Planning Incorporating Advanced Robot Control: A Parallel Supporting Architecture

Marcelo Martín, Ricardo Carelli
Universidad Nacional de San Juan
Av. San Martín 1109 (0)
5400 San Juan
ARGENTINA

Manuel M. Barata
Universidade Nova de Lisboa - UNINOVA
Quinta da Torre -2825 Monte de Caparica
PORTUGAL

Abstract: This paper presents a contribution to the integration of Advanced Robot Control and task planning areas of research. Advanced Robot Control Algorithms are discussed and a case study for a two- degree- of -freedom manipulator is presented. An approach to integrating advanced controllers and task planning for planning and control parameter evaluation as well as approaches to the real-time execution of the controller, are made. The application of parallel architectures for implementation of real-time Advanced Controllers is explored and discussed. The application of transputer based architectures is presented and a methodology for exploring the parallelism of the control algorithm is explained.

Keywords: Robot Control, Parallel Processing, Task Planning

1.0 Introduction

Task planning activities mostly consider robots as machines equipped with conventional controllers. However, for tasks with high demands on speed, load and interaction forces. Advanced Robot Control will be required. These types of controllers take into account the dynamic model of the robot, and have, in some cases, adaptive and learning capabilities. There are several solutions to this problem that can be found in the related bibliography [1-7]. For example, some of these controllers are classified in: Inverse Dynamics, Impedance Control, Hybrid Control, Adaptive Control, Robust Control.

The implementation of this kind of a Robot Controller is highly demanding on computing power, thus affecting both the process of selection of the controller by simulation and the real time execution of the controller. A parallel architecture suitable to executing the above Control Algorithms is proposed. This architecture is used at two levels:

i) Planning and control parameter evaluation, and
ii) Real time execution of the controller.

The results of applying transputers for supporting the base processing hardware platform are also presented in this paper.

2.0 Advanced Robot Control Algorithms

Advanced dynamics robot control requires complex non-linear algorithms to be solved for planning or real time execution. The objective of our work is to consider that during planning, simulation has to be done to select the controller algorithm and related controller parameters as well as during on-line execution, the sampling periods are about 1 ms. This work intends to exploit the potential parallelism of the control algorithm and to implement it by using transputer based systems.

2.1 Robot Dynamics Model

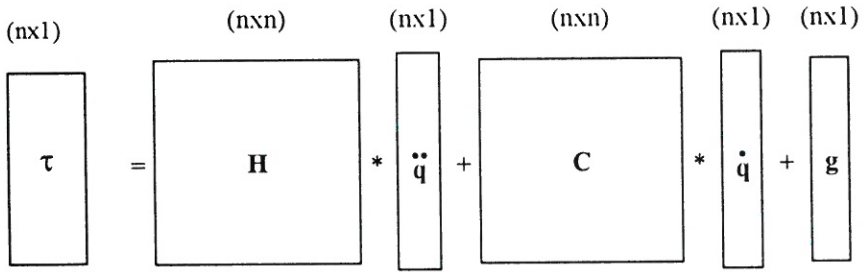
The Euler-Lagrange non-linear multivariable coupled dynamic equation for an n-link rigid robot [1] is ,

$$\tau = \mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) \qquad (1)$$

where:

$\mathbf{H(q)}$	: Inertia Matrix
$\mathbf{C(q,\dot{q})}$	: Centripetal - Coriolis Torques matrix
$\mathbf{g(q)}$	: Gravitational torques vector
$\mathbf{q}$	: Joint position vector
τ	: Joint torques vector

The matrix structure of equation (1) is:



Matrix $\mathbf{H(q)}$ is positive definite and symmetric. The dynamic model of equation (1) can be expressed linearly in terms of a suitable selected set of robot and load parameters [2]:

$$\tau = \Phi(q,\dot{q},\ddot{q})\theta$$

where θ is the (m x 1) parameter vector and Φ is a (n x m) signal matrix. This property is used in the design of adaptive and robust controllers.

When the robot interacts with the environment, the interaction force $\mathbf{F_e}$ at the end-effector is added to the dynamic equation as $\mathbf{J^T(q)F_e}$ to represent the interaction in terms of the equivalent joint torques,

$$\tau = \mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \mathbf{J^T(q)F_e}$$

where $\mathbf{J(q)}$ is the Jacobean matrix.

