0 & -\lambda_\rho\rho_d & 0 & 1+{\rho_d}^2 & 0 \\ 0 & 0 & 0 & \rho_d & 0 & -1 \end{pmatrix} \tag{12}$$

with $\lambda_\rho = -z_d/({z_d}^2 - r^2)$. Since this matrix is of full rank 4, we can choose $W = L^T_{\underline{P}|\underline{P}=\underline{P}_d}$. We therefore have:

$$\underline{e} = W^{+}(\underline{P} - \underline{P}_d) + (\mathbb{I}_6 - W^{+}W)\, \underline{g}_s^T \tag{13}$$

where:

$$W^{+} = \begin{pmatrix} \lambda_\rho/2l & 0 & -\lambda_\rho/2l & 0 \\ 0 & 0 & 0 & 0 \\ -1/2\lambda_\rho\rho_d & 0 & -1/2\lambda_\rho\rho_d & 0 \\ 0 & -1/2\rho_d & 0 & 1/2\rho_d \\ -(1+{\rho_d}^2)/2l & 0 & (1+{\rho_d}^2)/2l & 0 \\ 0 & -1/2 & 0 & -1/2 \end{pmatrix} \tag{14}$$

$$(\mathbb{I}_6 - W^+W) = \begin{pmatrix} (1+{\rho_d}^2)^2/l & 0 & 0 & 0 & \lambda_\rho(1+{\rho_d}^2)/l & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \lambda_\rho(1+{\rho_d}^2)/l & 0 & 0 & 0 & \lambda_\rho^2/l & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{15}$$

with $l = \lambda_\rho^2 + (1+{\rho_d}^2)^2$.

In order that the camera turns around the cylinder, we have specified a secondary task consisting in moving the camera with a constant velocity V_x along its $\vec{x}$ axis. The secondary cost function is then:

$$h_s = \frac{1}{2}\beta_x(x - x_0 - V_x t)^2 \tag{16}$$

where β_x is a positive scalar weight. We obtain:

$$\underline{g_s}^T = \begin{pmatrix} \beta_x(x - x_0 - V_x t) \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \frac{\partial \underline{g_s}^T}{\partial t} = \begin{pmatrix} -\beta_x V_x \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{17}$$

and the desired camera velocity T_c is finally given by:

$$T_c = -\lambda\, \underline{e} + \begin{pmatrix} \beta_x V_x (1+{\rho_d}^2)^2/l \\ 0 \\ 0 \\ 0 \\ \beta_x V_x (1+{\rho_d}^2)\lambda_\rho/l \\ 0 \end{pmatrix} \tag{18}$$

Now, by choosing $\beta_x = l/(1+{\rho_d}^2)^2$, the translational camera velocity along $\vec{x}$ axis will have the desired value V_x when $\underline{e} = 0$.

Note that the effect of the camera translational motion along its $\vec{x}$ axis is perfectly compensated by a rotational motion (see (18)) such that the cylinder always appears at its specified position in the image. From the form of $\mathbb{I}_6 - W^+W$ in (15), we can see that an other trajectory tracking can be performed: it consists in moving the camera along the $\vec{y}$ axis. Experimental results showing the realization of this task will be presented in Section 5.1

2.4 Towards implementation

From the point of view of programming, these algorithms have two specific features. First, they have an equational nature: they express relations between various flows of data, in a declarative way. In particular, the iterative aspect in the control loop (at each instant) is completely implicit. Second, they are synchronous: the equations involve values of the different quantities within the same instant. Classical programming methods are not well adapted to specify and program such algorithms. Asynchronous imperative languages require the explicit management of low level aspects of the implementation (like the sequencing of computations imposed by data dependencies). Furthermore, there is no well-founded support or model of the temporal aspects. Hence, we use the synchronous data flow language SIGNAL, providing the adequate high-level of abstraction for specification, as well as a coherent model of time.

In the following, we first briefly present the synchronous language SIGNAL, then we show how the control law presented before can be specified in this language.

2.5 Data Flow Equations in SIGNAL

SIGNAL [24] is a synchronous real-time language, data flow oriented (*i.e.,* declarative) and built around a minimal kernel of operators. This language manipulates signals, which are unbounded series of typed values, with an associated clock determining the set of instants when values are present. For instance, a signal `X` denotes the sequence $(\mathtt{x}_t)_{t\in T}$ of data indexed by time t in a time domain T. Signals of a special kind called `event` are characterized only by their clock *i.e.,* their presence. Given a signal `X`, its clock is noted `event X`, meaning the event present simultaneously with `X`. The constructs of the language can be used in an equational style to specify relations between signals *i.e.,* between their values and between their clocks. Systems of equations on signals are built using a composition construct. Data flow applications are activities executed over a set of instants in time: at each instant, input data is acquired from the execution environment. Output values are produced according to the system of equations considered as a network of operations.

The kernel of the SIGNAL language is based on four operations, defining elementary processes, and a composition operation to build more elaborate ones.

- *Functions* are instantaneous transformations on the data. For example, signal Y_t, defined by the instantaneous function f in: $\forall t,\ Y_t = f(X_{1_t}, X_{2_t}, \ldots, X_{n_t})$ is encoded in SIGNAL by: `Y := f{ X1, X2,..., Xn}`. The signals `Y`, `X1`,..., `Xn` are required to have the same clock.
- *Selection* of a signal `X` according to a boolean condition `C` is: `Y := X when C`. The operands and the result do not generally have identical clock. Signal `Y` is present if and only if `X` and `C` are present at the same time and `C` has the value `true`; when `Y` is present, its value is that of `X`.
- *Deterministic merge*: `Z := X default Y` defines the union of two signals of the same type. The clock of `Z` is the union of that of `X` and that of `Y`. The value of `Z` is the value of `X` when it is present, or otherwhise that of `Y` if it is present and `X` is not.
- *Delay Operator*, a "dynamic" process giving access to past values of a signal, will be presented in Section III-C.

Composition of processes is the associative and commutative operator "`|`" denoting the union of the underlying systems of equations. In SIGNAL, for processes P_1 and P_2, it is written: `(| P1 | P2 |)`.

Hierarchy, modularity and re-use of processes are supported by the possibility of defining process models, and invoking instances.

The SIGNAL compiler performs the analysis of the consistency of the system of equations, and determines whether the synchronization constraints between the clocks of signals are verified or not. This is based on an internal representation featuring a graph of data dependencies between operations, augmented with temporal information coming from the clock calculus. If the program is constrained so as to compute a deterministic solution, then executable code can be automatically produced (in C or FORTRAN). The complete programming environment also contains a graphical, block-diagram oriented user interface where processes are boxes linked by wires representing signals, as illustrated in Fig. 3.

