Proceedings of the
ANS Fifth Topical Meeting on

ROBOTICS AND REMOTE SYSTEMS

Knoxville, Tennessee / April 25 - 30, 1993

VOLUME I

Publication Date: April, 1993

Sponsored by the American Nuclear Society - Robotics and Remote Systems Division
American Nuclear Society - Oak Ridge/Knoxville Chapter

Co-Sponsored by the
Oak Ridge National Laboratory (ORNL)
Electric Power Research Institute (ERPI)
The Utility/Manufacturers Robot Users Group (UMRUG)

In cooperation with
The International Atomic Energy Agency (IAEA)
The Institute of Electrical and Electronic Engineers (IEEE)
Robotics and Automation Society

Proceedings of the
Fifth ANS Topical Meeting on Robotics and Remote Systems
Knoxville, TN (April 1993).

DYNAMIC FAULT RECONFIGURABLE INTELLIGENT CONTROL ARCHITECTURES FOR ROBOTICS

Ian D. Walker and Joseph R. Cavallaro
Dept. of Electrical and Computer Engineering
Rice University, Houston, TX 77251
Tel: (713) 527-4020, FAX: (713) 524-5237

Abstract

In this paper we describe new progress in our development of an Intelligent Control Framework for robots which dynamically reconfigures itself to cope with faults in either sensors or joint hardware. The Framework is configured to allow the incorporation of new approaches for on-line critiquing of user plans and commands within the framework. We discuss integration of the two components above to produce an Intelligent Robot Operating System which can tolerate failures or unexpected actions from both the logical (user) world and the physical (manipulator) world and continue operation where possible.

1 Introduction

Intelligent control is a concept that has been advocated for some time, and has generated much interest. The idea of integrating intelligence and concepts from the logical (discrete) worlds of Computer Science, Artificial Intelligence, etc. with those of the physical world of traditional Dynamics and Control, is hugely appealing, and has many key applications. The challenge of Intelligent Control is to produce a body of knowledge which is independently useful to, and usable by, both groups.

This work involves formal research into fusion of discrete and continuous (logical and physical) systems. We are developing fault tolerant robot architectures (at both logical and physical levels, and at the interaction levels between the two) [19, 20, 21]. Robotics has for some time been recognized as an area in which Intelligent Control can play a key role [14].

For example, in remote applications with fault tolerance, a robot could continue its assigned tasks without jeopardizing the mission or damaging the working environment. We have designed generalized robot fault tolerant algorithms which utilize the advantages of whatever structure exists [19, 20]. Our algorithms [21] detect and tolerate robot motor and internal sensor failures. Kinematically redundant robots or robots with multiple sensors per joint can use the algorithms to obtain more extensive failure detection and tolerance capabilities. These algorithms can, however, also provide basic detection capabilities for robots which have little or no redundancy with which to tolerate the failures. In order to modularize these fault detection and fault tolerance algorithms and enable the algorithms to more easily adapt to a wide variety of robot structures, this paper builds on a multilayer intelligent control framework [21] (see Figures 1 and 2) which extends the hybrid dynamical system framework proposed in [8].

Our new framework is divided into servo, interface, supervisor, critic, and user layers. The servo layer is the continuous robot system and its normal controller. The interface layer monitors the servo layer for sensor or motor failures using analytical redundancy-based fault detection tests. The discrete event supervisor performs fault tolerance at the manipulator level, and protects the robot from internal failures. The critic level analyzes the robot, the environment, and the user plan. Analysis of the user plan may be performed by consistency checking at the controller level [24], or by using methods of conformance testing on finite state machines, as discussed later in this paper. This level protects the robot from inappropriate user commands, and dynamically modifies the fault tolerance routines. The overall system, which is viewed as a new robot operating system, assures tolerance to robot, environmental, and operator failures by defining and enforcing appropriate protocols.

Our involvement with the space program [7] at NASA Johnson Space Center, and nuclear hazardous waste cleanup [2, 5] at Sandia National Labs, has demonstrated the need for reliable robots which can withstand system failures without requiring immediate repairs. The main results of this work have been to develop a multi-layered control framework which provides fault tolerance for a variety of robots and to introduce new sensor-based fault tolerance techniques for robotics using the concept of analytical redundancy. The emphasis is on a modular system that will be able to use the existing structure of a robot as the basis for fault detection and tolerance, without requiring specific types or numbers of joints, motors, or other components. We have implemented a proto-

type of the Intelligent Robot Kernel (Figure 1) at Rice University [18] and at Sandia National Labs.

