ORCCAD : towards an open robot controller computer aided design system

Antoine Joubert, Daniel Simon

UNITÉ DE RECHERCHE
INRIA-SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 1396

Programme 6
Calcul scientifique, Modélisation et
Logiciels numériques

ORCCAD : Towards an Open Robot Controller Computer Aided Design system

Antoine JOUBERT
Daniel SIMON

Février 1991

RR-1396

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET AUTOMATIQUE

ORCCAD: Towards an Open Robot Controller Computer Aided Design system

ORCCAD: Un système de CAO pour Contrôleur Ouvert en Robotique

Antoine JOUBERT
Isia/Ensmp Sophia-Antipolis
Daniel SIMON
Inria Sophia-Antipolis

Contents

Abstract

The development of new robotic tasks deals with many differents techniques and scientific fields. Automatic Control is involved in control laws analysis while Computer Science tools are used to produce efficient real-time programs.
Following the development of a full example, this report shows a methodology for the design of new robotics tasks and gives a review of the tools to be used at each step of the task development.
This work is partially supported by the Esprit II ARMS project 2637.

Résumé

Le développement de nouvelles applications en robotique pose de nombreux problèmes, depuis l'analyse des algorithmes de commande en boucle fermée répondants aux désirs de l'utilisateur jusqu'aux problèmes de répartition et d'implantation des programmes temps-réels correspondants sur des architectures complexes.
En s'appuyant sur le traitement complet d'un exemple, ce rapport propose une méthodologie de conception de tâches robotiques et passe en revue les différents outils nécessaires à chaque étape du développement.
Ce travail est en partie soutenu par le projet Esprit II ARMS 2637.

1 Introduction

A robotic process is made of a sequence of robot actions in order to achieve a given application like an assembly process. The creation of a robotic process needs to use a wide range of knowledge. Mechanics is used to define the task to perform. Automatic control is involved in control laws design. Real-time and distributed computing tools are needed to produce efficient runtime software.
Although the global performance of robotics systems have proceeded very quickly, the development of an efficient design methodology has generally not.

The implementation of a robotics process via a multi-processor robot controller is a complex problem which requires several steps such as:

- defining the task in a formal way
- choosing a suitable control law
- splitting it into a set of elementary sub-tasks
- defining the timing and the synchronization relations between these tasks
- mapping the set of tasks on a distributed architecture

A lot of degrees of freedom are then given to the designer of a new control law matching the end-user specification.

The structure of the hardware and software ressources used by an Open Robot Controller to achieve real time control of robotic systems has been studied in a previous work [BoS90] [BSS90]. In this work, a Robot Controller is defined as the set of computational ressources (hardware and software) involved in the on-line control of a robotic system. The control software is splitted into three layers, i.e. the Application layer accessed by the end-user of the system, the Control layer programmed by an Automatic Control scientist and the system layer which contains basic tools such as a real time operating system and a set of synchronization primitives.

Our purpose is to aid the implementation of robotics tasks via various robot controller architectures using a set of CAD tools, the ORCCAD system under development. These tools deals with the problems involved by programming tasks at the two first levels defined above.

Starting from the end-user point of view and continuing up to the actual implementation on the controller, the successive levels of ORCCAD are :

- End-user level:
  Definition of a robotic application in terms of a sequence of robot actions. Referenced actions may be found in a library, they have to be specified by the end-user otherwise.
- Automatic control level:
  Definition and analysis of a set of referenced action in terms of mathematical equations in a continuous time model.
- Implementation level:
  Transfer to discrete time and implementation on hardware of the referenced actions, the so called Robot-Tasks.

The different steps of a new robot-task creation are summarized in figure 1.

The next section presents the user point of view of the specification of a robotic process, consisting mainly in using a language to describe a sequence of referenced actions. The third section describes how a new action will be specified by a specialist of automatic control, and the two last sections show how ORCCAD helps to implement these specifications on an actual architecture.

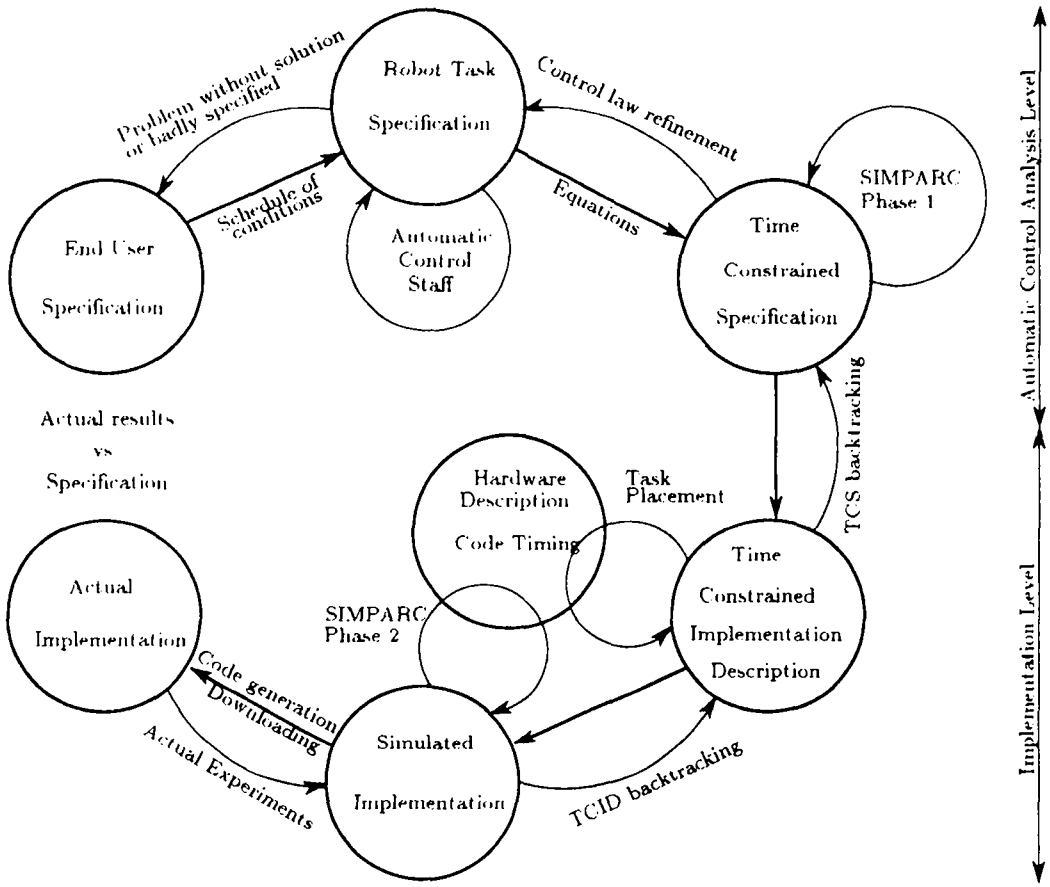

Figure 1: Robot-task creation process

2 Definition of a robotics process by an end-user

The end-user is for example working in a factory and his goal is to achieve a specific application, such as loading, deburring and unloading rough workpieces. He is not *a priori* a specialist in automatic control or in computer science.
The usual way to define such a process is to use a specific robotic programming language to describe the sequence of motions of a robot : the existing industrial robotic languages deals mainly with geometrical considerations and provide statements like : (using for instance the LM syntax [Gas87])

```
MOVE Gripper TO Frame CARTESIAN WITH SPEED=0.5;
```

or:

```
MOVE Gripper TO Target UNTIL FZ >= 0.01;
```

These languages present severe drawbacks to program advanced automation systems:

