IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'95
pp. 96–102, Pittsburgh, USA, August 1995

Real Time Active Visual Reconstruction using the Synchronous Paradigm

Éric Marchand, François Chaumette, and Éric Rutten

IRISA - INRIA Rennes - Université de Rennes 1
Campus universitaire de Beaulieu - 35042 Rennes Cedex, France
Email : {marchand, chaumett, rutten}@irisa.irisa.fr

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, for the specification of a system dealing with various domains such as robot control, computer vision and the programming of hierarchical parallel automata. More precisely, our application consists in the 3D structure estimation of a set of geometrical primitives using a camera mounted on the end effector of a six dof robot. At the level of camera motion control, the visual servoing approach is specified and implemented in* SIGNAL *as a function from sensor inputs to control outputs. The 3D reconstruction method is based on the "structure from controlled motion" approach. Its specification is made in parallel to visual servoing. We also present a perception strategy for connecting up several estimations, using time intervals and hierarchical structures for task preemption in* SIGNAL*. The integration of these techniques is validated experimentally by their implementation on a robotic cell.*

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] [7][9][19]. 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][4]. 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* [8] which is based on the task-function framework [18] 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 [5]. One way of interpreting the synchrony hypothesis consists in considering that computations produce values that are relevant within a single logical instant of time. A family of languages is based on this hypothesis [12]. 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 [14]. 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 such algorithms because they do not handle properly the simultaneousness of the values involved in equations. Therefore, we propose to use SIGNAL whose adequateness 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.

The remainder of this paper is organized as follows: Section 2 is devoted to image-based control loop description and 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.

2 Equational aspect of Visual Servoing

Two main approaches are currently used in robot control based on visual data [20]: 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][8][11][13][16]. 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

We first examine what data can be extracted from an image and incorporated in a vision-based control scheme.

Let us model a camera by a perspective projection. 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:

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

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 [8]:

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

where:

- $\mathcal{P}_i$ denotes the projection of $\mathcal{P}_s$ in the image ;
- 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 $\underline{X}$; and
- 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. 1). Function μ is therefore called the limb surface.

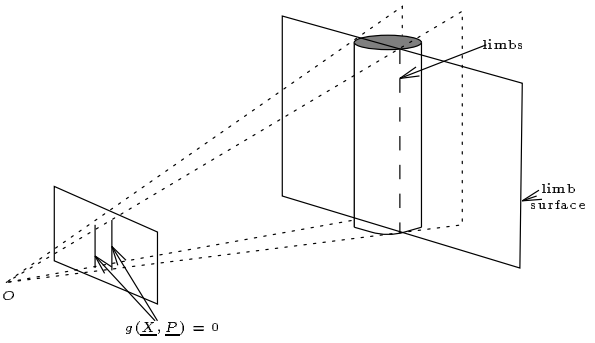


Figure 1: Image (g) of the primitive (h) and limb surface (μ) in the case of a cylinder

Let $T_c = (V, \Omega)^T$ be the camera kinematic screw. 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 [8] 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. In [8], 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 Task Function and Control

Embedding visual servoing in the task function approach [18] 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 [18] and, in the particular case of vision-based control, in [8]. We define a vision-based task:

$$\underline{e}_1 = C\ (\underline{P} - \underline{P}_d) \tag{5}$$

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 can be considered as the inverse jacobian related to the vision-based task and is defined as $C = W L_{\underline{P}}^{T^+}$, W being a full rank matrix such that Ker W = Ker L^T

When the vision-based task does not constrain all the 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 $\underline{g}_s^T$. A global task function $\underline{e}$, minimizing h_s under the constraint $\underline{e}_1 = 0$, takes the form:

$$\underline{e} = \ W^+ \underline{e}_1 + \left(\mathbb{I}_6 - W^+ W\right) \underline{g}_s^T \tag{6}$$

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 $\underline{P}_d$. For making $\underline{e}$ exponentially decrease and then behave like a first order decoupled system, we have [8]:

$$T_c = -\lambda \underline{e} - \widehat{\frac{\partial \underline{e}}{\partial t}} \tag{7}$$

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:

$$\widehat{\frac{\partial \underline{e}}{\partial t}} = W^+ \widehat{\frac{\partial \underline{e}_1}{\partial t}} + (\mathbb{I}_6 - W^+ W) \frac{\partial \underline{g}_s^T}{\partial t} \qquad (8)$$

The choice of the secondary cost function generally allows us to know $\partial \underline{g}_s^T / \partial t$. On the other hand, vector $\widehat{\partial \underline{e}_1 / \partial t}$ represents an estimation of a possible autonomous target motion. In our case, since we are interested in the 3D reconstruction of static scenes, we will assume that $\partial \underline{e}_1 / \partial t = 0$.

