ELSEVIER

Reliability Engineering and System Safety **46** (1994) 139–158

0951-8320/94/$7.00

Robotic fault detection and fault tolerance: A survey

M. L. Visinsky, J. R. Cavallaro & I. D. Walker*

Department of Electrical and Computer Engineering, Rice University, Houston, Texas 77251-1892, USA

(Received 20 December 1993; accepted 25 March 1994)

Fault tolerance is increasingly important for robots, especially those in remote or hazardous environments. Robots need the ability to effectively detect and tolerate internal failures in order to continue performing their tasks without the need for immediate human intervention. Recently, there has been a surge of interest in robot fault tolerance, and the subject has been investigated from a number of points of view. Ongoing research performs off-line and on-line failure analyses of robotic systems, develops fault-tolerant control environments, and derives fault detection and error recovery techniques using hardware, kinematic, or functional redundancy. This paper presents a summary of the current, limited, state-of-the-art in fault-tolerant robotics and offers some future possibilities for the field.

1 INTRODUCTION

The idea of fault tolerance, or the ability to detect and cope with system failures, has only during the last decade begun to spread into the field of robotics. One reason for this relatively recent move is that robotic applications have been extended from the accessible environments of laboratories and factories into the arenas of space, medical, nuclear and other hazardous environments. The dangerous, long-distance nature of these environments often makes it difficult, if not impossible, to send humans to repair a faulty robot. Communication delays between the robot and operator may be large enough to allow potentially tolerable faults to blossom into system-wide malfunctions, forcing the mission to abort. Robot malfunctions also have the potential to initiate larger accidents, especially in certain critical nuclear or hazardous waste operations. Typically, human operators are integral in monitoring such systems for problems,[1] but they are not completely reliable in detecting failures.[2,3] With fault tolerance built into the robot system, the robot is able to deal with failures autonomously. The risk to human life is reduced, because the need for humans to perform repairs or replacements decreases. By improving their reliability and dependability, fault tolerance also reduces the cost of the robots. Fault tolerance is still useful for industrial robots in that it decreases down-time, both by tolerating failures so as to increase the robot's lifespan and by identifying faulty components or subsystems to speed up the repair process. Fault tolerance also helps to protect the product being manufactured as well as any humans in the robot's environment.

This paper surveys the current state-of-the-art and research trends in robotic fault detection and fault tolerance specifically focusing on rigid-link robots with closed-loop feedback control systems, although most of the techniques could be adapted to other types of robot systems as well. Section 2 gives a brief overview of robotic manipulators and the associated failure concerns. The next three sections divide the problem into general levels: failure analysis, fault tolerance and fault detection. Figure 1 shows a flowchart of how these different levels work together to provide fault tolerance for the underlying robot system. The robot is a continuous-time system moving along a trajectory based on the torques sent to each motor by the robot controller. The fault-detection module monitors the robot response to the torques through the internal sensors and returns to the controller-trusted data on the robot status. If the fault-detection routines detect a failure (i.e. the robot is not responding as expected), the faulty data is eliminated or replaced by following some instinctive tolerance action. The fault-tolerance module monitors and verifies the actions taken by the fault-detection module. The fault-tolerance module

* To whom correspondence should be addressed.

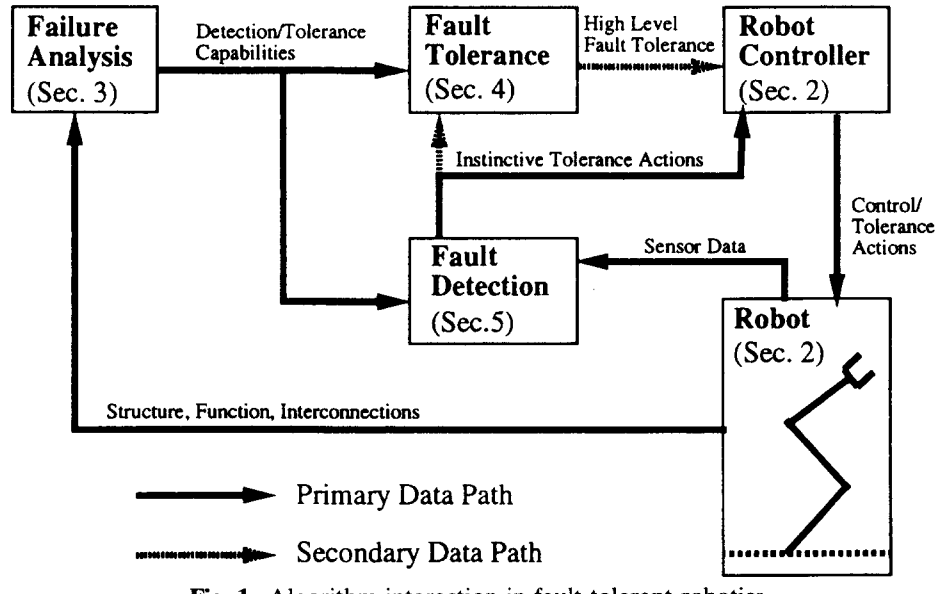

Fig. 1. Algorithm interaction in fault-tolerant robotics.

also draws from more long-term failure analyses to provide higher levels of fault tolerance to the controller.

Section 3 discusses on-line and off-line failure analysis techniques ranging from design-phase tools which improve the reliability of the system to operational-level programs that monitor the changing failure status of the robot and provide dynamic fault-tolerance options. Section 3 also presents a brief description of reliability standards and the need for more robot-specific data to improve the fault-tolerance analysis tools. Sections 4 and 5 discuss the current work in robotic fault tolerance and fault detection, respectively, and offer some options for future work. Section 5 further presents the derivation or robotic detection thresholds necessary to mask out the uncertainty inherent in robotic system models. Finally, we present our conclusions about the direction of robotic fault detection and tolerance research and the areas which need more attention.

2 GENERAL ROBOTIC SYSTEMS

2.1 Robot structure

In general, a robot manipulator is a mechanical structure consisting of links connected by joints. The joints may be either rotational or prismatic (see Fig. 2) and typically move around one axis (i.e. they have one degree of freedom).[4,5] Each joint is driven by an actuator, either directly or through a combination of gears, linkages or belts. The most common robot actuators are electric motors.[4]

Unlike the human arm and hand which together have over 30 degrees of freedom (DOFs), most robots working in three dimensions only have six DOFs, the minimum number of degrees necessary to position and orient the tool end of the robot (the end effector) anywhere in the three-dimensional workspace. Robots with more than this minimum number of joints can choose between several configurations of the joints to position the end effector at a specific workspace point. These robots are termed kinematically redundant. Kinematics[6] is the mathematical process relating the end effector position (x, y, z) and orientation (roll, pitch, yaw) to the joint configuration $(\theta_1, \ldots, \theta_n)$, where n is the number of joints. Figure 3 shows a kinematically redundant robot with its end

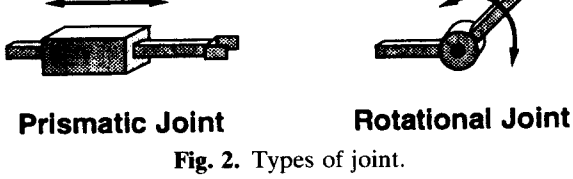

Fig. 2. Types of joint.

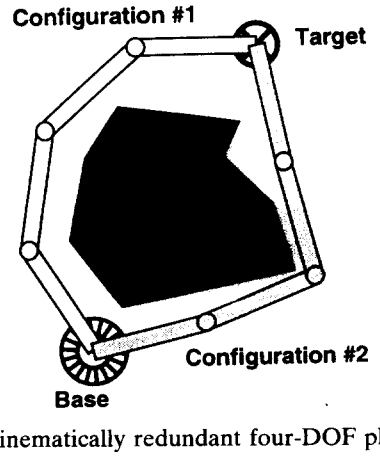

Fig. 3. Kinematically redundant four-DOF planar robot.

effector placed at a given point in the workspace using two different robot configurations. There are actually a multitude of configurations that would enable the robot to reach the desired point, although only two are shown. The robot shown is a four-DOF planar robot. Because it is constrained to move in the plane, the robot only needs two DOFs to freely position the end effector, therefore it has two extra DOFs. As indicated in the figure, kinematic redundancy can be used to maneuver the robot around obstacles. Section 4.2 discusses how kinematic redundancy can be further exploited to provide fault tolerance for the robot.

2.2 Robotic control

A computer-coordinated controller commands the robot to move through its workspace based on a plan or desired trajectory for the robot end effector. The plan is typically developed using an inverse kinematics algorithm which computes the next desired joint configuration given the current configuration and the next desired end effector position. The plan may be pre-computed off-line and stored in the controller's memory or computed dynamically during the assigned task. On-line computation allows more freedom in altering the trajectory to avoid obstacles or tolerate failures.

The robot dynamics are described by the following equation:

$$\underline{\tau} = [\hat{M}(\underline{\theta})]\ddot{\underline{\theta}} + \hat{\underline{N}}(\underline{\theta}, \dot{\underline{\theta}}), \tag{1}$$

where $\underline{\tau}$ is the joint torque vector, $[\hat{M}]$ is the inertia matrix, and $\hat{\underline{N}}$ represents coriolis and centripetal forces, gravity, friction, and other nonlinear effects. $[\hat{M}]$ and $\hat{\underline{N}}$ depend on the physical robot's parameters and are not known precisely by the controller. The robot controller uses estimates of the actual parameters along with sensed estimates of joint positions to calculate the estimated $[M]$ and $\underline{N}$. Section 5.2 discusses the problems that arise for fault detection due to the difference between the estimated parameters and the actual parameters. A computed-torque controller (proportional-derivative, or PD, control plus an outer feedback linearization loop) would then take the form:[6]

$$\underline{\tau} = [M(\underline{\theta})]\{\ddot{\underline{\theta}}_d + [K_P](\underline{\theta}_d - \underline{\theta}) + [K_D](\dot{\underline{\theta}}_d - \dot{\underline{\theta}})\} + \underline{N}(\underline{\theta}, \dot{\underline{\theta}}), \tag{2}$$

where $[K_P]$ and $[K_D]$ are system gain matrices with diagonal entries K_p and K_d corresponding to each joint. This type of controller uses position and velocity level feedback information to bias the controller so as to minimize transient errors or noise in the system. Larger system gains K_p and K_d reduce errors faster but may also overshoot the goal, leading to oscillations in the motion or actuator saturation. More detail can be found in a general robotics text.[6]

Using eqn (2), the controller computes the necessary torque to apply to each motor in order to move the robot from the given current position into the next desired position. Information about the current position or velocity of the robot joints is relayed to the controller from internal sensors (Fig. 4). The robot computer uses the sensors to determine the errors in joint positions, velocities, or forces so that the computer can compensate for these errors using feedback control as in eqn (2). Most robots, however, do not have an extensive number of internal sensors and may only have a position or velocity sensor at each joint. External sensors such as ultrasound systems, radar, laser rangefinders, and vision systems provide feedback about the surrounding robot environment and are used primarily for high-level planning purposes. More advanced robots have a force/torque sensor at the wrist joint of the end effector to monitor the contact forces when the end effector is manipulating an object.