2.2 Motion Control

The motion control problem can be stated as that of designing a controller law to compute the joint applied torques τ so that the following control objective should be verified:

$$\tilde{\mathbf{q}}(t) = \mathbf{q}(t) - \mathbf{q_d}(t) \rightarrow 0 \qquad \text{as } t \rightarrow \infty \qquad (2)$$

where $q_d(t)$ is the desired trajectory in joint coordinates. That is, the motion error tends asymptotically to zero.

Inverse dynamics control [1] compensates for the non-linearity of the dynamics model. Let us consider the following controller structure,

$$\tau = H(q)a + C(q,\dot{q})\,\dot{q} + g(q) \tag{3}$$

where **a** is a signal to be defined. Closed loop equation is obtained by equating (1) and (3),

$$H(q)\,\ddot{q} + C(q,\dot{q})\,\dot{q} + g(q) = H(q)\,a + C(q,\dot{q})\,\dot{q} + g(q) \tag{4}$$

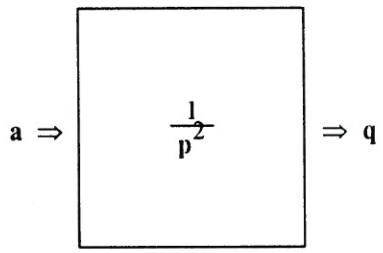
$$H(q)\,\ddot{q} = H(q)\,a$$

As $H(q)$ is invertible (it is definite positive),

$$\ddot{q} = H(q)^{-1}\,H(q)\,a \tag{5}$$

$$\ddot{q} = a$$

which represents a linear uncoupled system,



Now signal **a** can be defined as a linear **PD** controller,

$$a = \ddot{q}_d - K_V(\dot{q} - \dot{q}_d) - K_P(q - q_d) \tag{6}$$

where K_P and K_V are positive definite gain matrices.

Equating (5) and (6),

$$(\ddot{q} - \ddot{q}_d) + K_V(\dot{q} - \dot{q}_d) + K_P(q - q_d) = 0 \tag{7}$$

For K_V, K_P definite positive, equation (7) implies,

$$q(t) \to q_d(t),\ \dot{q}(t) \to \dot{q}_d(t) \quad with\ t \to \infty$$

that is, the control objective is verified.

Equations (3) and (6) represent the controller algorithm to be implemented. Equation (3) implies the on-line solution of the robot dynamics , which is numerically complex. We propose the Control System of Figure 1. It uses the T800/T805 processors for building the parallel processing system of the controller.

