22-24 May 1991

"Cankarjev dom" – Cultural and Congress Center

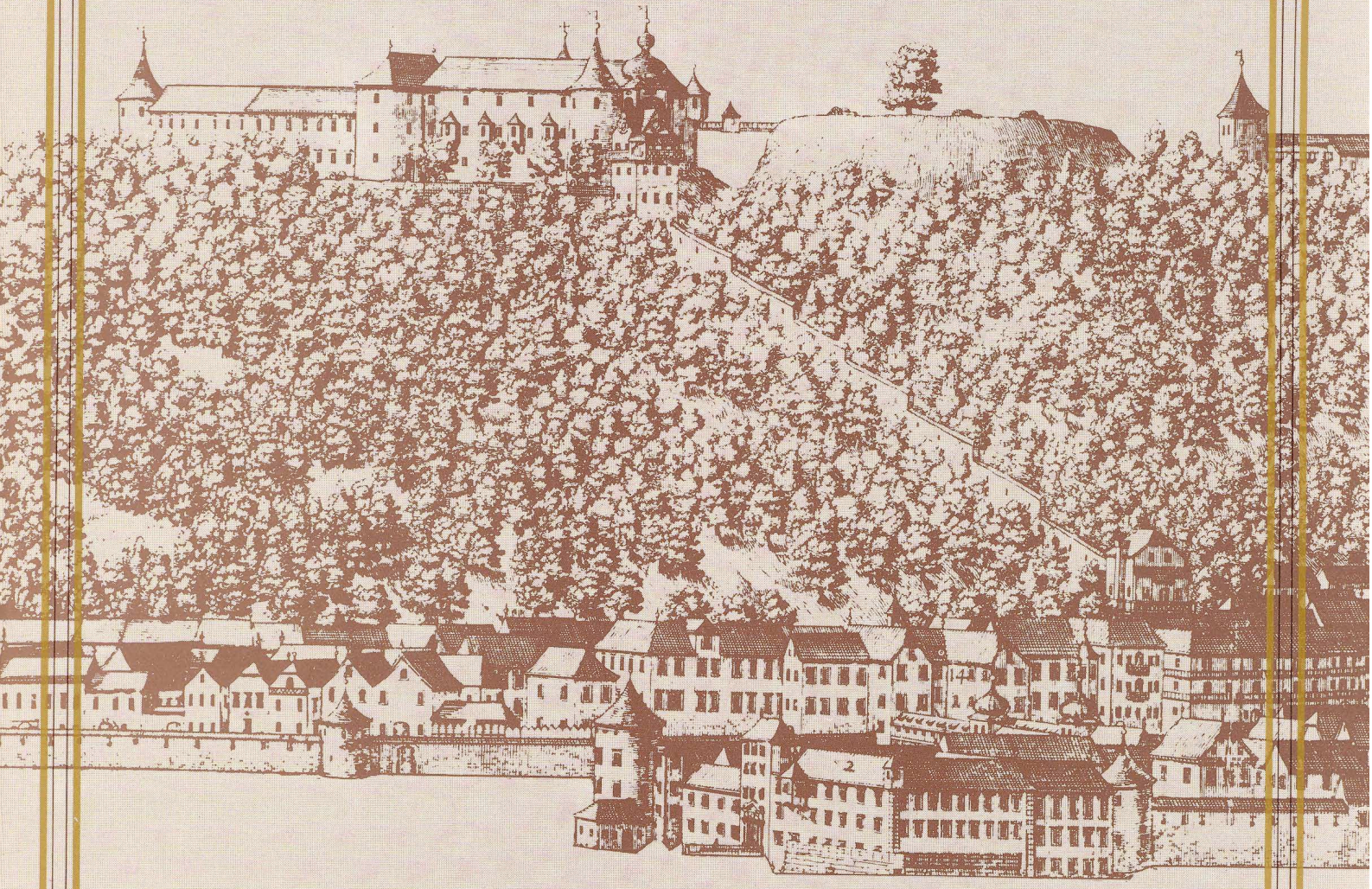
Ljubljana, Slovenia, Yugoslavia

melecon '91

MEDITERRANEAN ELECTROTECHNICAL CONFERENCE

ELECTRONICS, CONTROL AND ROBOTICS, TELECOMMUNICATIONS, COMPUTER TECHNIQUES, EDUCATION

Proceedings
Volume II



IEEE Catalog Number: 91CH2964-5
ISBN 0-87942-655-1 Softbound Edition
ISBN 0-87942-656-x Casebound Edition
ISBN 0-87942-657-8 Microfiche Edition
Library of Congress Number 90-86045

IEEE

Edited by
Baldomir Zajc
Franc Solina

ON THE EVALUATION OF COMPUTER SYSTEMS FOR ROBOT CONTROL

J.A. Tenreiro Machado, J.L. Martins de Carvalho and Alexandra M.S.F. Galhano
Faculty of Engineering of the University of Porto
Dept. of Electrical and Computer Engineering
4099 Porto Codex, Portugal

Abstract - The high computational burden posed by modern control algorithms precludes its industrial application using present day microcomputers. In this paper we evaluate the calculation load of different logical and arithmetic operations and the capabilities of several computer systems. This analysis reveals that some limitations can be alleviated through the adoption of techniques associated with the data representation. Such techniques achieve not only a more efficient management of the computational resources but also provide a deeper insight on developments towards future control architectures.

1 INTRODUCTION

In the last decades robot manipulator control has been an area of active research. These mechanical devices have complex dynamic phenomena which make difficult the development of efficient controllers. Although linear PID controllers are still used in industrial robots they are inappropriate for high performances. In fact, PID systems lead to limited path tracking accuracy and may exhibit vibrations at high speeds. The low efficiency of these systems motivated the appearance of controllers based on different concepts [1,2]. However, the high computational burden posed by many of these algorithms precludes its industrial application using present day microcomputers. While powerful monoprocessor controllers may be rather expensive, the alternative of a multiprocessor architecture [3,4] is still in a research stage and the total computational efficiency is probably far from desirable. This situation can be overcome through the development of control strategies more adapted to the computer structure. In fact, the greater the controller complexity the greater