2.6 Application to Visual Servoing

A robot control law, at the relatively lowest level, consists in the regulation of a task function, which is an equation $c = f(s)$ giving the value of the control c to be applied to the actuator, in terms of the values s acquired by the sensors. The control of the actuator is a continuous function f, more or less complex. Such a task can be composed of several sub-tasks, with a priority order. The implementation of such a control law is made by sampling sensor information s into a flow of values s_t, which are used to compute the flow of commands c_t: $\forall t, c_t = f(s_t)$. This kind of numerical, data flow computation

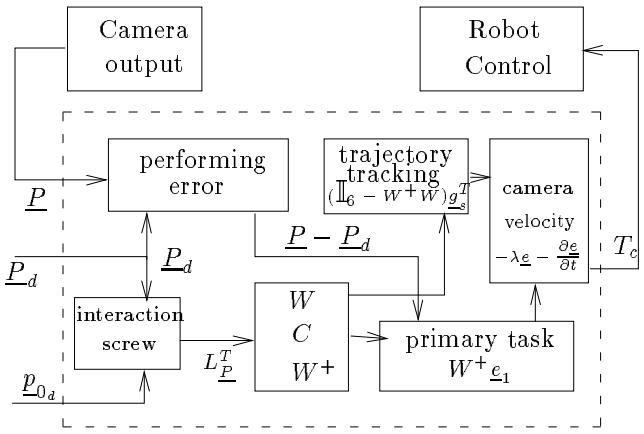

Figure 3: Modular description of a general visual servoing process,

is the traditional application domain of data flow languages in general, and of SIGNAL in particular. Furthermore, as indicated by the time index t in this schematical equation, the simultaneous presence of the values involved is adequately handled by the synchrony hypothesis.

A modular description of the visual servoing process (in the case where $C = WL_{\underline{P}}^{T}(\underline{P}_d, \underline{p}_{0\,d})$) is given in Fig. 3, also representing a block-diagram of the corresponding SIGNAL program. At a high level, the visual servoing process is composed of three different sub-modules:

- a `CAMERA_OUTPUT` module which provides a flow of image information at video rate: `P`.
- this information is received by the control module as input. This process computes the corresponding camera velocity `TC` using the task function approach.
- this camera velocity is transmitted to the `ROBOT_CONTROL` module.

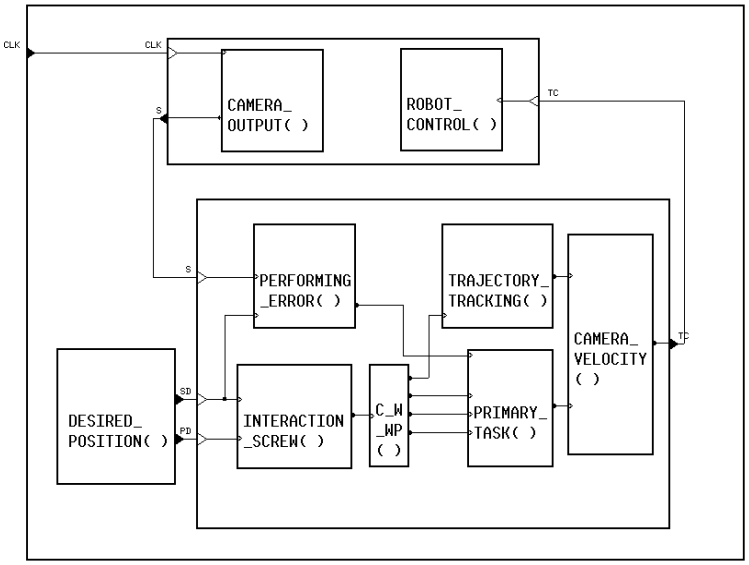

Figure 4: SIGNAL specification.

The control module itself is hierarchically decomposed into sub-modules : the `PERFORMING_ERROR` process computes the error $\underline{P} - \underline{P}_d$; the `INTERACTION_SCREW` process computes $L_{\underline{P}}^{T}(\underline{P}_d, \underline{p}_{0\,d})$; from the

output of the INTERACTION_SCREW module, a process computes the matrixes W, W^+ and C which are used with the PERFORMING_ERROR module to determine the camera velocity for the PRIMARY_TASK ($W^+\underline{e}_1$); a module performs a secondary task (here TRAJECTORY_TRACKING). This trajectory tracking is performed only when the error $\underline{P} - \underline{P}_d$ is less that some threshold ε, otherwise it is null. The final CAMERA_VELOCITY is then computed using the two flows of data coming from the PRIMARY_TASK and the secondary TRAJECTORY_TRACKING task.

In conclusion, a SIGNAL program of the control process can be written as in listing 1. Fig. 4 shows the same program presented with the graphic interface of SIGNAL.

Listing 1. The equation of the whole control process.

```
(| P := CAMERA_OUTPUT{}
 |(| L := INTERACTION_SCREW{pd,Pd}
   | error := PERFORMING_ERROR{P,Pd}
   | tau := PRIMARY_TASK{L,error}
   | traj := TRAJECTORY_TRACKING{pd} when error < epsilon default NULL_MOTION{}
   | Tc := CAMERA_VELOCITY{tau,traj}
   |)
 | SEND_VELOCITY{Tc}
 |)is
```

3 Data Flow Processes for Active 3D Reconstruction

The determination of the 3D description of a scene from 2D images is one of the main issues in computer vision. The work presented in this section is concerned with the processing of a sequence of images acquired by a moving camera to get an exact and complete description of geometrical primitives [7][10]. The camera motion will be performed using the visual servoing approach presented above. The estimation of the considered primitive will be achieved in parallel to the computation of the control law. Furthermore, we will see that performing the control law needs the use of previous values of the estimated parameters of the 3D primitive. This introduces the need for the other features of SIGNAL, in order to specify delays and parallelism.

3.1 3D Structure Estimation Using Dynamic Vision

The observability of the camera motion which is necessary for the 3D structure estimation characterizes a domain of research called dynamic vision. Approaches for 3-D structure recovery may be divided into two main classes : the discrete approach, where images are acquired at distant time instants [12][16][36] and the continuous approach, where images are considered at video rate [1][15][34][37]. The method presented here is a continuous approach which stems from the interaction matrix related to the considered primitive. Hence, this reconstruction method is embedded into the framework presented in Section 2.