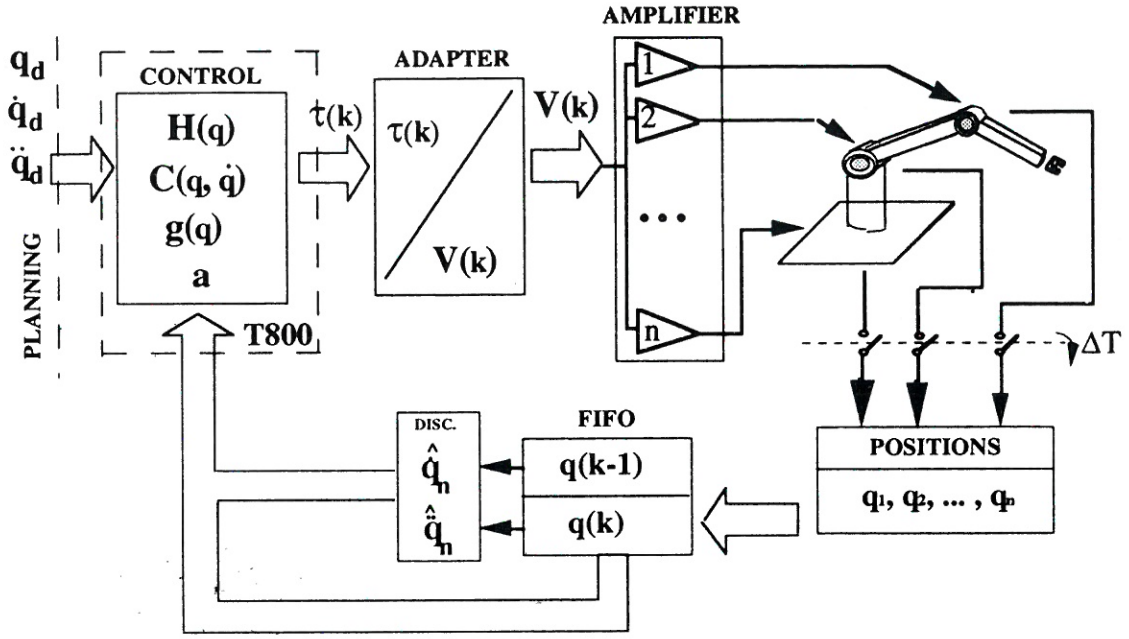


Figure 1
Block Diagram of the Control System

2.3 A General Controller Structure

The controller presented in section 2.2 is based on the exact knowledge of the robot inverse dynamics. More sophisticated algorithms which are designed to cope with uncertainties in robot dynamic parameters and unmodelled dynamics have been developed as adaptive controllers [2,3,4] and robust controllers [5,6,7]. Also in constrained motion control, the interaction forces have to be accommodated or regulated, and $\mathbf{F_e}$ as well as the desired force $\mathbf{F_d}$ have to be included in the controller algorithm.

In general, the controller structure can be expressed as follows:

$$\tau = \mathbf{H}(q)v_1 + \mathbf{C}(q,\dot{q})v_2 + g(q) + Kv_3 + J^T(q)F_e \qquad (8)$$

where $\mathbf{K}$ is a positive definite matrix and v_1, v_2, v_3 are functions of q, $\dot{q}$, q_d, $\dot{q}_d$, $\ddot{q}_d$, F_e and F_d.

2.4 Example: Two -Degree- of- Freedom Manipulator

The following equations represent the components of the controller equation (3) for a two-degree- of- freedom manipulator (DOF),

$$\mathbf{H}(q) = \begin{bmatrix} h_{11}(q_1,q_2) & h_{12}(q_1,q_2) \\ h_{21}(q_1,q_2) & h_{22}(q_1,q_2) \end{bmatrix}$$

$$\mathbf{C}(q,\dot{q}) = \begin{bmatrix} c_{11}(q_1,q_2,\dot{q}_1,\dot{q}_2) & c_{12}(q_1,q_2,\dot{q}_1,\dot{q}_2) \\ c_{21}(q_1,q_2,\dot{q}_1,\dot{q}_2) & c_{22}(q_1,q_2,\dot{q}_1,\dot{q}_2) \end{bmatrix}$$

$$g(q) = \begin{bmatrix} g_1(q_1,q_2) \\ \\ g_2(q_1,q_2) \end{bmatrix}$$

$$a = \begin{bmatrix} \ddot{q}_{1d} - k_{v1}(\dot{q}_1 - \dot{q}_{1d}) - k_{p2}(q_1 - q_{1d}) \\ \\ \ddot{q}_{2d} - k_{v2}(\dot{q}_2 - \dot{q}_{2d}) - k_{p2}(q_2 - q_{2d}) \end{bmatrix}$$

Velocities $\dot{q}_1$, $\dot{q}_2$ in the discrete domain can simply be obtained by,

$$\dot{q}(k) = \frac{q(k) - q(k-1)}{\Delta T}$$

3.0 Selection of Robot Controllers at Task Planning Level

In general terms, planning/programming is a set of activities that realizes a plan transformation from abstract levels into successive more detailed levels, ending by a list of executable task specifications.

The highest abstract specification can start with the product model where at least one assembly/operation step can naturally be associated with each component part. A more detailed description is obtained by assigning a precedence graph and a category to each operation step. Additional technological information, e.g. approaching orientation, movement speed, etc. can be added resulting in a more detailed process plan.

After having selected a specific cell (set of intervening agents), the task specification can be further refined until a description understandable by the cell controller is reached. All these steps require more or less well defined activity areas, e.g design, process planning, execution planning, in which several subsystems ("specialized planners" or "computer-aided" functions) have been and are being developed.

