
ROBOTICS AND REMOTE SYSTEMS FOR HAZARDOUS ENVIRONMENTS

EDITORS

Mo Jamshidi

Patrick J. Eicker



PTR Prentice HALL, ENGLEWOOD CLIFFS, NEW JERSEY 07632

Robotics and Remote Systems in Hazardous Environments,
(M. Jamshidi, P. J. Eicker, Eds.), Prentice Hall,
Englewood Cliffs, NJ, pp. 53-73, 1993.

3

ROBOTIC FAULT TOLERANCE: ALGORITHMS AND ARCHITECTURES

**Monica L. Visinsky, Ian D. Walker,
and Joseph R. Cavallaro**

Rice University

Fault tolerance is an essential factor in ensuring successful autonomous systems, especially for robots working in remote or hazardous environments. To avoid the cost and risk involved in sending humans into these environs and improve the chances of mission success, robots must be able to quickly detect and recover from internal failures without relying on immediate repairs or human intervention. In developing robotic fault tolerance algorithms, the possible failures within the robot and the interdependence of these failures must first be determined. One useful tool for performing these tasks is Fault Tree Analysis. The tree structures developed by this technique provide a flow chart of possible robot fault events and define the cause and effect relationships between the failures.

For robots, there exist a multitude of possible failure events ranging from stripped threads and loose chains to power failures and broken links. The robot itself may, however, be unable to differentiate between two different failures because it can only see the effect







of the failures. For example, if a motor is not driving a joint as it has been commanded to do by the controller, the controller would be unable to determine whether a chain in the gear-train has broken or an internal gear of the motor gear-box has worn down. All the robot can see is that the motor has stopped working. The fault trees must be arranged to reflect these functional capabilities of the robot to monitor its internal systems and respond to failures.

In other work, we have developed fault detection and fault tolerance algorithms for robots [16, 17]. Our current work [18] proposes an expert system framework for robotic fault tolerance which would use the fault trees described in this chapter. The integration of the trees within the framework is discussed in Section 3.8. Sections 3.1 through 3.3 give an overview of Fault Tree Analysis and some of the enhancements to this analysis technique. Computer programs which aid in fault tree construction are discussed in Section 3.4. Sections 3.5 and 3.6 present an example analysis of the fault trees for a specific robot, the Rice University Riceobot. Section 3.7 describes the transitional steps necessary to reduce the trees to reflect the functional fault detection capabilities (using the sensors) of the robot in preparation for designing the fault detection algorithms of [16] and [17].

3.1 ROBOTIC FAULT TREE ANALYSIS

Fault Tree Analysis (FTA) is a deductive method which identifies failure paths using a fault tree drawing or graphical representation of the flow of fault events [1, 2]. FTA is a well-known analysis technique often used in industry for computer control systems or industrial plants [6, 20]. Each tree event is a component failure, an external disturbance, or a system operation. Failure events are connected by logic symbols to create a logical tree of failures. The top event is the undesired event being analyzed and may be defined in different ways corresponding to structural or functional failure. In our analysis, the top event is the failure of the robot which is structurally considered a failure of any part of the robot (Section 3.5) and functionally considered a failure of the robot's ability to perform its specific function (Section 3.7). The basic symbols are explained in Table 3-1.

TABLE 3-1 FAULT TREE ANALYSIS SYMBOLS

| Symbol | Function |
|-------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
|  | All inputs required to produce output event. A fraction, x/y , in the gate means that more than x failures will produce the output event. |
|  | Any one input event causes the output event. |
|  | A malfunction which results from a combination of fault events through logic gates. |
|  | A fault event for which the causes are left undeveloped. |
|  | A basic fault event (includes component failures whose frequency and failure mode are known). |
|  | A suppressed tree. The tree is detailed in another figure. |

To develop the trees, the top event is broken down into primary events that can, through some logical combination, cause the top failure. This process is repeated to deeper levels until a basic event is reached. Many analyses, such as those in [14], have been performed to determine the criticality of individual failures. Their results can be used to enumerate the various failure events used in building the trees. Some conditions or causes may be left undeveloped if the probability that they will occur is small enough to be ignored.

3.2 FAILURE PROPAGATION/PROBABILITY ANALYSIS

The information available in the fault trees may be enhanced by a quantitative analysis of the failures. Failure rates are assigned to each input event and propagated up the tree based on the rules of the connecting logic gates. The output of an OR gate is the sum of the inputs. The resulting probability of the combined input events is greater than the probability of an individual input event. The output of an AND gate is the product of its inputs. The probability of all the events occurring is less than the probability of any one occurring. The AND gate represents a redundant measurement or capability and is more desirable in the tree since the probability of a failure decreases through the combination of lower level events.

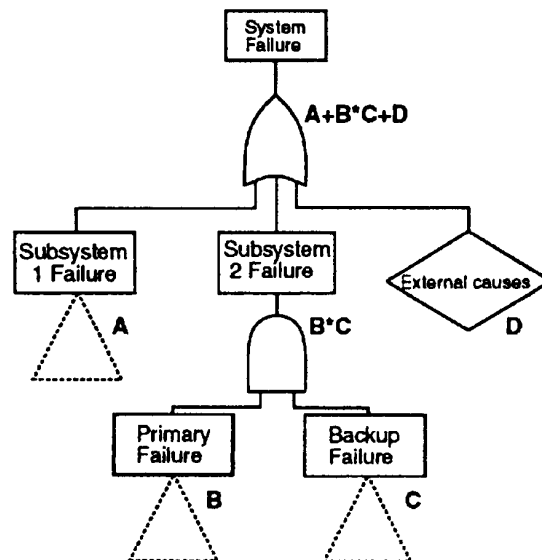


Figure 3-1: Fault Tree with Generic Failure Rates.

Figure 3-1 and Table 3-2 show an example of a fault tree with independent failure rates being combined up the tree. The rates are in failures per hour so that a failure rate of 1×10^{-4} would imply a possibility of one failure every 10,000 hours of operation. The

primary and backup systems of Subsystem 2 might be two functionally equivalent sensors in a single robot joint. When one sensor fails, a fault tolerance program can enable the other to take over the task of monitoring the joint. This functional redundancy increases the survivability of the system. A failure of the entire system occurs if either Subsystem 1 or 2 fails or if there is an external cause (such as a meteor damaging a robot link).