the computational requirements. Therefore, we may question about the feasibility of a given algorithm and which are the most adequate techniques to do the job. This article evaluates several Computer Control Systems (CCS's) and introduces techniques capable of render more efficient practical implementations. In this line of thought it is organized as follows. In section two we compare the computational load with respect to different logical and arithmetic operations of different CCS's. Section three presents general techniques amenable to control implementations and, finally, in section four conclusions are drawn.

2 ON THE EVALUATION OF CCS's

The evaluation of the computational load required by an algorithm and the capabilities of a given computer are essential steps in any preliminary study regarding its future implementation. Many of the algorithms suggested in the literature neglect these points, and no assessment is made about computational requirements. Some studies take into account the computational load based on the required number of sums and multiplications, while others only mention the maximum sampling frequencies achieved after the algorithm has been installed in a given system. Obviously such approaches are far from satisfactory, because they do not take into account all the "factors" involved. The difficulty of the problem lies precisely in the large number of factors involved, such as the type of processor, clock frequency, type and version of the programming language, type and accuracy of the operations, etc. The data displayed in Tables 1 and 2 clarifies these issues [5]. The tables show the range of variation of the calculation time for different operations running on several computers. Among the large number of possibilities we have chosen those combinations more relevant in controller implementation. Inspecting the results several conclusions can be drawn:

- Trigonometric operations are the most time consuming while logical and integer arithmetic operations are the fastest.
- The arithmetic coprocessor is essential to speed-up floating point operations (f.p.o.'s) but has no influence upon the computational time required by logical and integer arithmetic operations.
- For a given hardware, large variations of computing time may occur depending on the compiler being used.
- Reduced Instruction Set Computer (RISC) architectures appear to be far more efficient than conventional Complex Instruction Set Computers (CISC).