According to [8,9] Planning can be understood as an hierarchy of four levels:

1) Assembly /Graph Process Plan generation

2) Intermediate Process Plan

3) Cell-based Plan and

4) Component Level Program

The work described in this paper takes place at level four. The Component Level Program corresponds to an executable plan/program generated for each component. Additionally, the movements trajectory is to be decided by one of two possible ways: Automatic or Interactive. For those situations where the plan operation steps cannot be generated automatically, the Interactive method is used. Simulation at this level is "realistic", since it corresponds to an **emulation** of the physical cell. In the context of this paper we concentrate our work on the simulation of the robot controllers used by a specific cell.

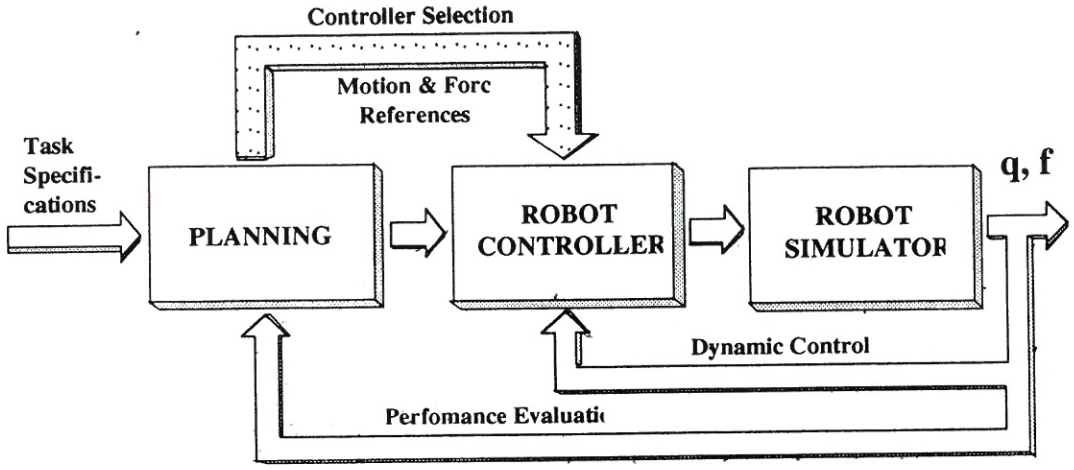


Figure 2
Selection of the Controller Structure and Controller Parameters During Planning.

In most of the robot based assembly tasks, the robot control parameters are assumed fixed and the task's description is a sequence of movement commands. For the operations of which demands on speed, load and interaction forces are high, the Advanced Control is necessary. Beside the trajectory, speed and load determination, it is also necessary to identify the optimized controller structure and controller parameters for the robot. According to our approach, for some of the robot's operations at the Component Level Program, the corresponding control parameters (the controller) are also specified.

The required controller is selected by simulation, as shown by the architecture in Figure 2. The controller under selection for the required force references, is evaluated by applying it to the robot simulator. This procedure enables an off-line selection of that controller which performs better and records it for real-time use.

4.0 Parallel Architecture

As a consequence of robot control algorithms become more complex and sample times become shorter, the computational requirements of real-time controllers are overcoming the microprocessors performance. Considering these aspects, engineers have sought alternative hardware solutions such as digital signal processors, specific microprocessors and transputer networks. In this paragraph we present the computation structure and the transputer-based implementation [10] of the inverse dynamics control of a two-degree-of-freedom robot. This is a simple example useful to present the implementation process. In practice, much more complex algorithms are to be considered, normally applied to robots of six degrees of freedom. Nevertheless, this simple controller has the same structure as the general controller of Eq.(8), so,the implementation presented here is an emblematic one.

A parallel implementation of controller structure (8) is considered and evaluated for motion inverse dynamics control, $v_1 = a$, $v_2 = \dot{q}$, $v_3 = 0$, $F_e = 0$, and the 2 DOF robot manipulator presented in section 2.4.