As previously stated, a geometrical primitive is defined by an equation $h(\underline{x}, \underline{p}) = 0$. Using the relation between the time variation of $\underline{P}$ in the image sequence and the camera velocity T_c, we are able to compute the value of the parameters $\underline{p}$ of the considered primitive [7][10].

First, from the resolution of a linear system derived from relation (4), we obtain the parameters $\underline{p}_0$ which represent the position of the limb surface:

$$\underline{p}_0 = \underline{p}_0(T_c, \underline{P}, \dot{\underline{P}}) \tag{19}$$

Then, knowing the position of the primitive in the image described by (3) and using geometrical constraints related to the considered primitive, we can estimate the parameters $\underline{p}$ which fully define its 3D configuration:

$$\underline{p} = \underline{p}(\underline{P}, \underline{p}_0) \tag{20}$$

From a geometric point of view, this approach leads to determine the intersection between the limb surface and a generalized cone, defined by its vertex located at the optical center and by the image of the primitive (see Fig. 5).

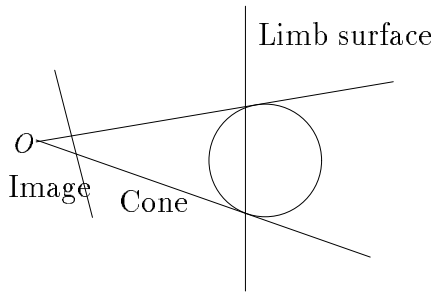

Figure 5: Continuous approach for 3D structure estimation.

This approach has been applied to the most representative primitives (*i.e,* point, straight line, circle, sphere and cylinder) [7][10]. Let us note that in the case of a cylinder, this method can be applied using the projection of only one limb in the image (let us however note that more precise results are obtained using the projections of the two limbs in the image). Such a method based on one single limb will be used to determine in Section 4.1 the nature of the observed primitive (cylinder or straight line).

3.2 3D Structure Estimation Using Active Vision

Active vision is defined in [5] as an intelligent data acquisition process. Since the major shortcomings which limit the performance of vision systems are their sensitivity to noise and their low accuracy, the aim of active vision is generally to elaborate control strategies for adaptively setting camera parameters (position, velocity,...) in order to improve the knowledge of the environment [3][31].

In our particular case, when no particular strategy concerning camera motion is defined, important errors on the 3D structure estimation can be observed. This is due to the fact that the quality of the estimation is very sensitive to the nature of the successive motions of the camera [15]. An active vision paradigm is thus necessary to improve the accuracy of the estimation results by generating adequate camera motions.

As seen on equation (19), the 3D structure estimation method is based on the measurement of $\dot{\underline{P}}$ the temporal derivative of $\underline{P}$. However, the exact value of $\dot{\underline{P}}$ is generally unreachable, and the image measurements only supply $\Delta\underline{P}$, the "displacement" of $\underline{P}$ between two successive images. Using $\Delta\underline{P}/\Delta t$ instead of $\dot{\underline{P}}$ generally induces errors in the 3D reconstruction. A sufficient and general condition that suppresses the discretization errors is to constrain the camera motion such that [10]:

$$\dot{\underline{P}} = 0, \text{ and } \dot{\underline{p}}_0 = 0, \forall t \tag{21}$$

These constraints mean that a fixation task is required. More precisely, the primitive must constantly appear at the same position in the image while the camera is moving.

Furthermore, the effect of the measurement errors on the estimation depends on the position of the projection of the primitive in the image. Therefore, the camera motion has to be constrained

in order to minimize the effects of these measurement errors. Such a minimization is obtained by a focusing task that consists in constantly observing the primitive at a particular position in the image. For example, a cylinder must appear vertical (or horizontal) and centered in the image.

A control law in closed-loop with respect to visual data is perfectly suitable to generate such a motion. In the visual servoing framework presented in Section 2, the focusing task can be expressed as the regulation of a primary task $\underline{e}_1 = C(\underline{P} - \underline{P}_d)$ to zero where $\underline{P}_d$ is the optimal position of the primitive in the image and where C is chosen as $C = L_{\underline{P}}^T(\underline{P}, \widehat{\underline{p}_0})$ (let us note that the interaction matrix is now updated at each iteration of the control loop using the measured value of $\underline{P}$ and the estimated value $\widehat{\underline{p}_0}$ of the parameters describing the limb surface). Then, a trajectory tracking has to be performed in order to realize the fixation task that suppresses the discretization error.

3.3 Parallel Dynamical Processes in SIGNAL

The described estimation scheme involves computations on the past values of signals, performed in parallel with the camera motion control. This introduces the need for constructs in the language enabling the expression of dynamical behaviors, as well as parallel ones. As mentioned in Section 2.5, the language comprises constructs enabling this:

- *delay* on the values of a signal gives access to the past value of a signal. For example, equation $ZX_t = X_{t-1}$, with initial value V_0 defines a dynamic process which is encoded in SIGNALby: `ZX := X$1` with initialization `ZX init V0`. Signals `X` and `ZX` have the same clock.

  Derived operators include delays on N instants (`$N`), and a `window M` operation giving access to a whole window in the past values (from times $t - M$ to t), as well as combinations of both operators.

  For example, a filter defined by equation $y_t = (x_t + x_{t-1} + x_{t-2})/3$ can also be written $y_t = (x_t + zx_t + zzx_t)/3, zx_t = x_{t-1}, zzx_t = x_{t-2}$, which is written in SIGNAL:

  ```
  (| Y := (X + ZX + ZZX)/3 | ZX := X$1 | ZZX := X$2 |).
  ```

- *parallelism* between processes is obtained simply with the composition operator "`|`", which can be interpreted as parallelism between processes communicating through the signals.

Let us also point out that the SIGNAL environment includes a proof system, called SIGALI, to verify dynamic properties of programs, involving state information (in the delayed signals) and transitions in reaction to occurrences of other signals.

3.4 Application to 3D Structure Estimation

1) Access to past values The estimation method used here is based on the use of the current and the past values of the position of the primitive in the image (*i.e* $\underline{P}_t$ and $\underline{P}_{t-1}$ to measure $\dot{\underline{P}}$). Furthermore, a measure of the camera position between these two instants t and $t-1$ is necessary to measure T_c (See relation (19)).

The past value of $\underline{P}$ and the camera velocity can be expressed using the delay operators. If `P` is a signal carrying the position of the primitive in the image and `Tc` the velocity of the camera, the estimation `p` of the 3D primitive parameters $\underline{p}$ is expressed as in Listing 2. Thus, the language structures meet the data flow nature of the estimation algorithm which uses at each time t the value of parameters $\underline{P}$ at time t and $t-1$.