2 Operating System Structure

In order to separate the various functions of a robotic system and to better monitor the interactions of the components, we use models from the design of computer operating systems. Figure 1 shows the high level interactions among the various layers of the intelligent control architecture. Operating system soft-

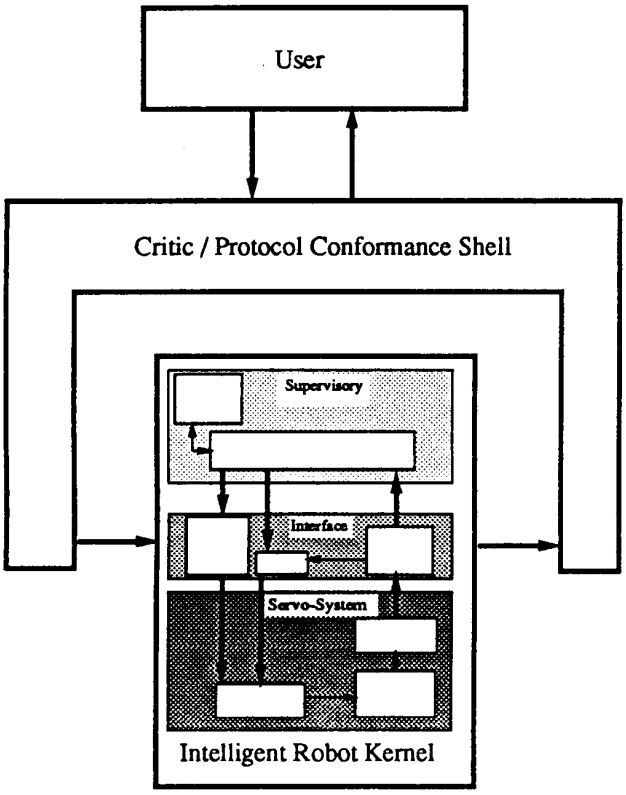

Figure 1: Fault Tolerant Robotic Operating System.

ware [12] allows for the scheduling and control of tasks desired by the user. The operating system presents a well-defined interface to the user and spares the user from having to know the particular details of the current configuration and underlying implementation. The user communicates with a "shell" which is wrapped around a "kernel". This analogy implies that the shell protects the kernel and analyzes and translates random user commands. In Figure 1, the user communicates commands to the critic/protocol conformance shell which validates the user commands and passes low-level information to the intelligent robot kernel shown in greater detail in Figure 2. As part of this standardization, the user and operating system agree on a language with a set of recognized commands. The commands are further defined in a specific protocol or standard. The designers of the system attempt to guarantee that the particular implementation will conform to the agreed upon protocol. Recently, there has been some work at Sandia National Labs to develop a robot-independent programming language [6].

We develop and formalize the protection capabilities of the shell for our example robot application. Computer operating systems provide various level of user privilege and enforce the concept of a "Super User" and a "Normal User." The shell understands that certain commands are potentially dangerous to the integrity of the system and are therefore privileged and can only be executed by the more knowledgeable "Super User." In our fault tolerant robot application, we develop a shell which contains a "critic." Following fault reconfiguration, this critic calculates the modified work space and checks for conformance to the original plan provided by the user. The critic also checks for obstacles and will halt the robot to protect it from possible damage. Only the super user can override the critic and force it to continue.

Many faults which arise in the system can potentially be handled by allowing for reconfiguration. We are developing an intelligent robot kernel which is described in Section 3 and is more general than the control structure developed in [2, 3]. This kernel is able to tolerate faults in sensors and joints. In a similar manner, operating systems provide a system of "signals" and "traps" for fault tolerance. For instance, in a generic computer system, a floating-point division by zero would corrupt a calculation and lead to an erroneous result. The IEEE Floating-Point arithmetic standard was developed to deal with such situations. The floating-point hardware signals or interrupts the operating system to indicate that an exception or error has occurred. The operating system receives the signal and executes a trap handler which normally aborts the user program and prints an error message. However, it is possible to create a trap handler that could deal more intelligently with the error by setting the result to a pre-defined value that the user code could interpret.