- they are closed, only the actions listed in the set of statements of the language are available (i.e., it is not possible to use visual servoing if only force sensing is provided). On the other hand, the complexity and variety of robotic applications are increasing.
- instructions like :

  ```
  MOVE Gripper TO Insertion_Axis WITH SPEED = -0.1*(FX + FY);
  ```

  allows the building of closed loop sensor based control, but with an unpredictable sampling rate for this loop. Therefore, it is not possible to set the gain of the loop. The behaviour of the robot will be unpredictable, with possibly destructive effects.
- a complex robotic system, like an automated assembly workcell, is made of several sub-systems (robots, sensors...) working in parallel: the language must provide a way to express this parallelism.

Our purpose is to provide a rich user-interface which can integrate various possibilities, such as using other information than only positions. For this purpose, we preconise the use of a language to construct the sequence of standard robot-tasks (R-T) which perform the desired application.

The Robot-Tasks are the actual implementation of control laws via the controller, such as position control in operational space, hybrid force/position control in the task space, visual servoing... Each robot-task can include various features, such as trajectory generation sub-task, observation sub-tasks, starting and ending conditions, signals to be emitted...

An example of an contour following task running in parallel with a belt advance task should be, using for instance an ESTEREL-like syntax [BeC85][ECM90]:

```
trap FAILURE in                                  %exception declaration
  [
     emit RUN_BELT;
     present WORKPIECE_IS_DETECTED               %end condition for RUN_BELT
        emit STOP_BELT;
  ]
  ||                                             %the 2 blocks run in parallel
  [
  do
   do
```

```
    do
      exec CONTOUR([1,0,0,],1,.5,.1);          %follow clockwise
    watching JOINT-LIMIT;
      exec CONTOUR([1,0,0,],-1,.5,.1);         %follow counterclockwise
   watching BACK-TO-INITIAL
            timeout emit STOP_POWER;
  watching SINGULARITY exit FAILURE;           %exception reached
  ]
handle FAILURE do
  emit TASK_FAILED;emit STOP_POWER;            %exception handler
end
```

In this simple program, the library robot-task $CONTOUR(N_p, C_k, f_d, v_d)$ is a complex hybrid position/force control task which parameters are the components of the normal vector to the plane of the contour, the contouring direction, the desired value of the normal force and the desired value of the velocity of the gripper. There must not be interference between the internal timing of such a real time control task (sampling rate of a control law) and the higher level description of the application (sequence of control laws to perform an application).

This is a modular approach, which can support any enhancement by addition of new robot-tasks, for instance when using new robots or new sensors, or when trying to solve new robotic problems involving new control law designs. A methodology is given to the automatic-control staff to add new robot-tasks in the library: This methodology is described in the next section.

3 Specification of robot-tasks by the automatic control staff

In order to create new robot-tasks, a formal specification must be achieved by the automatic-control staff. The end-user provides a task description in a language close to natural language. This description is made of several parts:

- the used ressources of the global system concerned by the task, for instance : Using a 6 degrees of freedom robot (with its physical characteristics), the wrist of which is equipped with a 6 components force sensor, moving in a given environment (obstacles, workpieces, trajectory to be followed...)

- the needed preconditions for the system before starting the task, for instance :
  A contact has been detected between the end effector of the robot and an unknown object in the environment by the force sensor.

- the actions during the task, for instance :
  Follow the contour of the object clockwise in a given plane at a given velocity exerting a specified normal force on this contour, keeping the end effector normal to the object.

- the events to be generated during the execution of the task and the actions to be taken upon reception of these events, for instance :
  the robot reaches a limit of its working space: emit JOINT-LIMIT
  the norm of the Task Jacobian reaches a given threshold: emit SINGULARITY

- the postconditions for the system after execution of the task, for instance:
  the robot comes back to the initial contact position: emit BACK-TO-INITIAL

Starting from this description, the automatic-control staff, using mathematical methods and expertise, gives a formal description of possible control laws achieving the specified action in a

continuous time model.
A systematic way for the synthesis of a control law matching the end-user specification is given by the Task-function approach [SEL91]: This approach focusses on the actions performed by the robots, expressed in terms of regulation. The problems related to the control of robots may be classified in two levels [Sam88]:
The "high" level consists of specifying the task to be performed and of choosing the vector of signals to be regulated in order to fulfill the user's objectives (choice of the task-function);
The "low" level consists of computing the control (actuators torques) for the regulation of the task-vector in connection with the available hardware.

In this approach, the task-space is defined as the set of variables to be regulated; this space may be the space of joint coordinates as in the classical approach but it may also be any other space representative of the action that we wish the robot(s) to perform. Let us cite for example the space of Cartesian coordinates, or the space of forces and torques measured at the end-effector level.
This formal specification is given in term of block-diagrams and/or equations. To this set of block-diagrams or equations are added a set of constraints on values (for instance, minimum gains to ensure the stability and/or the accuracy of the system, location of the sensors,...)

Example: a Hybrid Dynamic Position/Force control loop for contouring or deburring applications:
The goal of the control is to follow the contour of un unknown object in the plane P normal to a given vector N_p which contains the initial contact point with the end effector of the robot. The contour must be followed at a desired velocity v_d with the end effector exerting a normal force f_d on the object (primary goal). The k_6 axis of the end effector is kept normal at the contour and the i_6 axis of the end effector is kept parallel to the plane P (secondary goal). The contouring direction around N_p is choosen by the end-user. An alarm must be provided when the robot reaches a joint limit or a singularity, and a signal must be emitted when the end-effector comes back to the initial contact point.

The different steps of the design of a control law matching these requirements are detailed in appendix A. These design steps uses the the methodology given in [SEL91] and developed in [Cha90] and [EMS90] (figure 10) and leads to the following set of equations:

$$\epsilon = \begin{bmatrix} R_0^w \\ 0 \end{bmatrix} \begin{bmatrix} < i_{w|0}, T_0^6 - x_d > \\ < j_{w|0}, T_0^6 - x_d > \\ < N_{c|0}, R_0^6 f_{|6} > - f_d \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & I_3 \end{bmatrix} g_s^r$$

$$g_s^r = \alpha_1 < i_6, N_{c|0} > \begin{bmatrix} 0 \\ i_6 \times N_{c|0} \end{bmatrix} + \alpha_2 < j_6, N_{c|0} > \begin{bmatrix} 0 \\ j_6 \times N_{c|0} \end{bmatrix} + \alpha_3 < i_6, N_{p|0} > \begin{bmatrix} 0 \\ i_6 \times N_{p|0} \end{bmatrix}$$