TABLE 3-2 FAILURE RATES APPLIED TO FAULT TREES.

| Node in Figure 8-1 Tree | Failure Rate | Fault Free Example |
|-------------------------------------------------|--------------|----------------------------------------|
| Layer 2: Primary Backup | B | $2 \times 10^{-4} \text{ h}^{-1}$ |
| | C | $2 \times 10^{-4} \text{ h}^{-1}$ |
| Layer 1: Subsystem 1 Subsystem 2 External | A | $1 \times 10^{-4} \text{ h}^{-1}$ |
| | B*C | $4 \times 10^{-8} \text{ h}^{-1}$ |
| | D | $2 \times 10^{-5} \text{ h}^{-1}$ |
| Layer 0: Top Level | A+B*C+D | $1.2004 \times 10^{-4} \text{ h}^{-1}$ |

Analyzing failure probabilities and their propagation through the fault trees allows the fault tolerance designer to focus on the components which are more likely to fail in the robot and which are thus more central to the monitoring software and fault detection routines. Unfortunately, few actual failure rates are currently available for robot components. A probabilistic analysis has been performed using failure rates based on engineering judgement and insight for various generic servomanipulator structures [7], but only general failure rates were used and each applied to a wide range of components.

However, looking at relative orders of magnitude for failures can still help the designer weed out the unlikely failure events. For example, there is only a small possibility of a link breaking during a typical robot task. The chains in the gear-train assemblies, however, often become loose and would affect the performance of the motors. The probability of a link failure is thus very small while the probability of a loose chain is relatively large. The fault tolerance designer can therefore eliminate link failures from the analysis and focus on detecting the effects of loose chains (*i.e.* inefficient or unresponsive motors).

A numerical analysis provides a measure of the overall chance of a complete failure for each robot. The structure provided by the fault trees organizes the probabilities appropriately for the robot system and provides a simple map of how the events relate to each other. Using the trees, robots of significantly different origin and structure can be compared for fault tolerant abilities and survivability. The integrated expert system proposed in Section 8 will use robot fault trees to estimate the run-time health and capabilities of the system. It can further be used for off-line comparisons of robots or for suggesting or executing possible on-line recovery actions in the presence of failures.

3.3 FAULT TREE PRUNING

A suggested drawback of FTA is that there is no way to ensure that all the causes of a failure have been evaluated [2]. The designer tends to identify the important or most obvious events that would cause a given failure. However, the events that are not modeled normally have a low probability of occurring and can be ignored or treated as a single basic event without overly biasing the analysis.

Several failures may be interconnected creating lateral branches or cycles in the fault tree. In some robots, one motion at a joint may be coupled with another motion such that failure of either motion causes a failure of the other. To keep the trees simple, the two failures are merged into one with twice the probability of occurring. It is also sometimes difficult to determine the relationship between failures. For example, the failure of all internal feedback sensors at the elbow joint of a robot would make the robot blind to the elbow's position. The elbow has not actually failed but the robot is unable to detect the results of commands sent to the elbow. Thus, the sensor malfunctions do not contribute to a physical elbow failure but may cause a functional failure of the elbow by blinding the robot controller to that joint. The most appropriate position for the sensor subtrees in a structural analysis is thus in the computer system tree as the sensors affect the communication link between the computer and the robot. In a functional analysis (as in Section 3.7), the sensor subtrees would be located beneath the joint failures.

3.4 COMPUTER AIDED ANALYSIS

Computer programs have been developed which aid the operator in creating and analyzing fault tree structures [3, 5, 7, 12, 13]. Fussell [5] presents a method for designing automated fault tree constructors to aid in building the trees and to eliminate the possible logic or omission errors made by a human designer. While focused on a fault tree constructor for electrical systems, the discussion in [5] provides a good example of how to develop computer aids for general systems. Fussell also explains how to define the system and component information so the computer can easily construct the correct trees.

The Reliability, Availability, and Maintainability (RAM) Software described in [7] helps robot designers efficiently optimize for RAM characteristics. This program can also help in the run-time evaluation of the effects of failures on the robot system. The operator can input a failure to the program and, based on the resulting calculation of the unreliability of the system, determine if the system is still operating at acceptable levels for the current mission requirements.

A Markov or semi-Markov model approach to probability analysis has also been developed as in the PAWS/STEM [3] and CARE III [13] reliability analysis packages. These packages are capable of analyzing simpler fault trees as well as Markov chains, but they are unwieldy and not necessarily optimized to handle simpler structures [11]. These analysis tools were also not designed to take advantage of some of the commonality within robot structures to simplify the analysis.

The NASA developed Failure Effect Analysis Tool (FEAT) [12] is typically used to analyze general systems such as the helium flow control system for the space shuttle. FEAT uses a digraph representation of the fault events within a system. Digraphs are directed graphs in which nodes are connected by arrows and are very similar to the fault trees explained above. The digraph nodes are connected in and/or combinations using petrinet notations instead of logic gates. FEAT develops a useful fault structure which can be easily converted to the trees created in Fault Tree Analysis (and *vice-versa*).

These computer programs determine what failures are likely to occur as the result of a given failure and can help in isolating the possible causes of the failure. They indicate the loss of functionality or redundancy in a system and thus aid in determining the fault tolerant capabilities of the system. The programs do not, however, specifically reveal the fault detection capabilities and limitations of the system. The system is only capable of detecting failures of elements which have associated sensors in the physical system (see Section 3.7).

This means that in tracking the causes of a given failure, the analysis programs cannot specifically isolate the cause unless every element in the system has a monitoring sensor and the program is capable of interpreting the sensor reading for an error.

Fault analysis programs can, at best, isolate a failure to a set of system components. The programs are useful, however, in directing repair of the system to the appropriate area thus decreasing the down time of the system. Fault analysis routines can be enhanced to provide information on available redundancy and system connectivity useful in fault tolerance. The automated tree constructors discussed above would aid a robot system operator in creating a database of information on the system which can then be used by an intelligent fault tolerance program for the robot (see Section 3.8).

3.5 ROBOT FAULT TREE ANALYSIS EXAMPLE

As an example of the analysis of general robots, this section presents an overview of the analysis of an arm of the Rice University Riceobot. The arm has eight degrees of freedom: three motions in the shoulder (z translation, pitch, and yaw), two motions in the elbow (roll and pitch), and three motions in the wrist (roll, pitch, and yaw). The results obtained from the Riceobot apply to most general robots especially since the Riceobot has a wide variety of commonly used link, joint, and motor arrangements. A similar analysis has been performed for generic servomanipulator structures composed of three subsystems: a master/slave arm, a hoist, and a viewing system [7]. Extensive fault trees were developed for various possible combinations of these subsystems and various top-event failures.

One drawback of focusing on the Riceobot is that the robot has only one optical encoder per joint. Its feedback sensor capability is limited and a failure of an encoder completely blinds the robot to that joint. However, as additional sensors would increase the survivability of each joint by increasing the fault detection and tolerance capabilities, analyzing the Riceobot can be considered a worst-case scenario where sensors are concerned. This analysis thus provides insight into base case algorithms for fault detection by revealing the important focal points for failures in the robot.