In order to provide a better perspective of these properties, we have decided to measure the frequency of calculation for an adequate benchmark. Because extrapolation from generic benchmarks are questionable [6,7], we selected a benchmark capable of reflecting the requirements associated with robot control. In this line of thought, Fig. 1 shows the frequencies of calculation of the inverse dynamics for the six degrees of freedom PUMA 560 manipulator and for the systems mentioned in Tables 1 and 2. The inverse dynamics equations already include the simplifications presented in [8]. Some improvements introduced by the authors have reduced the number of required operations to 5 sines, 6 cosines, 89 sums and 151 multiplications (Table 3). In this chart the vertical coordinate f represents the "absolute" computing frequency, while the horizontal coordinate f/f_{CLOCK} (f_{CLOCK}: clock frequency of the processor) corresponds to a "normalized" calculation rate. The results reveal that computation speed-up through the acceleration of f_{CLOCK} - which is, essentially, technological dependent - is less significant than the improvement attained by the optimization of the processor architecture ("measured" by the horizontal coordinate). Moreover, the order of magnitude of f show that, by the time the whole control algorithm is implemented, which contains several sub-algorithms such as kinematics, dynamics, control and trajectory planning, the computational load can easily reach levels incompatible with the use of high sampling frequencies. The development of techniques for the implementation of these algorithms is the subject of the next section.

3 TECHNIQUES TO IMPROVE THE PERFORMANCES OF CCS's

Modern robot control algorithms pose very stringent computational requirements. Technological progress makes available systems with ever increasing performances but, the truth is that their use as robot controllers may not be economically feasible. Therefore, in

CH2964-5/91/0000-0892$01.00

TABLE 1. Computation times for several systems

Computer	IBM PS/2 8530-021	IBM PS/2 8560-061	IBM PS/2 8570-A21	IBM PS/2 8530-021	IBM PS/2 8560-061	IBM PS/2 8570-A21	IBM PS/2 486/25 P.	INMOS T800-20
OS	MSDOS 3.3	MSDOS 3.3	MSDOS 3.3	MSDOS 3.3	MSDOS 3.3	MSDOS 3.3	MSDOS 3.3	
Clock	8 MHz	10 MHz	25 MHz	8 MHz	10 MHz	25 MHz	25 MHz	20 MHz
Proc./ /Coproc.	8086	80286	80386	8086/ /8087	80286/ /80287	80386/ /80387	80486	IMS T800-20
Language	TC V2.0	TC V2.0	TC V2.0	TC V2.0	TC V2.0	TC V2.0	TC V2.0	Occam 2
Operation	Computation Time (μsec)							
ADD fp	540-660	223-251	83-93	52-55	46-50	9-11	1.7-3.7	1.8-2.0
MULT fp	870-940	293-299	110-114	57-63	51-53	9-11	1.7-3.7	2.5-2.6
SIN fp	3500-4300	939-1175	335-418	300-360	236-269	27-39	11-22	38.8-40.7
ADD int	8.3	4.3	1.2	8.3	4.3	1.2	0.5	0.66
MULT int	22.5	5.2	1.5	22.5	5.2	1.5	1.1	2.54
IF	9.1	4.5	1.1	9.1	4.5	1.1	0.47	0.922