The fault tolerant kernel which is described below dynamically responds to errors in the robot system and can continue the task in many cases. Other previous research in surviving robot failures concentrates on providing fault tolerance by duplicating components [16, 25] or using kinematic redundancy [10, 13]. The problems in using these schemes are that too much complexity in redundant components may be added for system specifications, or existing robots, like the shuttle RMS, may not be kinematically redundant. Our algorithms provide levels of fault detection and fault tolerance to robots of arbitrary design.

3 Fault Tolerant Kernel

Our previous work has concentrated on robot fault tolerance at the manipulator level (controller and path planning (kinematics)). This represents the inner fault tolerant kernel of Figure 2, and provides tolerance of failures within the robot itself.

We have designed generalized robot fault tolerant algorithms which utilize the advantages of whatever structure exists [18, 19]. This interfaces with our related work in the design of high speed VLSI control architectures for robotics [4, 22, 23]. Our algorithms [19, 21] detect and tolerate robot motor and internal sensor failures. Kinematically redundant robots or robots with multiple sensors per joint can use the algorithms to obtain more extensive failure detection and tolerance capabilities. These algorithms do, however, also provide basic detection capabilities for robots which have little or no redundancy with which to tolerate the failures.

In order to modularize these fault detection and fault tolerance algorithms and enable the algorithms to more easily adapt to a wide variety of robot structures, we developed a multilayer intelligent control framework ([21], see Figure 2) similar to the hybrid dynamical system framework proposed in [8]. The system is initially divided into three layers: a continuous servo layer, an interface monitor layer, and a discrete supervisor layer. The servo layer consists of the normal robot controller and the robot. The interface layer contains the planner and provides fault detection and some basic fault tolerance for the system. The supervisor layer provides higher level fault tolerance and general action commands for the robot.

The interface layer monitors the internal robot sensors and sends only trusted data to the controller for feedback control. At the supervisor level, a target destination for the robot is determined based on the operator assigned task and is sent to the interface-level planner to develop the robot trajectory. The supervisor layer receives a response from the interface when the action has been completed or a failure has occurred within the robot. The supervisor always alerts the operator of failures and can provide information on possible causes of the failure based on a database of fault trees [11, 19] for the robot. However, only if the mission becomes impossible due to a specific failure does the supervisor require immediate intervention and repair from the operator. Failures which can be tolerated by the system are reported but are expected to be repaired at a later time.

4 Critic/Protocol Conformance Shell FSM's

The shell which is shown in Figure 1 is composed of two finite state machines, the User/Executive FSM and the Critic FSM. The User/Executive FSM handles the interaction with the user and initializes the subordinate FSM's. The Critic FSM is responsible for providing the protection and privilege services in the proposed intelligent control architecture.

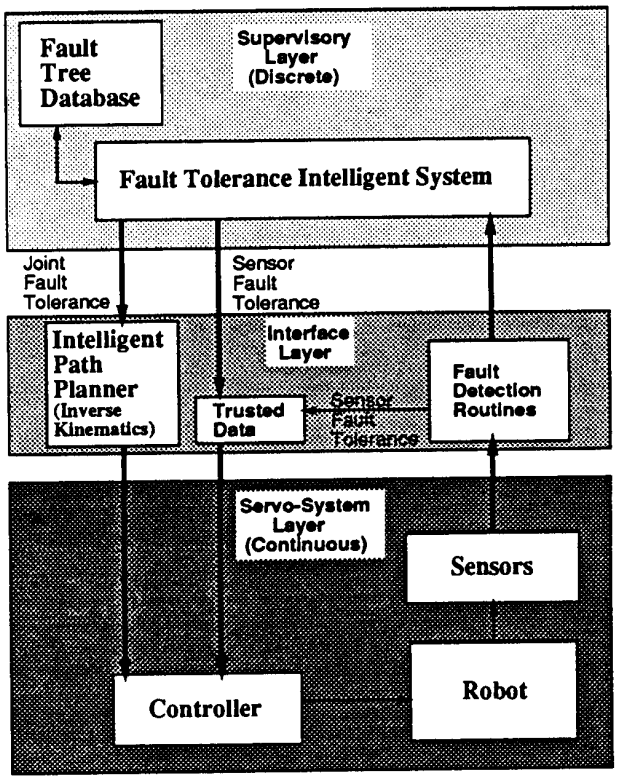

Figure 2: Fault Tolerant Intelligent Robot Kernel.