3.5.1 Overall Robot Failure

Several fault trees have been developed from an inspection of the robot structure and are reproduced in the following pages. The top event is the failure of the entire robot (Figure 3-2). In this section, we concentrate on the physical interconnections of the robot and define a robot failure as the failure of any component within the robot. This approach allows us to analyze the whole robot structure, evaluating how failures interact and enumerating the possible causes of various failures within the system. The primary causes of a robot failure are power failure, computer system failure, or a failure in any of the joints.

3.5.2 Joint and End Effector Failures

The Riceobot has two directly driven motions: the shoulder z-direction motion (Figure 3-3) and the wrist roll motion (Figure 3-5). The fault trees for these motions are quite simple since only the failure of the motor plays an important role in the failure of the motion. External causes of failures such as collisions or overly heavy loads are infrequent in a controlled environment or in the vacuum and zero gravity of space. The torques and inertias must be watched closely, however, so as not to put too much stress on the joints.

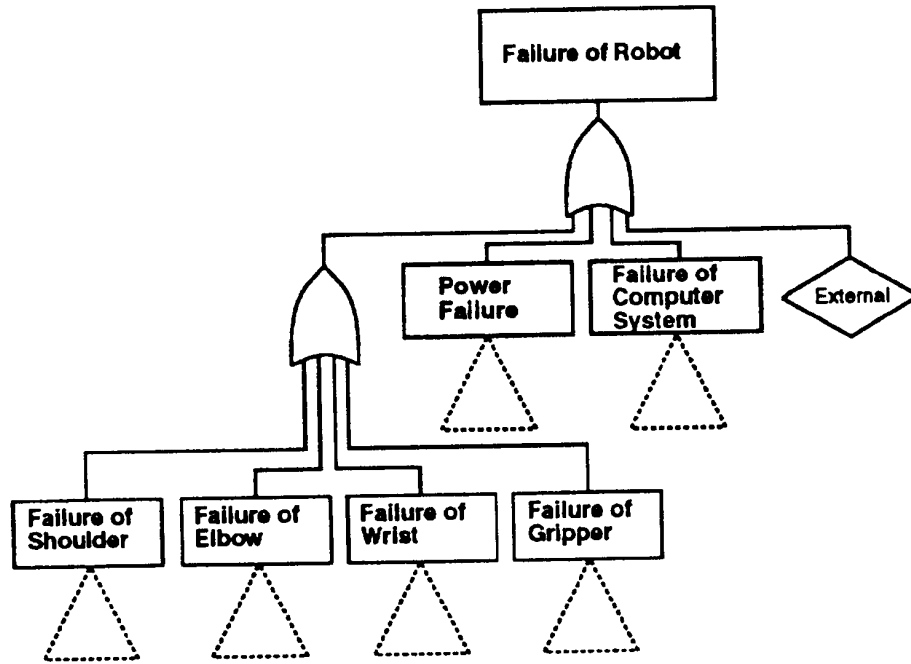


Figure 3-2: Riceobot Fault Tree.

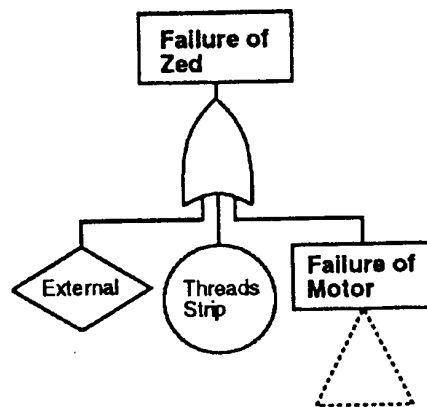


Figure 3-3: Shoulder Z-axis Motion Fault Tree.

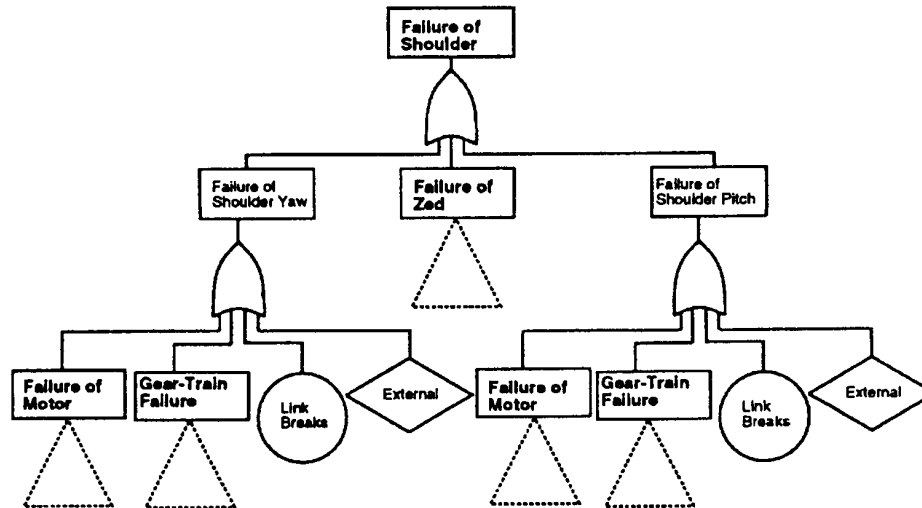


Figure 3-4: Shoulder Joint Fault Tree.

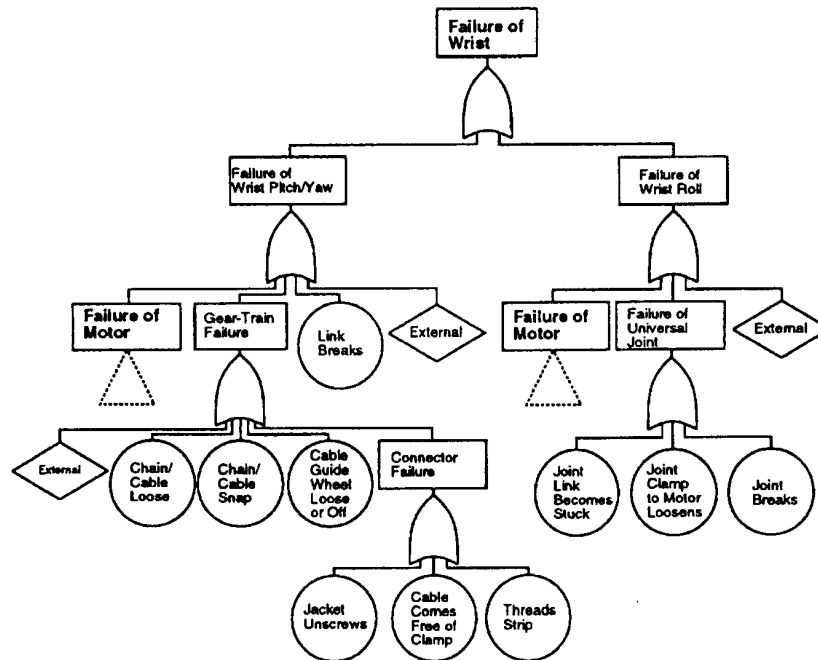


Figure 3-5: Spherical Wrist Fault Tree.