2.3 Robotic failure

Figure 5 shows the simulated path of a PUMA robot after it experienced a simple frozen encoder failure in Joint 0 (providing the rotation about the base) at time 0·5. The desired path is also presented using a wire-frame model of the robot and a lighter trajectory line. The robot begins deviating from the desired path almost immediately after the failure and the failed motion is a looping trajectory that ends with the robot moving off to the left of the desired position. Figure 6 shows a simulation of an encoder failure in Joint 1 with the same desired trajectory. This simulation reveals that failures can cause the robot to quickly accelerate into wild motions that may cause considerable damage in real-life as the robot tries to

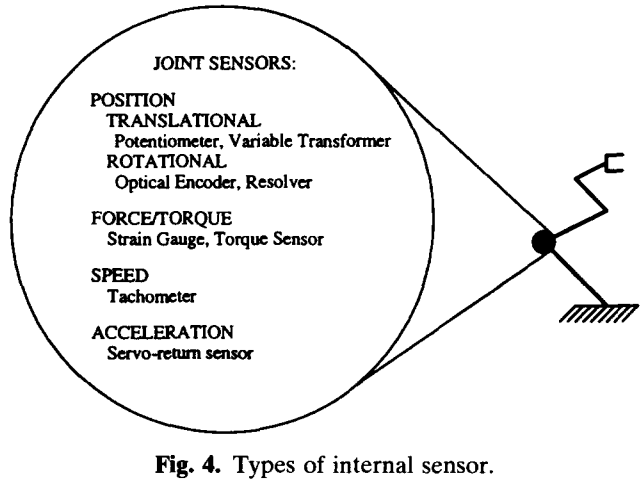

Fig. 4. Types of internal sensor.

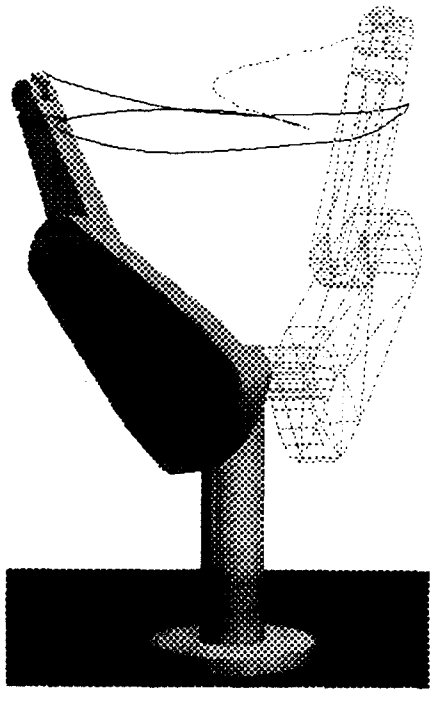

Fig. 5. PUMA with Joint 0 encoder failure (solid) vs desired position (wire-line).

pass through the floor or the robot itself. The well-being of the sensors is thus considered critical for the robot both for completing its task and in providing fault detection and tolerance capabilities.

All sensors typically have some form of noise or inaccuracy associated with their readings. This error is compounded by the inaccuracies between the chosen model parameters and the actual robot parameters. These errors (normal in fault-free operation) must be hidden by thresholds as discussed in Section 5.2 in order to focus the fault-detection routines on real failures instead of on artifacts caused solely by modeling inaccuracies. Further, if a sensor is damaged and the failure remains undetected, the controller will receive incorrect information about the joint and will be significantly in error. A frozen failure mode in which the sensor produces a constant value is fairly common. Other possible failure modes include free-spinning, biased, or erratic sensors.[7,8] If multiple sensors are used per robot joint, the sensors can be chosen in such a way as to be made resistant to many different failure modes which may be fatal to a single type of sensor in the joint.[8]

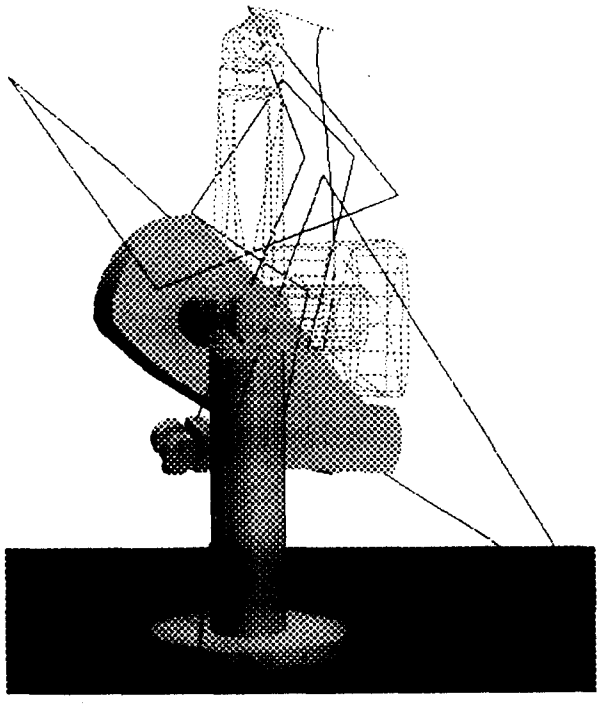

Fig. 6. PUMA with Joint 1 encoder failure (solid) vs desired position (wire-line).

Because the number of sensors is limited, robots are not able to isolate all the possible failures in the system. 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 the motor itself has failed or the gear-train connecting the motor to the joint has suffered a failure. All the algorithms can detect is that, according to the sensors, the desired motion of the joint is no longer being produced. Sensors that are strategically placed throughout the system maximize the robot's fault-tolerance capability by monitoring the components or modules whose failures would most affect the ability of the robot to continue its tasks.[9,10] The failures of subcomponents in these critical modules are assumed to precipitate a failure of the whole module. Tolerance actions are then geared towards alleviating the results of the larger failure of the whole critical module rather than the failures of the internal subcomponents. In essence, the module is treated like a black box for immediate tolerance concerns. The actual cause of the failure can be determined using post failure analysis tools at the leisure of the supervisor or operator (see Section 3).

Motors are also obviously critical to the robot task and the ability to lock a motor in the event of a failure is important in supporting fault-tolerance schemes. A failed joint which is allowed to swing freely pulls dynamically at the other joints and draws them off course. An important point here is that robot controllers have high update rates (on the order of a few milliseconds) and failures affecting the controller blossom into serious system-wide failures within a few of these updates. Thus, there needs to be quick or instinctive fault tolerance at the level of the internal failures. If a motor fails and locks in place, it will not affect the other joints through coupling although it will stop contributing to the completion of the task. If not already an automatic response to failures through springs or automatic brakes, locking may be effectively implemented as a command from the

controller or operator to initiate the joint brakes. Once a joint has been locked, the paths of surviving joints can then be modified to take over the workload of the failed joint and move the end effector to its goal.

Table 1 lists the critical robot components and some of the failure modes associated with them. The possible responses to the various failure modes depends on the goal of the system. If a fault detection and tolerance system is designed to provide fast fault tolerance (as might be necessary for a robot performing delicate or mission-critical tasks far from human intervention), the response to all the failure modes would be to eliminate the faulty component from the working set.[11] With sensors, any deviation from the expected reading (outside of modeling errors) would be considered a failure and the data would be derived from some other sensor. If all the sensors failed for a joint or the motor failed, the joint would be locked in place and its workload redistributed to other joints. If the fault detection and tolerance system has more time to analyze failures, the responses to the various failure modes could vary. For example, a biased sensor can still be part of the working set if the bias is removed from the sensor reading before the result is sent to the controller. If a dual actuator[12,13] is used in which two motors work together to produce the output for a single joint, a run-away failure in one motor can be counteracted by driving the other motor in reverse. This unequal load sharing is only possible, however, if the gears are designed to be self-locking or nonbackdrivable.[12]

When robots are easily accessible for maintenance and repair, such as in laboratory or industrial applications, a major concern is the safety of the robot as it interacts with humans and its environment. Failure in these arenas typically implies an externally evident problem in performing the desired task or avoiding obstacles. A significant amount of past robotic fault-tolerance research has resulted in improving the control programs and providing additional software, not so much to tolerate or find solutions to these external failures as to limit their effects on the robot.[14–19] The cause of such an external failure may lie within the robot structure but more often is the result of human error, either in initializing, programming, controlling or working around the robot.[15] Proper training obviously would be one means of decreasing these human errors. Other safety suggestions point to the proper placement of warning signals and barriers around the robot's work area, emergency stop buttons both on and off the robot, and limits on the robot's speed and range of motion to prevent any dangerous robot actions which might take a human operator by surprise.[20] Discussions on how to deal with external task errors that arise when a payload is inadvertently dropped or undergoes some changes due to improper task planning are also available.[18]

As robots become increasingly prevalent in more hazardous and inaccessible environments, fault-tolerant research has begun to focus on the internal failures of the robot—failures of the robot's mechanical or software systems.[13,21–23] While human and environmental safety is still a concern for these remote robot arenas, the focus shifts to enabling the robot to successfully and autonomously cope with internal failures so as to prolong its working life and maintain as much of its functionality as possible without reinstating the risk to human life or requiring immediate human intervention. To support these internal fault-tolerant capabilities, robotic methods of detecting and isolating failures must be developed and perfected. The detection algorithms must account for the uncertainty inherent in robot models and dynamics. Once detected, a failure must be tolerated in some manner through the use of redundancy or by graceful degradation of functionality. More intelligent

Table 1. Critical components and failure modes

Component	Failure modes	Possible solutions
Joint sensor	Frozen	Ignore sensor and find alternate data
	Biased	Remove bias from results
	Run-away	Ignore sensor and find alternate data
	Spike	Ignore spike value
Joint motor	Frozen (constant output)	Halt motor, redistribute work load
	Zero (on output)	Halt motor, redistribute work load
	Run-away	Counteract if have dual-motor
	Random (unknown/variant output)	Halt motor, redistribute work load
Power supply	Surge	Surge protector
	Zero	Back-up power supply
Control computer	Software error	
	invalid command	Ask operator for new command
	wrong value	Use alternate software modules
	Hardware error	Back-up processors or redistribute to parallel processors

programs can analyze the overall and long-term effects of the failure to improve fault tolerance and enable the robot to cope with transient or temporary failures. Dynamic analysis of the causes and effects of failures can assist the operator in coping with future problems, completing the current task and focusing future repairs. The following section reviews various off- and on-line analysis tools that have been applied to robotics.