TABLE 2. Computation times for several systems

Computer	DECSTATION 3100	Apollo 3500	SUN 3-60		SUN 4-110	SUN 4-60 SPARC St.1	AViiON AVX 300	IBM S/6000 Pow.St.530
OS	Ultrix 3.1	BSD4.2 IX	Unix 4.2		Unix 4.3.2	SUN 4.0.3	DG/UX 4.2	AIX V3.1
Clock	16.7 MHz	25 MHz	20 MHz		14.3 MHz	20 MHz	16.7 MHz	25 MHz
Proc./ /Coproc.	MIPS 2000/ /2010	68030/ /68882	68020/ /68881		SPARC	SPARC	88100	IBM Power Syst/6000
Language	System C	System C	System C	GNU V1.25	System C	System C	System C	System C
Operation	Computation Time (μsec)							
ADD fp	0.16-0.33	7.3-7.5	18-19	11-13	3-4	1.9-2.1	2.2-2.5	0.13-0.15
MULT fp	0.16-0.33	7.3-7.5	15-16	11-12	3-4	2.1-2.2	2.3-2.8	0.13-0.14
SIN fp	5.0-8.3	23-25	17-20	20-25	5-7	4.0-4.3	22-25	2.3-3.2
ADD int	0.024	0.4	1.2	0.5	0.9	0.63	2.7	0.013
MULT int	0.024	0.6	3.3	0.5	2.8	1.7	2.7	0.013
IF	0.055	1.3	1.6	1.6	1.2	6.2	0.67	0.034

TC: Turbo C. Precision of the floating point (fp) operations: 8 bytes, Integer (int) operations: 4 bytes