The other motions of the Riceobot depend on some form of gear-train assembly to allow the spatially separated motor to drive the joint (Figures 3-4, 3-5, and 3-6). Failure of the gear-train can be caused by events as simple as a loosening of the chain or cable and, in practice, has had a high probability of occurring. (A special ellipse shaped guide was used to take up the slack in the chain.) Most robot chains are similar to bicycle chains and are unlikely to snap, but the chain/cable assembly used for the Riceobot spherical wrist uses very delicate chains. The likelihood of these chains snapping could be quite high depending on how strongly the motor is driving the joint.

While the gear-train subtrees are presented as immediate causes of joint failures within the definition of Fault Tree Analysis, the effects of a failed gear-train are actually detectable only through the inability of the motor to perform its assigned task. A faulty gear-train does not cause the motor to fail, it simply affects the motion of the joint by directly affecting the output drive capability of the motor. In fault detection algorithms, a geartrain failure would be undetectable as a separate event and would instead be a possible cause of a detected motor failure (see Section 3.7).

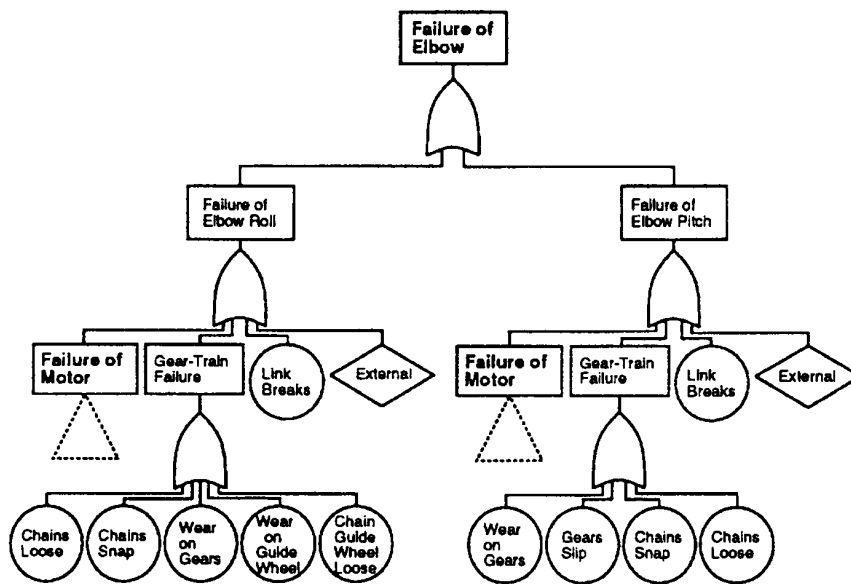


Figure 3-6: Elbow Joint Fault Tree.

The robot has a gripper type end effector which is driven by a motor through an arrangement of mechanical levers (Figure 3-7). The levers transform the rotary motion of the motor into translational motion to open and close the gripper. While it is unlikely the levers will break, the probability of becoming stuck is relatively high. If the levers fail in this manner, the gripper fails and undo stress is put on the motor trying to move the grippers.

If the limit switch for the gripper breaks, there is typically a hard stop which will prevent the gripper from opening too far. The robot will not be able to detect the problem immediately except by noting that the end effector motor is straining but not producing any motion.

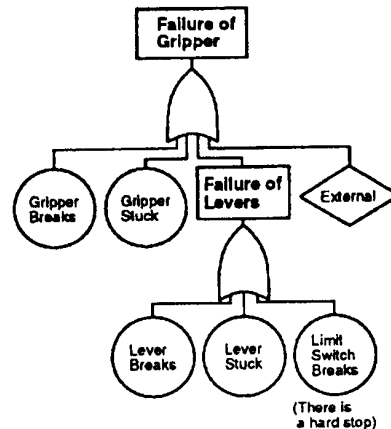


Figure 3-7 : Gripper Fault Tree.

3.5.3 Motor and Sensor Failures

Detecting motor failures has been an important focus so far in our analysis. The probability of a motor failure is dependent on the type of motor used (Figure 3-8). The Riceobot contains both brushless DC and stepper motors. Each motor has a gear box which may fail due to gear slippage or wear. A power failure (Figure 3-9) affects all motors as well as any other electrically driven part in the robot, but each motor could lose power separately if its specific power cables break. Some motors have internal sensors which detect motor failures by noting when the drive current surges or when a limit in the range of motion has been reached. Motor failures may also be detected based on their effect on the joint sensors of the robot. A motor failure could, however, conceivably be the cause of a sensor failure (Figure 3-10) because many sensors are mounted on the shafts of the motors as is the case for the Riceobot. If the motor breaks, it may do so in such a way as to damage or arrest the sensor output.

Sensors are also affected by incorrect calibration, external noise or vibrations although some of the errors due to these causes are absorbed by the feedback control algorithms. Each sensor could lose its power lines or have a faulty connection to the amplifier cards. Many sensor failures result in a frozen failure mode with the sensor producing a constant value. Other possible failure modes produce free-spinning, biased, or erratic sensors [14]. Sensors are critical to fault detection as they are generally the only window into the robot world for the controller. Errors in the sensors thus need to be detected quickly in order for fault tolerance algorithms to reliably allow the robot to continue its tasks.

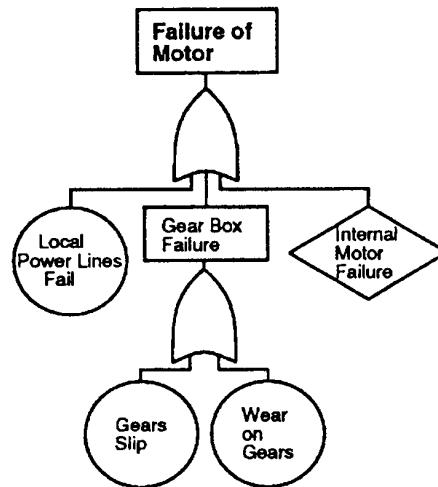


Figure 3-8: Motor Fault Tree.