3 FAILURE ANALYSIS TOOLS

In order to develop robotic fault-tolerance algorithms based on the existing structure of a robot, the possible failures must be itemized in some manner and their interdependence determined. There are many existing tools for analyzing engineering systems in relation to reliability and failures. Some of these tools provide a static pre-failure or post-failure analysis of the system to give insight into a system's reliability, fault detection and tolerance capabilities, and maintenance needs. Performing a Failure Mode, Effects, and Criticality Analysis (FMECA), for example, helps the designer focus on the critical failures and failure modes for the system in order to improve the design and make the system more reliable from the start.[24] Off-line analysis of systems after failures have occurred can further direct repair efforts by enumerating or isolating the possible cause(s) of a given failure and can help in preventive maintenance by pointing out future effects of the failure. These off-line analysis tools are also used in training human operators and preparing them for 'what if' situations[25] to prevent potential accidents.

On-line analysis tools run concurrently with the system to help speed up detection and tolerance of failures. On-line analyses check fault-tolerance actions for correctness or effectiveness. These on-line tools might include standard list databases of solutions (derived from FMECA-type tools discussed in Section 3.1) or dynamic structural databases (such as the fault trees in Section 3.2) to provide dynamic alternative tolerance solutions. In the following, we discuss the relatively few applications of both off-line and on-line analysis tools in robotic fault detection and fault tolerance and offer some possible extensions for future work.

3.1 FMECA

Failure Mode, Effects and Criticality Analysis (FMECA) techniques address reliability issues in the design phase. There are several defined steps in the technique, but the result is that the designer enumerates the possible failures and their failure modes and then tabulates the effects of each failure and mode combination on the system into standardized FMECA charts.[26] Qualitatively, the FMECA tables include causes of failures, system effects, safeguards, and recommended design changes and operating actions to reduce the effect of each failure mode for each component. The data can then be manipulated to quantify the frequency of failure modes, equipment damage, cost of business interruption and cost of repair. The important failure modes which significantly affect the reliability of the system are identified using the qualitative and quantitative evaluations. The entire FMECA process can be reiterated once the recommended changes have been made for the critical reliability problems in order to test the results and possibly enumerate the next batch of reliability issues. Because of time and financial constraints on the design, it is usual to consider only the effects of primary failure modes on system performance.[27]

The FMECA technique has been applied to a four-degree-of-freedom, cartesian co-ordinate industrial robot system in order to improve the design and reliability of the system.[24] The system was evaluated under both severe and favorable operating conditions in order to determine bounds on the reliability over the possible range of application conditions. One interesting result of this study is that, under both the severe and favorable conditions, a relatively small group of failure events contribute a large percentage of the overall system failure rate and financial risk. Focusing on eliminating the effects of these few failure events will greatly improve the overall system reliability. In addition, correcting a few of the high risk failure events will often correct other events not under consideration at that time.[24] Another interesting point is that operator error and other peripheral events such as power loss and software errors contribute the most to the unreliability of the system. The designer has moderate to no control over the design of these peripheral systems. Possible solutions to these problems are discussed elsewhere in this survey (Sections 2.2 and 4.4). By using FMECA during the design and operational stages, the robotic community can accumulate a database of failure information for a variety of robots and keep track of all the operating experience available (see Section 3.4). This will help lead to the design of more reliable robots and the accumulation of much needed failure rate information for the quantitative studies of failures discussed in Section 3.3.

One of the limitations of FMECA for robotic applications is the difficulties involved in completely analyzing robots with multiple-event cut sets (i.e. parallel or redundant systems).[24,27] Typically, FMECA only considers single-event failures. To analyze systems with parallel configurations in which several

initiating events must be simultaneously present to cause a larger failure, FMECA would have to include additional failure modes representing all the possible combinations of the appropriate single-failure modes. This leads to an unwieldy amount of analysis for systems with many redundant components. To limit the time commitment, the designer can focus on those failure combinations which will lead to significant system degradation. Also, redundancy typically increases the reliability of the system, and the designer can focus the FMECA on the single-event cut sets which typically contribute the most to system degradation.[24]

3.2 Fault trees

The rigor of FMECA relies upon the skill and experience of the system analyst.[28] A full description of the structural interactions of failure modes in the system can also be derived using the more involved technique of Fault Tree Analysis (FTA). FTA is a deductive method in which failure paths are identified using a fault tree drawing or graphical representation of the flow of fault events.[29] Each event in the tree is a component failure, an external disturbance, or a system operation. The events are connected by logic symbols to create a logical tree of failures. FTA is a well-known analysis technique often used in industry for computer control systems and large industrial plants. In robotics, fault trees have recently been used in defining robot fault-tolerance capabilities[30–32] and more dynamically in providing an on-line database of failure information for real-time fault detection and tolerance algorithms.[33–35]

Figure 7 shows (in reduced scale) a top-level fault tree for a hazardous waste retrieval manipulator. The proposed robot is intended to retrieve waste from within an underground storage tank. The waste will be removed using an end effector equipped with high pressure water jets and a suction device to remove the sludge. The tank may contain obstacles such as vertical riser pipes. Several failure scenarios are considered critical. Among these are the manipulator striking a vertical riser, striking the bottom of the tank, or a fault preventing removal of the arm from the tank. Fault trees have been developed for these scenarios. Figure 7 shows a simplified fault tree for the specific event description of the manipulator colliding with a riser in the rank. The top portion of the fault tree is shown in detail in Fig. 8. The subtree for the operator/controller failure which triggers an event in the top tree is shown in Fig. 9. More detailed versions of these trees exist,[32] and note that the triangles beneath the failure events represent suppressed trees.

The above trees were used for a qualitative analysis of a preliminary manipulator design, and show sufficient detail to highlight the key subsystem failure events. Fully detailed fault trees (down to the individual component level and enumerating all possible failure modes) are useful in understanding and mapping the possible structural interactions of failures within the robot. Automated tree constructors such as the Fault Tree Compiler[36] aid a robot system designer in creating the tree database for the system. However, most robot structures are fairly simple and have generic subparts, making it possible to develop the trees by hand. As in FMECA, the effort involved in FTA can be minimized by limiting the analysis to the critical failure events which will cause the most damage or which have the highest probability of occurring. Those events with a negligible chance of occurring or which have a limited impact on the overall system can be eliminated from the trees.

The Failure Environment Analysis Tool (FEAT)[37] developed at NASA presents another possible method of structurally defining the failure interactions of robot systems. FEAT uses a digraph representation of the fault events within a system. Digraphs are directed graphs in which nodes are connected in AND/OR combinations using petri-net notation instead of logic gates. The digraph structure can easily be converted to the trees created in Fault Tree Analysis (and vice-versa). FEAT has typically been used to analyze more general systems, such as the helium flow control system for the space shuttle, but could be used for robotics as well.

Computer programs using trees or digraphs can help qualitatively in isolating the possible causes of a failure and determining what failures are likely to occur as the result of a given failure. These programs assist the designer in determining the fault-tolerant capabilities of the system. The programs do not, however, reveal all of the fault-tolerant limitations of the system. The analysis is only capable of detecting failed elements which have associated sensors in the physical system and that have been enumerated in the tree or digraph structure. This means that in tracking the causes of a given failure, the analysis programs cannot specifically isolate the cause unless the failed element has a monitoring sensor and the program is capable of interpreting the sensor reading as an error. As mentioned in Section 2.2, robots typically do not have an abundance of internal sensors, therefore the analysis program must be able to interpret the varying effects of different element failures on the available sensors in order to isolate the cause. Even if the failure cannot be isolated, the analysis programs are useful in directing repair of the system to a localized subsystem and thus decreasing the system down time.

3.3 Quantitative analysis

The information available in the structural analysis may be further enhanced by a quantitative analysis of

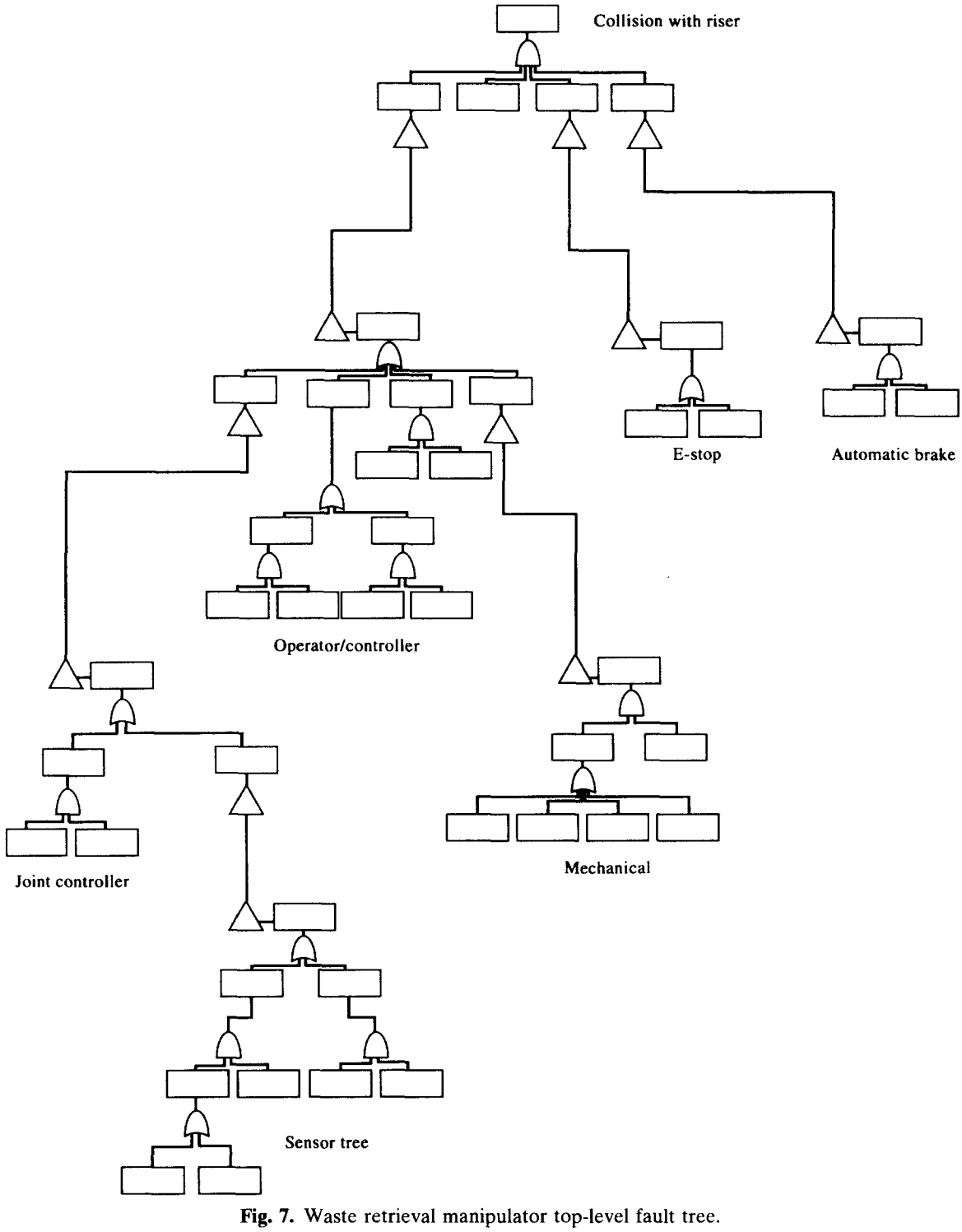

Fig. 7. Waste retrieval manipulator top-level fault tree.