Besides, we smooth the output of this process, by computing the mean value of the current estimation and of the two previous ones. As already stated, such a filter can also be expressed in SIGNAL with the delay operator.

Listing 2. Using the DELAY operator for the estimation.

```
(| p := ESTIMATION{P,ZP,ZTc}
 | ZP := P$1
 | ZTc := Tc$1
 |)
```

2) Parallelism The estimation process is added to the control process of Section 2 in such a way that it is executed in parallel with the control law, as shown in Fig. 6. Textually and schematically, we have

$$(|\ \texttt{TC := CONTROL\{P,P}_d\texttt{,p}_0\texttt{\}}\ |\ \texttt{p := ESTIMATION\{P,ZP,TC\}}|).$$

3) Synchronous aspect We give here another example of the interest of the synchronous hypothesis. The interaction matrix is computed for each new value provided by the measurement and the estimation processes. In the previous section (visual servoing without estimation), assumptions on the shape of the primitive had to be made; here the geometrical structure of the primitive is estimated on-line. According to the synchrony hypothesis, the value at instant t of the interaction matrix is updated using the estimated parameters and the current position of the primitive in the image at the same logical instant t.

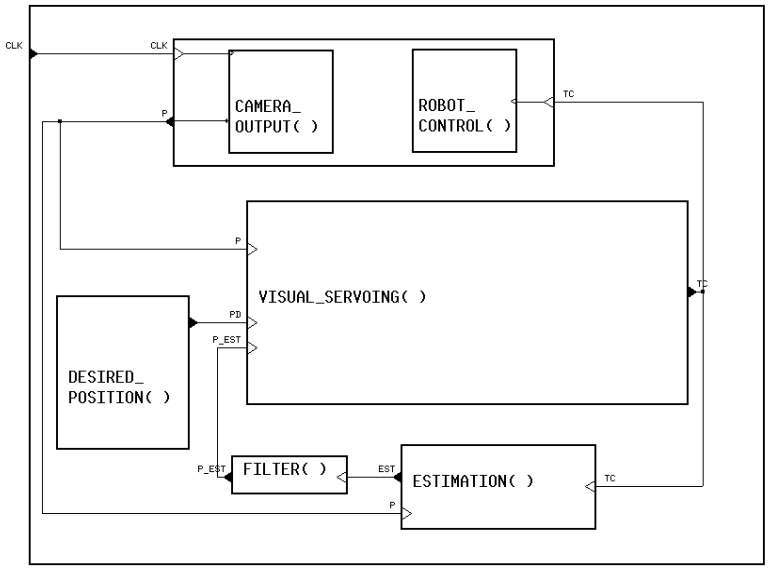

Figure 6: Control and estimation in parallel.

4 Task Sequencing for Scene Reconstruction

We are now interested in investigating the problem of recovering a precise description of a 3D scene containing several objects using the visual reconstruction scheme presented above. As already stated, this scheme involves focusing on and fixating at the considered primitive in the scene. This can be done on only one primitive at a time, hence reconstructions have to be performed in sequence. We present in this section the specification of such a sequencing which is stated in terms of a hierarchical parallel automaton [25].

Sequencings of data flow tasks are handled in an extension to SIGNAL using the notion of time interval. This enables the specification of hierarchical interruption structures, associating data flow computations to execution intervals, and making transitions from the one to the other in reaction to events.

4.1 Vision Tasks for Complex Scene Reconstruction

In order to successively perform the estimation of the 3D structure of each primitive of the scene, a database containing 2D visual data is first created. With this database, a selection process focuses on a chosen primitive, and after a recognition process which estimates the nature of the considered primitive (segment or cylinder), an optimal estimation of its 3D structure is performed. After the reconstruction of the selected primitive, the database is updated, then, a new selection is done. The scene reconstruction process ends when the database is empty. We now detail the different steps involved in this strategy.

1) Selection of a primitive We assume that the scene is only composed of polyhedral objects and cylinders, so that the contours of all the objects projected in the image plane form a set of segments. The first step in the whole scene reconstruction process is to build a 2D database composed of the set of segments in the image for the initial camera position. The database is simply obtained by extracting the edges in the image with a Shen Castan filter [32], and applying a Hough transform on the edge image which computes the equation of the different segments. A weight is given to each element of the database. This weight is function of the length and the position of the corresponding segments in the image in accordance with a given strategy. The segment with the highest weight is extracted from the database, then, an optimal estimation based on this segment is performed.

2) A maximum likelihood ratio test for primitive recognition The only information we initially have on the considered scene is composed by the set of 2D segments. We assume that these segments correspond to the projection in the image of either a limb of a cylinder, or of a 3D segment. Since the structure estimation method is specific to each kind of primitives, a preliminary recognition process is required. In order to obtain a robust criterion, we have developed the following method [25].

To determine the nature of the observed primitive, we first assume that it is a cylinder, and a one limb-based estimation is performed. When this estimation is done, two competing hypotheses can be acting, respectively:

- H_0: the observed primitive is a straight line. This hypothesis implies that we have to find a radius r close to 0 ;
- H_1: the observed primitive is a cylinder. This hypothesis implies that we have to find $r = r_1$ with $r_1 > 0$;

A maximum likelihood ratio test is used to determine which one of these two hypotheses is the right one. Let us denote L_0 and L_1 the likelihood functions associated with hypothesis H_0 and H_1. Assuming that the cylinder radius follows a Gaussian law of mean r and variance σ^2, we obtain after N estimations $r_i, i = 1 \ldots N$:

$$L_0 = (\frac{1}{2\pi\sigma^2})^{\frac{N}{2}} e^{-\frac{\sum_{i=1}^{N} {r_i}^2}{2\sigma^2}} \text{ , and } L_1 = (\frac{1}{2\pi\sigma^2})^{\frac{N}{2}} e^{-\frac{\sum_{i=1}^{N} (r_i - r_1)^2}{2\sigma^2}} \tag{22}$$

The likelihood ratio ξ is given by $\xi = \log \frac{L_1}{L_0}$ [9]. Substituting for expressions given in (22) in this equation leads to:

$$\xi = -\frac{1}{2\sigma^2}(\sum_{i=1}^{N} (r_i - r_1)^2 - \sum_{i=1}^{N} {r_i}^2) \tag{23}$$

The resulting criterion for determining the nature of the primitive can be stated as follows:

$$\max_{r_1} \xi \begin{array}{c} H_1 \\ > \\ < \\ H_0 \end{array} \zeta$$

where ζ is a predetermined threshold. The optimal parameter $\hat{r}_1$ must satisfy the relation $\frac{\partial \xi}{\partial r_1} = 0$, which leads to $\hat{r}_1 = \bar{r}$ where $\bar{r} = \frac{1}{N} \sum_{i=1}^{N} r_i$. Using this relation, ξ can finally be expressed in the simple form:

$$\xi = \frac{N \bar{r}^2}{2\sigma^2} \tag{24}$$

Clearly, hypothesis H_1 (cylinder) is selected versus hypothesis H_0 (segment) if the obtained value for likelihood ratio ξ is greater than ζ (this threshold can be easily determined by experiment). Indeed, when the primitive is a segment, the reconstruction process using one limb gives a low radius, with a very high variance (leading to a small value of ξ). On the other hand, when the primitive is a cylinder, the estimated radius is close to its real value and its variance is small (leading to a high value of ξ).