2.3 Towards Implementation

From the point of view of programming, visual servoing has two specific features. First, it has an equational nature: it expresses 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, it is 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.

2.4 Data Flow Equations in SIGNAL

SIGNAL [14] 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. 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}`.

Down-sampling of a signal `X` according to a boolean condition `C` is: `Y := X when C`. 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 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.

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. If the program is constrained so as to compute a deterministic solution, then executable code can be automatically produced. 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. 2.

2.5 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. 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 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 SIGNAL description of the visual servoing process is given in Fig. 2, 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 data at video rate ;
- these data are received by the control module as input. This process computes the corresponding camera velocity using the task function approach ;
- the camera velocity is transmitted to the `ROBOT_CONTROL` module.

The control module itself is hierarchically decomposed into sub-modules which compute the error $\underline{P} - \underline{P}_d$, the interaction screw $L_{\underline{P}}^T$, etc.

2.6 Visual Servoing Results

The whole application presented in this paper has been implemented with SIGNAL on an experimental testbed com-

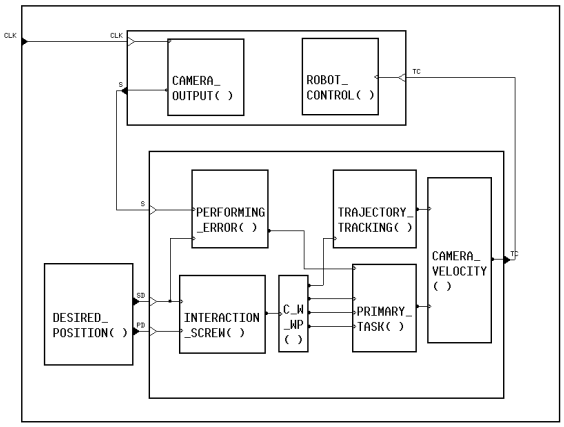


Figure 2: Graphical SIGNAL specification of visual servoing.

posed of a CCD camera mounted on the end effector of a six degrees of freedom cartesian robot.

We here present the results of the realization of the positionning task with respect to a cylinder. We want the cylinder to appear centered and vertical in the image. We have also specified successive trajectory tracking along $\vec{x}$ camera axis after the convergence of the vision-based task.

Fig. 3.a represents the initial image acquired by the camera and the selected cylinder (note the superimposed white lines). Fig. 3.b contains the image acquired by the camera after the convergence of the vision-based task. In Fig. 3.d 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. 3.c represent the evolution, at each iteration of the control law, of the translational and rotational components of the camera velocity. 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.

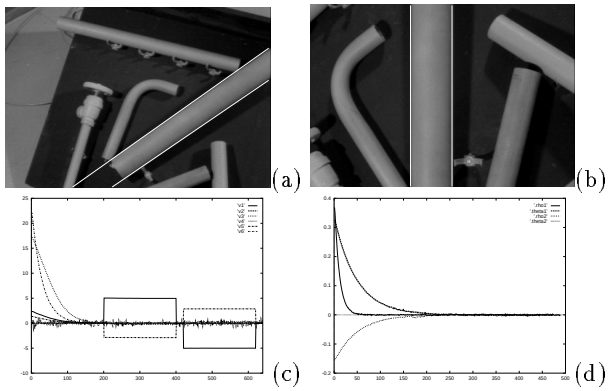


Figure 3: Positionning with respect to a cylinder.

3 Data Flow Processes for Active 3D Reconstruction

The work presented in this section concerns with the processing of a sequence of images acquired by a moving camera to get an exact and complete description of geometrical primitives [6]. 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.