the failures. Failure probabilities could be assigned to each input event and propagated appropriately up through the system. Quantitative analysis provides a measure of the possibility of a selected failure event within the robot. The structure provided by fault trees or digraphs could organize the probabilities for the robot system. Using the annotated structures, robots of significantly different origin and structure can be compared quantitatively for fault-tolerant abilities and survivability. Because of limited empirical data on failures in robot parts, the actual probability numbers are not well-defined or trusted in robotic fields. Further, the fault trees used for robotic analysis may be less detailed than in conventional reliability engineering and contain simplifications such as assuming all failure events to be independent. In a quantitative analysis of the fault trees, the issues of common-cause failures[38,39] and common mode or

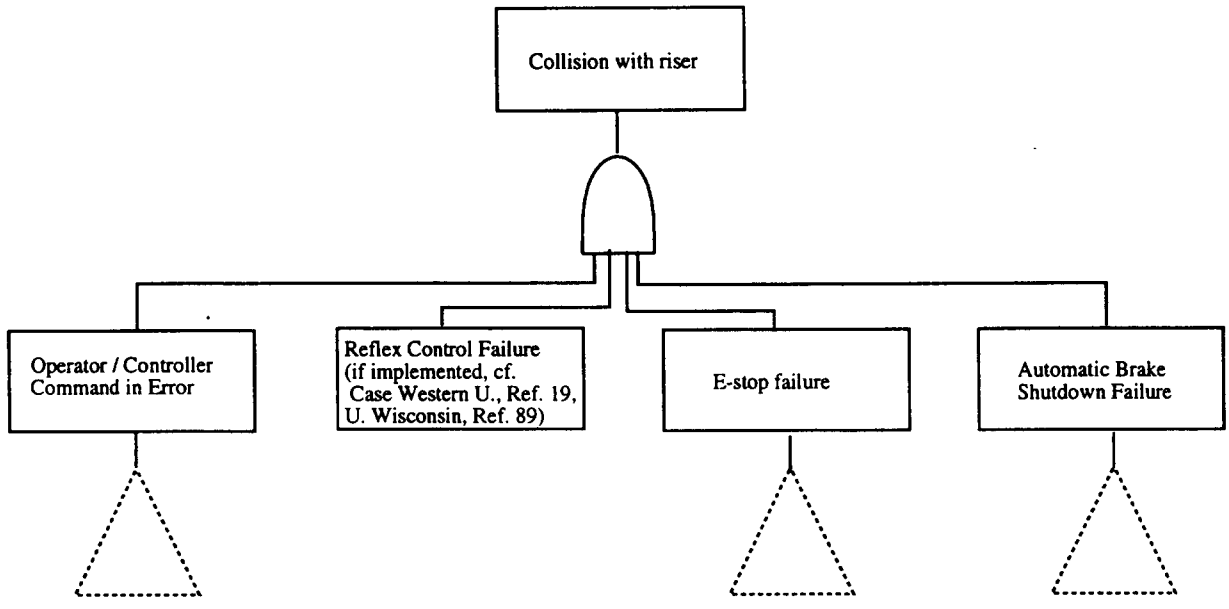

Fig. 8. Detail of top portion of fault tree.

repeated event failures[40] add complexity to the analysis. Additionally, the analysis of M-out-of-N event gates[41] in a fault tree, such as in modeling voting in redundant sensor subsystems, leads to complications in determining cut sets for quantitative analysis. The trees and probabilities, while not perfectly accurate, do provide relative information for performing analyses and comparisons. As more robot-specific failure data is accumulated and as reliability engineering practices expand into robotics, the numbers and trees should become more accurate and trusted.

Static analysis of failure probabilities and their propagation through the system allows the robot 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.[30,42] Even looking at relative orders of magnitudes for failures can help the designer eliminate unlikely failure events. For example, there is only a small possibility of a link breaking during a typical robot task, but gear-train belts often become loose and affect the performance of the motors. The probability of a link failure is thus very small while the probability of a loose belt is relatively large. The fault-tolerance designer can therefore eliminate link failures from the analysis and focus on detecting the effects of loose belts (i.e. inefficient or unresponsive motors). This example shows, however, the importance of collecting empirical data on robot failures, possibly using the FMECA technique described above,[24] in order to develop and understand the failure probabilities. Section 3.4 discusses further the available data and the need for more extensive studies.

3.4 Reliability data and standards

This section discusses the application of standards to robot reliability and surveys the literature of relevant existing standards. The standards documentation spans several different categories. There are handbooks[43-45] and parts specifications[46,47] useful in the characterization of components for a system. Some sources provide specific reliability data for parts which may be used in robotic systems (e.g. Ref. 48). Other documents describe procedures and programs[49,50] which are useful for design, analysis or system operation. Additionally, data item description documents (e.g. Ref. 26) provide standardized report generation procedures which are useful for system specification.

The scope of standards is quite broad and includes issues beyond hardware design. For instance, there are also standards which deal with software quality,[51] safety requirements[52] and a proposed standard for reliability.[53] One of the more widely used military standards handbooks is MIL-HBDK-217F: Reliability of Electronic Equipment.[43] This handbook provides tables to calculate failure rates for a number of electronic components from resistors and capacitors, to switches and relays, to motors and resolvers. The failure rates are also based on the environment in which the component is expected to be used. Because robotic fault tolerance is a relatively new concern, there has been little empirical information gathered on the mean-time-to-failure (MTTF) of common components used in a robotic application. The unique coupling effects in robot joints, for example, might actually cause a higher incidence of failure in the joint motors due to the increased stress

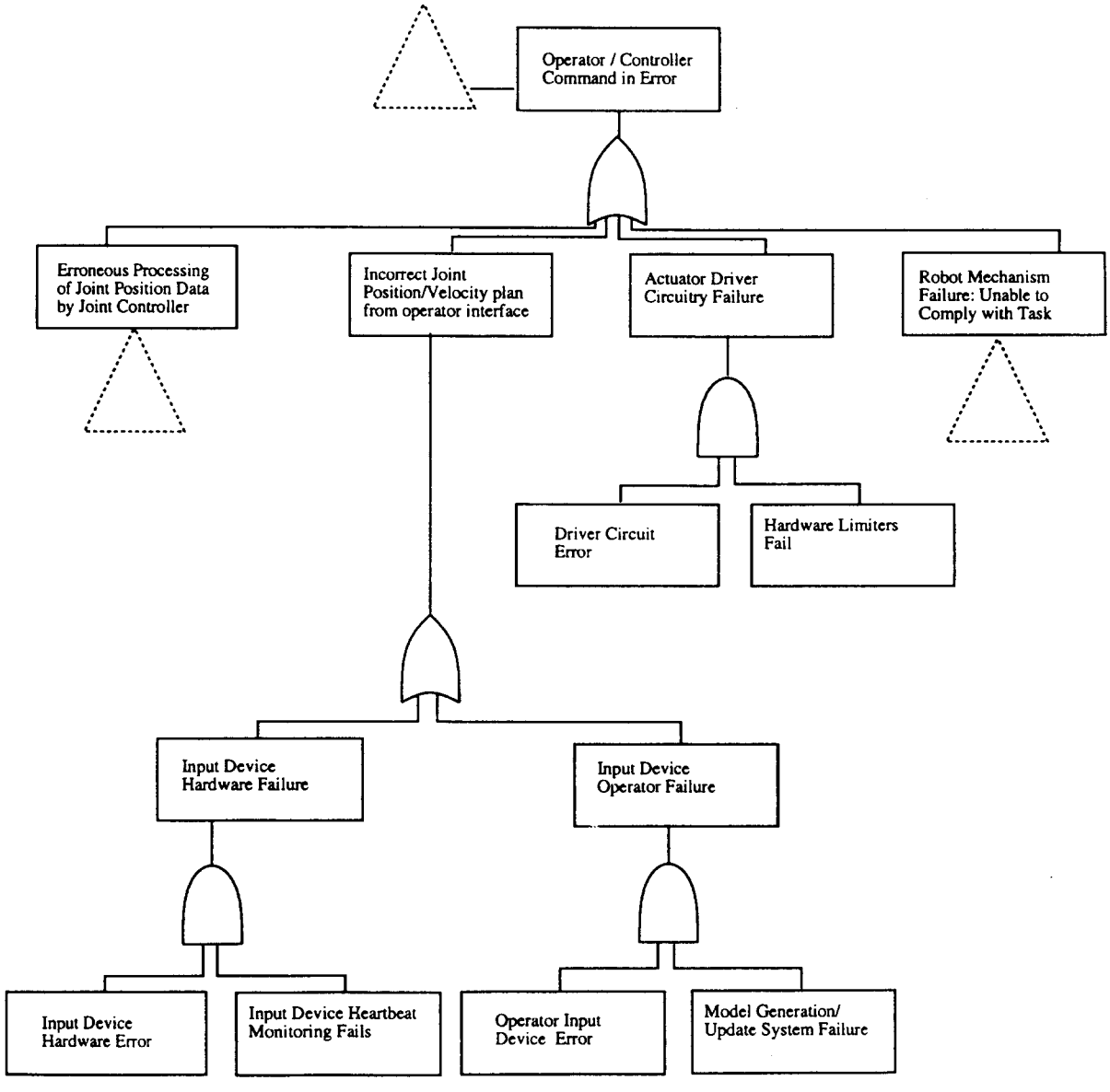

Fig. 9. Detail of operator/controller portion of fault tree.

on the joint. The current standards also, as mentioned above, do not discuss the effects of specific robot environments on the failure rates. In addition, most standards deal with non-nuclear environments, and further studies are underway[54] for hazardous waste sites and nuclear environments to determine how radiation and other hazardous concerns affect the robot components. Thermal effects on component reliability are, however, considered in the reliability handbook[43] and are useful in robotic applications, especially for space robotics. NASA has published a standard for reliability[55] which references the data in MIL-HDBK-217F. In the NASA document, tables are given which further derate components for space use beyond the factors given in MIL-HDBK-217F.

3.5 Expert systems and long-term analyses

Expert systems provide a flexible, intelligent means of controlling fault-analysis tools in conjunction with fault-tolerance algorithms. Expert systems handle large databases, like those typical of failure analysis tools, well and are easily updated and flexible enough to apply to a wide range of robotic systems. The capabilities of expert systems also allow for the addition of embedded programs which may run as spawned or secondary processes, enabling the fault-tolerant system to perform more time-consuming analyses in the background while still responding quickly to failures. The expert system can further keep track of the global system status and fault-tolerance structure and can enhance the fast, instinctive fault-tolerance actions with more global actions.