$$x_d(0) = G(q(0))$$

$$N_{c|0} = \frac{(R_0^6 f_{|6} - f_{T|0})}{||R_0^6 f_{|6} - f_{T|0}||}$$

$$f_{T|0} = < f_{|0}, v_{|0} > \frac{v_{|0}}{||v_{|0}||}$$

$$v_{|0} = J_{L|0}(q)\dot{q}$$

$$i_w = \frac{v_{|0}}{||v_{|0}||} \qquad j_w = -N_c \times i_w$$

$$R_0^w = \begin{bmatrix} i_{w|0} & j_{w|0} & -N_{c|0} \end{bmatrix}$$

$$N_{cp_{|0}} = \frac{N_{c_{|0}} - < N_{p_{|0}}, N_{c_{|0}} > N_{p_{|0}}}{||N_{c_{|0}} - < N_{p_{|0}}, N_{c_{|0}} > N_{p_{|0}}||}$$

$$V_{t_{|0}} = (N_{cp_{|0}} \times N_{p_{|0}}) v_d C_k$$

$$x_d^{k+1} = (1 - \beta) x_d^k + \beta x^k + \Delta V_t^k$$

$$\Gamma = -k\ diag(M) J_{|0}^{-1} G(\mu De + J_{|0}\dot{q} + R_0^{wT} J_{L_{|0}} \dot{q})$$

The corresponding block-diagram is given by figure 2.
It must be pointed out that the boxes in the block-diagram represents mathematical functions

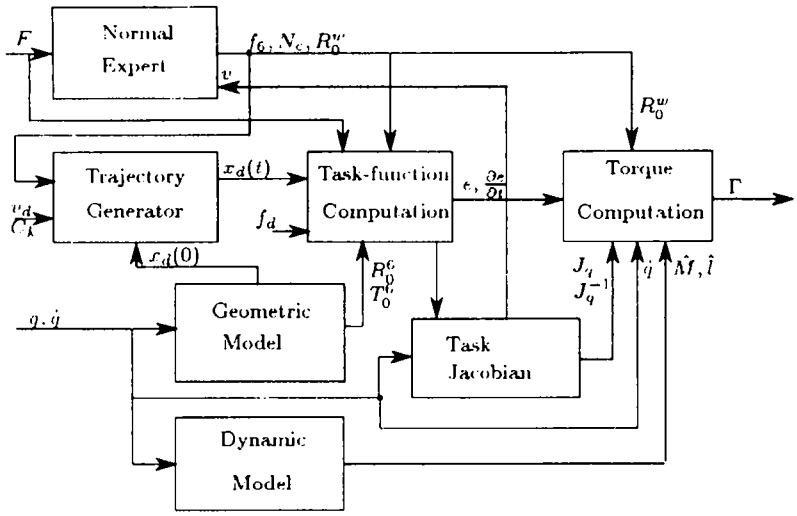

Figure 2: Block-diagram representation

which in general do not depends on the hardware. For instance, in the above example, we only need to know the number of d.o.f. of the manipulator. Until this step, the control laws generated from the end-user specification present an high level of genericity.

This formal specification will be used to define the time-constrained specification.

4 Time constrained specification of a robot task

At this stage, a cutting up is made of formal specifications into elementary programming units, the so-called Module-Tasks (M-T). They are most often periodic tasks.

The M-T's communicate with each other using private events acting onto oriented typed ports, using a communication primitive choosen in a set of eight [MSBB89]. Some of these communication primitives are blocking ones, allowing possible synchronization faults during the execution of the program which is often made of several multi-clocked loops.

The time-constrained specification (TCS) is made of the formal specification of the control law to which assertions about timing issues are added. Assertions are made about the bounds of execution time of the M-T's, their activation period and the kind of synchronization primitives used between them.

These assertions about the timing of the M-T's may be either provided from scratch, for basic automatic control analysis, or from experience of existing programs running on an available target.

The components of the TCS represents programs corresponding to a choice of hardware and algorithms for which a computation time and synchronization primitives have to be defined or assumed, allowing thus a first step towards a numerical implementation.

Starting from the TCS, ORCCAD will provide a verification of the synchronization of the network of M-T's (i.e. deadlock detection...). The model of the synchronization scheme used for this verification must take into account the interleaving of temporal events checking the activity of the tasks and the instants of data exchange. Issues from the theory and practice of synchronous languages and their associated verification tools should be useful for this analysis [BLL88][HCRP89]. In particular, it is expected that the clock calculus provided by the compiler of the data flow language SIGNAL [LGB86] will give usefull information about the tasks timing.

Very often, the M-T will be a reusable piece of code such as Inertia matrix or geometrical Jacobian of a given manipulator, communicating by their I/O ports. This decomposition of an R-T into a network of M-T will be generally close to the block-diagram representation as it is shown in figure 3.

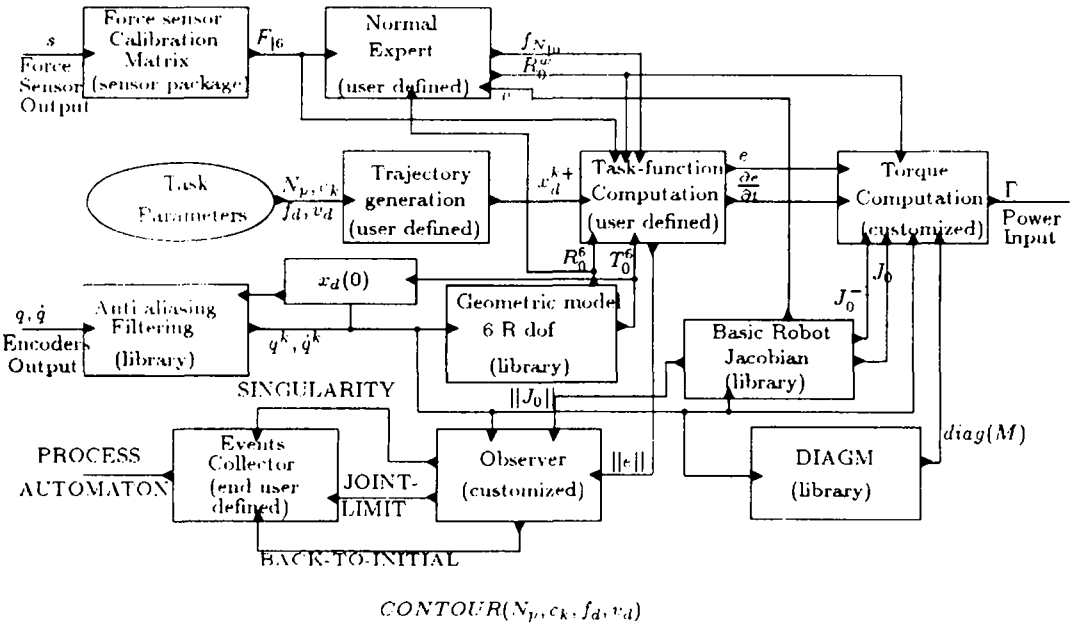

Figure 3: Module-tasks decomposition

The temporal characteristics of the components of the Robot-Task are the sampling period and bounds of execution time of the M-T's and the kind of synchronization primitive used by each port. They will have to be handled by a Graphic Editor used to design the control law. This editor, under development, will be used all along the design process by adding new properties to the components of the Robot-Task at each step. An example of module-task design with its associated properties is given by figure 4. (this example has been drawn using the existing Vanguard schematic editor for electronic boards design).