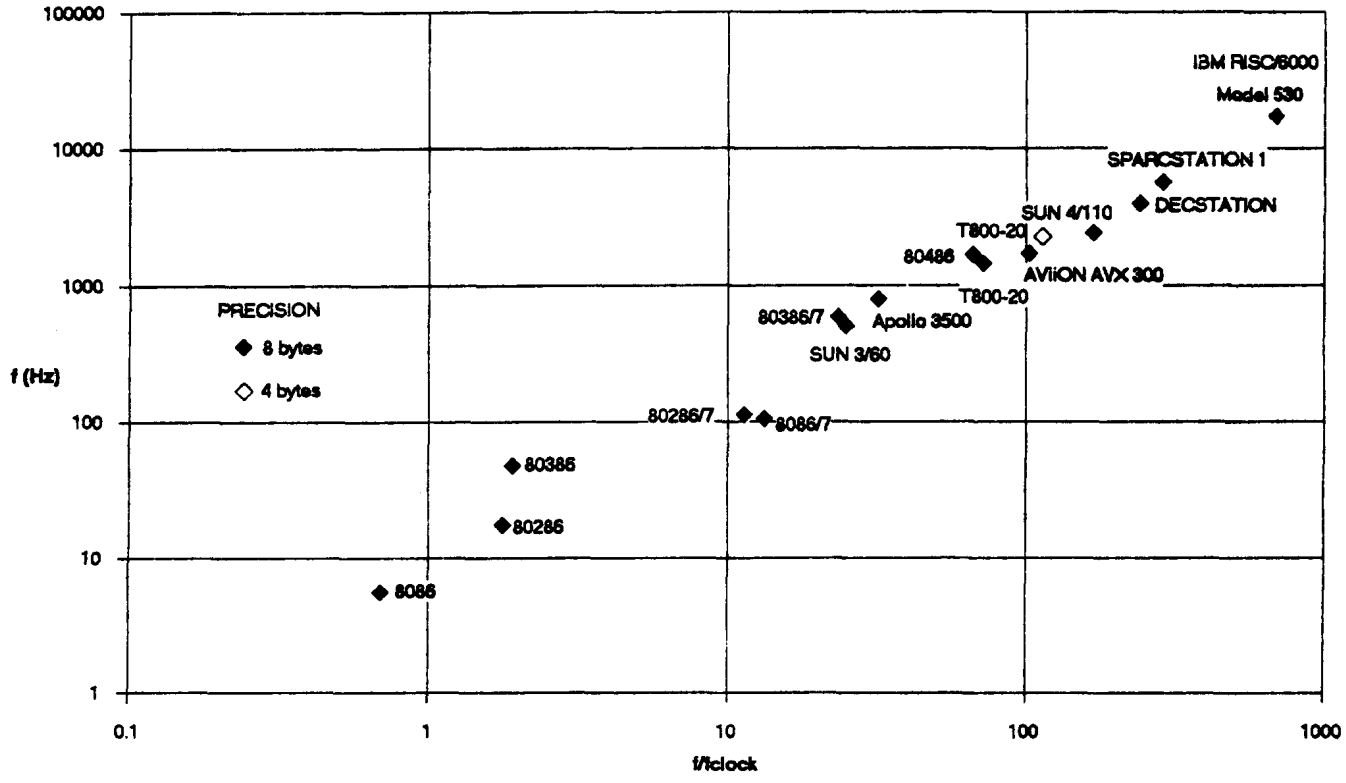


Figure 1: Chart of the "absolute" computing frequencies f for PUMA 560 dynamic equations versus the "normalized" index f/f_{clock} for the computers considered in Tables 1 and 2.

TABLE 3. Dynamic equations of the PUMA 560 robot manipulator

$S_2=SIN(p_2)$
$C_2=COS(p_2)$
$S_3=SIN(p_3)$
$C_3=COS(p_3)$
$S_4=SIN(p_4)$
$C_4=COS(p_4)$
$S_5=SIN(p_5)$
$C_5=COS(p_5)$
$p_{23}=p_2+p_3$
$S_{23}=SIN(p_{23})$
$C_{23}=COS(p_{23})$
$C_{223}=COS(p_2+p_{23})$

$SS_{23}=S_{23}*S_{23}$
$SC_{23}=S_{23}*C_{23}$
$C_2S_{23}=C_2*S_{23}$
$C_2C_{23}=C_2*C_{23}$
$S_4S_5=S_4*S_5$
$C_4S_5=C_4*S_5$
$S_4C_5=S_4*C_5$
$C_4C_5=C_4*C_5$
$C_{23}C_4C_5=C_{23}*C_4C_5$

$m_1=-8.4*S_{23}$
$m_2=-0.134*C_{23}$
$m_3=-0.0025*S_5$
$m_4=[0.00164+0.0003*(1-2*S_4*S_4)]*S_{23}-0.0025*C_{23}*C_4S_5$
$m_5=-(0.0025*C_5+0.000642)*C_{23}*S_4$
$m_6=0.6*SC_{23}-0.0213*(1-2*SS_{23})$

$a_{12}=0.69*S_2+m_2+0.0238*C_2$
$a_{13}=m_3-0.00397*S_{23}$
$a_{22}=0.333+0.372*S_2-0.011*C_2$
$a_{24}=-0.00125*S_4S_5$
$a_{25}=0.00125*C_4C_5$

$b_{112}=-2.76*S_2*C_2+0.744*C_{223}+m_6$
$b_{113}=0.744*C_2C_{23}+0.022*C_2S_{23}+m_6$
$b_{114}=-0.0025*SC_{23}*S_4S_5+0.00086*C_4S_5-0.00248*C_2C_{23}*S_4S_5$
$b_{115}=-0.0025*(SS_{23}*S_5-SC_{23}*C_4C_5)$
$\quad -0.00248*C_2*(S_{23}*S_5-C_{23}C_4C_5)+0.00086*S_4C_5$
$b_{214}=m_4+0.00248*S_{23}*C_4S_5$
$b_{215}=m_5+0.00248*S_{23}*S_4C_5$
$b_{223}=0.022*S_{23}+0.744*C_{23}$
$b_{224}=-0.00248*C_{23}*S_4S_5$
$b_{225}=m_3+0.00248*(C_{23}*C_4C_5-S_{23}*S_5)$
$b_{415}=-0.000642*S_{23}*C_4$
$b_{425}=0.000642*S_4$