3.1 Structure from controlled motion

The observability of the camera motion which is necessary for the 3D structure estimation characterizes a domain of research called dynamic vision. Approaches for 3D structure recovery can be divided into two main classes: the discrete approach, where images are acquired at distant time instants [7][10] and the continuous approach, where images are considered at video rate [1][9][19]. The method presented here is a continuous approach which stems from the interaction matrix related to the considered primitive. More precisely, we use a "*structure from controlled motion*" method which consists in constraining the camera motion in order to obtain a precise and robust estimation of 3D geometrical primitives such as points, straight lines and cylinders [6].

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 [6].

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:

$$\underline{p}_0 = \underline{p}_0(T_c, \underline{P}, \dot{\underline{P}}) \tag{9}$$

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:

$$\underline{p} = \underline{p}(\underline{P}, \underline{p}_0) \tag{10}$$

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.

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 [9]. An active vision paradigm [3][4] is thus necessary to improve the accuracy of the estimation results by generating adequate camera motions.

As seen on equation (9), 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 [6]:

$$\dot{\underline{P}} = 0, \text{ and } \dot{\underline{p}}_0 = 0, \forall t \tag{11}$$

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 effects of the measurement errors on the estimation depend 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.

A control law in closed-loop with respect to visual data is perfectly suitable to generate camera motion ensuring such constraints. In the visual servoing framework presented in Section 2, the focusing task can be expressed as the regulation of $\underline{e}_1 = C(\underline{P} - \underline{P}_d)$ where $\underline{P}_d$ is the optimal position of the primitive in the image. Then, a trajectory tracking has to be performed in order to realize the fixation task that suppresses the discretization error.

Note that this approach has been applied to the most representative primitives (*i.e,* point, straight line, circle, sphere and cylinder) [6].

3.2 Parallel Dynamical Processes and 3D Structure Estimation

Access to past values The structure estimation method 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 (9)). These past values can be easily expressed using the SIGNAL **delay** operator: the past value $\underline{P}_{t-1}$ of $\underline{P}_t$, with initial value $\underline{P}_0$ is encoded in SIGNAL with `ZP := P$1`.

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 by :

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

Thus, the language structures meet the data flow nature of the estimation algorithm.

Parallelism. Parallelism between processes is obtained simply with the composition operator "|", which can be interpreted as parallelism with signals carrying instantaneous communication between processes. Thus, 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. Textually and schematically, we have:

```
(| Tc := CONTROL{P,Pd,p}
 | p := ESTIMATION{P,ZP,ZTc}|)
```

3.3 Results in the case of a Cylinder

We use the proposed 3D reconstruction method to estimate the parameters describing the 3D configuration of a cylinder. More details about this derivation can be found in [6]. In order to obtain a non-biased and robust estimation, the cylinder must always appear centered and horizontal or vertical in the image sequence during the camera motion (which here consists in a translation along $\vec{x}$ camera axis).

Fig. 4 reports the error between the true value of the cylinder radius and its sucessive estimated value. Let us note that it 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). As far as depth is concerned, the standard deviation is less than 1.5 mm (that is 0.15%).

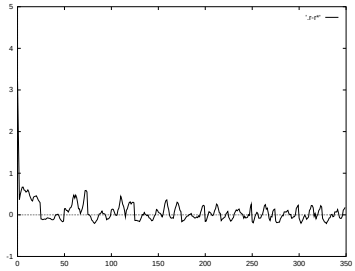


Figure 4: Error between the real and estimated radius of the cylinder selected in Fig. 3

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 [15].

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 A Hierarchical Parallel Automaton as Controller

We assume that the scene is only composed of polyhedral objects and cylinders, so that the contours form a set of segments in the image. The first step in the whole scene reconstruction process is to build a 2D database containing this set of segments. A first segment is selected from the database and a recognition process is performed. Indeed, a 2D segment may correspond to the image of either

a cylinder limb, either a 3D segment. The recognition is based on a preliminary structure estimation and a statistical test described in [15]. If a cylinder is recognized, the results obtained with the preliminary estimation are used to predict the position of the second limb in the image and a robust estimation based on the two limbs is performed. Finally, the length of the primitive is obtained by moving the camera along the primitive axis in order to successively observe its two vertices [15]. After this last process, a new segment is selected and the previous steps are repeated until all the segments of the database have been treated. Let us also note that, in parallel with an optimal estimation, a coarse estimation of other primitives can also be realized (coarse since the camera motion is not adequate for these primitives).