In robotics, a number of useful fault-analysis techniques are currently being developed such as optimal configuration (cautious robot posture) algorithms,[56] workspace degradation computations[57] and trend analyses.[58] These could easily be incorporated into an expert system for dynamic use in the

robot. For example, the expert system can spawn a process to determine the area reachable by the end effector with the robot in its current state.[12,57] Using the reported changes to the reachable workspace as failures occur, the expert system decides if the target position is still reachable. If the robot can no longer reach its assigned goal, the supervisor signals an abort, alerts the operator, then waits for operator directions. Trend analysis can be used to verify the more instinctive, real-time reactions to failures and decrease the possibility of permanently shutting down a working component due to a false alarm.[58] This more long-term analysis style can also be used to detect slowly moving failures such as drifts in sensor readings.

An expert system can also provide a more interactive connection to the operator for assistance in determining other possible tolerance actions. The expert system can dynamically provide the operator with information on possible causes of failures by automatically traversing the appropriate fault trees or other database structures to accumulate a list of components which may have caused a given failure. An operator could then use the list for future inspections and repairs. When pruning the analysis database of failed components, the system may also search for indications that a component is likely to fail because it is linked either structurally or functionally to a previously failed component. In the ideal case, advanced detection routines could then use this information to be more alert to the possibility of failures in these components while monitoring the system for failures. An expert system thus provides the intelligence necessary to dynamically control and interpret the fault tree information, freeing the operator for other tasks.

The ability to include quantitative information has been added to a dynamic, expert system-based fault tree supervisor for robots.[33,35] Status information is displayed along with a graphical representation of the fault trees to focus an operator's attention on the faulty components as failures occur. On-line analysis could also be useful in directing task-orientated decisions about which joint or component to use for a given task by promoting the use of those systems which are less likely to fail. The robot would then mimic optimal configurations[56] or utilize alternate trajectories so as not to put undue stress on systems which are injured or 'sore'.

Expert systems are capable of intelligently handling the coupled, often chaotic world of robotics. For the German space robotics experiment ROTEX,[59] an expert system is used to monitor multiple built-in test equipment and other diverse hardware and software signals to watch for failures and summarize the great variety of information for the operator. In the Decision Support System (DESSY),[60] the expert system addresses the problems of incorrect, noisy and missing data as well as lags and irregularities in data due to state transitions. DESSY monitors the operational state of the status positioning mechanism and retention latches for the Space Shuttle Payload Deployment and Retrieval Systems (PDRSs) which contain the shuttle's Remote Manipulator System (RMS). DESSY is able to keep track of the system state, even through periods of lost contact when insufficient data is transmitted, using context-sensitive pattern recognition. Maintaining the expected state enables DESSY immediately to tell if a failure has occurred when the data comes back on line as well as predict possible problems in the plan. This is an important feature, especially for remote robots where communication delays may be long enough to hide a failure or the response to an invalid command from the operator until the failure has caused major damage to the system or environment. Rules in the DESSY expert system are disabled when data is unreliable (similar to a human ignoring bad data) and can be reinstated if reliable data returns, allowing for not only graceful degradation but also graceful recovery. Graceful recovery greatly improves a system which experiences temporary failures or false alarms by automatically bringing components that are no longer faulty back on line and into the detection and tolerance loop.

4 FAULT TOLERANCE

After analyzing a robot system for fault-tolerance capabilities and responses to failures, fault-tolerance algorithms can be developed. Robotic fault tolerance has been able to utilize the fault-tolerance suggestions from other fields and combine them with actions derived from the unique attributes of robots. As mentioned earlier, these algorithms need to focus on the components which are the most critical to the robot's survival and control—mainly the sensors and motors. Due to the coupled, nonlinear dynamics of the actual robot (eqn (1)), there is little time in which to tolerate failures before the robot swings out of control, thus some automatic or instinctive reactions to failures should be built into the system. Longer on-line analyses of the failures and subsequent tolerance actions can be postponed until the initial danger has been averted.

4.1 Structural redundancy

Most of the previous work on fault tolerance of mechanical failures in robots has concentrated on those algorithms which rely on duplicated parts for

their fault-tolerant abilities. This type of hardware or structural redundancy has been developed for fault tolerance in many other applications such as computers and large plant systems. The structural redundancy schemes generally focus on faults in one specific part of the robot (mechanical failure in a motor, sensor, etc.). Recent research[12,13] has explored methods of duplicating motors in a robot joint. The two motors in a joint work together to provide one output velocity for the joint. When one of the motors breaks, the other one takes over the faulty motor's functions while adjusting to any transients introduced into the system by the failed motor. If the robot is performing a time-critical or delicate task, fault tolerance must allow the robot to get a run-away motor under control quickly before any damage to the environment or the robot occurs.

The fault-tolerant advantages of hardware redundancy have also led to adding extra parallel structures, such as a backup robot arm or leg,[22] in order to allow many different reconfiguration possibilities in the presence of a failure. Redundant components offer an obvious solution to the reconfiguration problem by providing a backup if one of the components fails. As in triple modular redundancy with computers, redundancy may also give the robot system multiple components to check and vote among, thus improving fault detection. Control systems can further use redundant components to average out faulty data or noise.[61] In robotics, however, the amount of physical redundancy is often limited by cost, size and power considerations. Alternative methods of tolerating failures, such as those discussed in the next few sections, help complement the limited physical redundancy of robot systems.

4.2 Kinematic fault tolerance

Many robots in the emerging generation have the advantage of being kinematically redundant. That is, the robot has more degrees-of-freedom (DOFs) or motions than necessary to position and orient the end effector within its workspace, allowing the robot to choose between multiple joint configurations for a given end effector position (as in Fig. 3). A minimum of six DOFs are necessary for a robot to freely position and orient its end effector in three dimensions. If a robot has eight joints (and thus eight DOFs), it can use the extra two DOFs to provide different approaches to a given point or to avoid obstacles in the task space. This natural redundancy can be used to create fault-tolerant algorithms which use the alternate congfiurations to aid in positioning a robot with failed joints.[23,56] In essence, the positioning workload of the failed joint is redistributed to the remaining joints. These algorithms do not require the addition of extra motors, sensors, or other components to the robot but use the existing structure to provide fault tolerance. Kinematic redundancy can also be artificially induced by constraining the task space of the robot.[62] For example, if the end effector orientation is not important in a three-dimensional workspace task (i.e. only position is necessary), a six-DOF robot which typically has no kinematic redundancy would have three extra DOFs which could be used to provide alternative configurations in failure situations. Thus, large, expensive robots with many degrees-of-freedom and complex controllers are not necessary if the task space can be reduced in failure situations to free up the basic robot joints and enable some level of kinematic redundancy.

If all the sensors for a joint fail, the robot becomes blind to that joint and must shut down the motor. This is the same tolerance response as that for a real motor failure. With kinematic redundancy, it is possible for the robot to reconfigure its model of itself, redistribute the workload of a failed joint, and continue working in a reduced form. The effect of joint failure on the remaining dexterity of a kinematically redundant manipulator has been quantified.[63] An optimal initial configuration of redundant arms can also be calculated to maximize fault tolerance while minimizing the degradation of the system in the event of a failure.[56] This method provides fault tolerance if the robot is near this initial configuration and arranges its joints to mimic the fault-safe configuration as closely as possible.

Other fault-tolerance studies of kinematically redundant robots[62,64] have examined methods of determining which areas of the robot workspace are the most fault tolerant. Because joints are locked in place when they fail, the robot's reachable workspace shrinks as failures occur. Different failures limit the workspace in different ways. One goal of these studies is to find ways to quantify how the failures affect the range of motion of the robot. The best fault-tolerant locations in the workspace can be determined by noting the areas of the workspace that can be reached by many different failure configurations.[62] Tasks that require a high degree of fault tolerance can then be placed in these smaller, more fault-tolerant regions of the workspace and the robot will be better able to reconfigure itself to continue the tasks in the presence of failures. The manipulator joint motion instead of the task space can also be constrained so that the worst-case configuration (one that allows no fault tolerance) is avoided.[64]

4.3 Functional redundancy

Many existing robots such as the Space Shuttle Remote Manipulator System (RMS) do not have identical backup (structurally redundant) components or joints. Currently for the RMS, the astronaut

operator is a key element in detecting and evaluating sensor or joint faults.[1] The rudimentary automatic RMS fault-detection algorithms compare sensor readings to the expected data and shut down the entire robot if the difference between the two values is outside of a pre-defined constant threshold.[65] Because the robot makes no attempt to tolerate the failure, however, the operator must manually dock or completely jettison a faulty RMS. With the addition of fault-tolerance algorithms, many currently critical faults which force the system to shut down immediately could be tolerated, allowing the RMS to automatically restructure its systems into a workable configuration. Because the RMS and many other existing robots are structurally limited, however, in that they do not have redundant hardware or kinematic redundancy, techniques have been developed that use the existing structure for fault tolerance by using functionally equivalent data from dissimilar components.[8,23] For example, if there are two different sensors such as an encoder and a tachometer per joint (each performing a distinct function), the position information of a faulty encoder can be replaced with position information computed by integrating the working tachometer for that joint.[11] There will be issues related to the initial position for the integration; however, this slight error is preferable to using data from the faulty sensor. The redundant information from nonhomogeneous surviving sensors can thus provide basic sensor fault tolerance for the robot. Faulty data can also be replaced with computed estimates of the system state.

Fault detection and tolerance could also be improved using an external sensor such as the vision system. For a robot with one sensor per joint, the additional joint angle information from the vision system would help distinguish between a sensor error and a real joint failure by providing a third opinion. Using the vision system for this task, however, increases the load on the image processing software and will hinder the system's ability to perform its normal vision tasks. Further alternatives to structural and kinematic redundancy for fault-tolerance algorithms need to be explored.