$c_{21}=-0.5*b_{112}$

$v_1v_1=v_1*v_1$
$v_1v_2=v_1*v_2$
$v_1v_3=v_1*v_3$
$v_1v_4=v_1*v_4$
$v_1v_5=v_1*v_5$
$v_2v_2=v_2*v_2$
$v_2v_3=v_2*v_3$
$v_2v_4=v_2*v_4$
$v_2v_5=v_2*v_5$
$v_3v_3=v_3*v_3$
$v_3v_4=v_3*v_4$
$v_3v_5=v_3*v_5$

$v_2v_2v_3v_3=v_2v_3+0.5*v_2v_2$
$v_2v_4v_3v_4=v_2v_4+v_3v_4$
$v_2v_5v_3v_5=v_2v_5+v_3v_5$

$T_1=(2.57+1.38*C_2*C_2+0.3*SS_{23}+0.744*C_2S_{23})*w_1$
$\quad +a_{12}*w_2+a_{13}*w_3$
$\quad +b_{112}*v_1v_2+b_{113}*v_1v_3+b_{114}*v_1v_4+b_{115}*v_1v_5$
$\quad +(0.267*S_{23}-0.00758*C_{23})*v_2v_3v_3v_3$
$\quad +(0.69*C_2+0.134*S_{23}-0.0238*S_2)*v_2v_2$

$T_2=a_{12}*w_1+(6.79+0.744*S_2)*w_2+a_{23}*w_3$
$\quad +b_{214}*v_1v_4+b_{215}*v_1v_5+b_{223}*v_2v_3v_3v_3$
$\quad +b_{224}*v_2v_4v_3v_4+b_{225}*v_2v_5v_3v_5$
$\quad -0.5*b_{112}*v_1v_1$
$\quad -37.2*C_2+m_2+1.02*S_2$

$T_3=a_{13}*w_1+a_{23}*w_2+1.16*w_3+a_{24}*w_4+a_{25}*w_5$
$\quad +m_4*v_1v_4+m_5*v_1v_5+m_3*v_2v_3v_3v_3-0.0025*S_4C_5*v_4*v_5$
$\quad +c_{21}*v_1v_1-0.5*b_{223}*v_2v_3-0.00125*C_4S_5*(v_4*v_4+v_5*v_5)$
$\quad +m_1+0.25*C_{23}$

$T_4=a_{24}*w_2+0.2*w_4$
$\quad +b_{214}*v_1v_2-b_{214}*v_1v_3+b_{415}*v_1v_5+b_{425}*v_2v_3v_3v_5$
$\quad -0.5*(b_{114}*v_1v_1+b_{224}*v_2v_2)$
$\quad +0.028*S_{23}*S_4S_5$

$T_5=a_{25}*w_2+0.18*w_5$
$\quad -b_{215}*v_1v_2-b_{215}*v_1v_3-b_{415}*v_1v_4-m_5*v_2v_3v_3v_3$
$\quad -b_{425}*v_2v_4v_3v_4$
$\quad -0.5*(b_{115}*v_1v_1+b_{225}*v_2v_2)$
$\quad -0.028*(C_{23}S_5+S_{23}*C_4C_5)$

$T_6=0.19*w_6$

the sequel we present a set of techniques to improve the performances of CCS's, which are, to a large extent "hardware independent" [9]. We group these techniques into six categories:

- Programming in assembly language.
- Calculation with low precision.
- Use of memory.
- Multirate sampling.
- Preview techniques.
- Dedicated compilers.

These techniques are discussed in the next subsections.

3.1 Programming in Assembly Language

Programming in assembly language is a well known technique and constitutes a natural starting point to improve the performances of any CCS. In fact, this method allows a considerable optimization of the object code. However, modern control algorithms are very complex and that makes their programming in assembly very time consuming. This is the reason why only a few researchers have adopted this strategy [10,11] as an alternative to the programming in high level languages such as FORTRAN or C.