This kind of strategy involves the use of several subsystems (such as the different tasks described above). Achieving the complete operation requires a dynamic scheduling of these elementary subsystems. Other approaches formalize reactive behaviors of vision "guided" robot with Discrete Event Systems (DES). 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 or an optimal estimation with a coarse estimation).

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, etc. Each state of our automaton 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. 5). 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.

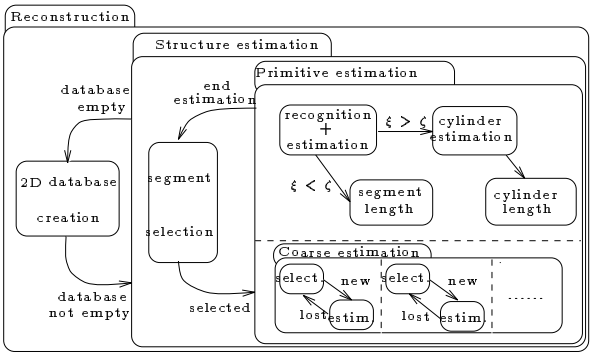


Figure 5: Hierarchical parallel automaton for the application.

4.2 Sequencing Data Flow Tasks

This section introduces recent extensions to SIGNAL: handling tasks execution over time intervals and their sequencing [17]. A data flow application is executed from an initial state of its memory at an initial instant α, and ends due to an external event ω. Time intervals have been introduced in SIGNAL in order to enable their association with processes.

Tasks consist in associating a given process of the application with a given sub-interval of $]\alpha,\omega]$ on which it is executed. Inside the task interval, the task process is active *i.e.*, present and executing normally. Outside the interval, the process is inexistent. The processes associated with intervals can themselves be decomposed into sub-tasks associated with sub-intervals. Hence, the specification of *hierarchies* of complex behaviors is possible.

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. The SIGNAL language enables that 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.

4.3 Application to Visual Reconstruction Strategy

The visual reconstruction process based on the hierarchical parallel automaton has been implemented using the notion of task and time intervals defined above. The source code, in SIGNAL (see Listing 1), 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. We illustrate these points by concrete examples:

Listing 1. Part of the SIGNAL program

```
Process Structure_estimation
(| I_C := ] end_estimation, selected]  init inside
 | I_R := comp I_C
 | SEGMENT_SELECTION  each I_C
 | PRIMITIVE_ESTIMATION  each I_R  |)
Process Primitive_estimation
(| OPTIMAL_ESTIMATION
 | (| COARSE_ESTIM_1 | ... | COARSE_ESTIM_n |)|)
```

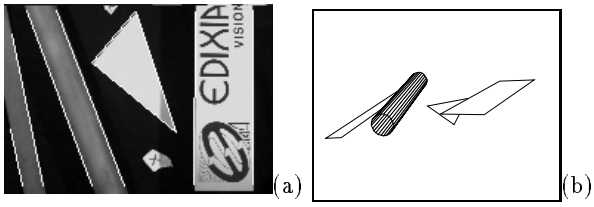
Termination. A data-flow process defines, like our vision tasks, a behavior, but not a termination: this aspect must be defined separately. One way of deciding on termination of a task is to apply criteria for reaching a goal. Let us consider the case of a visual servoing task: when the desired value $\underline{P}_d$ is acquired by the sensor, the task is considered to have reached its goal, hence it ends. So, we have to minimize the error $(\underline{P} - \underline{P}_d)$. 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.

Parallelism. Parallelism between two tasks is transparent to the programmer using the composition operator. This is the case, for example, of the control 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.

4.4 Complex Scene Reconstruction

The example reported here (see Fig. 6a) deals with a scene composed of a cylinder, a triangle, a rectangular polygon, and an oblong plinth (on the left of the image) which may look like a cylinder. The first three objects lie in the ground plane, the last one is in another plane located at 20 cm from the first one and parallel to it. Fig. 6b represents a view of the 3D reconstructed scene after the end of the automaton execution.



(a) (b)

Figure 6: (a) Scene observed from the initial position of the robot, (b) a view of the reconstructed scene

5 Conclusion

The goal of this paper was to show that synchronous languages are suitable to specify and to implement vision tasks at different levels: camera motion control, estimation algorithms, and perception strategies. The data flow framework is particularly appropriate for the specification of visual servoing because of the equational and data flow nature of the closed-loop control laws which can be implemented as control functions between sensor data and control outputs. The association of time intervals with data flow processes in order to form tasks, and the sequencing of these data flow tasks allow the specification of more complex strategies. Hence, the whole application can be specified in SIGNAL, from the discrete event driven sequencing down to the servoing loop.