4.1 Detailed Controller Computations

The sequence of calculations towards obtaining the inverse dynamics control of equation (3) for the 2 DOF robot, and for sampling instant **k**, is given in the following set of expressions:

$$\dot{q}_1 = \frac{q_1(k) - q_1(k-1)}{\Delta T} \qquad \text{(e1)}$$

$$\dot{q}_2 = \frac{q_2(k) - q_2(k-1)}{\Delta T} \qquad \text{(e2)}$$

$$a_1 = \ddot{q}_{1d} - k_{v1}(\dot{q}_1 - \dot{q}_{1d}) - k_{p1}(q_1 - q_{1d}) \quad (e3)$$

$$a_2 = \ddot{q}_{2d} - k_{v2}(\dot{q}_2 - \dot{q}_{2d}) - k_{p2}(q_2 - q_{2d}) \quad (e4)$$

$$h_{11} = m_1 d_1^2 + m_2 (l_1^2 + d_2^2 + 2 l_1 d_2 \cos q_2) + j_1 + j_2 \quad (e5)$$

$$h_{12} = h_{21} = m_2 (d_2^2 + l_1 d_2 \cos q_2) + j_2 \quad (e6)$$

$$h_{22} = m_2 d_2^2 + j_2 \quad (e7)$$

$$c_{11} = - m_2 l_1 d_2 \dot{q}_2 \sin q_2 \;\; ; \;\; c_{22} = 0 \quad (e8)$$

$$c_{12} = - m_2 l_1 d_2 \sin q_2 (\dot{q}_1 + \dot{q}_2) \quad (e9)$$

$$c_{21} = m_2 l_1 d_2 \dot{q}_1 \sin q_2 \quad (e10)$$

$$g_1 = g_2 + (m_1 d_1 + m_2 l_1) g \cos q_1 \quad (e11)$$

$$g_2 = g\, m_2\, d_2 \cos (q_1 + q_2) \quad (e12)$$

$$m_2 = M + m\, L_2 \quad (e13)$$

$$d_2 = \frac{(m L_2 \, dL_2 + M l_2)}{m_2} \quad (e14)$$

$$j_2 = jL_2 + I + M(l_2 - d_2)^2 + mL_2 (d_2 - dL_2)^2 \quad (e15)$$

$$A_1 = h_{11}\, a_1 + h_{12}\, a_2 \quad (e16)$$

$$A_2 = h_{21}\, a_1 + h_{22}\, a_2 \quad (e17)$$

$$B_1 = c_{11}\, \dot{q}_1 + c_{12}\, \dot{q}_2 \quad (e18)$$

$$B_2 = c_{21}\, \dot{q}_1 \quad (e19)$$

$$\tau_1 = A_1 + B_1 + g_1 \quad (e20)$$

$$\tau_2 = A_2 + B_2 + g_2 \quad (e21)$$

where: j_1, j_2, jL_2, m_1, m_2, mL_2, l_1, l_2, d_1, d_2, dL_2 are robot parameters, **I**, **M** are load parameters and **g** is the gravity acceleration.

Some computations as those given by expressions (e7), (e13), (e14), and (e15), are to be done off-line and only once for a given robot and load condition. On-line computations, on the other hand, are resumed each sampling period.

Data for computations come from:

- Robot sensors.
- External sensing (e.g. force sensors).
- Off-line calculation.
- Intermediate calculation. These data condition the algorithm parallelism.

4.2 Parallelism of the Computations

According to the given constraints, the set of expressions (e1) to (e21) presented in the previous section, must have an evaluation time less than the sampling period ΔT = 1ms. Before determining whether one processor is enough or not, let us explore the parallelism that is implicit in this set of expressions.

On constructing an evaluation precedence graph, we identify four levels, or stages of expression evaluation. The expressions of each stage are independent of the relative order of the evaluation:

Level 1: (e1), (e2), e(5), e(6), e(12)

Level 2: (e3), (e4), (e8), (e9), (e10), (e11)

Level 3: (e16), (e17), (e18), (e19)

Level 4: (e20), (e21)

The total evaluation time is the sum of the times needed for each level's evaluation. As the expressions at each level are evaluation free, the maximum parallelism grain corresponds to the assigning of one processor to the evaluation of each expression. We can estimate the computation time required by each level as well as the total computation time by assigning one processor to the evaluation of each expression.