The validity of the TCS can then be checked by a simulation achieving the computation of each M-T and taking into account the temporal constraints defined in the previous step.The validity of the TCS according to the end user specification can be then checked looking at the outputs or state variables of the simulated model of the continuous system under control.

All the different M-T's used by an R-T do not have to be computed at the same rate; those involved in feedforward computation may be slowed with respect to those involved in the

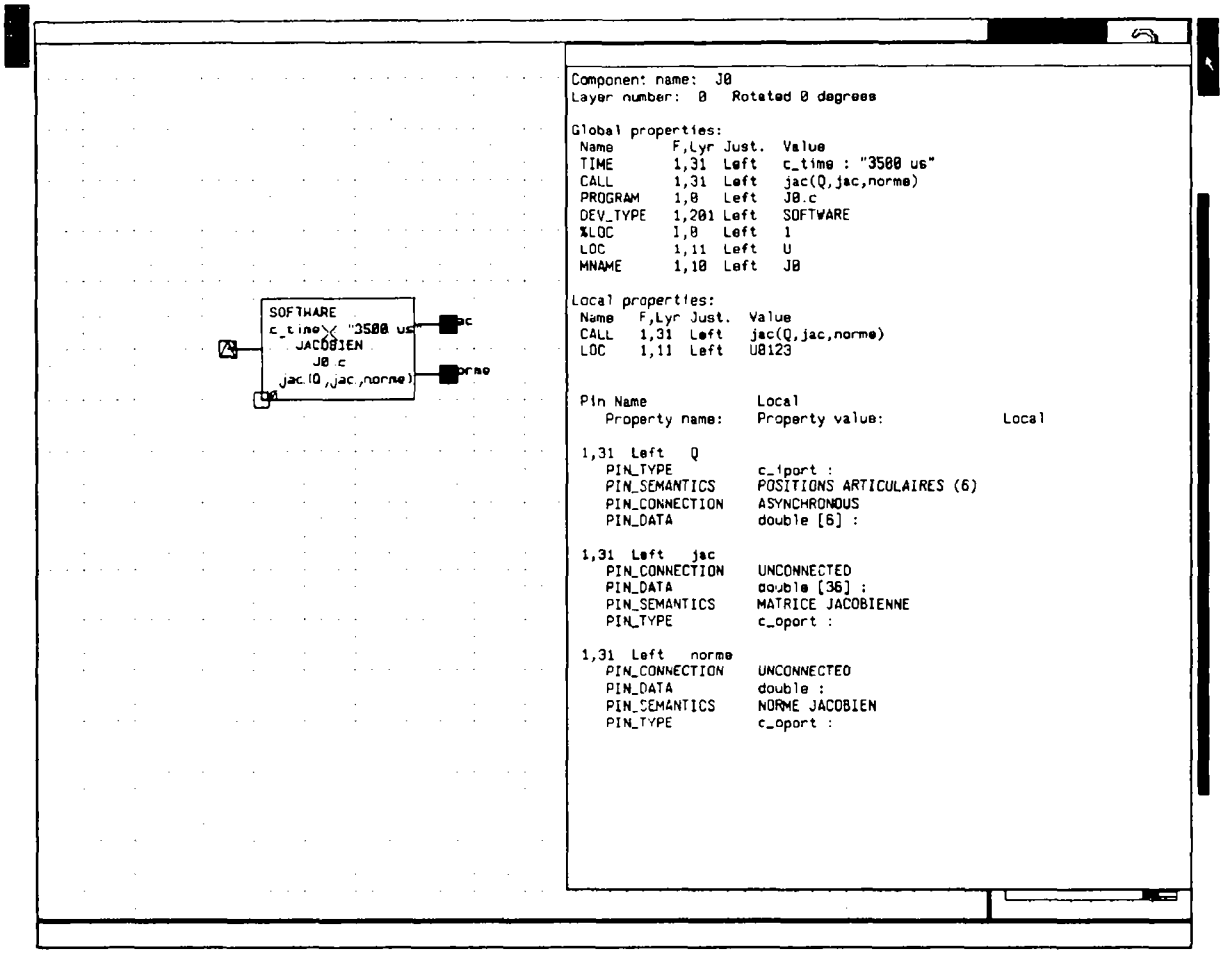

Figure 4: An example of Module-task design

error feedback computation. At the same time, many solutions may be choosen to compute the M-T: for instance, the inertia matrix of the manipulator may sometimes be approximated by a constant matrix : no theorical tools exists now to predict the best solution in each robotic application, and a first simulation pass could give answers to such questions.

This first simulation pass will use a reduced set of functionalities of the SIMPARC simulator [Bor90] (figure 6), i.e. mainly the time management kernel, the objects corresponding to the M-T's, the model of the continuous plant and the continuous systems integration package. A detailed description of the target architecture is not necessary at this stage.

Let us recall that SIMPARC [Bor90] is made of two cooperating parts (figure 5):

- A discrete time event driven simulator, fed with a model of the target (processors, communication links) and with a model of the programs (i.e. the actual code of the M-T's extended with timing information about the bounds of the computation time of blocks of code, including a model of the real time operating system).
- A continuous time integration package providing a simulation of the continuous plant (using models of robots and sensors) driven by the control inputs computed by the discrete time simulator.

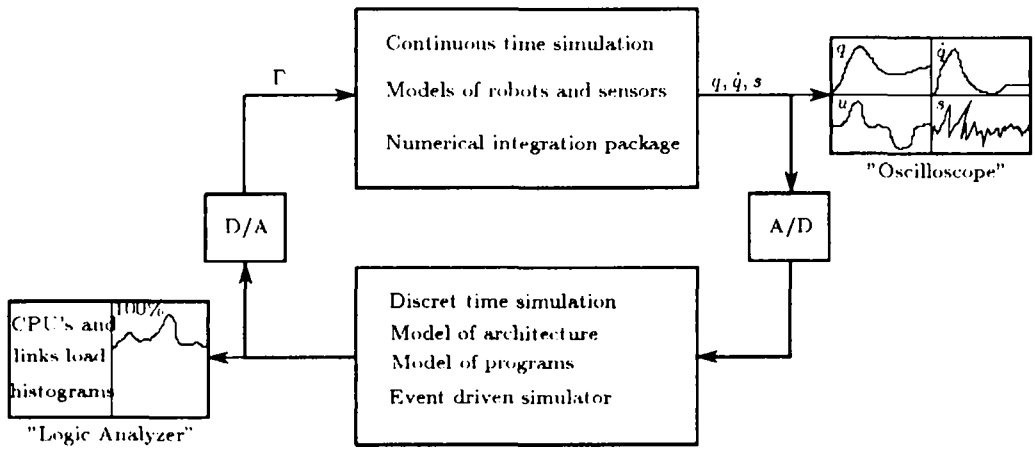

Figure 5: a SIMPARC overview

Once the TCS is validated, the implementation of the Robot-Task on a target architecture can be investigated.

