

Fault Tolerant Algorithms and Architectures for Robotics

D.L. Hamilton, M.L. Visinsky
J.K. Bennett, J.R. Cavallaro, and I.D. Walker
Department of Electrical and Computer Engineering
Rice University
Houston, Texas 77251-1892 USA
(713) 527-4020

Abstract

As robot tasks in space, nuclear, and medical environments become more widespread, the issues of reliability and safety for robots are becoming more critical. Attempts to address these issues have resulted in a recent surge of activity in robot fault tolerance. We concentrate on fault tolerance in the robot controller, and highlight the importance and potential of multiprocessor control architectures from the fault tolerance perspective. The issue of performance versus reliability is discussed. This paper also summarizes other work by our group at Rice University in the area of fault tolerance for robotics.

Introduction

There is much ongoing interest in deployment of robots in remote and hazardous environments. Applications include hazardous waste management and clean-up [6] and space-based operations [5]. In these spheres of operation, the use of humans to perform tasks is limited to short periods of time, and the work is often dangerous as well as difficult. The inherent problems in having humans perform tasks safely in these environments make robots an attractive, if not mandatory, alternative. However, the remoteness of such environments, the safety-critical tasks required, and the difficulty of repairing the robots after failure make fault tolerance and reliability more critical issues than in conventional applications [7]. Recently, there has been increasing interest in robot fault tolerance, and the subject has been investigated from a number of points of view. Ongoing work includes analysis of redundant systems [16, 17, 21, 23, 32], modular software environments [3, 20, 29], fault tolerant control environments [12, 13, 22, 24, 31], error recovery [8, 19], and fault detection [11, 26, 27, 33].

However, there appears to have been little work in utilizing multiprocessor architectures to obtain increased reliability. Multiprocessor controllers have been proposed for increased performance in robotics, but the opportunity for fault tolerance inherent in multiprocessor architectures has not been fully exploited. We are performing ongoing research in this area [12, 13]. In this paper, we present a discussion of the issues involved in designing reliable parallel multiprocessor controllers for robotics, together

Hardware Redundancy

The number of processors available to do work determines the number of processor failures that can be tolerated. The processors working together and comparing iterative results are called the “working set”. When a failure occurs, the working set size is reduced, and the existing hardware is reconfigured. The maximum number of processor failures that can be tolerated is m , where $n - m = 2$ and n is the number of processors originally in the working set. This allows two processors to continue working and comparing after all others have failed. If a disagreement occurs with only two processors in the working set, it is not possible to determine which processor is faulty. When this happens, the robot can discontinue operation, or one processor can be chosen to continue working based on a log of previous maintenance or results of self-testing by each processor. Either option is no worse than the case for a uniprocessor controller, in which the failure would not be detected at all. In our architecture, the working set size is initially the number of degrees of freedom of the robot. Thus, the number of failures that can be tolerated will be two less than the degrees of freedom [13].

Software Redundancy

There are two basic options for providing software redundancy. The simplest is to have all processors execute the same code to perform an operation. As long as all processors are fault-free, they should produce the same results. The results of intermediate calculations are compared, and a working set is derived for sending data to the robot. The number of comparisons that take place per iteration is dependent upon the level of fault tolerance desired. Exact agreement in the data must be maintained in this case.

Another option is to have each processor perform the same operation, but in a different way. For a robot controller system, for example, the processors could receive input from different types of sensors. Each would make calculations based on the data received from its corresponding sensor. This approach takes into account the possibility of generic faults in the software by using different versions of the software to produce the same result. This approach is similar to the approach used in the design of fault

Speedup Techniques

Robotic systems are typically compute-bound. That is, system performance is directly related to how fast the kinematics and dynamics computations can be performed. Most of the literature discussing the use of parallel algorithms and multiprocessors for robot control have the sole goal of providing faster responses from the controller [1, 14, 18, 28]. Real-time control is mandatory for a robot to be useful, and fast control offers greater possibilities for robot usage. Parallel algorithms exist to solve both the forward and inverse dynamics and kinematics [4, 10, 30]. There are also parallel algorithms for computing the inertia matrix [9, 15]. Currently, we are using a parallel dynamics algorithm only, because the inverse dynamics is by far the most computationally intensive part of robot control. Furthermore, a general approach to parallelizing the code, such as ours, could be applied to the kinematics, as well as the dynamics.