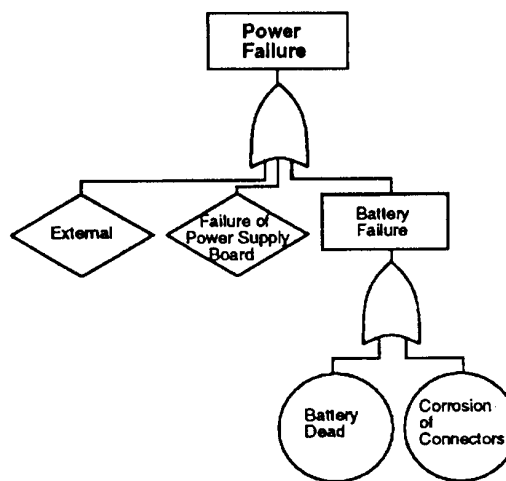


Figure 3-9: Power Supply Fault Tree.

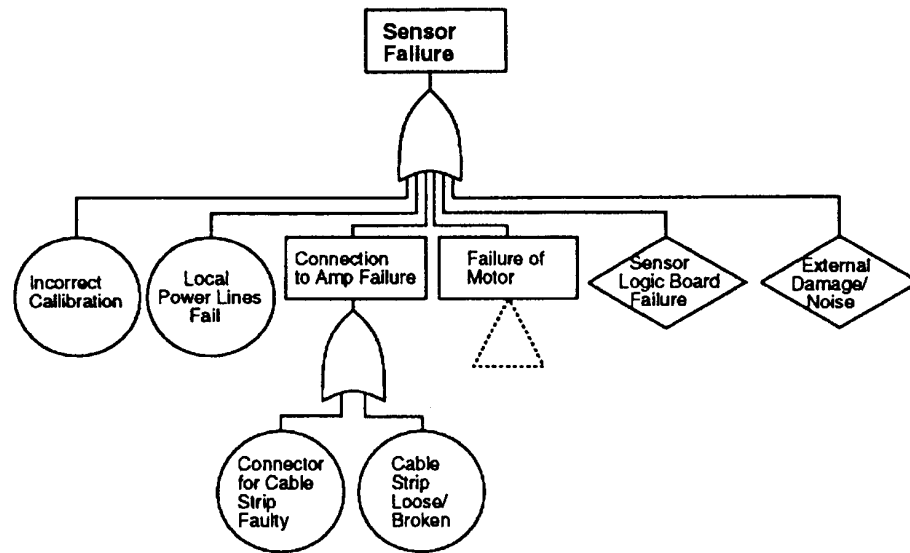


Figure 3-10: Internal Sensor Fault Tree.

3.5.4 Computer System Failures

The computer control system of the Riceobot consists of three main parts: (1) amplifiers which read from the optical encoders and drive the motors, (2) servo control chips which store information about the different motors and convert the desired angles into currents for each motor, and (3) an on-board host computer which is programmed in the C language and computes the desired angles for the desired motions (Figure 3-11). These three parts each contain at least one board filled with TTL chips, capacitors, power transistors, resistors, and other analog and digital circuit components. A failure within the host computer may corrupt the commands sent to the robot. By distributing the control algorithms amongst several processors [8], it becomes possible for the robot to withstand failures in the host computer by applying fault tolerance schemes to the set of processors. The robot cannot, however, currently withstand the failure of all the servo controllers or all the amplifiers because it would no longer be able to communicate with or, more importantly, monitor the joints. Failure of an entire board is thus generally terminal to the robot unless backup boards exist.

Detecting a non-terminal failure in the computer system requires some form of testing circuitry or the ability to poll components to see if they are still alive [9]. The IEEE standard 1149.1 Test Access Port may be incorporated into any VLSI chip on the boards and could be used for active testing. Correction code bits can be used to check data transfers and could identify a bus failure if the bits were consistently wrong. If no internal testing capabilities are provided, the robot could possibly use its vision system or some other external sensor

to check the status lights on the various boards. If something went wrong, the robot could even attempt to press the reset buttons on these boards to try to get them running again. A frequent problem with the Riceobot was that the servo controller boards often needed to be reset. The robot would not be able to move to reset the board if all the servo controllers were down, but if it could still move one arm, it could conceivably reset the boards.

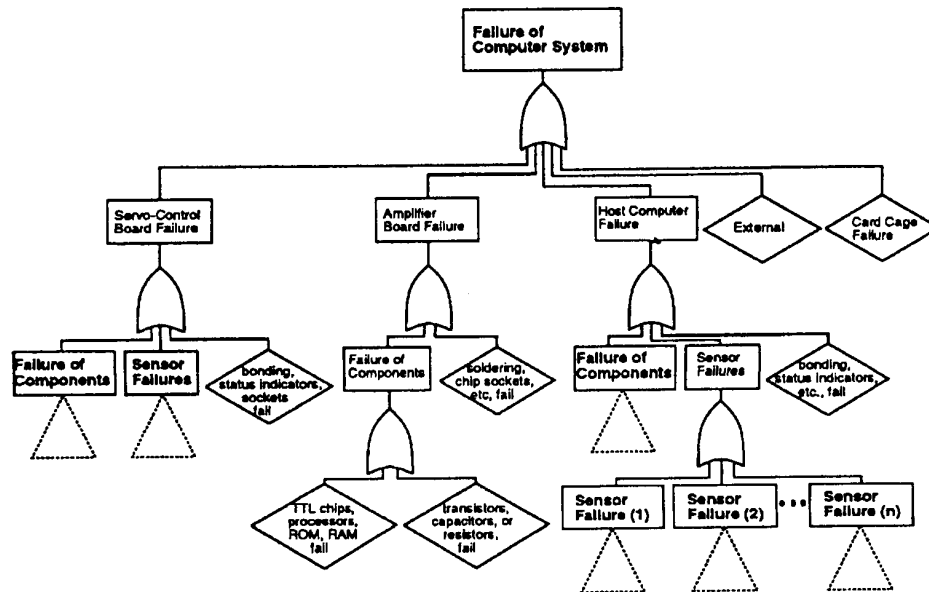


Figure 3-11: Computer System Fault Tree.

3.6 RICEOBOT FAULT DETECTION

In analyzing the fault trees, important information is learned about the fault detection capabilities of the Riceobot. If the power supply fails, the robot will fail completely unless it has a backup power source since it will no longer be able to control the motors. The robot will also be inoperative if the amplifier or servo control boards fail completely without a backup. Motor failures and other failures in the mechanical robot can only be detected by sensors. A human teleoperator can detect certain failures in the robot if they drastically affect the robot's performance [4]. However, to detect failures quickly without depending on human senses, the robot must glean information about its health from the internal and external sensors where internal sensors are part of the internal control feedback loop and external sensors are used to provide feedback about the environment.