5 Implementation of a robot-task

Starting from the TCS, a Time-constrained implementation description (TCID) of the task is defined. The TCID is made of the TCS in which each M-T is evaluated as a global number of instructions to be achieved and an amount of communication with other M-T's. This valuation should be automatically provided by the simulation phase of the TCS.

A description of the hardware structure of the target machine is also given to ORCCAD using the graphic editor (figure 7): An evaluation of the feasibility of the implementation of the Robot-Task on the target is then done, mainly in terms of computation power and links load. In case of failure, an evaluation of the minimal computational power required to meet the real time constraints of the R-T has to be determined.

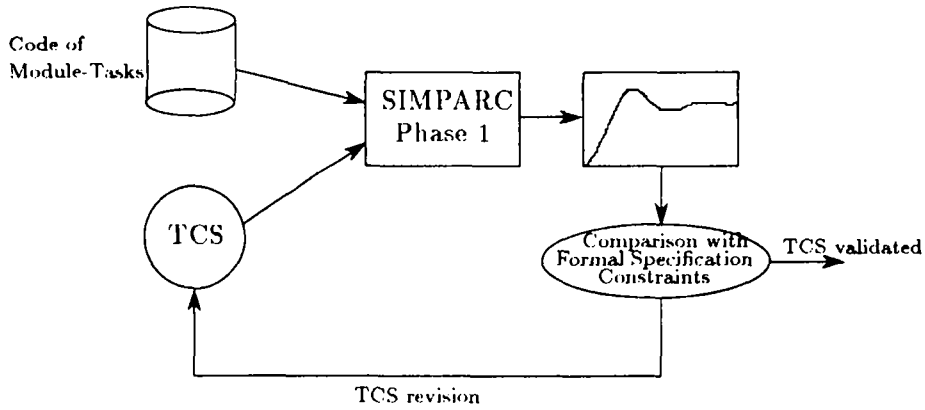

Figure 6: Simulation Phase1

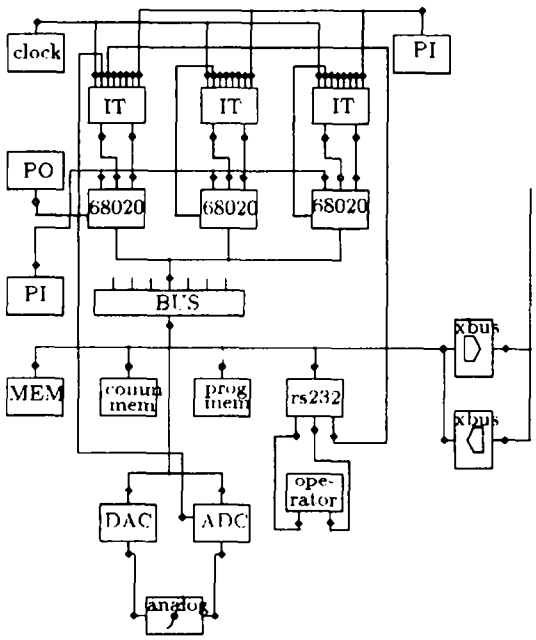

Figure 7: Hardware description

Then ORCCAD will provide tools to help finding a suitable mapping of the set of M-T on the set of available computational resources of the architecture. This tool either can give a proposition of mapping in some cases or an interactive load verification during mapping for more difficult cases. It must take into account constraints like CPU's loads and links loads (figure 8). Some tasks may also be pre-affected to some specific boards (for instance I/O devices or specific computation devices).

Only static mapping of the M-T's on the architecture will be considered for simplicity and compatibility with respect to the high speed real time constraints of the robot controller.

Mapping a set of tasks on a distributed architecture with respect to some optimal criterion is well known to be a N-P complete problem. Many heuristic algorithms have been investigated, and a lot of work have been published within this field [AnP87] [ZRS87]. Software tools exists for dealing with such problems, for example SYNDEX [LaS89] which emphasizes the real time properties of distributed signal processing programs.

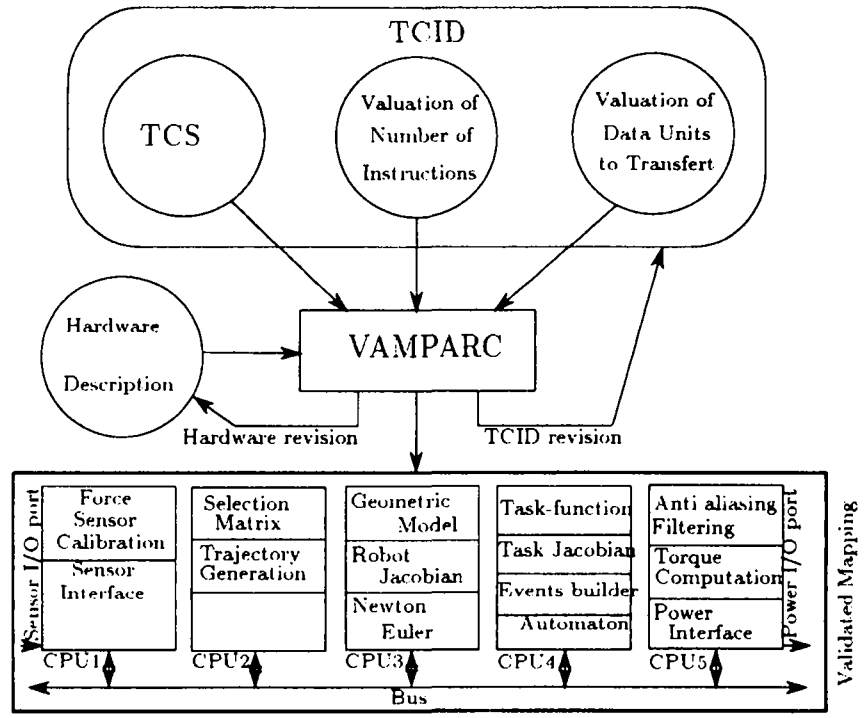

Figure 8: Tasks Allocation

An additional interesting feature would be the ability to concatenate several M-T's into a unique "real-time" task (in the sense of Real Time Operating Systems). Such a feature should improve the run time efficiency by decreasing the context switching and intertask communication overhead, despite a loss of modularity and reusability. An obvious criteria to concatenate two tasks is that they must have the same sampling rate and run in sequence.

The SIMPARC simulator will then simulate the whole execution of the tasks on the target architecture according to all the real time constraints, such as time sharing on the processors, interrupt handling or contention of internal buses...

The code running in the simulator is exactly the same as the code to be loaded on the actual architecture, including the additional system code used allowing the Operating System to manage the real-time tasks. The only difference lies in the need for additional information about

assumed or measured execution time for instruction blocks in the simulated code. Therefore, these tasks do not have to be written twice and the simulator can be used as a debugging tool for the M-T's.

Getting this timing information at the source level of a high level programming language is a difficult problem, due to their strong dependency upon either the target, the compiler or the context [PaS89].