One of the variations in controller architectures is in the memory structure (data communication). Both shared memory multiprocessors and message-passing multiprocessor architectures have been utilized as robot controllers. Due to the interdependence of joint information in robotics control, all processors must have access to data pertaining to every joint. Thus, we utilize shared memory because our goal is to provide the greatest efficiency in a fault-tolerant system. A distributed private memory architecture would require explicit communication between processors due to the data interdependence, which would adversely affect the response time of the controller [12].

Multiprocessor Architecture

Multiprocessor control architectures have been presented as the solution to the processor fault tolerance problem, as well as the controller latency problem. However, the notion of addressing both issues with a single architecture has not been considered. We are using a multiprocessor architecture to provide controller speedup and processor fault tolerance. Intuitively, solutions to these two problems conflict. The fastest program would have each processor executing different code so that the entire task would be completed in minimum time. Use of redundant code and additional code for data comparisons hinders the performance gains of such an algorithm. However, it is possible to combine the two efforts to achieve an efficient, fault tolerant control architecture. If processor fault tolerance is most important, then more time can be dedicated to ensuring that the control processors are fault-free. On the other hand, if fast controller response is more critical, fewer precautions will be taken to recover from failures so that the computations complete sooner. A good balance of efficient parallel code and redundancy can provide better performance than a uniprocessor robot controller.

The speedup technique that we are investigating, called zero-order prediction, was originally proposed by Binder and Herzog [2]. In this method, old values of data are used in performing the dynamics calculations. This reduces the computation time by eliminating the time spent waiting for previous iterations to complete. However, using old values also adds some error to the resulting control solution. Binder and Herzog demonstrated that

Therefore, the tasks being executed on a processor are those associated with the corresponding joint. For the case where a task corresponds to more than one joint, tasks are assigned so that the processors will complete all of their tasks at nearly the same time. The Newton-Euler algorithm that we use is more difficult to parallelize because of its recursive structure, and will require a different processor assignment. In this algorithm, a single task is a smaller portion of a complete equation. Making the calculation of the angular velocity, for example, of one joint into a single task would force the processors to operate serially. Instead, a matrix/vector multiplication that is only part of the calculation of the angular velocity is delegated to a single processor. A second processor performs another portion of the angular velocity calculation.

Previously, we implemented several Lagrangian-based control programs to demonstrate the validity of our approach [12, 13]. The first, a serial program, was used as a benchmark for the study. Following that program was a multiprocessor controller in which each processor executed exactly the same algorithm and compared calculation results at designated points. This program was slower than the serial program because there was no overlapping of tasks and data comparison code was added. The third program developed was a parallel algorithm in which the tasks were divided among the processors in the working set. Only at certain points would the processors perform the same operation and compare results for failure detection. This control program executed faster than the serial controller and detected processor failures as well [12].

Through comparison of the first and third programs described, we showed how a parallel algorithm with some redundant code can be much more efficient than a serial algorithm. This work utilized the task decomposition method described for the Lagrangian-based controller. However, no additional speedup techniques were used. In our current work, we use zero-order prediction for improved performance [2]. Processor fault tolerance can be implemented as it was in our preliminary work. Using multiple control programs, we are investigating the performance/fault tolerance tradeoff that exists when faster control and processor fault tolerance are provided in a single architecture.