Acknowledgment

This work was partly supported by the MESR within project VIA (Vision Intentionnelle et Action) and under contribution to student grant. The authors thank Samuel Ketels and Florent Martinez who have implemented in part some of the experimental work.

References

[1] G. Adiv. – Inherent ambiguities in recovering 3D motion and structure from a noisy flow field. – *IEEE Trans. on PAMI*, 11(5):477–489, May 1989.

[2] P.K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman. – Automated tracking and grasping of a moving object with a robotic hand-eye system. – *IEEE Trans. on Robotics and Automation*, 9(2):152–165, April 1993.

[3] Y. Aloimonos. – Purposive and qualitative active vision. – In *Proc. of ICPR*, pages 346–360, New Jersey, 1990.

[4] R. Bajcsy. – Active perception. – *Proc. of the IEEE*, 76(8):996–1005, August 1988.

[5] A. Benveniste and G. Berry. – Real-time systems designs and programming. – *Proc. of the IEEE*, 79(9):1270–1282, September 1991.

[6] F. Chaumette, S. Boukir, P. Bouthemy, and D. Juvin. – Optimal estimation of 3D structures using visual servoing. – In *CVPR'94*, pages 347–354, Seattle, USA, June 1994.

[7] C. Chien and J.K. Aggarwal. – Model construction and shape recognition from occluding contour. – *IEEE Trans. on PAMI*, 11(4):372–389, February 1989.

[8] B. Espiau, F. Chaumette, and P. Rives. – A new approach to visual servoing in robotics. – *IEEE Trans. on Robotics and Automation*, 8(3):313–326, June 1992.

[9] B. Espiau and P. Rives. – Closed-loop recursive estimation of 3D features for a mobile vision system. – In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 1436–1443, Raleigh, North Carolina, April 1987.

[10] O. Faugeras. – *Three-dimensionnal computer vision: a geometric viewpoint.* – MIT press, 1993.

[11] J.T Feddema, C.S.G. Lee, and O.R. Mitchell. – Weighted selection of image features for resolved rate visual feedback control. – *IEEE Trans. on Robotics and Automation*, 7(1):31–47, February 1991.

[12] N. Halbwachs. – *Synchronous programming of reactive systems.* – Kluwer, 1993.

[13] K. Hashimoto, editor. – *Visual Servoing: Real Time Control of Robot manipulators based on visual sensory feedback.* – World Scientific Series in Robotics and Automated Systems, Vol 7, World Scientific Press, Singapore, 1993.

[14] P. Le Guernic, M. Le Borgne, T. Gautier, and C. Le Maire. – Programming real time application with SIGNAL. – *Proc. of the IEEE*, 79(9):1321–1336, September 1991.

[15] E. Marchand and F. Chaumette. – Real time estimation of 3D environment with an active vision system. – In *Proc. of the Int. Workshop on Intelligent Robotic Systems*, pages 311–318, Grenoble, France, July 1994.

[16] N. Papanikolopoulos, B. Nelson, and P.K. Khosla. – Full 3D tracking using the controlled active vision paradigm. – In 7^{th} *IEEE Symposium on Intelligent Control*, pages 267–274, Glasgow, Scotland, August 1992.

[17] E. Rutten and P. Le Guernic. – The sequencing of data flow tasks in SIGNAL. – In *Proc. of the ACM SIGPLAN Workshop on Language, Compiler and Tool Support for Real-Time Systems*, Orlando, Florida, June 1994.

[18] C. Samson, B. Espiau, and M. Le Borgne. – *Robot Control: the Task Function Approach.* – Clarendon Press, Oxford, England, 1991.

[19] A.M. Waxman, B.K. Parsi, and M. Subbarao. – Closed-form solutions to image flow equations for 3D structure and motion. – *International Journal of Computer Vision*, 1(3):239–258, October 1987.

[20] L.E. Weiss, A.C. Sanderson, and C.P. Neuman. – Dynamic sensor-based control of robots with visual feedback. – *IEEE Journal of Robotics and Automation*, 3(5):404–417, October 1987.