Nevertheless, this information should be available from the cross-compilation tools used to program and debug the Module-Tasks.

This very detailed simulation can be useful for the detection of problems which are highly coupled to hardware constraints such as deadline meeting for periodic tasks or communication links overload. A comparison between the outputs of the continuous plant and the original specifications of the end-user also has to be made.

Once validated by simulation, the code can be downloaded on the target with a good probability of success. If the simulation results are not satisfactory, the sources of failure will be analyzed in order to make changes in the mapping or in the hardware, or even in the TCS if no TCID can be implemented (figure 9).

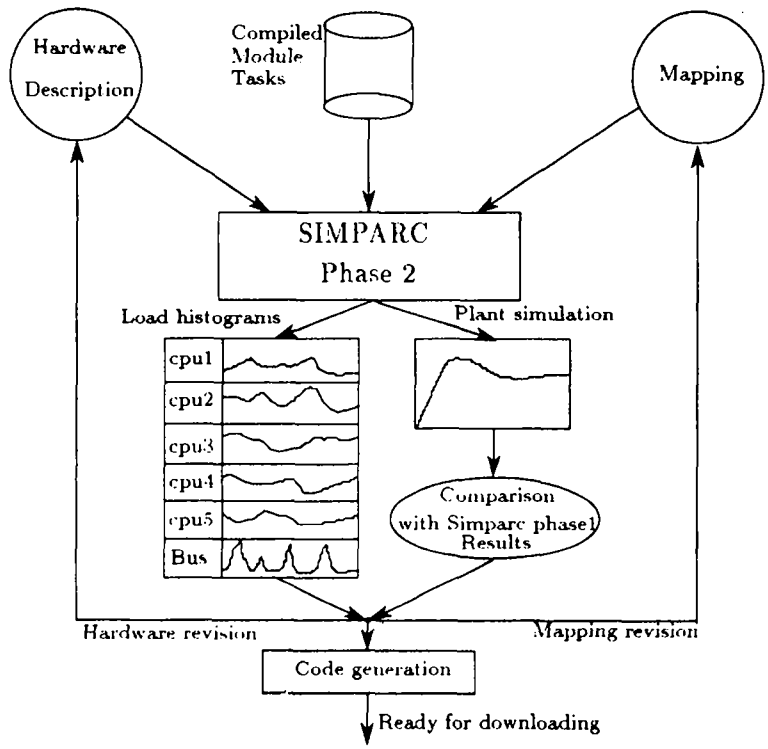

Figure 9: Simulation Phase2

6 Summary

ORCCAD will provide an advanced environment to design and program Robot-Tasks. This design process involves many different techniques and gives a lot of degrees of freedom to successive designers.

The end-user specification is first handled by the Automatic Control staff in order to design a control law. This stage seems to be very difficult to automatize, but should leads to a collection of generic control laws.

A central tool of the system is the hybrid simulator SIMPARC. This simulator handles both a continuous time simulation of the plant and an event driven simulation of the control laws. It might be used at several stages of the design process. In particular, it will be used to achieve a detailed simulation of the real control programs at runtime.

Some work remains to be done in order to check the synchronization scheme of the network of elementary tasks. Software tools will be used in order to help tasks mapping and code generation for different target architectures.

The basic man-machine interface will be an object oriented graphic editor. Building the Robot-Task proceeds by a progressive enrichment of the schemes of the control laws. Using the same graphic editor at each step ensures coherency and propagation of modifications along the design process.

Using such a CAD system can save a lot of time in the development of a robotic process. This avoids an unusual locking up of the hardware and make easier the tuning of control laws. Experiments safety will also be improved by checking for main mistakes during simulation.

Such a development tool thus appears to be necessary in the off-line software environment of a modern Open Robot Controller.

A Design of $CONTOUR(N_p, C_k, f_d, v_d)$

A.1 Notations

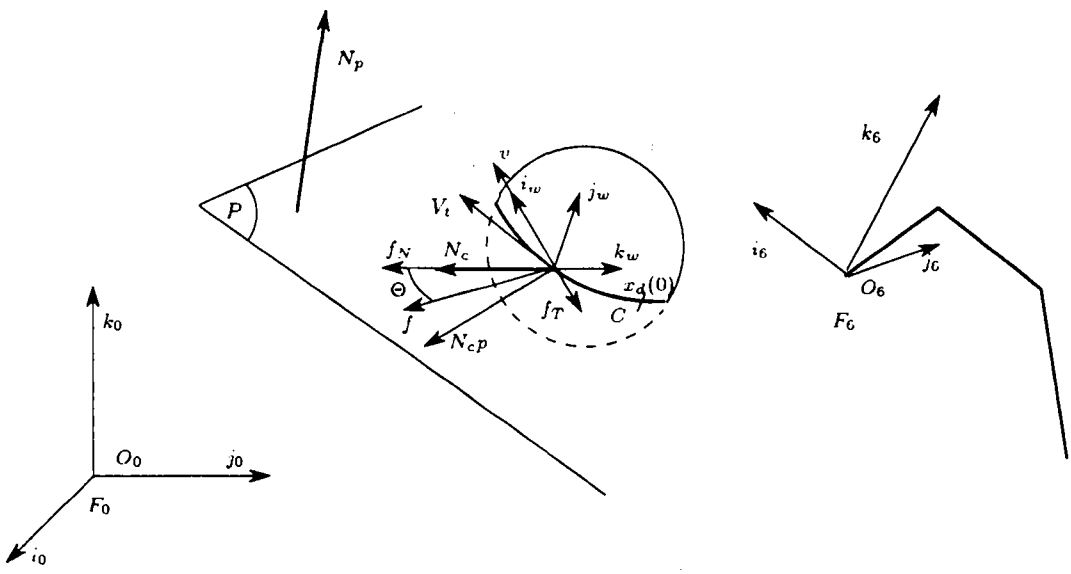

Figure 10: Framework of the contouring task

$F_0 = [O_0, i_0, j_0, k_0]$ fixed frame in the cartesian space
$F_6 = [O_6, i_6, j_6, k_6]$ frame fixed to the end effector, O_6 is located at the end-point of the tool
$F_w = [O_6, i_w, j_w, -N_c]$ task frame, defined with respect to the normal at the contour and the velocity of O_6
$M_0^6 = \begin{pmatrix} R_0^6 & T_0^6 \\ 000 & 1 \end{pmatrix}$ homogeneous transformation matrix from F_0 to F_6
R_6^w rotation matrix from F_6 to F_w
(with the subscript convention $v_{|i} = R_i^j v_{|j}$, $v_{|i}$ being the expression of the vector v in the frame F_i)