Robot controllers may also have the task of easing the transition through failure-inducing singularities in the robot's trajectory.[66] A congfiuration is termed singular if the robot is configured in such a way as to hinder motion in one direction without rapid changes in one or more joint positions. The joint velocities of a manipulator become extremely high, even for kinematically redundant manipulators, when the robot must move through or close to one of these singular congfigurations while trying to keep its end effector moving on the desired trajectory. Fault detection routines might interpret these jumps in the joint velocities as failures in the robot and erroneously shut down a fault-free system. The optimal damped least-squares technique used in the Singularity Robust Inverse (SRI) algorithm[66–68] ensures feasible joint velocities with minimum end-effector deviation from the specified trajectory. By restricting the joint speed, this inverse kinematics scheme enables the manipulator to avoid drastic joint motions at or near singular congfigurations and helps eliminate false alarms in the fault-detection algorithms. Limiting the motion of the robot also improves the fault tolerance by eliminating the possibility of the robot failing in these singular configurations which would disable motion in a certain direction.[56]

4.4 Robot computer fault tolerance

Additional robot fault tolerance research has focused on improving the reliability of the robot computers. This research area can draw heavily from the fault-tolerance work performed in the field of computer science.[69,70] In hardware, systems without redundant components handle faults by allowing a graceful degradation in functionality or speed. The computer science literature discusses time redundancy[71] in which a computational cycle is lengthened so the remaining fault-free part (or parts) will have enough time to handle the tasks of a faulty component. Other systems use set-switching or processor-switching schemes[71] for reconfiguration. Set-switching removes an entire row, column, or diagonal of the processor array to isolate the faulty processor. The algorithm is then modified to deal with the new dimensions of the mesh. Only a few errors can be tolerated before the mesh is reduced to an unusable size. In processor-switching (typically used in multiprocessor systems), fault-free components are collected to form a basic subpart of the desired system configuration until the full configuration is achieved. This method may, however, require many extra interconnections between components.

Robots are inherently parallel structures as they have many subunits, such as a joint with an internal sensor, repeated throughout the robot. The matrix structure of eqn (1) indicates that the robot equations are parallelizable, although special care must be taken due to the coupled nature of the joints. This leads to an opportunity for parallel implementation of the robotic algorithms which can be exploited in the architectural design[72] and which also provides a valuable foundation for tolerance of robot processor failures. With several computers or processing elements working in parallel, the robot controller can tolerate failures by redistributing the workload of a failed processor to the surviving set.[73] Note that this same method is used to tolerate other types of component failures as well (Sections 4.1 and 4.2).

In order to limit the deterioration of the processor

array, a fault-tolerant processing array specifically for robotic applications[74] was designed to isolate just the faulty processor and its communication links and then reassign the faulty processor's data to the fault-free neighbors. If a processor is chosen to perform the extra calculations, it uses its idle cycles (necessary for the systolic propagation of information through the array) to work on the faulty processor's data. Fault-free processors modify their communication path to route data around a faulty processor if it is a near-neighbor. The hardware must be adjusted as each processor needs extra registers to handle the data from a faulty near-neighbor. This reconfiguration scheme uses no additional spare processors, exploits the idle time inherent in the data computations, and can handle many disjoint faults in the processor array.[75]

Fault tolerance of software errors in robot computers is also currently under exploration.[17,19] Software errors might arise due to human error, incorrect software, or failed hardware. In any case, erroneous commands misdirect the robot and may lead to potentially life-threatening situations.[15,76] To avoid these problems, software commands are continually monitored and pre-tested for actions which might endanger the robot or environment specifically by causing collisions.[77] Such erroneous commands are rejected and alternatives are sought. This type of fault tolerance is termed 'protective autonomy'[3] in nuclear engineering based robotics. Another slightly different approach to software fault tolerance is to have a variety of redundant software modules which are identical only in their functionality.[8] With this algorithmic diversity, the same voting and tolerating techniques used with hardware redundancy can be employed for fault tolerance to eliminate a faulty module or average the results and minimize noise in the data. Arithmetic codes are used to find and correct errors in the matrix computations performed in robot kinematics.[78] Check bits and error correction codes help monitor data transmissions and allow a reconstruction of the original data if the transmission line is faulty.

5 FAULT DETECTION

To initiate fault-tolerance schemes, a failure must first be detected by the robot system. Disciplines such as reliability engineering, process control, and computer science have developed a wide variety of fault detection techniques. Hardware redundancy of system components is used in voting schemes such as triple modular redundancy,[70] in which the components all perform the same task and compare their results. If one of the components in the set is faulty and its result does not agree with the results of the other two components, the faulty component is voted out of the final decision and the correct result is passed on to the rest of the system. If the system is well defined, hypothesis testing can be used for fault detection. Each failure hypothesis is enumerated through such techniques as Kalman filtering and can then be used in matching faulty systems to indicate the most likely hypothesis.[61] Kalman filters may also be used in detecting system failures by comparing the actual measured system outputs with the corresponding expected output derived from a model of the system behavior. Jumps in the filter readings are indicative of a failure. Residual-based parity vectors[79] indicate failures with specific signatures or unique patterns of a residual that become biased away from the fault-free value in the presence of failures.

While the methods from other fields highlight possible approaches for fault detection, developing methods specifically for robotics is hampered by the limitations and special characteristics of robots. Robotics can make use of any redundant component both in fault detection and fault tolerance, but the level of redundancy is typically limited by special considerations. In medical applications, for example, robots need to be small enough to perform delicate operations and use as little power as possible while not endangering the patient. The weight of space-based robots is constrained by the cost involved in lifting them out of Earth's gravity. In nuclear plants, robot components must be specially treated to resist harmful radiation, and there are also a restrictive number of sensor types that are suitable for the radioactive environment. The typically small number of sensors used in robots further limits the ability of the detection routines to isolate failures within the system. While a human operator can be used to augment failure detection routines, the human typically responds slowly, can only isolate failures that drastically affect the robot's performance,[2] and has difficulty monitoring large robots with many joints.[3] To detect failures quickly without depending on a human's senses, the robot must rely only on the information it can glean from its existing sensors.

The on-line time to analyze the system for alternative tolerance options is also limited. Robots respond to failures very quickly and with wild motions that endanger both the robot and its environment. Hypothesis testing, which may take quite a while to reach a solution, is therefore inadvisable for robotic application unless the hypotheses can be pre-computed off-line and stored in a comparison database for on-line use. To be able to compute all the important hypotheses off-line, the robot application, task and environment should be well defined. This might be possible for robots such as those used in assembly lines with highly repetitive tasks and generally stable environments. For robots

performing a variety of tasks or with less structured environments, there is an extremely large number of possible failure situations. Hypotheses for all the failures may not be possible and, as in FMECA, the analysis may need to be limited to only those failure situations that are the most critical.

In addition to being dynamic, robots are difficult to model precisely. Robots are inherently nonlinear, but their dynamic models tend to be linearized to simplify the controller. Errors also exist between the actual and modeled system parameters. The inaccuracies are complicated by noise and changes due to the surrounding environment. Kalman filters and parity vectors, which are highly sensitive to modeling errors[61,80] are thus not as useful for robotic fault tolerance. These methods also may be more sensitive to one failure over another. New methods of masking out the errors are being developed to enable correct detection and identification of robot failures.[81] This will be discussed further in the next few sections.

5.1 Analytical redundancy

The first step in developing fault detection algorithms is to identify as many useful independent tests as possible while eliminating redundant checks. There are established results for the generation of such detection tests or residuals for general dynamic systems using the concept of analytical redundancy.[61] Analytical redundancy is a concept for failure detection and isolation which uses only the available sensor components in a system to generate residuals from which failures can be identified. Thorough reviews of the various methods of analytical redundancy exist.[61,79,82] By comparing the histories of sensor outputs versus the actuator inputs, results from dissimilar sensors can be compared at different times in order to check for failures. Analytical redundancy provides a useful mathematical approach for determining the various redundancies that are relevant to the failures under consideration.[79] These methods can be applied to robotic systems to develop the tests necessary for robotic fault detection.

Analytical redundancy uses the available redundancy (either hardware redundancy or functional redundancy of unlike components) of the system to provide fault-detection tests. It is therefore a useful technique for fault-tolerant robots due to the limited physical redundancy in robotics. An adaptation of analytical redundancy specifically for robotic applications[23] is summarized below. The results were developed along the lines of the mathematical characterization of analytical redundancy for dynamic systems[79] modeled as

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + W(t) \\ y(t) &= Cx(t) + V(t), \end{aligned} \tag{3}$$

where x is the state vector for the system, A is the system dynamics matrix, B is the input distribution matrix, u is the known input (or control) signal, y is the output measurement signal, C is the measurement matrix, and W and V are input and output noise vectors arising from nonlinearities and inaccurate sensors.

The robot used in the analysis contained two sensors per joint, a position sensor and a velocity sensor. These sensors were chosen to be an encoder and a tachometer which enables an examination of the structure for temporal redundancy[79] between two different types of sensors. For each joint, the number of actuators, q, is 1. The robot dynamic relations are derived from ($\underline{x} = [\theta \dot{\theta}]^T$:

$$\underline{x}(k+1) = \begin{bmatrix} 1 & (\Delta t) \\ 0 & 1 \end{bmatrix} \underline{x}(k) + \begin{bmatrix} 0 \\ (\Delta t) \end{bmatrix} u(k), \tag{4}$$

and

$$\underline{y}(k) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \underline{x}(k). \tag{5}$$

The variable θ is the angular position of each joint and Δt is the length of time between each iteration k. The input $u(k)$ is the control acceleration sent by the controller for iteration k. Table 2 shows the robot detection tests resulting from the use of the analytical redundancy methods on eqns (4) and (5).

Two apparent problems with the analytical redundancy tests are that the sensors are not perfect (for example, the encoder truncates values) and some sensed values must be differentiated to compare them to the values from functionally equivalent sensors (the encoder would be differentiated twice to compare it to the computed acceleration). The inherent loss of precision and noise in the sensors are compounded through the differentiation. The fault tolerance algorithms further the problem by replacing faulty sensors with data from functionally equivalent sensors (i.e. when an encoder fails, its data is replaced by the position information derived from integrating tachometer readings). To eliminate the differentiation problem, the encoder reading can be compared to the expected position derived from the planner or to the integrated tachometer reading which is more stable. The RMS uses similar tests to detect encoder failures[1,65] but takes several time steps to verify the error. Because robot controllers are generally designed to adjust to small errors in the system, fault-tolerance designers will typically accept the slight increase in noise caused by differentiation/integration in order to avoid the disastrous consequences of allowing faulty data into the system. The next section discusses how sensor noise as well as system modeling errors can be further masked out in the detection tests using thresholds.

Table 2. Analytical redundancy derived detection tests

Encoder reading vs tachometer reading	$\frac{\theta(k+1)-\theta(k)}{\Delta t}=\dot{\theta}(k)$
Encoder reading vs computed acceleration	$\frac{(\theta(k+2)-\theta(k+1))-(\theta(k+1)-\theta(k))}{\Delta t^2}=u(k)$
Tachometer reading vs computed acceleration	$\frac{\dot{\theta}(k+1)-\dot{\theta}(k)}{\Delta t}=u(k)$
Tachometer reading vs computed jerk	$\dddot{\theta}(k)=\frac{(u(k+1)-u(k))}{\Delta t}$

5.2 Modeling uncertainty—fault-detection thresholds

In their pure form, the ideal tests derived from analytical redundancy cause an unacceptable number of false alarms in robotic fault detection due to the sensor and modeling errors inherent to the fault-free system which arise, for example, from linearization of the robot equations and inaccuracies in model parameters such as link inertia or mass. An acceptable bound or threshold for the difference between the desired value and the sensor reading must be chosen to mask out these inaccuracies.[83] Typically, in robotics, the thresholds have been determined empirically by monitoring a fault-free run and setting the threshold larger than the noted maximum deviation from the desired path.[1] This method results in a constant threshold based on a specific trajectory. The effects of the modeling errors and sensor noise, however, fluctuate dynamically as the motion of the robot changes and as failures occur. The constant threshold can lead to many false alarms in this dynamic situation as it was not designed for all possible situations. This was demonstrated using the RMS fault-detection system on the STS-49 mission (May 1992) of the Space Shuttle. On this flight, one consistency check triggered 13 false alarms because a number of special operating conditions such as unusual hand controller inputs and loading on the manipulator were not taken into account.[65] The thresholds therefore need to be dynamic in order to cope with the ever-changing robot status while remaining small enough to catch any failure-induced errors that the feedback controller would be unable to handle.