3) Optimal estimation In order to get the 3D spatial structure of a primitive, we obviously use the 3D reconstruction method described in Section 3. Dealing with the cylinder case, this method can be applied using only the projection of one limb [7]. A two limbs based estimation provides a more robust and precise estimation. However, it is impossible without *a priori* knowledge on the structure of a cylinder to determine the position in the image of its second limb. To solve this problem, we use the results obtained with the one limb-based estimation involved in the previous recognition process (if the primitive has been selected as a cylinder). These results are good enough to predict the position of the second limb in the image by projecting the 3D estimated cylinder in the image. Then, after a simple matching step, a robust estimation based on the two limbs can be performed. Each optimal estimation ends when all the primitive parameters have been accurately computed with a sufficient precision. In parallel with an optimal estimation, we can also realize a coarse estimation of other primitives selected in the 2D database. It is coarse since the camera motion is not adequate for these primitives. Therefore this leads to inaccurate results. Each coarse estimation ends when the corresponding segment gets out from the image or when the optimal estimation ends. The interest of this estimation is that it provides 3D informations about the scene which can be used if the application does not necessitate a precise estimation for all the primitives present in the scene (such an application is for example the grasping of an object, whose 3D location has to be accurately determined, while avoiding obstacles, whose 3D exact location around this object is generally less critical).

Furthermore, the optimal estimation described in Section 3 considers that the primitive has an infinite length. In order to determine its length, the vertices of the primitives have to be observed in the image, which generally implies a complementary camera motion. For accuracy issues, this motion is performed in the direction of the primitive axis, at a constant range, and until one of the two endpoints of the primitive appears at the image center. Once the camera has reached its desired position, the 3D position of the corresponding end point is computed as the intersection between the primitive axis and the camera optical axis. A motion in the opposite direction is then generated to determine the position of the other endpoint. Such a camera motion, based on visual data, is again performed using the visual servoing approach described in Section 2.

4.2 A Hierarchical Parallel Automaton as Controller

This kind of complex robotics strategy involves the use of several subsystems (such as the different tasks described in the previous section). Achieving the complete operation requires a dynamic scheduling of these elementary subsystems. An object oriented approach, based on the ESTEREL synchronous

language [8] and the ORCCAD system, has been presented to model such a controller in [33][13]. Other approaches formalize reactive behaviors of vision "guided" robot with Discrete Event Systems (DES) [4][27]. Since SIGNAL is an equational synchronous language based on DES, programming such a state transition network with this language remains in the DES framework and enables us to use the same formal tools. Furthermore, it allows us to specify combinations of tasks. Indeed, we can combine the effects of several tasks executed in parallel (*e.g.,* a primary vision-based task combined with a trajectory tracking. Another example used here is the coarse estimation of some primitives performed in parallel with the optimal estimation of an other primitive).

We thus have developed a method for connecting up several estimations based on the definition of a hierarchical parallel automaton. This automaton is able to connect up the different stages of the reconstruction process: selection, focusing, optimal estimation of the selected primitive and concurrently, coarse estimation. Each state of our automata is associated with a certain task such as the creation or the update of the database, the structure estimation process, the camera motion control using visual servoing, etc (see Fig. 7). The transitions between the states are discrete events and are function of the image data, the value of the estimated parameters of the primitives, and the state of the database.

A framework to schedule such tasks and to program such automata is now presented.

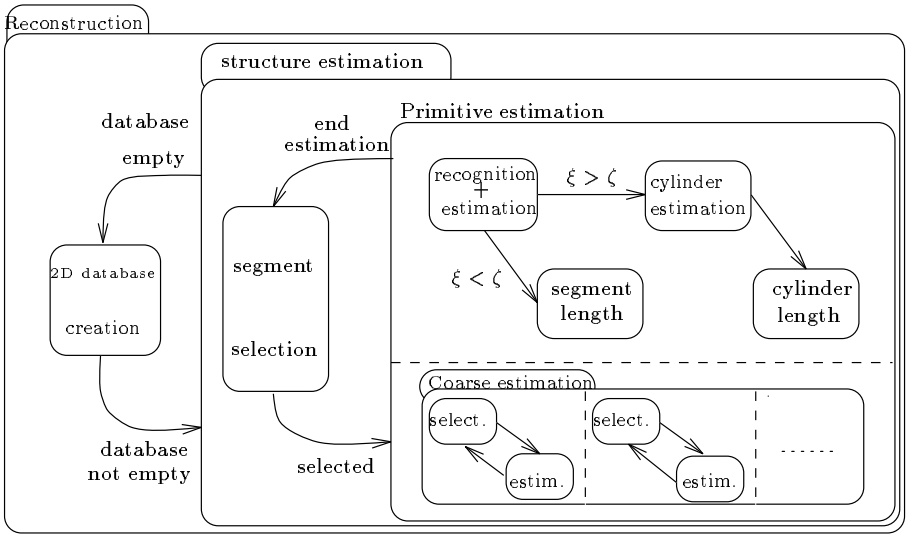

Figure 7: Hierarchical parallel automaton for the application.

4.3 Sequencing Data Flow Tasks in SIGNAL

This section introduces recent extensions to SIGNAL, handling tasks execution over time intervals and their sequencing [28]. A data flow application is executed from an initial state of its memory at an initial instant α, which is before the first event of the reactive execution. A data flow process has no termination specified in itself; therefore its end at an instant ω can only be decided in reaction to external events or the reaching of given values. Hence, ω is part of the execution, and the time interval on which the application executes is the left-open, right-closed interval $]\alpha,\omega]$.

1) Time intervals They are introduced in order to enable the structured decomposition of the interval $]\alpha,\omega]$ into sub-intervals as illustrated in Fig. 8, and their association with processes [28]. Such a sub-interval `I` is delimited by occurrences of bounding events at the beginning `B` and end `E`: `I := ]B, E]`. It has the value `inside` between the next occurrence of `B` and the next occurrence of `E`, and `outside` otherwise.