3.2 Calculation with Low Precision

The calculation with low precision has been one of the main alternatives to the programming in assembly language. The most common technique consists in giving up f.p.o.'s in favour of fixed point arithmetic calculations [12,13]. At a first sight it would seem more practical to reduce only the accuracy of the f.p.o.'s. However, this is somewhat deceptive because many of the high level languages implement low accuracy f.p.o.'s on the basis of operations with standard accuracy. Therefore, and contrary to our expectations, there is no improvement of the computation times. Clearly, if the high level language has distinct arithmetic libraries for different accuracies then computation can be speeded up [7] (Fig. 1).

3.3 Use of Memory

This technique transfers the calculation load to memory evaluations. A common procedure consists of replacing trigonometric f.p.o.'s by memory look-up tables [10,13]. The use of memory evaluations can be generalized to a greater portion of the algorithm however, the application of this technique to CCS's has not received much attention to date with the exception of a few controllers based on learning strategies [14,15]. Therefore, a large field of research remains open.

3.4 Multirate Sampling

The effects of the finite sampling frequency upon the performances of a CCS are very important. In a control system, made of several feedback and feedforward paths, it is natural to expect different bandwidths along them. Therefore, it is reasonable to allocate different sampling (and computing) rates to such paths. This strategy forms the basis of multirate sampling [16,17]. In this way the computing power is assigned to each loop in accordance to its real needs, allowing therefore a more rational management of the computer resources.

3.5 Preview Techniques

Because microprocessors have limited computational capabilities the maximum allowable sampling frequency of the controller is bounded above. At the same time this gives rise to a time delay between the instant sensors are read and the control command is issued. Preview techniques attempt to minimize this detrimental effect. Their application to the CCS's for robot manipulators has shown promising results [18] yet, it is still in a research stage.

3.6 Dedicated Compilers

The implementation of a control algorithm assumes the representation of system variables through real numbers. To these real numbers corresponds a computer internal representation by means of floating point structures. This representation, requiring several bytes, is in sharp contrast with the 8 to 16 bit accuracy of standard A/D and D/A converters. The authors developed a method for implementing control algorithms [19,20] that eliminates this discrepancy, by optimizing the chain:

Data Collection -> Processing -> Control Action

This method adopts an uniform accuracy for all variables that is, both for the computer internal and external variables. Once the admissible range of a variable is known, quantization levels can be defined as a function of the accuracy of the A/D and D/A converters. To each of these levels, an integer binary code is assigned and the algorithm can be processed by Boolean operations upon the bits representing the variables. Furthermore, the Boolean operations are optimized by means of Binary Decision Diagrams, that are very efficient, once converted to IF_THEN_ELSE structures as shown by Tables 1 and 2.

This method is still in a research stage however, it has already suggested interesting extensions to RISC computational structures and parallel architectures. In the first case the extension is motivated by the fact that processing by means of BDD's only requires microprocessors with a reduced number of instructions; in the second case the advantage stems from the simplicity of task scheduling to the processors in parallel.

4 CONCLUSIONS

The high computational burden posed by modern cobntrol algorithms precludes its industrial application using present day microcomputers. This situation can be overcome through the development of control strategies more adapted to CCS's. The analysis of the computational requirements and the evaluation of the computer capabilities reveals that limitations are alleviated through the adoption of techniques associated with the data representation. Furthermore, the use of these techniques achieves not only a more efficient management of the computational resources but also provides a deeper insight on developments towards future CCS's.

REFERENCES

[1] J.Y.S. Luh, M.W. Walker and R.P.C. Paul: Resolved-Acceleration Control of Mechanical Manipulators, IEEE Trans. Automat. Contr., vol. 25, no. 3, pp. 468-474, June 1980.

[2] E. Freund: Fast Nonlinear Control with Arbitrary Pole-Placement for Industrial Robots and Manipulators, The Int. J. Robotics Research, vol. 1, no. 1, pp. 65-78, Spring 1982.

[3] J.Y.S. Luh and C.S. Lin: Scheduling of Parallel Computation for a Computer-Controlled Mechanical Manipulator, IEEE Trans. Syst., Man, Cybern., vol. 12, no. 2, pp. 214-234, March/April 1982.