A variety of algorithms focus on computing thresholds for fault detection in uncertain dynamic systems.[84–88] One such method is the Reachable Measurement Intervals (RMI) algorithm developed and analyzed for aircraft systems by Horak *et al.* in various publications.[80] RMI provides conditions for finding the extreme possible system outputs $y(t)$ under fault-free conditions for each input $u(t)$ and state $x(t)$ given bounded parameter variations in the coefficient matrices of the dynamic equations (A, b and W from eqn (3)). The procedure is started at a known state in the past history of the trajectory and run for a specific window of iterations to reach the current time. This allows RMI to take into account the global effects of the parameter uncertainty.

RMI, however, is designed for systems with primarily linear dynamics and thus requires special tailoring to fit the nonlinear dynamics of robot manipulators. The RMI equations become unreasonably complex to implement for robots of useful size.[81] These complex equations must be recomputed at each iteration for each set of uncertain parameter extremes in order to determine the maximum and minimum output. Increasing the number of joints in the robot geometrically increases the number of comparisons needed during the run.

In addition, a history of the effects are built-up by beginning the algorithm at a given distance in the past and working back up to the present iteration, computing the maximum and minimum for each step. Thus, the time needed to compute the RMI bounds for robots with many uncertain parameters or multiple joints prohibits a real-time response to failures for the detection algorithms. If the robot cannot make a fast decision about a possible failure, the errors rapidly escalate and endanger the robot and the environment. It only takes a few iterations of the controller before the robot begins to swing wildly out of control.

RMI also requires a large amount of memory to maintain the various matrix sets used in finding the maximum and minimum variances. Some savings are possible due to repetition within the matrices, but the memory requirements may be too large to justify the autonomy needed for quick, on-board fault detection. The memory space is typically limited in robotics due to the cost and the constraints on space and power. This is especially true of space robotics where most of the on-board memory is dedicated to the robot control algorithms.

To overcome these concerns, a new model-based threshold algorithm (ThMB) which still utilizes the idea of finding the maximum variance possible due to bounded parameter errors at each state iteration was developed as an alternative to the RMI concept.[81]

This new method exploited the similarities between RMI and analytical redundancy. With ThMB, only two time steps of history were needed for adequate fault detection in robots. ThMB thus represents a local optimization as compared to the global optimizations of the RMI method. It is possible that a smaller error at the current iteration will ultimately result in a larger error later in the trajectory than the error currently used in the calculations. Despite this problem, the use of only local history allows ThMB to perform significantly fewer calculations at each iteration and it is thus able to produce a threshold much faster than RMI (the window used in one implementation of RMI took 100 internal iterations to produce the thresholds for each robot controller cycle[81]). This fast response is essential for robotic fault detection.

6 CONCLUSIONS

Robot fault tolerance is a rapidly expanding area of research, especially as robots move into more hazardous or remote environments. The potential exists to make a fully autonomous robot system that can survive for long periods on its own without succumbing to internal failures. However, this task of developing a self-sufficient, reliable robot is hindered by the special aspects of robotics such as the speed at which the robot responds to failures, the modeling inaccuracy, and the complex dynamics and joint coupling. Research is currently underway to explore ways to overcome many of these problems, but few of the new ideas have been applied to actual robots. Operational experience must be gained to determine how often and in what manner robots fail as well as to test out the new algorithms for fault detection and tolerance in real-time systems.

In this survey, we present many of the common tools used in reliability analysis, fault detection, and fault tolerance along with the limited applications in robotics. We also discuss how some of these tools may not fit the robotic problems in their current form and offer some suggestions for improvement to engender interest and research in defining better methods for robotic needs. Table 3 summarizes the current research in robotics on the various aspects of fault detection and tolerance and shows under which categories the current robotic work referenced in this survey fall.

Table 3. Summary of robotic fault detection and tolerance research

Research topic	References
Robotic failure analyses	
Failure impact studies	2, 57
Reliability prediction	7, 24, 30, 42
Fault trees	25, 31–33, 37
Expert systems	35, 59, 60
Fault detection	
Current systems	1, 11, 65, 83
Modeling inaccuracies	81, 88
Fault tolerance	
Human/robot hazards	15, 20, 76
Planning error recovery	9, 14, 16, 18, 77, 78
Software control	10, 17, 19, 89
Hardware redundancy	
robot mechanical systems	12, 13, 22, 31
robot computer systems	72–75
Kinematic redundancy	21, 56, 62–64, 66
Functional redundancy	8, 81

ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation under grants DDM-9202639 and MSS-9024391, by DOE Sandia National Laboratory Contract number 18-4379A and by DOE contract DE-AC04-94AL8500, and by a Mitre Corporation graduate fellowship and a National Science Foundation graduate fellowship RCD-9154692.

REFERENCES

1. PDRS MAL 2102. *Payload Deployment and Retrieval System Malfunction Workbook.* NASA Technical manual, NASA Johnson Space Center, Houston, TX, Apr. 1988.
2. Draper, J. V., Handel, S. & Hood, C. C., The impact of partial joint failure on teleoperator task performance. In *Proc. Fourth ANS Topical Meet. on Robotics and Remote Systems,* Albuquerque, NM, Feb. 1991. American Nuclear Society, LaGrange Park, IL, 1991, pp. 433–9.
3. Larcombe, M. H. E. & Halsall, J. R., *Robotics in Nuclear Engineering.* Graham & Trotman Ltd, London, 1984.
4. Poole, H. H., *Fundamentals of Robotics Engineering.* Van Nostrand Reinhold, NY, 1989.
5. Sandler, B. Z., *Robotics: Designing the Mechanisms for Automated Machinery.* Prentice Hall, Englewood Cliffs, NJ, 1991.
6. Spong, M. W. & Vidyasagar, M., *Robot Dynamics and Control.* John Wiley & Sons, Inc., NY, 1989.
7. Tangorra, F., Montgomery, A. D. & Grasmeder, R. F., 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, Jan.–Feb. 1987–88.
8. Toye, G., Management of non-homogeneous functional modular redundancy for fault tolerant programmable electro-mechanical systems. Doctoral thesis, Department of Mechanical Engineering, Stanford University, Stanford, CA, Jul. 1989.
9. Anderson, R. J., SMART: A modular architecture for robotics and teleoperation. In *1993 IEEE Int. Conf. Robotics and Automation,* Atlanta, GA, May 1993. IEEE Computer Society Press, Los Alamitos, CA, 1993, pp. 416–21.