4.1 User/Executive Interaction FSM

The top level FSM which describes the user interaction with the robot is shown in Figure 3. The robot returns to the executive idle, EX_Idle, state to await input from the operator and the user plan is input into a command queue. We plan to recognize a command language, such as MOVE, APPROACH, OPEN_GRIPPER, CLOSE_GRIPPER, DEPART, MOVE_HOME, and STOP, as described in [6]. The executive main loop takes a command from the queue and when ready to run enables the critic FSM. If the critic FSM reports successful completion, then the next command is taken from the job queue until a STOP or done command appears in the queue. The executive then returns to the EX_Idle state.

When the executive starts the critic FSM with the EX_CR_Start signal, a number of actions occur to control the servo. These actions are outlined in Figure 3. The state transition diagrams in Figures 3 and 4 show in greater detail the hierarchy among these finite state machines. Only the most important start, done, and failure signals are shown in Figure 3.

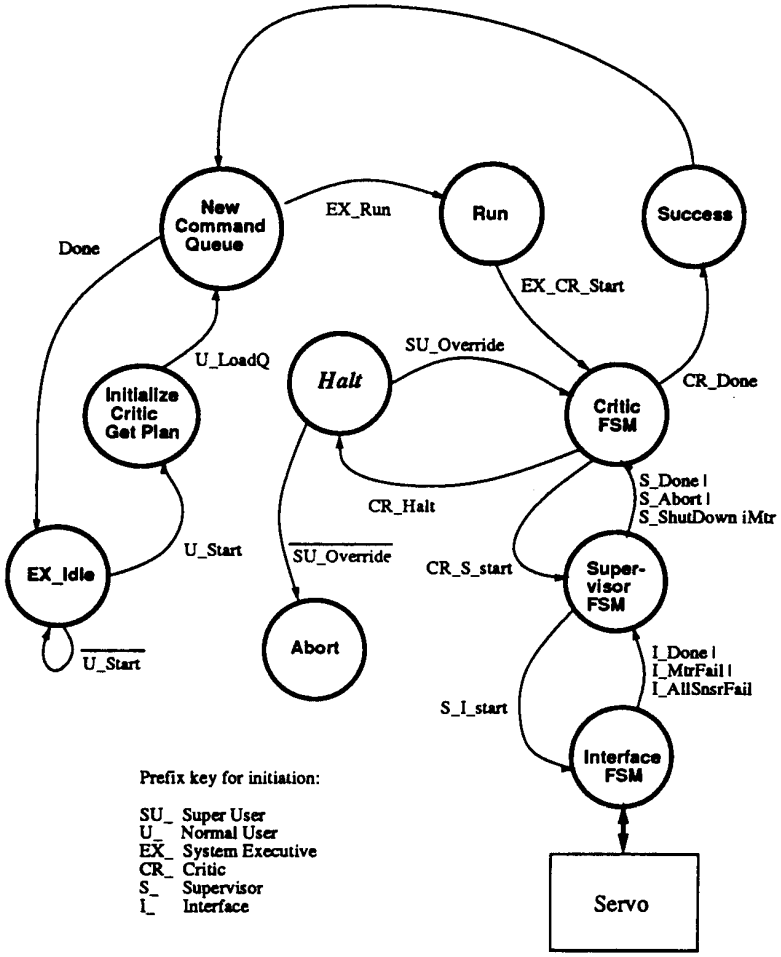

Figure 3: Top Level State Diagram for Full System showing User/Executive Interaction.

If a problem occurs in the robot which cannot be resolved through the lower level fault reconfiguration capabilities, the critic FSM will output a halt signal, CR_Halt. This protection mechanism will stop the robot and require user intervention. A "Super User" may override the critic warning and cause the robot to resume a possibly dangerous task. A "Normal User" may not override the critic and the task will abort. Additional levels of privilege could be added to such a system.