In practice the maximum parallelism grain is not used because of the communications' cost between the processors. It is necessary to establish a threshold where the communications' work is much lower than the processors' work.

Figure 3 presents the flow chart for a 2 DOF robot controller calculus, where the previous levels and corresponding expressions are shown.

4.3 Results Obtained

Considering the four sets of expressions presented in the previous section, each level computation time was determined, as the total computation time was, using only one transputer. The determination of computation times was possible by an OCCAM program containing all the expressions. They were programmed in sequence and after making sure that no other task was running during their evaluation. For each set of expressions of levels 1 to 4 the computation time was measured using the transputer timer 0. Using one T800 20Mhz and only its internal memory, the results obtained are the following:

Level 1: 98 μs

Level 2: 20 μs

Level 3: 10 μs

Level 4: 4 μs

Total: 132 μs

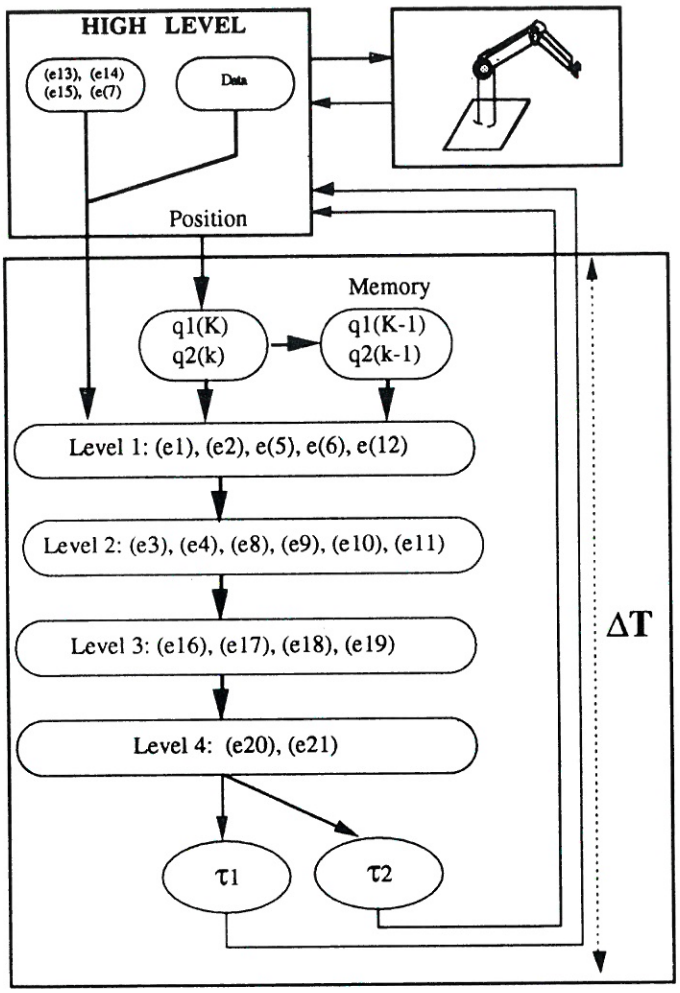


Figure 3
Flow Chart for 2 DOF Robot Controller Calculus.

The total time resulted to be 132 μs, much lower than the 1 ms total time, that means for this specific case only one transputer of this type is enough!

4.4 Some Considerations about Parallel Architecture

The previous sections analysis did not take into consideration the problem of tasks' distribution and of the required communications' times . In fact this problem turns up only when distributing the computations by a set of processors. The given example has brought about the conclusion that only one processor is enough to do the computations. The time required for data communication between several levels is thus saved, because data are stored in global variables allocated to the processor's memory. Similarly, the time necessary for tasks' management is also saved; the processor evaluates the expressions in sequence.

An immediate way of distributing the computations illustrated in Figure 3 by several processors is, for example, assigning one processor to the evaluation of each level. However this distribution makes no point since only one processor will be active during the evaluation of each level. This is because of each level processor starting work on the moment of the previous one finishing it. Similarly the next processors have to wait until the current one is terminated.