With only one sensor at each joint, the Riceobot represents the worst case scenario for detecting sensor and joint failures. This situation provides a foundation for the fault detection algorithms developed in [16]. The only option available to the fault detection

software is to compare the sensed angles with the calculated desired angles. After accounting for a predetermined threshold to mask any precision errors in the calculations or sensing equipment, any difference between the sensed and desired values must be considered the result of a failure. The computer is, however, unable to differentiate between a sensor failure and an actual joint failure due, for example, to a frozen motor. Because the Riceobot is kinematically redundant, it is possible for the robot to reconfigure its model of itself and distribute the workload of the failed joint to the surviving joints. The Riceobot thus provides a base case for fault detection, but can still utilize the more advanced fault tolerance routines proposed in [16].

Fault detection for the Riceobot could be improved using an external sensor such as the vision system. With the computer calculation and the internal sensor reading, the additional joint angle information from the vision system would help distinguish between a sensor error and a real joint failure. The Riceobot could then function in the presence of one sensor failure. Using the vision system for this task, however, increases the load on the image processing software and may hinder the system's ability to perform its normal vision tasks.

3.7 ANALYZING FAULT TREES FOR FAULT TOLERANCE CAPABILITIES

The fully detailed, structural fault trees are useful in understanding and mapping the possible structural interactions of failures within the robot. To emphasize fault detection capabilities, however, the trees must be modified to focus on the functional interaction of failures. For example, the full analysis of the Riceobot has revealed that the health of motors and sensors are an important consideration in the survivability of robots. The fault detection designer must appropriately readjust the structural fault trees to reflect the loss of functionality when a sensor or motor is affected by some internal failure. The tree events thus should be grouped into events which can be detected by the robot through the sensors or intelligent fault detection algorithms.

Failure events may not structurally cause a failure in a specific component but might still limit the robot's ability to use that component (see also discussion in Section 3.3). A geartrain failure does not break the motor to which the chain is connected but does result in the robot being unable to use the motor. All the failure events which would affect the performance of a component should therefore be relocated beneath that component's failure event. For example, the "Gear-Train Failure" and "Failure of Universal Joint" subtrees in Figure 3-5 would each be combined through an OR gate with the existing subtrees of the appropriate motor failures to create new subtrees for the motor failure events. Because the new subtrees are combined through an OR gate, the probability of failure of the complete tree is not affected. All failure events with low probabilities such as links breaking or external failure events are removed from the analysis to simplify the original fault tolerance algorithm design. This does decrease the probability of failure of the tree slightly, but the change is small. Figure 3-12 illustrates these steps using the "Failure of Wrist Roll" subtree of Figure 3-5.

Robots which have redundant or backup components are better able to maintain their functionality in the presence of failures in these redundant parts. A robot with multiple sensors per joint, for example, can tolerate the failure of all but one of the sensors and still provide information about the joint to the controller. Fault tolerant algorithms enable the robot to reconfigure its internal model of its structure to reflect this shift in responsibility. If a robot is kinematically redundant, it can tolerate the failure of the redundant joints and still maintain its full range of motion although the workspace (reachable) area is reduced.

If the faulty motion or joint can be stabilized (*i.e.*, by locking the joint), other motions of the robot still provide some degree of motion to the robot. As the Riceobot has two redundant degrees of freedom, it can tolerate as many as two joint failures without losing its range of motion. The top level structural fault tree (Figure 3-2) is thus modified as in Figure 3-13 to reflect this functional redundancy. Changes such as these help focus the development of fault detection and reconfiguration strategies on the more critical system failures [16, 18].

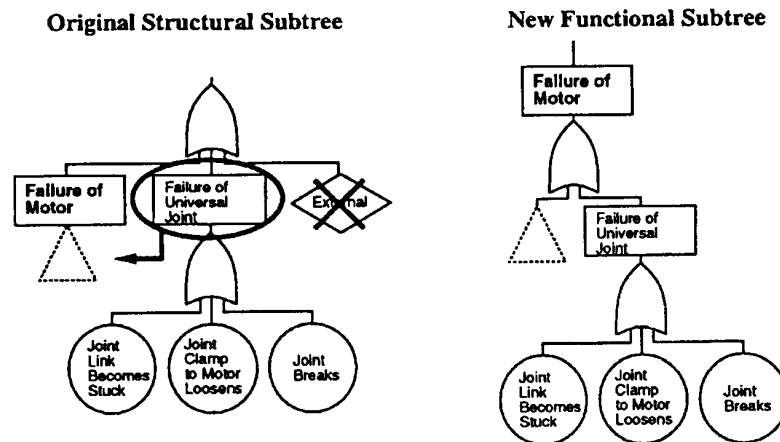


Figure 3-12: Restructuring Trees by Analysis of Fault Tolerant Ability.

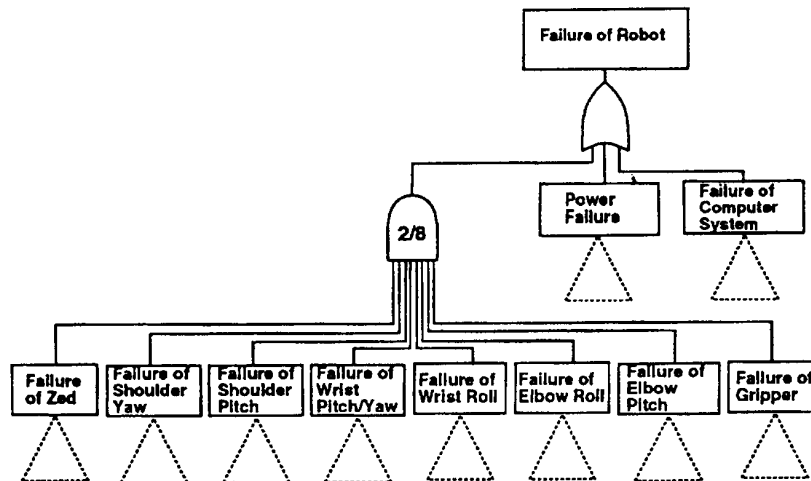


Figure 3-13: Functional Top-Level Fault Tree for Riceobot.

The full Fault Tree Analysis technique develops the trees needed in a fault tree database in order to allow an expert system to determine possible causes of detected failures. The database can then report the list of possible causes to an operator for future inspection and repair. However, the trees produced by fault tree analysis display the structural interconnection of failures and do not always reveal the functional affects of failures. By performing the above reduction analysis, the fault tolerance algorithm designer can determine the significant paths within the detailed structural fault trees for detecting and tolerating the more critical functional failures of the robot system. The pruned trees illustrate the structural support available for developing fault detection and fault tolerance algorithms.