N_p	normal to the plane P	3*1
C_k	flag for contouring direction	integer
f_d	desired normal force	scalar
v_d	desired contouring velocity	scalar
ϵ	error vector	6*1
$F = [f, \tau]$	contact screw	6*1
f	measured contact force	3*1
W	partition matrix	6*3
N_c	normal to the contour	3*1
α	secondary goal gain	scalar
$x_d(0)$	initial contact point	3*1
$x(t)$	position of O_6	3*1
$q(t)$	joint coordinates	6*1
d	distance between O_0 and the plane P	scalar
f_N	normal contact force	3*1
f_T	tangential contact force	3*1
$v_{\|0}$	velocity of O_6	3*1
V_t	desired velocity of O_6	3*1
N_{cp}	normalized projection of N_c on P	3*1
c_i	constraints defining the secondary goal	scalars
Γ	control torque	6*1
$\hat{M}$	estimated inertia matrix of the robot	6*6
J_q	estimated task Jacobian	6*6
j_t	estimation of $\frac{\partial e}{\partial t}$	6*1
$J_{L_{\|0}}$	translational Jacobian	3*6
$\hat{l}$	centrifugal and Coriolis forces	6*1
$\mu, k,$	scalar gains	
G, D	matrices of gains	6*6

A.2 From specification to control law

- 3 dimensioned error function to be regulated (error vector of the primary goal):

$$\epsilon_1 = \begin{bmatrix} < i_w, O_d O_6 > \\ < j_w, O_d O_6 > \\ < N_c, f > - f_d \end{bmatrix}$$

$< .,. >$ denoting the scalar product of vectors. O_d is the on-line computed point of the contour to be tracked with the velocity v_d
This formulation states that we wish to track the projection of O_d in the plane $[i_w, j_w]$ while regulating the measured force around f_d in the orthogonal direction given by N_c.
The sensed object is motionless with respect to F_0, so the derivative of the primary task-function with respect to time may be written as:

$$\dot{\epsilon}_1 = \frac{\partial \epsilon_1}{\partial \bar{r}} \frac{d\bar{r}}{dt} = L^T V_{6/0}$$

where $V_{6/0}$ is the (6×1) velocity screw of F_6 with respect to F_0. L is the (6×3) interaction matrix which states the variation of the ϵ_1 with respect to the relative motion between the gripper and the target. $V6/0$ and L are computed at the contact point O_6.
Assuming that the contour to follow is smooth enough to consider that the task frame is

slowly varying, the interaction matrix associated with e_1 may be estimated by :

$$\hat{L} = \begin{bmatrix} i_w & j_w & -\lambda \hat{N}_c \\ 0 & 0 & 0 \end{bmatrix}$$

where λ is the unknown stiffness along the direction of N_c. A possible choice for a matrix function W such as $Range(W^T) = Range(L)$ and $L^T W^T > 0$ is therefore:

$$W = \begin{bmatrix} i_w^T & 0 \\ j_w^T & 0 \\ -\hat{N}_c^T & 0 \end{bmatrix}$$

and the partition matrices between the primary task-function and the secondary goal can be computed as (W^+ denotes the pseudo inverse of W):

$$W_1 = W^+ = \begin{bmatrix} R_0^w \\ 0 \end{bmatrix} \qquad W_2 = I_6 - W^+ W = \begin{bmatrix} 0 & 0 \\ 0 & I_3 \end{bmatrix}$$

- Computation of the global task function (expressed in F_0):

$$e = \begin{bmatrix} R_0^w \\ 0 \end{bmatrix} \begin{bmatrix} < i_{w|0}, T_0^6 - x_d > \\ < j_{w|0}, T_0^6 - x_d > \\ < k_{w|0}, R_0^6 f_{|6} > - f_d \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & I_3 \end{bmatrix} g_s^{\bar{r}}$$

where $g_s^{\bar{r}} = \frac{\partial h}{\partial \bar{r}}$ is the gradient of a cost function $h_s(\bar{r}, t)$ to be minimized and $\bar{r}$ is the location of the gripper in the cartesian space.

- Preliminary calculations
storage of the initial contact position $[O_0 O_d]_{|0}(0) = x_d(0) = G(q(0))$, G is the geometric model of the manipulateur
computation of the distance $d = (O_0, P) =< N_{p|0}, x_d(0) >$

- Normal vector estimation:

$$\hat{N}_{c|0} = \frac{(R_0^6 f_{|6} - f_{T|0})}{||R_0^6 f_{|6} - f_{T|0}||}$$

The tangential force f_T can be computed by a "normal expert" assuming that a realistic model of friction is available[Mer87]; an other way to compute f_T might be:

$$f_{T|0} =< f_{|0}, v_{|0} > \frac{v_{|0}}{||v_{|0}||}$$

$v_{|0}$ being the measure of the velocity of the end point of the robot computed in F_0 given by $v_{|0} = J_{L|0}(q)\dot{q}$, $J_{L|0}(q)$ being the translational Jacobian of the manipulator, expressed in the frame F_0.

- Computation of the task frame:

$$i_w = \frac{v}{||v||} \qquad j_w = -\hat{N}_c \times i_w$$

where $\times$ denotes the cross product of vectors

$$R_0^w = \begin{bmatrix} i_{w|0} & j_{w|0} & -\hat{N}_{c|0} \end{bmatrix}$$

- the tool follows the contour at a velocity v_d:
  projection of the normal N_c on the plane P (normalized):

$$\hat{N}_{cp|0} = \frac{\hat{N}_{c|0} - < N_{p|0}, \hat{N}_{c|0} > N_{p|0}}{||\hat{N}_{c|0} - < N_{p|0}, \hat{N}_{c|0} > N_{p|0}||}$$

  desired velocity of displacement:

$$V_{t|0} = (\hat{N}_{cp|0} \times N_{p|0}) v_d C_k$$

  with $C_k = 1$ for clockwise contouring and -1 otherwise

- Position to track, in continuous time:

$$x_d(t) = x_d(0) + \int_0^t V_t(s) ds$$

  V_t is computed in F_0. A stable implementation in discrete time is given by, $(.)^k = (.)(k\Delta)$, Δ being the sampling period for the desired position computation loop:

$$x_d^{k+1} = (1 - \beta) x_d^k + \beta x^k + \Delta V_t^k; 0 < \beta < 1$$

- Cost function to minimize:

$$h_s = \frac{1}{2} \sum_i \alpha_i c_i^2, \alpha_i > 0$$

$$g_s^{\bar{r}} = \sum_i \alpha_i c_i \frac{\partial c_i}{\partial \bar{r}}$$

  These functions are defined by the constraints c_i weighted by the scalars α_i.