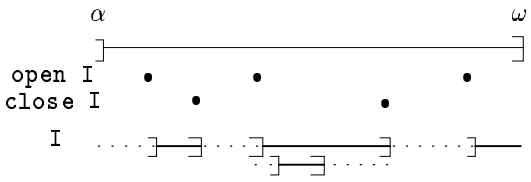

Figure 8: Time intervals sub-dividing $]\alpha,\omega]$.

Like $]\alpha,\omega]$, sub-intervals are left-open and right-closed. This choice is coherent with the behavior expected from reactive automata: a transition is made according to a received event occurrence and a current state, which results in a new state. Hence, the instant where the event occurs belongs to the time interval of the current state, not to that of the new state.

The operator `compl I` defines the complement of an interval `I`, which is `inside` when `I` is `outside` and conversely. Operators `open I` and `close I` respectively give the opening and closing occurrences of the bounding events. Occurrences of a signal `X` inside interval `I` can be selected by `X in I`, and reciprocally outside by `X out I`. In this framework, `open I` is `B out I`, and `close I` is `E in I`.

2) Tasks They consist in associating a given (sub)process of the application with a given (sub)interval of $]\alpha,\omega]$ on which it is executed. Tasks which are active on $]\alpha,\omega]$ represent the default case: they are *remanent* throughout the whole application. Inside the task interval, the task process is active *i.e.*, present and executing normally. Outside the interval, the process is inexistent *i.e.*, absent and the values it keeps in its internal state are unavailable. In some sense, it is out of time, its clock being cut.

Tasks are defined by the process `P` to be executed, the execution interval `I`, and the starting state (current, or initial) when (re-)entering the interval. More precisely, the latter means that, when re-entering the task interval, the process can be re-started at its current state at the instant where the task was *suspended* (*i.e.*, in a temporary fashion); this is written in SIGNAL `P on I`. Alternately, it can be re-started at its initial state, as defined by the declaration of all its state variables, if the task was *interrupted* (*i.e.*, aborted in a definitive fashion); this is written `P each I`. The processes associated with intervals can themselves be decomposed into sub-tasks. Hence, the specification of *hierarchies* of complex behaviors is possible.

3) Task control Task control is achieved as a result of constraining intervals and their bounding events, and associating activities to them. *Parallelism* between several tasks is obtained naturally when tasks share the same interval, or overlapping intervals. *Sequencing tasks* then amounts to constraining the intervals of the tasks. Using `on` and `each`, as defined above, enables to control activities and more elaborate behaviors can be specified. Hence, it is possible to specify hierarchical parallel automata or place/transition systems.

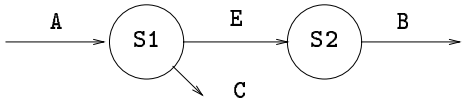

Figure 9: Transitions between states.

Each time interval holds some state information, and events cause transitions between these states. In the simple *timeout* behavior illustrated in Fig. 9, a transition leads from state `S1` to state `S2` on the occurrence of an event `E`, except if the event `C` occurs before, leading elsewhere. This can be coded

by two intervals such that the closing of the one, on the occurrence of event `E`, is the opening of the other, as in listing 3.

Listing 3. Example of time intervals

```
(| S1 := ]A , E default C]  | S2 := ]E in S1, B] |)
```

An encoding of intervals and tasks into the SIGNAL kernel is implemented as a pre-processor to the SIGNAL compiler, called SIGNAL*GTi*. Data flow and sequencing aspects are both in the *same language* framework, thus relying on the *same model* for their execution and the verification of correctness of programs. An approach related to ours integrates ARGOS (hierarchical parallel automata) with LUSTRE (data flow) [23]; we have tried to specify sequencing in a more declarative style.

4.4 Application to the Reconstruction Strategy

The visual reconstruction process based on the hierarchical parallel automaton proposed in Sections 4.1 and 4.2 has been implemented using the notion of task and time intervals defined above. This automaton is able to sequence the different vision tasks (selection, focusing, optimal estimation of the selected primitive and concurrently, coarse estimation of the other ones) and to provide a robust estimation of the spatial organization of the scene [29].

Once the automaton specification is completed, programming such an automaton is quite easy using the presented tasks sequencing framework. The source code, in SIGNAL, of the application is very close to the specification because programming is performed via the specification of constraints or relations between all the involved signals. At this step of the description, we show in Listing 4 a part of the SIGNAL code to illustrate the feasibility of encoding such a hierarchical parallel automaton with our approach based on time interval description. We illustrate these points by concrete examples:

1) Termination A data-flow process defines, like our vision tasks, a behavior, but not a termination : this aspect must be defined separately. One way of deciding on termination of a task is to apply criteria for reaching a goal : when a certain value $\underline{P}_d$ is acquired by the sensor, the task is considered to have reached its goal, hence it ends. Let us consider the case of the length computation. We have a trajectory tracking task, and the goal is defined by a certain position of the endpoint of the primitive in the image (in fact, it must appear in the center of the image). In fact, we have to minimize the error $(\underline{P} - \underline{P}_d)$ between the current position of the endpoint and its desired position. The goal is reached with a precision ε when condition $||\underline{P} - \underline{P}_d|| \leq \varepsilon$ is satisfied. The evaluation of this condition must be performed at all instants: hence, this evaluation is another data flow treatment. The instant when the condition is satisfied can be marked by a discrete event, which, causing termination of the task, can also cause a transition to another task at a higher level of the reactive sequencing. In this sense, this event can be used to specify the end of the execution interval of the task. Evaluation of such conditions can be made following a dynamic evolution: a sequence of modes of evaluation of $\underline{P}$, or of the criterion, can be defined, becoming finer (and possibly more costly) when close to interesting or important values.

2) Parallelism Parallelism between two tasks is transparent to the programmer using the composition operator. This is the case, for example, of the coarse estimation processes and the optimal estimation process. To perform these estimations, they both use the same information (*i.e.*, the measure of camera velocity, the measures performed in the image at current and previous instants), in such a way, according to the synchronous hypothesis, that they can use it at the same logic instant. In fact, we have here a parallelism of specification, and the compiler monitors all the synchronization and communication problems.