4.2 Critic FSM

The concept of a critic that will make judgments and recommendations to the user has been used in a number of automated systems. For example, systems exist for the analysis and criticism of text and VLSI circuit designs [15]. In Figure 4, a critic implemented as a finite state machine is described. The critic is initialized with the user plan by the executive FSM each time a new task is begun. The critic is responsible for the safety of the robot system and monitors the user's plan and the intelligent robot kernel.

After initialization, when the executive sends a start signal, EX_CR_Start, the critic moves from the idle state, CR_Idle, to initialize the supervisor to carry out the next command in the user plan. Obstacle checking [3] is then performed until the supervisor indicates that the command is complete through S_Done. If the obstacle checking routine reports a problem to the critic, then the critic issues the CR_Halt signal and moves to the ERROR-HALT state. The override or abort options are as described in the above section on the user interface.

A novel contribution of this paper is the integration of detection and threshold algorithms with dynamic fault reconfiguration of the robot sensors and/or joints. In response to the supervisor signals S_Abort or S_ShutDowniMtr, fault reconfiguration is

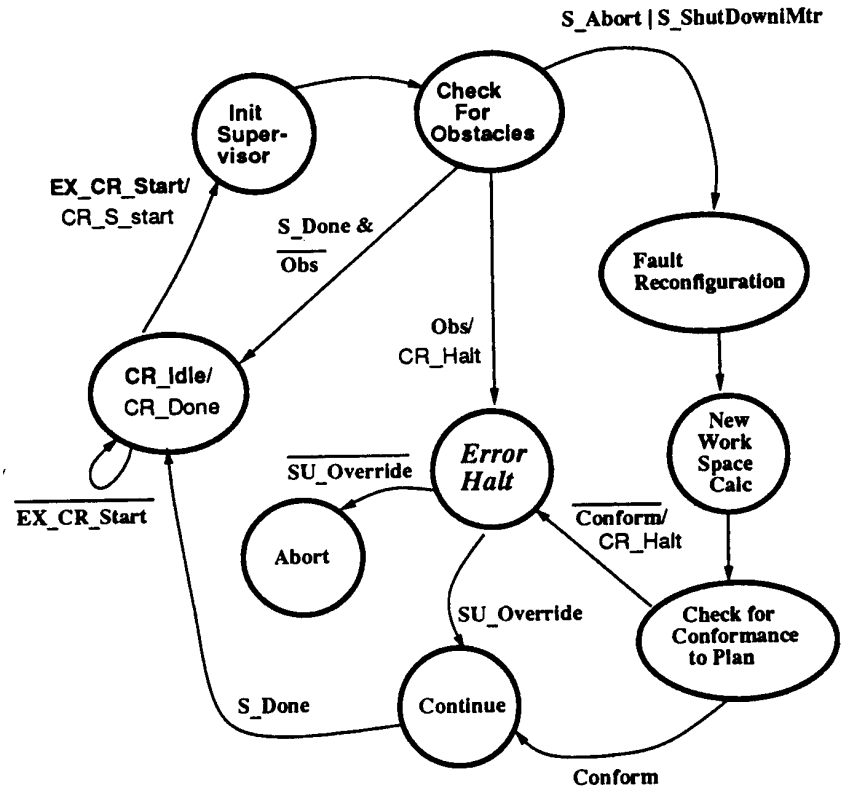

Figure 4: State Diagram for Critic.

carried out in the critic and the supervisor. The critic will monitor the reachable workspace of the robot throughout its operation [13]. While the supervisor waits for a signal in the PROCESS state, the critic can spawn a process to determine the area reachable by the end effector with the robot in its current state. When motor failures occur and a joint is shut down, the workspace shrinks due to the immobility of the failed joint. The critic computes the changes to the reachable workspace [13] and decides if the target position is still reachable by the robot. If the robot can no longer reach its assigned goal, the critic signals an abort and alerts the user. The critic then waits in the ERROR-HALT state for direction from the User or Super User.

The process of checking to see if the fault reconfigured robot can still complete its plan will be approached as a conformance testing problem as discussed in Section 6. The user plan was developed with a model of the robot which contained a specific number of sensors and joints. When faults cause joints to be lost, the reduced robot can be viewed as a subset of the original robot. Test sequences will be developed to determine if the reduced robot conforms to the original robot specification with respect to the user's original plan.