- Constraints defining the secondary goal:
  the i_6 axis of the tool remains parallel to the normal at the contact point:

$$c_1 =< i_6, \hat{N}_{c|0} >= 0 \qquad \frac{\partial c_1}{\partial \bar{r}} = \begin{bmatrix} 0 \\ i_6 \times \hat{N}_{c|0} \end{bmatrix}$$

$$c_2 =< j_6, \hat{N}_{c|0} >= 0 \qquad \frac{\partial c_2}{\partial \bar{r}} = \begin{bmatrix} 0 \\ j_6 \times \hat{N}_{c|0} \end{bmatrix}$$

  the i_6 axis of the wrist keeps parallel to the plane P:

$$c_3 =< i_6, N_{p|0} >= 0 \qquad \frac{\partial c_3}{\partial \bar{r}} = \begin{bmatrix} 0 \\ i_6 \times N_{p|0} \end{bmatrix}$$

- A general form of the control torque is:

$$\Gamma = -k\hat{M} J_q^{-1} G(\mu D \hat{\epsilon} + J_q \dot{\hat{q}} + j_t) + \hat{l}$$

  with $\hat{M}$ being an estimation of the inertia matrix of the manipulator, μ,G, k and D being gains, J_q an estimation of the Jacobian matrix $\frac{\partial \epsilon}{\partial q}$ of the task, j_t being an approximation of the derivative of the task-function with respect to time $\frac{\partial \epsilon}{\partial t}$ and $\hat{l}$ includes an estimation of the Coriolis and centrifugal forces.
  Hybrid position/force control tasks are generally performed at a rather low speed, with

few dynamical effects, while force control requires a high sampling rate. Moreover, we assume that the estimation of the task frame is accurate enough to insure that $\frac{\partial e}{\partial q} J_0^{-1} > 0$.
The following choices are thus made:
$\hat{M} = diag(M)$ with a slow sampling rate
$\hat{l} = 0$
$j_t = \frac{\partial e_1}{\partial t} = R_0^{w\,T} J_{L_{|0}} \dot{q}$ (giving feedforward only for the primary goal)
$J_q = J_{|0}$ (the basic Jacobian of the manipulator)
$J_q^{-1} = J_{|0}^{-1}$
These choices and the value of the gains will have to be verified by simulation.

- Events to be generated:
  BACK-TO-INITIAL when a desired position is reached, SINGULARITY when the norm of the Task jacobian reaches a given threshold, JOINT-LIMIT when a specified joint limit is reached.

A.3 Module-Tasks to be used

Task name	Function	Input ports	Output ports
J0	basic Jacobian	q_i^k , $i = 1,6$	$J_{\|0}$ (6*6), $\|\|J_{\|0}\|\|$
Jinv0	basic inverse Jacobian	q_i^k , $i = 1,6$	$J_{\|0}^{-1}$ (6*6)
GM	Direct Geometric Model	q_i^k , $i = 1,6$	M_0^6 (4*4), $\bar{r}$ (6*1)
FN	Normal Expert	$f_{\|6}, v_{\|0}, R_0^6$	$f_{N_{\|0}}$ (3*1) ,R_0^w (3*3),$\hat{N}_{c_{\|0}}$ (3*1)
Calib	Calibration matrix of the force sensor	s_i , $i = 1,6$	$F_{\|6}$ (6*1)
Control	Control Torque vector	$\epsilon, \dot{q}, diag(M), J_{\|0}, J_{\|0}^{-1}$	Γ (6*1)
DiagM	diagonal elements of M	q_i^k , $i = 1,6$	$M_{ii}, i = 1,6$
TrackXd	Trajectory generator	$\hat{N}_{c_{\|0}}$	x_d^{k+1}
TF	task function	$M_0^6, R_0^w, x_d, f_{\|6}$	ϵ (6*1), $\|\|\epsilon\|\|$
AAF	Anti-aliasing filter	$q_i, \dot{q}_i$, $i = 1,6$	$q_i^k, \dot{q}_i^k$, $i = 1,6$
OBS	observer	$x, \|\|J_0\|\|, \|\|\epsilon\|\|$	SING., JOINT., BACK.

References

[AnP87] Andre F., Pazat J.L.: *Le placement de taches sur des architectures paralleles*
INRIA Research Report n^o 614, February 1987.

[BLL88] Benveniste A., Le Goff B., Le Guernic P.: *Hybrid Dynamical Systems Theory and the language SIGNAL*
INRIA research Report n^o 838, April 1988.

[BeC85] Berry G., Cosserat L.: *The Synchronous Programming Language ESTEREL and its Mathematical Semantics*
Proc. Seminar on Concurrency, Springer-Verlag LNCS 197, 1985

[BSS90] Borrelly J.J., Samson C.and Simon D.: *Specification of an Open Robot Controller*
Esprit II ARMS project. Periodic Progress Report n^o2, June 1990.

[Bor90] Borrelly J.J.: *SIMPARC : SImulator for Multi-Processor Advanced Robot Controller*
ARMS project internal working document, April 1990.

[BoS90] Borrelly J.J., Simon D.: *Propositions d'Architecture d'un Contrôleur Ouvert pour la Robotique*
INRIA Research Report n^o 1304, October 1990

[PaS89] ChangYun Park, Alan C. Shaw: *A Source-Level Tool for Predicting Deterministic Execution Times for Programs*
Technical Report n^o 89-09-12, Department of Computer Science and Engineering, FR-35, University of Washington, Seattle.

[Cha90] Chaumette F.: *La commande referencee vision: une approche des problemes d'asservissements visuels en robotique*
These de l'Universite de Rennes I, 19 Juillet 1990

[ECM90] Espiau B., Coste-Maniere E.: *A synchronous approach for control sequencing in robotics applications*
IEEE Workshop on Intelligent Motion Control, Istambul, August 1990.

[EMS90] Espiau B., Merlet J.P., Samson C.: *Force feedback control and non-contact sensing: a unified approach*
Romansy, Cracovie, July 1990

[Gas87] Gaspart P.: *Langages de programmation de la Robotique*
Hermes, Paris, 1987

[HCRP89] Halbwachs N., Caspi P., Raymond P., Pilaud D.: *Programmation et Verification des Systemes Reactifs a l'aide du langage flot de donnees synchrone LUSTRE*
IMAG/LGI Research Report, January 1990.

[Jou88] Joubert A.: *Description et simulation d'Algorithmes et d'Architectures distribuées*
Thèse de Doctorat de l'Université de Rennes I, Octobre 1988.

[LaS89] Lavarenne C., Sorel Y.: *SYNDEX: un environnement de programmation pour multiprocesseur de traitement du signal*
INRIA Technical Report n^o 113, November 1989.

[LGB86] Le Guernic P., Benveniste A.: *Real-Time, Synchronous, Data-Flow Programming: the language* SIGNAL *and its Mathematical Semantics*
INRIA Research Report n^o 533, June 1986

[MSBB89] Mejia M.,Simon D.,Belmans P.,Borrelly J.J.: *Mecanismes de synchronisation dans un systeme robotique reparti*
Seminaire Franco-Bresilien sur les systemes informatiques repartis, Florianopolis, Bresil. Sept 89

[Mer87] Merlet J.P.: *C-surface theory applied to the design of an hybrid force-position controler*
IEEE International Conference on Robotics and Automation, Raleigh, 1987

[Sam88] Samson C.: *Commande de robots manipulateurs rigides.Synthèse et analyse.*
Techniques de la robotique Tome 1:Architectures et Commandes,Hermès,Paris,19 88.

[SEL91] Samson C., Espiau B., Le Borgne M.: *Robot Control: the Task-function approach*
Oxford University Press, 1991

[ZRS87] Zhao W., Ramamritham K., Stankovic J.A.: *Preemptive Scheduling Under Time and Resource Constraints*
IEEE Transactions on Computers, vol C-36, n^o 8, August 87

Imprimé en France
par
. l'Institut National de Recherche en Informatique et en Automatique

ISSN 0249-6399