Listing 4. Program for the application

```
(| I_DBC := ] when(database_empty), when(not database_empty)] init inside
 | I_EST := comp I_DBC
 | DATABASE_CREATION each I_DBC
 | STRUCTURE_ESTIMATION each I_EST |)
Process Structure_estimation
(| I_C := ] length_estim, Segment_chosen ] init inside
 | I_R := comp I_C
 | SEGMENT_SELECTION each I_C
 | PRIMITIVE_ESTIMATION each I_R |)
Process Primitive_estimation
(| OPTIMAL_ESTIMATION
 | (| COARSE_ESTIMATION_1 | ... | COARSE_ESTIMATION_n |)|)
Process Coarse_estimation_i
(| I_R_s := ] Find_new_segment, Segment_lost ]
 | COARSE_ESTIMATION each I_R_s |)
Process Optimal estimation
(|(| I_recognition := ] close I_L_C default close I_L_S, Accuracy_reached ]
   | Accuracy_reached := when (|prec_r |< ε_r)
   | isDroite := RECOGNITION each I_recognition |)
 |(| I_CYL_EST := ] (not isDroite) when close I_recognition, Accuracy_reached_c ]
   | Accuracy_reached_c := when |prec_c |< ε_c |
   | CYLINDER_ESTIMATION each I_CYL_EST)
 |(| I_L_S := ] isDroite when close I_recognition, end_seg_length_estim ]
   | LENGTH_SEGMENT each I_L_S |)
 |(| I_L_C := ] close I_CYL_EST, end_cyl_length_estim ]
   | LENGTH_CYLINDER each I_L_C |)
  lenght_estim := end_cyl_length_estim default end_seg_length_estim |)
```

5 Experimental Results on a Robotic Cell

The whole application presented in this paper has been implemented with SIGNAL on an experimental testbed composed of a CCD camera mounted on the end effector of a six degrees of freedom cartesian robot (see Fig. 10). The image processing part is implemented in C and performed on a commercial image processing board (EDIXIA IA 1000). The implementation of the control law, the 3D structure estimation and the automata implemented using the SIGNAL language run on a SPARC Station 10. Fig. 10 shows our robot architecture.

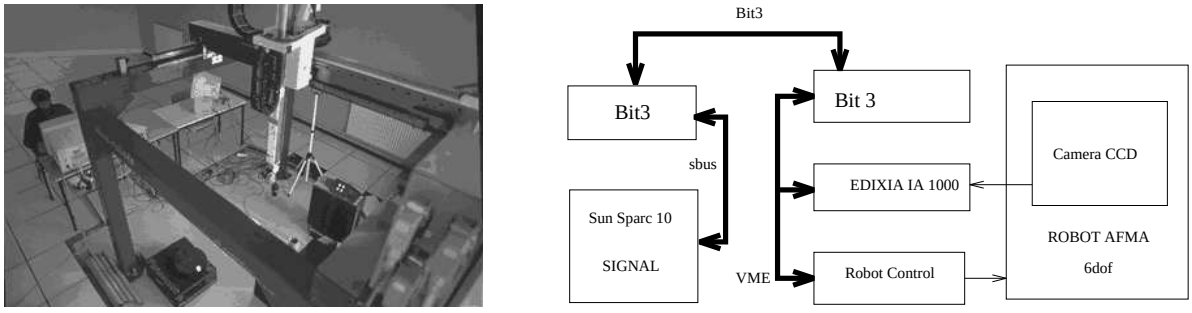

Figure 10: Experimental cell (camera mounted on a 6 dof robot) and architecture

Experimental results related to the realization of real-time visual servoing, cylinder structure estimation using active vision and scene reconstruction are presented.

5.1 Visual Servoing Results

We here present the results of the realization of the positionning task with respect to a cylinder, described in Section 2.3.