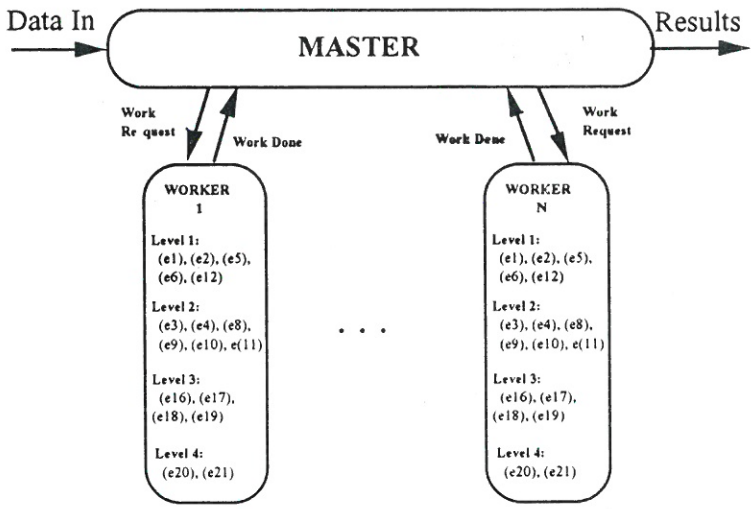


Figure 4
Computer Farm Model

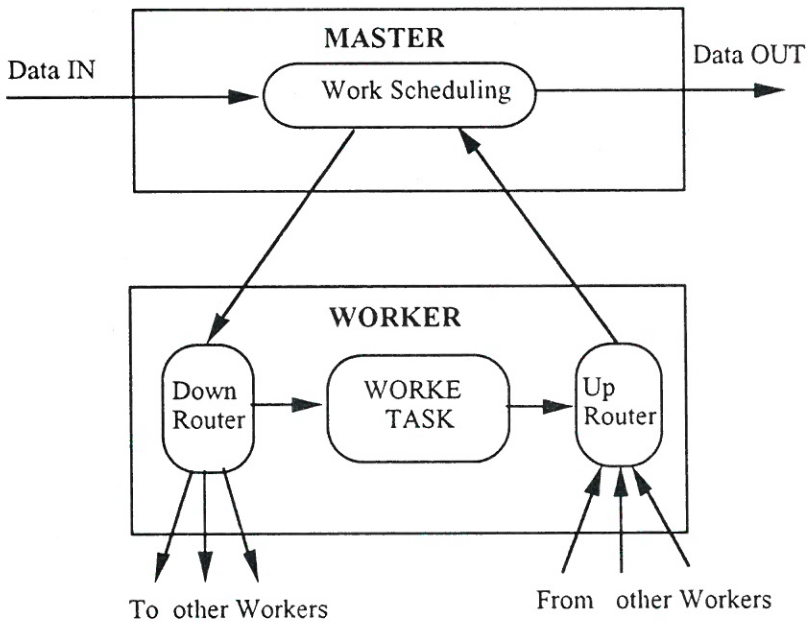


Figure 5
Transputer Implementation of the Computer Farm Model Using the InMos Links

This is a problem of load balance. To overcome this problem the *computer farm* model may be selected [11]. The *computer farm* parallelism is suitable to the algorithms executing the same set of operations on a set of different blocks of data, and the several executions do not require knowledge of each other. In a *computer farm* there are only two kinds of tasks: master and worker. The task master distributes packs of data to be processed to the workers and assembles the corresponding results. The tasks worker, for each pack of work data received, does the necessary processing work and sends back the result. The time required for a worker to process the received pack depends on the nature of the data and is not necessarily equal for all packs. This enables the balance of load, i.e. as soon as a worker processes a pack, it is ready for the next one, independently of the others.

In Figure 4 is presented the *computer farm* model corresponding to our problem. All the processors work in parallel during the evaluation of each level. The *master* processor does the work of operations scheduling and level sequencing. It sends continuously, to each *worker*, a message containing the data and the operation code

to be executed. The resulting computations are sent back by the *worker* to the *master* and the worker remains idle until receiving another work message. In this case each worker is able to execute a set of operations scheduled by the master, that means it stores internally the required program. If we assume that all processors store the same program, this system can be classified as an **MDSI** (multiple data single instruction) parallel machine.