Other Ongoing Fault Tolerance Work at Rice

In addition to the work just described, we have developed a new overall fault tolerance framework [25] that contains an expert system and a hierarchical software control environment. The framework, incorporating a fault tree database, covers all levels from the operator to the robot itself, including both system control software and sensor/motor hardware. An intermediate level software critic layer protects the robot from incorrect operator commands [29] and monitors the reduced reachable workspace as joints fail. Fault detection [27] and fault tolerance [26] are performed at a lower level interface layer covering the robot sensor and actuator hardware. Status signals from the interface level propagate back up to the fault tree database and critic layer. Portions of this integrated testbed have been implemented on the Segment 5 manipulator at the Robotics and Automation Center at Rice University.

controller, and pointed out the value of multiprocessor controllers for fault tolerance as well as performance.

Acknowledgements

This work was supported in part by the National Science Foundation under grants MSS-9024391 and DDM-9202639, by DOE Sandia National Laboratory Contract #18-4379A, by NASA Graduate Fellowship NGT-70251, and by NSF Graduate Fellowship RCD-9154692.

References

- [1] R.L. Andersson. Computer Architectures for Robot Control: A Comparison and a New Processor Delivering 20 Real Mflops. In *1989 IEEE International Conference on Robotics and Automation*, pages 1162–1167, Scottsdale, AZ, 1989.
- [2] E.E. Binder and J.H. Herzog. Distributed Computer Architecture and Fast Parallel Algorithms in Real-Time Robot Control. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-16(4):543–549, 1986.
- [3] P. T. Boissiere and R. W. Harrigan. An Alternative Control Structure for Telerobotics. *Proceedings of the NASA Conference on Space Telerobotics*, pages 141–150, January 1989.
- [4] C.L. Chen, C.S.G. Lee, and E.S.H. Hou. Efficient Scheduling Algorithms for Robot Inverse Dynamics Computation on a Multiprocessor System. In *1988 IEEE International Conference on Robotics and Automation*, pages 1146–1151, Philadelphia, PA, 1988.
- [5] A. Cohen and J. D. Erickson. Future Uses of Machine Intelligence and Robotics for the Space Station. *IEEE Journal of Robotics and Automation*, RA-1(3):117–123, Sept 1985.
- [6] Department of Energy, Washington, DC. *Environmental Restoration and Waste Management Robotics Technology Development Program Robotics 5-Year Plan*, 1990. DOE/CE-0007T, Vol. 1-3.
- [7] B.S. Dhillon. *Robot Reliability and Safety*. Springer-Verlag, New York, 1991.
- [8] B.R. Donald. *Error Detection and Recovery in Robotics*. Springer-Verlag, New York, 1989.
- [9] A. Fijany and A.K. Bejczy. A Class of Parallel Algorithms for Computation of the Manipulator Inertia Matrix. In *1989 IEEE International Conference on Robotics and Automation*, pages 1818–1826, Scottsdale, AZ, 1989.
- [10] A. Fijany and A.K. Bejczy. Parallel Algorithms and Architecture for Computation of Manipulator Forward Dynamics. In *1991 IEEE International Conference on Robotics and Automation*, pages 1156–1162, Sacramento, CA, 1991.
- [11] B. Freyermuth. An Approach to Model Based Fault Diagnosis of Industrial Robots. In *1991 IEEE International Conference on Robotics and Automation*, pages 1350–1356, Sacramento, CA, April 1991.
- [12] D. L. Hamilton, J. K. Bennett, and I. D. Walker. Parallel Fault-Tolerant Robot Control. In *Proc. SPIE Conference on Cooperative Intelligent Robotics in Space III*, pages 251–261, Boston, MA, November 1992.
- [13] D. L. Hamilton, J. K. Bennett, and I. D. Walker. Simulation of a Reliable Parallel Robot Controller. In *Proc. NASA International Simulation Technology Conference (SIMTEC '92)*, pages 321–327, Houston, TX, November 1992.
- [14] J.Y. Han and C.Y. Wang. Modeling and Performance Evaluation of Multiprocessor Systems for Real-Time Nonlinear Robot Control. In *1989 IEEE International Conference on Robotics and Automation*, pages 1016–1021, Scottsdale, AZ, 1989.