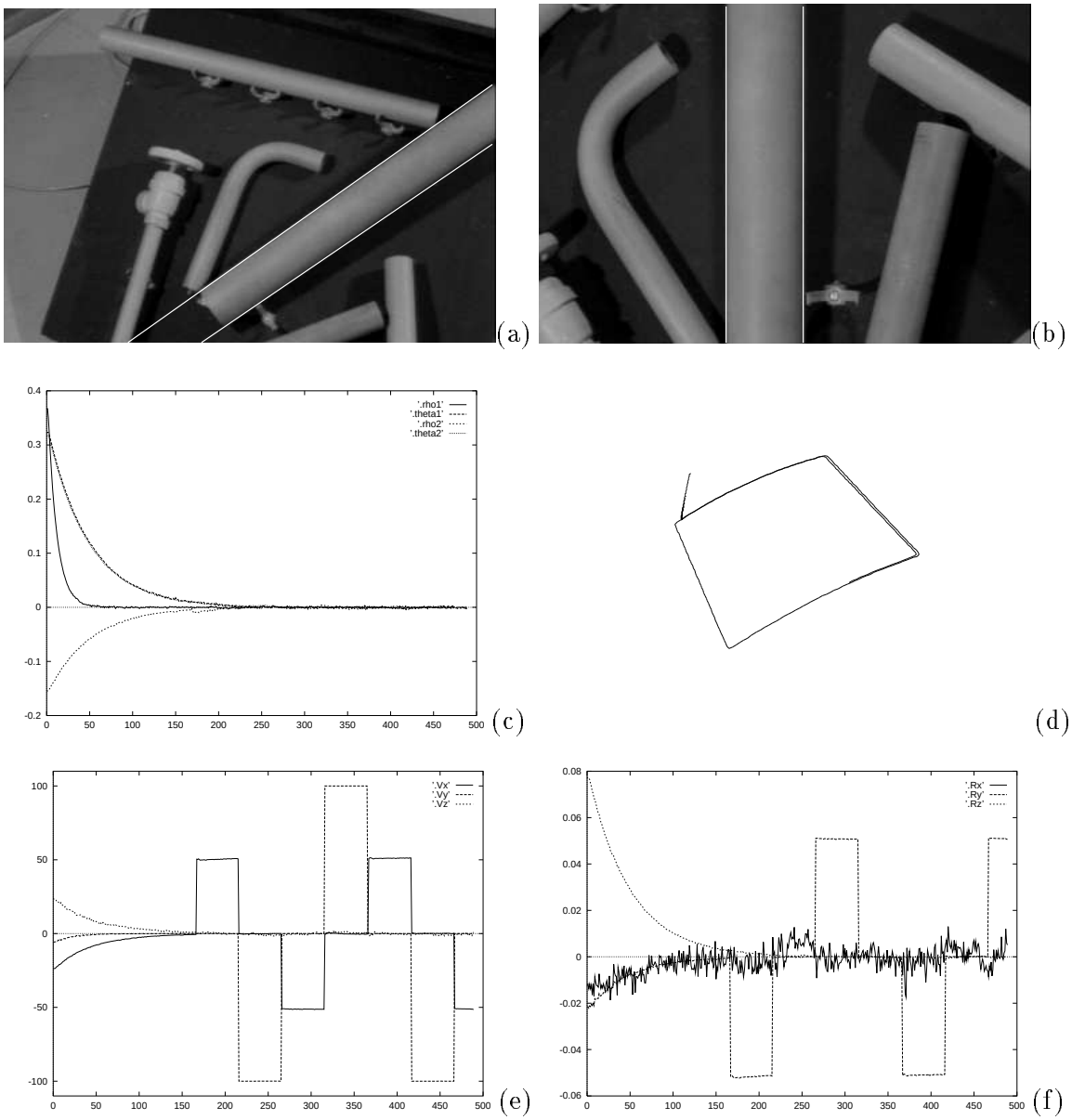

Figure 11: Positionning with respect to a cylinder and trajectory tracking.(a) initial image (b) image acquired after convergence (c) error $\underline{P} - \underline{P}_d$ (d) camera trajectory (e) translational and (f) rotational components of the camera velocity

Let us recall that we want the cylinder to appear centered and vertical in the image (note the sperimposed white lines). After the convergence of the vision-based task, a first trajectory tracking around the cylinder is performed along the $\vec{x}$ axis (as described in Section 2.3). A second trajectory is considered consisting in moving the camera along the axis of the cylinder (*i.e.*, with a velocity V_y along the camera $\vec{y}$ axis. The secondary tasks is then: $h_s = \frac{1}{2}(y - y_0 - V_y t)^2$ which leads to a camera velocity given by $T_c = -\lambda \underline{e} + (0, V_y, 0, 0, 0, 0)^T$). The complete trajectory tracking consists in moving the camera with a velocity $V_x = 5cm/s$, $V_y = 10cm/s$, $-V_x$ and $-V_y$ successively as drawn in Fig. 11.d.

Fig. 11.a represents the initial image acquired by the camera and the selected cylinder. Fig. 11.b contains the image acquired by the camera after the convergence of the vision-based task. In Fig. 11.c are plotted the four components of $\underline{P} - \underline{P}_d$. Let us point out the exponential decay of these evolutions during the convergence phase (iteration 0 to 170). The graphics shown in Fig. 11.e (respectively Fig. 11.f) represent the evolution, at each iteration of the control law, of the translational (resp. rotational) components of the camera velocity T_c. Let us note that a rotational motion compensates for the translational motion along $\vec{x}$ axis, and makes the cylinder be static in the image plane during the trajectory tracking.

5.2 Estimation of the Structure of a Cylinder

A cylinder can be represented by an equation of the type:

$$h(\underline{x}, \underline{p}) = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - (ax + by + cz)^2 - r^2 = 0, \quad (25)$$

$$\text{with} \begin{cases} a^2 + b^2 + c^2 &= 1 \\ ax_0 + by_0 + cz_0 &= 0, \end{cases}$$

where a, b, c represent the direction of its axis and x_0, y_0, z_0 are the coordinates of a point belonging to the cylinder axis. We use the proposed 3D reconstruction method to estimate those parameters. More details about this derivation can be found in [10]. In order to obtain a non-biased and robust estimation, as already stated, the cylinder must always appear centered ($\rho_1 = -\rho_2$) and horizontal ($\theta_1 = \theta_2 = \frac{\pi}{2}$) or vertical ($\theta_1 = \theta_2 = 0$) in the image sequence during the camera motion (which here consists in a motion V_x as in the previous Section).

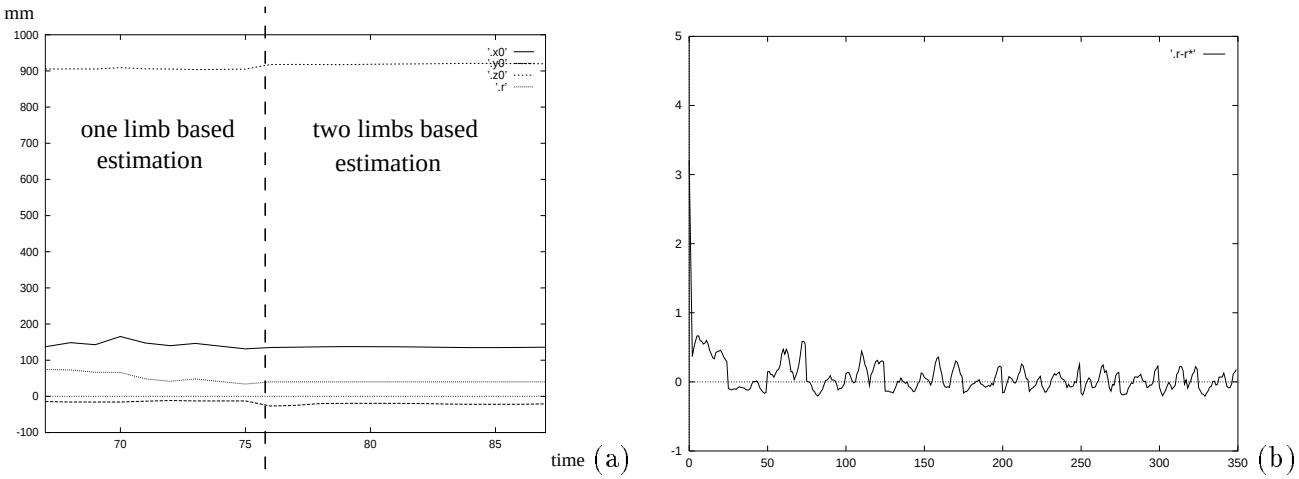

Figure 12: Estimation of the parameters of a cylinder in the camera frame

Fig. 12.a and Fig. 12.b show the evolution of the estimation of the parameters of the cylinder displayed in Fig.11.b. Fig. 12.a shows coordinates x_0, y_0, z_0 and radius r . It is divided into two parts: the estimation based on one limb and, then, the estimation based on the two limbs of the cylinder. This second estimation is far better than the first one. Let us note that the cylinder radius r is determined with an accuracy less than 0.5 mm whereas the camera is one meter away from the cylinder (and even less than 0.1 mm with good ligthting conditions). Fig. 12.b reports the error between the true value of the radius and its estimated value. As far as depth z_0 is concerned, the standard deviation is less than 1.5 mm (that is 0.15%).

Fig. 13 shows first the coarse estimation of the parameters of a second cylinder (located on the left of the image given in Fig. 11.b) obtained while performing in parallel an optimal reconstruction of the central cylinder, then the optimal reconstruction of the second cylinder. The coarse estimation