For applying the *computer farm* model efficiently, it is necessary to balance the time required for doing the work and the time required to sending/receiving the work/results messages. Otherwise the distribution network becomes the bottleneck of the system.

The implementation of the *computer farm* model using transputers and their communications links is presented in Figure 5. Each worker transputer executes three tasks: distribution of work messages, collecting of work results and the worker task. Using 20 Mbits for transmission, one byte takes 550 ns to be transmitted from one processor to another.In case of a network of 7 workers, the transmission of one byte from the master to the worker takes more than 1.1 μs to be transmitted. For a specific application it is necessary to evaluate whether the implementation of the *computer farm* using the serial InMos Links is suitable or not, considering the size of the work/result messages and the total computation time required.

For those situations where the serial communications add a significant delay, a shared memory based communication has to be selected.

5.0 Conclusions and Future Work

A contribution to the integration of the Advanced Robot Control and the task planning areas of research is presented. Advanced Robot Control Algorithms are discussed and a case study for a two- degree- of- freedom manipulator is presented. An approach to integrating advanced controllers and task planning was made. This approach is a contribution to planning and control parameter evaluation as well as an approach to the real-time execution of the controller.

The application of parallel architectures for implementation of real-time Advanced Controllers is explored and discussed. The application of transputer- based architectures is presented and a methodology for parallelism of the control algorithm is presented.

The results obtained are very encouraging for the authors to pursue this work. The future work is two- fold: Exploitation of the presented parallelism methods for robots with more degrees of freedom; The performance evaluation of the transputer based parallel architectures for these aims will also be pursued.

Acknowledgments

The work presented in this paper and its continuation would not be possible without the support of the following organizations: EC-Flexys, CYTED-D and CONICET.

REFERENCES

1. SPONG, M ,W. and VIDYASAGAR, M., **Robot Dynamics and Control** JOHN WILEY, New York, 1989.
2. ORTEGA, R.and SPONG, M. W. , **Adaptive Motion Control of Rigid Robots: A Tutorial**, AUTOMATICA, December 1989.
3. SLOTINE, J.-J. E. and LI, W., **On the Adaptive Control of Robot Manipulators**, INT. J. ROBOT RESEARCH , Vol. 6, No.3, Fall 1987, pp.49-59.
4. KELLY, R., CARELLI, R. and ORTEGA, R. , **Adaptive Motion Control Design of Robot Manipulators : an Input-Output Approach**, INT. JOURNAL OF CONTROL , Vol. 50, No.6, 1989, pp.2563-2581.
5. ABDALLAH, C. et al, **Survey of Robust Control for Rigid Robots**, IEEE CONTROL SYSTEM MAG. , Vol.11, No.2, February 1991, pp.24-30.

6. SU, C. Y., LEUNG, T. P. and STEPANENKO, Y. , **Real-Time Implementation for Regressor-Based Sliding Mode Control Algorithm for Robotic Manipulators**, IEEE TRANS. ON INDUSTRIAL ELECTRONICS , Vol. 40, No.1, February 1993, pp.71-79.
7. SPONG, M.W., **On the Robust Control of Robot Manipulators**, IEEE TRANS. ON AUTOMATIC CONTROL, Vol. 37, November 1992.
8. CAMARINHA-MATOS, L.M. and SASTRON, F. , **Information Integration for CIM Planning Tools**, *CAPE '91* - 4th IFIP Conference on Computer Applications in Production and Engineering, Bordeaux, 10-12 September 1991.
9. CAMARINHA-MATOS, L M. and PITA H., **Interactive Planning of Motion and Assembly Operations**, IEEE International Workshop on Intelligent Motion Control, Istanbul, 20-22 August 1990.
10. FLEMING, P., JONES, D. and JONES, S., **Parallel Processing in Real-time Control: Applications, Architecture and Implementation**, COMPAR 88, Bangor, 1988, pp. 325-326.
11. PRITCHARD, D.J., **Mathematical Models of Distributed Computation, Parallel Programming of Transputer Based Machines**, IOS Amsterdam Springfield (Ed.), 1988, pp.25-36.