- [18] F. Naghdy, C.K. Wai, and G. Naghdy. Multiprocessing Control of Robotic Systems. In *1988 IEEE International Conference on Robotics and Automation*, pages 975–977, Philadelphia, PA, 1988.
- [19] A. K. Pradeep, P. J. Yoder, R. Mukundan, and R. J. Schilling. Crippled Motion in Robots. *IEEE Transactions on Aerospace and Electronic Systems*, 24(1):2–13, January 1988.
- [20] D.B. Stewart, R.A. Volpe, and P.K. Khosla. Integration of Real-Time Software Modules for Reconfigurable Sensor-based Control Systems. In *Proc. 1992 IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 325–332, Raleigh, NC, July 1992.
- [21] D. Tesar, D. Sreevijayan, and C. Price. Four-Level Fault Tolerance in Manipulator Design for Space Operations. In *First International Symposium on Measurement and Control in Robotics*, volume 3, page J3.2.1, Houston, TX, June 1990.
- [22] Y. Ting, S. Tosunoglu, and D. Tesar. A Control Structure for Fault-Tolerant Operation of Robotic Manipulators. In *Proceedings 1993 IEEE International Conference on Robotics and Automation*, pages 684–690, Atlanta, GA, 1993.
- [23] G. Toye. Management of Non-Homogeneous Functional Modular Redundancy for Fault Tolerant Programmable Electro-Mechanical Systems. Ph.D. Thesis, Stanford University, July 1989.
- [24] K.S Tso, M. Hecht, and N. Marzwell. Fault-Tolerant Robotic System for Critical Applications. In *Proceedings 1993 IEEE International Conference on Robotics and Automation*, pages 691–696, Atlanta, GA, 1993.
- [25] M.L. Visinsky, J.R. Cavallaro, and I.D. Walker. Expert System Framework for Fault Detection and Fault Tolerance in Robotics. *International Journal of Computers and Electrical Engineering*, 1994. to appear.
- [26] M.L. Visinsky, I.D. Walker, and J.R. Cavallaro. Layered Dynamic Fault Detection and Tolerance for Robots. In *Proceedings 1993 IEEE International Conference on Robotics and Automation*, pages 180–187, Atlanta, GA, 1993.
- [27] M.L. Visinsky, I.D. Walker, and J.R. Cavallaro. New Dynamic Model-Based Fault Detection Thresholds for Robot Manipulators. In *Proceedings 1994 IEEE International Conference on Robotics and Automation*, San Diego, CA, 1994. to appear.
- [28] I.D. Walker and J.R. Cavallaro. Parallel VLSI Architectures for Real-Time Control of Redundant Robots. In *Proceedings of the Fourth ANS Topical Meeting on Robotics and Remote Systems*, pages 299–310, Albuquerque, NM, 1991.
- [29] I.D. Walker and J.R. Cavallaro. Dynamic Fault Reconfigurable Intelligent Control Architectures for Robotics. In *Proceedings 1993 Fifth American Nuclear Society Meeting on Robotics and Remote Handling*, pages 305–312, Knoxville, TN, 1993.
- [30] W. Wang, K. Chen, Y. Lai, and C. Liu. Implementation of a Multiprocessor System for Real-Time Inverse Dynamics Computation. In *1989 IEEE International Conference on Robotics and Automation*, pages 1174–1179, Scottsdale, AZ, 1989.
- [31] T. Wikman and W. Newman. Reflex Control for Robot System Preservation and Reliability. In *Proceedings of Fourth International Symposium on Robotics and Manufacturing*, pages 979–986, Sante Fe, NM, November 1992.
- [32] E. Wu, M. Diftler, J. Hwang, and J. Chladek. A Fault Tolerant Joint Drive System for the Space Shuttle Remote Manipulator System. In *1991 IEEE International Conference on Robotics and Automation*, pages 2504–2509, Sacramento, CA, April 1991.
- [33] J. Wunnenberg and P.M. Frank. Dynamic Model Based Incipient Fault Detection Concept for Robots. In *Proceedings of 1990 IFAC*