5 Intelligent Robot Kernel FSM's

The intelligent robot kernel shown in Figure 2 is composed of two further finite state machines, the Supervisor FSM and the Interface FSM. The Supervisor FSM receives commands from the state machines in the shell and communicates with the Interface FSM which deals with the continuous-time Servo-System.

Thus in total there are four FSM's, one for each layer. In this paper we have concentrated on the higher level layers and their interactions. The structure of the lower level FSM's are similar, and the full details are reported in [21].

6 Protocol Conformance and Testing

In Sections 4 and 5 the finite state machines for the intelligent control architecture have been described. One of the key problems in designing a hybrid system is in testing and verifying a particular software implementation. The interaction of the executive, critic, supervisor, and interface state machines can be viewed as a set of rules or protocol which define the system.

For the purpose of analysis, the state machines may be combined through their respective communication signals. The resulting finite state machine may then be viewed as a directed graph consisting of a vertex set and an edge set. A number of graph algorithms [9] may then be used to find particular paths through the graph.

Of particular importance in mapping the ideal finite state machines into practical software algorithms, is the determination of a set of test sequences to verify the conformance of the implementation to the model. Recently, an optimization technique for conformance testing using the Chinese postman graph tours has been reported [1]. This technique was developed for the testing of finite state machines used in communication protocols for telephone switching [17]. We plan to generate test sequences for our intelligent robot architecture using this technique. This test set is composed of test sequences such that the total set checks every edge of the graph or transition in the state machine. By running this test set it is possible to verify that the particular implementation conforms to the state machine model.

Other areas of application include the testing of our fault tree / fault reconfiguration algorithms and in the work space to plan conformance check. In particular, it will be important to generate time optimal test sets to verify the mapping of adaptive threshold and fault detection algorithms for different robots with a variety of sensor and motor configurations.

7 Conclusions and Future Work

In this paper we have described the initial stages of a five level Reconfigurable Intelligent Robot Control Architecture. The architecture invokes fault tolerance at both the manipulator and planning levels. The key structures are an interface level and a critic level. The interface level performs fault isolation and recovery (if possible) for manipulator and sensor failures, notifying the operator and supervisor levels as desired. We have recently implemented a prototype of the Intelligent Robot Kernel at Rice University [18] and at Sandia National Labs to support the goals of the environmental restoration and waste management robotics technology development program.

The critic level performs higher level consistency checking of the plan given the current robot state. The overall system is analogous to an operating system, providing a consistent interface to the user, and protecting the robot system from inappropriate tasks. Users can be assigned different levels of privileges, allowing certain users to override the critic in some cases. The architecture can be used to implement a number of interesting fault detection and tolerance schemes currently being investigated in the literature.

In future work, the operating system concepts for the overall architecture will be formalized. This will involve translations of user plans into the appropriate Finite State Machine. Initially, we intend to use the robot independent language developed at Sandia National Laboratories as our model user language. We will formally define the robot protocol (to be consistent with the both user and critic) which the user plan must conform to.

The critic layer of our framework will be developed and verified. We will use the formal techniques of conformance testing to formally check the protocols (critiquing the plan). Effective verification will require contributions from both the logical (conformance testing of finite state machines) world and the physical (fault detection using robot and sensor dynamics) world. This represents a new application of proven techniques to a hybrid (logical and physical) application.

Acknowledgments

This work was supported in part by DOE Sandia National Laboratory Contract #18-4379A, and in part by the National Science Foundation under grants MIP-8909498, MSS-9024391, and DDM-9202639.

References

[1] A. V. Aho, A. T. Dahbura, D. Lee, and M. Ü. Uyar. An Optimization Technique for Protocol Conformance Test Generation Base on UIO Sequences and Rural Chinese Postman Tours. *IEEE Transactions on Communications*, 39(11):1604–1615, November 1991.

[2] P. T. Boissiere and R. W. Harrigan. Telerobotic Operation of Conventional Robot Manipulators. *IEEE International Conference on Robotics and Automation*, pages 576–583, April 1988.

[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] A. S. Deo, J. R. Cavallaro, and I. D. Walker. New Real-Time Robot Motion Algorithms using Parallel VLSI Architectures. In *Proc. Fifth SIAM Conference on Parallel Processing for Scientific Computing*, pages 369–375, Houston, TX, March 1991.