[4] T. Watanabe, M. Kametani, K. Kawata and K. Tetsuya: Improvement in the Computing Time of Robot Manipulators Using a Multimicroprocessor, J. Dynamic Syst. Measurement, Contr., Trans. ASME, vol. 108, no. 3, pp. 190-197, September 1986.

[5] J.A.T. Machado and J.L.M. de Carvalho: Microprocessor-Based Controllers for Robotic Manipulators, in Microprocessors in Robotic and Manufacturing Systems (to be published) Kluwer Academic Publishers (Spyro Tzafestas, editor).

[6] W.J. Price: A Benchmark Tutorial, IEEE Micro, vol. 9, no. 5, pp. 28-43, October 1989.

[7] Lies, damned lies and benchmarks in The Transputer Applications Notebook: Systems and Performance, INMOS, pp. 258-279, 1989.

[8] B. Armstrong: O. Khatib and J. Burdick, The Explicit Dynamic Model and Inertial Parameters of the PUMA 560 Arm, Proc. 1986 IEEE Int. Conf. on Robotics and Automation, San Francisco, CA, 1986.

[9] J.A.T. Machado, J.L.M. de Carvalho and A.M.S. Galhano: Computer System Evaluation in Robot Control, IEEE Int. W. on Intelligent Motion Control, Istanbul, Turkey.

[10] M.W. Spong, J.S. Thorp and J.M. Kleinwaks: Robust Microprocessor Control of Robot Manipulators, Automatica, vol. 23, no. 3, pp. 373-379, 1987.

[11] S.H. Murphy and G.N. Saridis: Experimental Evaluation of Two Forms of Manipulator Adaptive Control, Proc. 26th IEEE Conf. on Decision and Control, Los Angeles, CA, 1987.

[12] C.H. An, C.G. Atkeson, J.D. Griffiths and J.M. Hollerbach: Experimental Evaluation of Feedforward and Computed Torque Control, Proc. 1987 IEEE Int. Conf. on Robotics and Automation, Raleigh, NC, 1987.

[13] T. Suehiro and K. Takase: A Manipulation System Based on Direct-Computational Task-Coordinate Servoing, Robotics Research: The Second International Symposium, MIT Press, 1985.

[14] J.S. Albus: A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC), J. Dynamic Syst. Measurement, Contr., Trans. ASME, vol. 97, pp. 220-227, 1975.

[15] S. Kawamura, F. Miyazaki and S. Arimoto: Realization of Robot Motion Based on a Learning Method, IEEE Trans. Syst., Man, Cybern., vol. 18, no. 1, pp. 126-134, January/February 1988.

[16] J.A.T. Machado and J.L.M. de Carvalho: Engineering Design of a Multirate Nonlinear Controller for Robot Manipulators, J. Robotic Systems, vol. 6, no. 1, pp. 1-17, February 1989.

[17] M. Kircanski, M. Vukobratovic, N. Kircanski and T. Timcenko: A New Program Package for the Generation of Efficient Manipulator Kinematic and Dynamic Equations in Symbolic Form, Robotica, vol. 6, Part 4, pp. 311-318, October 1988.

[18] K. Yoshimoto and H. Sugiuchi: Trajectory Control of Robot Manipulator Based on the Preview Tracking Control Algorithm, Robotics Research: The Second International Symposium, MIT Press, 1985.

[19] J.A.T. Machado, J.L.M. de Carvalho, J.A.S. Matos and A.M.C. Costa: An Efficient Computational Scheme for Robot Manipulators, in Proc. IEEE Int. Symp. on Intelligent Control, Arlington, Virginia, 1988.

[20] J.A.T. Machado, J.L.M. de Carvalho, J.A.S. Matos and A.M.C. Costa: Robot Manipulator Dynamics - Towards Better Computational Algorithms, in Proc. IFAC Symp. on Robot Control, Karlsruhe, FRG, 1988.