3.8 INTEGRATION OF FAULT DETECTION AND FAULT TREES

The fault trees can be modularly combined with robotic fault detection and fault tolerance algorithms in a multilayer intelligent control framework [16, 18] which can more easily adapt to a wide variety of robot structures. The full fault tolerant control of a robot system can be 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 maintains a fault tree database and 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. The interface runs two continuous functions (sensor monitoring and inverse kinematics) and returns discrete event or failure information to the supervisor. The interface layer isolates the supervisor layer and the operator from the continuous control algorithms of the plant. The operator need only provide the final target for the robot and an initial map of which sensors monitor each joint. The operator receives a signal when the robot completes the task or when there is a failure. Operators are only required to provide direction when a failure situation makes the task impossible, although they may intervene at any time.

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 the database of fault trees 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.

3.8.1 Fault Detection Layer

The interface layer can contain several levels of fault detection and basic fault tolerance. In the example of Figure 3-14, Level A monitors the difference between various sensor readings for each joint and checks the changes in each sensor reading to determine which sensor has failed. Fault tolerance of a single sensor failure in a joint is then provided by automatically voting the faulty sensor out of the results. The detection routine, at this

point, sends only the data received from surviving sensors on to the other detection levels and the controller. Level B receives the information from Level A on which sensors have failed. It then keeps track of the inaccuracy of the working sensors compared to the desired values. If Level A has reported all but one sensor is faulty and Level B notices that the remaining sensor's error exceeds an acceptable threshold or if Level B notices that the errors of all sensors exceed the threshold, Level C is notified of a joint failure and the controller is now blind to that joint (see Figure 3-15). Level C performs the appropriate joint fault tolerance procedures within the planner based on Level B's report. Thus, Level A detects, isolates and recovers from single sensor failures; Level B detects and isolates multiple sensor or motor failures; and Level C recovers from motor failures.

As an example of the interaction of the layers in the framework, let us assume that, for some reason, the operator has directed the robot to perform a task using as controller feedback data only the values derived from the encoder reading at each joint. At some point in the run, the encoder for Joint 2 fails and the test monitoring this sensor is violated. The interface sends a signal which tells the supervisor that encoder 2 has failed and the interface knows of no more sensors to use. Figure 3-14 shows the supervisor subsequently removing the encoder for Joint 2 from a fault tree database. The database reveals that a tachometer is still available to functionally replace the encoder as they are connected by an AND gate and the failure does not move any farther up the tree. The supervisor can therefore command the interface to read the information from this alternate as long as the operator does not request an abort.

3.8.2 Fault Tree Database

The supervisor layer of the framework receives the failure reports from the fault detection algorithms and prunes the fault trees accordingly. The fault tree database then relays information back to the detection routines on which components are still available as replacements. Increasing the redundancy of the motors or sensors as in [15] and [19] would provide a wider range of survivor choices for the detection algorithms.

As discussed in previous sections, the full, structural fault trees would provide the expert system with a map of failure interactions. Given a failure, the expert system could traverse down the appropriate subtree to accumulate a list of components which could have caused the failure. The list would then be sent to an operator for future inspections and repairs. The trees would also direct the detection routine on which component or subsystem failures might occur given a specific failure. When pruning the trees of failed components, the database would search for interactions which might indicate that another component is likely to fail because it is linked either structurally or functionally to the failed component. More advanced detection routines could then be more alert to the possibility of failures in these components while monitoring the system for failures.

In Figure 3-15, a Motor 0 failure is detected by the detection routines in the interface layer. The interface sends a signal to alert the supervisor to the motor failure. The trees show that the failure would propagate through the OR gate all the way up to the top level AND gate. The database supervisor therefore removes the entire Joint 0 subtree. Fault tolerant options are still available, however, at the joint level as only one joint has failed so far and the tree indicates that this four link, planar robot can withstand as many as two joint failures. In signaling the interface layer to shut down the faulty joint, the supervisor initiates the joint fault tolerance scheme present in the planner routine [16].

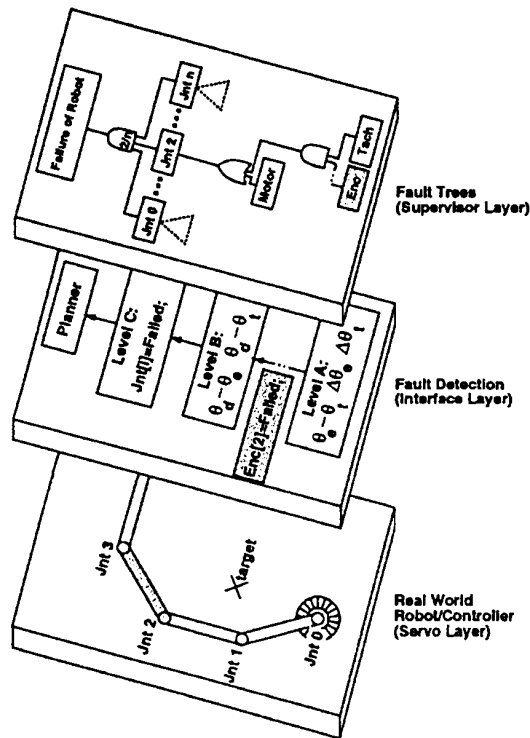


Figure 3-14: Propagation of a Sensor Failure Among Framework Layers.

Maciejewski's work on kinematic fault tolerance [10] can also be integrated into the fault tree database to provide the planner with information on new optimal configurations for the robot in the presence of failures. When the fault trees are pruned due to detected failures, a new optimal configuration can be drawn up to provide a more dynamic preventive maintenance measure. The optimal configurations will also keep the robot away from singularities where the fault detection routine picks up the inconsistent motion of the robot as failures. The fault trees can also recompute the failure probabilities of various subsystems based on the failure reports from the detection algorithm. The probabilities could be sent to the planner which would then reorganize the plan to try to avoid overloading parts with high probabilities of failing.

3.9 CONCLUSION

This chapter has presented an analysis technique for failures within a robotic system. We have discussed the basics of fault tree analysis and presented examples of trees derived

from a robot system. Fault tree analysis in conjunction with an analysis of the functional relationship of the failures in the system, provides the fault tolerance designer with information about the fault detection and recovery capabilities of the system. The trees are used statically to initially develop the fault detection and fault tolerance algorithms for a specific system. The fault trees can also be used dynamically with an intelligent fault tolerance controller to provide a database of run-time fault detection and recovery data which can reveal functional replacements for faulty components or indicate possible causes or effects of failures.