[5] Department of Energy, Washington, DC. *Environmental Restoration and Waste Management Robotics Technology Development Program Robotics 5-Year Plan.* DOE/CE-0007T, Vol. 1-3.

[6] P. J. Eicker, D. J. Miller, and D. R. Strip. Intelligent Systems and Technologies for Manufacturing. *AT&T Technical Journal*, pages 10–22, November/December 1991.

[7] W. Fisher and C. Price. *Space Station Freedom External Maintenance Task Team Final Report.* NASA Johnson Space Center, 1990.

[8] A. Göllü and P. Varaiya. Hybrid dynamical systems. In *IEEE Conference on Decision and Control*, pages 2708–2712, 1989.

[9] A. Gibbons. *Algorithmic Graph Theory.* Cambridge University Press, Cambridge, UK, 1985.

[10] A. A. Maciejewski. Fault Tolerant Properties of Kinematically Redundant Manipulators. In *1990 IEEE Conference on Robotics and Automation*, pages 638–642, Cincinnati, OH, May 1990.

[11] A. L. Martensen and R. W. Butler. The Fault-Tree Compiler. NASA TECHNICAL MEMORANDUM 89098, NASA Langley Research Center, January 1987.

[12] J. Peterson and A. Silberschatz. *Operating System Concepts.* Addison-Wesley, Reading, MA, 1983.

[13] 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, Jan 1988.

[14] G. N. Saridis. Intelligent Robotic Control. *IEEE Transactions on Automatic Control*, AC-28(5):547–557, May 1983.

[15] B. G. Silverman. Survey of Expert Critiquing Systems: Practical and Theoretical Frontiers. *Communications of the ACM*, 35(4):106–127, April 1992.

[16] D. Tesar, D. Sreevijayan, and C. Price. Four-Level Fault Tolerance in Manipulator Design for Space Operations. In *Proc. of the First International Symposium on Measurement and Control in Robotics*, volume 3, page J3.2.1, Houston, TX, June 1990.

[17] M. Ü. Uyar, A. Lapone, and K. K. Sabnani. Algorithmic Verification of ISDN Network Layer Protocol. *AT&T Technical Journal*, pages 17–31, January/February 1990.

[18] M. L. Visinsky, J. R. Cavallaro, and I. D. Walker. Expert System Framework for Fault Detection and Fault Tolerance in Robotics. In *Proceedings of Fourth International Symposium on Robotics and Manufacturing*, pages 793–800, Sante Fe, NM, November 1992.

[19] M. L. Visinsky, I. D. Walker, and J. R. Cavallaro. Fault Detection and Fault Tolerance in Robotics. In *Proceedings of NASA Space Operations, Applications, Research Symposium*, pages 262–271, Houston, TX, July 1991.

[20] M. L. Visinsky, I. D. Walker, and J. R. Cavallaro. Chapter 8: Robotic Fault Tolerance: Algorithms and Architectures. In M. Jamshidi and P. Eicker, editors, *Robotics and Remote Systems in Hazardous Environments.* Prentice Hall, Englewood Cliffs, NJ, 1993. In Press.

[21] M.L. Visinsky, J.R. Cavallaro, and I.D. Walker. A Dynamic Fault Tolerance Framework for Remote Robots. Technical Report 9211, Department of Electrical and Computer Engineering, Rice University, Houston, TX, August 1992.

[22] I. D. Walker and J. R. Cavallaro. Parallel VLSI Architectures for Real-Time Control of Redundant Robots. In *Proc. Fourth ANS Topical Meeting on Robotics and Remote Systems*, pages 299–309, Albuquerque, NM, February 1991.

[23] I. D. Walker and J. R. Cavallaro. Parallel VLSI Architectures for Real-Time Kinematics of Redundant Robots. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 1993. Accepted for Special Issue on Computational Aspects of Robot Kinematics.

[24] 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.

[25] E. Wu, M. Diftler, J. Hwang, and J. Chladek. A Fault Tolerant Joint Drive Systems for the Space Shuttle Remote Manipulator System. In *1991 IEEE International Conference on Robotics and Automation*, pages 2504–2509, Sacremento, CA, April 1991.