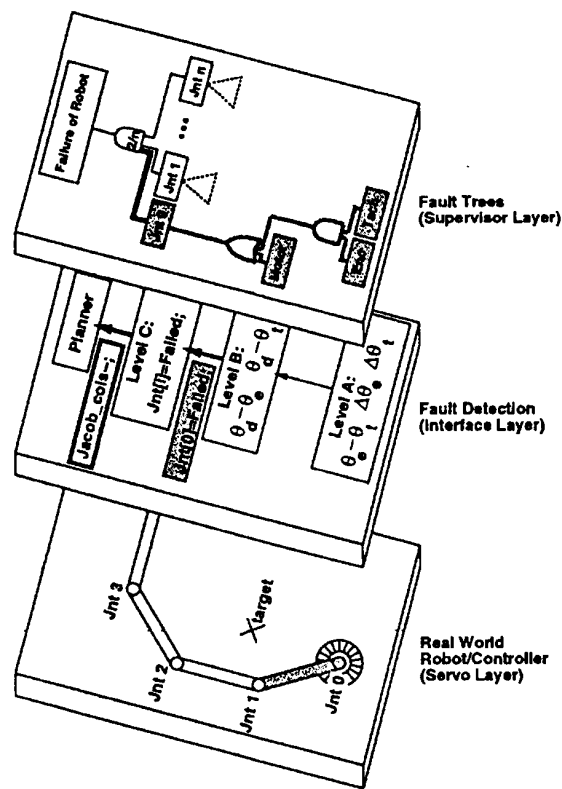


Figure 3-15: Propagation of a Joint Failure Among Framework Layers.

ACKNOWLEDGEMENTS

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

Further support was provided by a Mitre Corporation Graduate Fellowship and NSF Graduate Fellowship RCD-9154692.

REFERENCES

1. Barlow, R. E., and H. E. Lambert. "Introduction to Fault Tree Analysis." *Reliability and Fault Tree Analysis: Theoretical and Applied Aspects of System Reliability and Safety Assessment*. R. E. Barlow, J. B. Fussell, and N. D. Singpurwalla, editors, pp. 7-35. Society for Industrial and Applied Mathematics, 1975.
2. Bloch, H. P., and F. K. Geitner. *An Introduction to Machinery Reliability Assessment*. Van Nostrand Reinhold, 1990.
3. Butler, R. W., and P. H. Stevenson. "The PAWS and STEM Reliability Analysis Programs." *NASA Technical Memorandum 100572*. NASA Langley Research Center, March 1988.
4. Draper, J. V., S. Handel, and C. C. Hood. "The Impact of Partial Joint Failure on Teleoperator Task Performance." *Proceedings of the Fourth ANS Topical Meeting on Robotics and Remote Systems*. Albuquerque, NM, January 1991, pp. 433-439.
5. Fussell, J. B. "Computer Aided Fault Tree Construction for Electrical Systems." In *Reliability and Fault Tree Analysis: Theoretical and Applied Aspects of System Reliability and Safety Assessment*. R. E. Barlow, J. B. Fussell, and N. D. Singpurwalla, editors, Society for Industrial and Applied Mathematics, 1975, pp. 37-56.
6. Goldberg, J. A. "Survey of the Design and Analysis of Fault-Tolerant Computers." *Reliability and Fault Tree Analysis: Theoretical and Applied Aspects of System Reliability and Safety Assessment*. R. E. Barlow, J. B. Fussell, and N. D. Singpurwalla, editors, Society for Industrial and Applied Mathematics, 1975, pp. 687-731.
7. Guthrie, V. H., and D. K. Whittle. "RAM Analysis Software for Optimization of Servomanipulator Designs." *DOE Small Business Innovative Research (SBIR) Program Report JBFA-101-89*. JBF Associates, Inc., Knoxville, TN, March 1989. Performed for Oak Ridge National Laboratory.
8. Hamilton, D. L., J. K. Bennett, and I. D. Walker. "Parallel Fault Tolerant Robot Control." *1992 SPIE Conference on Cooperative Intelligent Robotics in Space*. Boston, MA, November 1992. Submitted.
9. Lien, J. C., and M. A. Breuer. "A Universal Test and Maintenance Controller for Modules and Boards." *IEEE Transactions on Industrial Electronics*. 36(2):231-240, May 1989.
10. Maciejewski, A. A. "Fault Tolerant Properties of Kinematically Redundant Manipulators." *1990 IEEE Conference on Robotics and Automation*. Cincinnati, OH, May 1990, pp. 638-642.

11. Martensen, A. L., and R. W. Butler. "The Fault-Tree Compiler." *NASA Technical Memorandum 89098*. NASA Langley Research Center, January 1987.
12. Stevenson, R. W., and R. W. McNenny. "Fault Isolation in Space Station Freedom Systems from the Space Station Control Center." *Joint Applications in Instrumentation Process and Computer Control (JAIPCC) 1992 Symposium: Technologies for New Exploration*. Clear Lake, TX, March 1992.
13. Stiffler, J. J., and L. A. Bryant. "CARE III Phase II Report—Mathematical Description." *NASA Contractor Report 3566*. NASA Langley Research Center, 1982.
14. Tangorra, F., A. D. Montgomery, and R. F. Grasmeyer. "Independent Orbiter Assessment Analysis of the Remote Manipulator System." *NASA Working Papers NAS 1.26:185585 and NAS 1.26:185592*. NASA STS Orbiter Projects Office, January-February 1987-88.
15. Tesar, D., C. Sreevijayan, and C. Price. "Four-Level Fault Tolerance in Manipulator Design for Space Operations." *First International Symposium on Measurement and Control in Robotics*. Volume 3, pg. J3.2.1, Houston, TX, June 1990.
16. Visinsky, M. L. "Fault Detection and Fault Tolerance Methods for Robotics." Master's thesis, Department of Electrical and Computer Engineering, Rice University, Houston, TX, December 1991.
17. Visinsky, M. L., I. D. Walker, and J. R. Cavallaro. "Fault Detection and Fault Tolerance in Robotics." *Proceedings of NASA Space Operations, Applications, Research Symposium*. Houston, TX, July 1991, pp. 262-271.
18. Visinsky, M. L., I. D. Walker, and J. R. Cavallaro. "A Dynamic Fault Tolerance Framework for Remote Robots." *IEEE Transactions on Robotics and Automation: Special Issue on Space Robotics*. 1993, submitted.
19. Wu, E., M. Diftler, J. Hwang, and J. Chladek. "A Fault Tolerant Joint Drive Systems for the Space Shuttle Remote Manipulator System." *1991 IEEE International Conference on Robotics and Automation*. Sacramento, CA, April 1991, pp. 2504-2509.
20. Young, J. Using the Fault Tree Analysis Technique. *Reliability and Fault Tree Analysis: Theoretical and Applied Aspects of System Reliability and Safety Assessment*. R. E. Barlow, J. B. Fussell, and N.D. Singpurwalla, editors, Society for Industrial and Applied Mathematics, 1975, pp. 827-848.