10. Stewart, D. B., Volpe, R. A. & Khosla, P. K., Integration of real-time software modules for reconfigurable sensor-based control systems. In *Proc. 1992 IEEE/RSJ Int. Conf. Intelligent Robots and Systems,* Rayleigh, NC, Jul. 1992. IEEE, Piscataway, NJ, 1992, pp. 325–32.
11. Visinsky, M. L., Dynamic fault detection and intelligent fault tolerance for robotics. Doctoral thesis, Department of Electrical and Computer Engineering, Rice University, Houston, TX, Apr. 1994.
12. Sreevijayan, D. & Tesar, D., On the design of fault tolerant robotic manipulator systems. Master thesis, Mechanical Engineering Department, University of Texas, Austin, TX, June 1992.
13. Wu, E., Diftler, M., Hwang, J. & Chladek, J., A fault tolerant joint drive system for the space shuttle remote manipulator system. In *1991 IEEE Int. Conf. Robotics and Automation,* Sacremento, CA, April 1991. IEEE Computer Society Press, Los Alamitos, CA, 1991, pp. 2504–9.
14. Farah, J. J. & Kelley, R. B., The planning coordinator for robust error recovery and dynamic on-line planning of robotic tasks. In *Proc. 1992 IEEE/RSJ Int. Conf. Intelligent Robots and Systems,* Raleigh, NC, July 1992. IEEE, Piscataway, NJ, 1992, pp. 1262–9.
15. Kumamoto, H., Sato, Y. & Inoue, K., Hazard identification and safety assessment of human–robot systems. In *Engineering Risk and Hazard Assessment,* Vol. 1, ed. A. Kandel & E. Anvi. CRC Press, Inc., Boca Raton, FL, 1988, pp. 61–80.
16. Simmons, R. G., Monitoring and error recovery for autonomous walking. In *1992 IEEE/RSJ Int. Conf. Intelligent Robots and Systems,* Raleigh, NC, July 1992. IEEE, Piscataway, NJ, 1992, pp. 1407–12.
17. Tso, K. S., Hecht, M. & Marzwell, N. I., Fault tolerant robotic system for critical applications. In *1993 IEEE Int. Conf. Robotics and Automation,* Atlanta, GA, May 1993. IEEE Computer Society Press, Los Alamitos, CA, 1993, pp. 691–6.
18. Valavanis, K. P., Jacobson, C. A. & Gold, B. H., Integration control and failure detection with application to the robot payload variation problem. *J. Intelligent and Robotic Systems,* **4** (1991) 145–73.
19. Wikman, T. S., Branicky, M. S. & Newman, W. S., Reflexive collision avoidance: A generalized approach. In *1993 IEEE Int. Conf. Robotics and Automation,* Atlanta, GA, May 1993. IEEE Computer Society Press, Los Alamitos, CA, 1993, pp. 31–6.
20. Dhillon, B. S., *Robot Reliability and Safety.* Springer-Verlag, NY, 1991.
21. Maciejewski, A. A., The design and control of fault tolerant robots for use in hazardous or remote environments. In *Proc. Fourth ANS Topical Meet. on Robotics and Remote Systems,* Albuquerque, NM, Feb. 1991. American Nuclear Society, LaGrange Park, IL, 1991, pp. 633–42.
22. Ting, Y., Tosunoglu, S. & Tesar, D., A control structure for fault tolerant operation of robotic manipulators. In *1993 IEEE Int. Conf. Robotics and Automation,* Atlanta, GA, May 1993. IEEE Computer Society Press, Los Alamitos, CA, 1993, pp. 684–90.
23. Visinsky, M. L., Walker, I. D. & Cavallaro, J. R., Layered dynamic fault detection and tolerance for robots. In *1993 IEEE Int. Conf. Robotics and Automation,* Atlanta, GA, May 1993. IEEE Computer Society Press, Los Alamitos, CA, 1993, pp. 180–7.
24. Walker, D. A., Creating reliability in the design process: An evaluation approach, design tool, and example application. Master thesis, Department of Mechanical Engineering, University of Tennessee, Knoxville, TN, Aug. 1990.
25. Pack, G. L. & Wadsworth, D. B., Failure environment analysis tool applications. NASA technical memorandum 10477: *Research and Development Annual Report* 1992, NASA Johnson Space Center, Houston, TX, Aug. 1993.
26. DI-R-7085A. *Failure Mode, Effect, and Criticality Analysis Report.* Department of Defense, Washington, DC, Sept. 1984.
27. Smith, D. J., *Reliability and Maintainability in Perspective: Practical, Contractual, Commercial, and Software Aspects.* Wiley, NY, 1985.
28. Thomson, J. R., *Engineering Safety Assessment: An Introduction.* Wiley, NY, 1987.
29. Bloch, H. P. & Geitner, F. K., *An Introduction to Machinery Reliability Assessment.* Van Nostrand Reinhold, NY, 1990.
30. Guthrie, V. H. & Whittle, D. K., 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.)
31. NASA, Computer based control system noncompliance report for computer independent hazard control system. Report, NASA Goddard Flight Center, Greenbelt, MD, Sept. 1991.
32. Walker, I. D. & Cavallaro, J. R., Failure mode analyses of the Hanford manipulator. Technical report TR-9402, Department of Electrical and Computer Engineering, Rice University, Houston, TX, Mar. 1994.
33. Visinsky, M. L., Cavallaro, J. R. & Walker, I. D., Expert system framework for fault detection and fault tolerance for robots. In *Robotics and Manufacturing: Recent Trends in Research, Education, and Applications. Proc. Fourth Int. Symp. Robotics and Manufacturing,* Santa Fe, NM, Nov. 1992. ASME Press, NY, 1992, pp. 793–800.
34. Visinsky, M. L., Walker, I. D. & Cavallaro, J. R., Robotic fault tolerance: Algorithms and architectures. In *Robotics and Remote Systems for Hazardous Environments, Vol.* 1. Prentice Hall Publishing Co., Englewood Cliffs, NJ, 1993, pp. 53–73.
35. Visinsky, M. L., Cavallaro, J. R. & Walker, I. D., Expert system framework of fault detection and fault tolerance in robotics. *Int. J. Computers and Electrical Engng,* **20**(5) (1994) 421–36.
36. Martensen, A. L. & Butler, R. W., The fault-tree compiler. NASA technical memorandum 89098, NASA Langley Research Center, Hampton, VA, Jan. 1987.
37. Stevenson, R. W. & McNenny, R. W., Fault isolation in space station Freedom systems from the space station control center. In *NASA Joint Applications in Instrumentation Process and Computer Control (JAIPCC) 1992 Symposium: Technologies for New Exploration,* Clear Lake, TX, Mar. 1992, NASA Johnson Space Center, Houston, TX, 1992.
38. Mosleh, A., Special issue on dependent failure analysis: Guest editorial. *Reliability Engineering and System Safety,* **34**(3) (1991) 243–8.
39. Mosleh, A., Common cause failures: An analysis methodology and examples. *Reliability Engineering and System Safety,* **34**(3) (1991) 249–92.
40. Rai, S., A direct approach to obtain tighter bounds for large fault trees with repeated events. In *Proc. IEEE Reliability and Maintainability Symp.,* Anaheim, CA, Jan. 1994. IEEE, Piscataway, NJ, 1994, pp. 475–80.
41. Radke, G. E., Jr, & Evanoff, J., A fast recursive

algorithm to compute the probability of M-out-of-N events. In *Proc. IEEE Reliability and Maintainability Symp.*, Anaheim, CA, Jan. 1994. IEEE, Piscataway, NJ, 1994, pp. 114-17.
42. Guthrie, V. H., Gebbie, J. W. & Paula, H. M., Bravo 2·0: Reliability, availability, and maintainability (RAM) analysis and risk assessment software. *DOE Small Business Innovative Research (SBIR) Program Report, Phase II—Final Report JBFA-111-92,* JBF Associates, Inc., Knoxville, TN, Jan. 1993. (Performed for Oak Ridge National Laboratory.)
43. MIL-HDBK-217F. *Reliability Prediction of Electronic Equipment.* Department of Defense, Rome Laboratory, Griffiss Air Force Base, NY, Jan. 1990.
44. NUREG-0492. *Fault Tree Handbook.* Nuclear Regulatory Commission, Washington DC; 1981.
45. ANSI/IEEE STD 352-1987. *IEEE Guide for General Principles of Reliability Analysis of Nuclear Power Generating Station Safety Systems.* IEEE Power Engineering Society, NY, 1987.
46. MIL-STD-1553B. *Aircraft Internal Time Division Command/Response Multiplex Data Bus.* Department of Defense, ASD, Wright-Patterson Air Force Base, OH, Sept. 1978.
47. MIL-M-8609B. *Motors, Direct-Current, 28 Volt System, Aircraft General Specification for.* Department of Defense, Dec. 1987, Notice 1.
48. IEEE STD 500-1984. *IEEE Guide to the Collection and Presentation of Electrical, Electronic, Sensing Component, and Mechanical Equipment Reliability Data for Nuclear-Power Generating Stations.* IEEE Power Engineering Society, NY, 1984.
49. MIL-STD-1629A. *Procedures for Performing a Failure Mode, Effects and Criticality Analysis.* Department of Defense, NAEC, Lakehurst, NJ, Nov. 1980.
50. MIL-STD-882B. *System Safety Program Requirements.* Department of Defense, AFSC, Andrews Air Force Base, Washington DC, Mar. 1984.
51. DOD-STD-2168. *Defense System Software Quality Program.* Department of Defense, ARDEC, Picatinny Arsenal, NJ, Apr. 1986.
52. ANSI/RIA R15.06-1986. *American National Standard for Industrial Robots and Robot Systems—Safety Requirements.* ANSI/Robotics Industries Association, Ann Arbor, MI, 1986.
53. BSR/RIA R15.05-3-199X. *Proposed American National Standard for Industrial Robots and Robot Systems—Guidelines for Reliability Acceptance Testing.* Robotics Industries Association, Ann Arbor, MI, 1993.
54. Tulenko, J. S., Ekdahl, D., Utley, L. & Hamilton, H., On-line annealing processes for hardening electronic components in mobile robots for radiation environments. In *Proc. ANS Conf. Remote Systems Technology,* San Francisco, CA, 1991. American Nuclear Society, LaGrange Park, IL, 1991, pp. 183–6.
55. NASA-TM-4322. *NASA Reliability Preferred Practices for Design and Test.* NASA Office of Safety and Mission Quality, Washington, DC, Apr. 1991.
56. Maciejewski, A. A., Kinetic limitations on the use of redundancy in robotic manipulators. *IEEE Transactions on Robotics and Automation,* **7**(2) (1991) 205–10.
57. Pradeep, A. K., Yoder, P. J., Mukundan, R. & Schilling, R. J., Crippled motion in robots. *IEEE Transactions on Aerospace and Electronic Systems,* **24**(1) (1988) 2–13.
58. Nelson, K. S. & Hadden, G. D., Trend recognition and failure prediction of the attitude determination and control system of the space station Freedom. In *Advances in the Astronautical Sciences,* Vol. 78. Univelt Inc., San Diego, CA, 1992, pp. 149–60. Also in *Proc. 15th Ann. Rocky Mountain Guidance and Control Conf.,* Keystone, CO, Feb., 1992.
59. Hotop, H. J. & Strube, C. D., The health and fault management expert system for the ROTEX Robot. In *40th Congr. Int. Astronautical Federation,* Malaga, Spain, Oct. 1989. American Institute of Aeronautics and Astronautics, Washington DC, 1989.
60. Land, S. A., Malin, J. T. & Culp, D. R., Monitoring system with tolerance for real-time data problems. In *Proc. IEEE 9th Conf. on Artificial Intelligence for Applications,* Orlando, FL, March 1993. IEEE Computer Society Press, Los Alamitos, CA, 1993, p. 460.
61. Stengel, R. F., Intelligent failure-tolerant control. *IEEE Control Systems Magazine,* **11**(4) (1991) 14–23.
62. Paredis, C. J. J. & Khosla, P. K., Synthesis methodology for task based reconfiguration of modular manipulator systems. In *Proc. Sixth Int. Symp. Robotics Research,* Hidden Valley, PA, October 1993. MIT Press, Cambridge, MA, 1993.
63. Maciejewski, A. A., Fault tolerant properties of kinematically redundant manipulators. In 1990 *IEEE Conf. Robotics and Automation,* Cincinnati, OH, May 1990. IEEE Computer Society Press, Los Alamitos, CA, 1990, pp. 638–42.
64. Lewis, C. L. & Maciejewski, A. A., Optimization of the dynamic performance of redundant robots in the presence of faults. In *Robotics and Manufacturing: Recent Trends in Research, Education, and Applications Proc. Fourth Int. Symp. Robotics and Manufacturing,* Santa Fe, NM, Nov. 1992. ASME Press NY, 1992, pp. 279–84.
65. Zanaty, F., Consistency checking techniques for the space-shuttle remote manipulator system. *Spar J. Engng and Technol.,* **2** (1993) 40–9.
66. Deo, A. S. & Walker, I. D., Robot subtask performance with singularity robustness using optimal damped least squares. In *1992 IEEE Int. Conf. Robotics and Automation,* Nice, France, May 1992. IEEE Computer Society Press, Los Alamitos, CA, 1992, pp. 434–41.
67. Nakamura, Y. & Hanafusa, H., Inverse kinematic solutions with singularity robustness for robot manipulator control. *ASME J. Dynamic Systems, Measurement, and Control,* **108** (1986) 163–71.
68. Wampler, C. W., Manipulator inverse kinematic solutions based on vector formulation and damped least-squares methods. *IEEE Transactions on Systems, Man, and Cybernetics,* **16**(1) (1986) 93–101.
69. Nelson, V. P. & Carroll, B. D., (eds), *Tutorial: Fault-Tolerant Computing.* IEEE Computer Society Press, Los Alamitos, CA, 1987.
70. Nelson, V. P., Fault-tolerant computing; Fundamental concepts. *IEEE Computer,* **23**(7) (1990) 19–25.
71. Chean, M. & Fortes, J. A. B., A taxonomy of reconfiguration techniques for fault-tolerant processor arrays. *IEEE Computer,* **23**(1) (1990) 55–67.
72. Hamilton, D. L., Bennett, J. K. & Walker, I. D., Parallel fault-tolerant robot control. In *1992 SPIE Conf. Cooperative Intelligent Robotics in Space III,* Boston, MA, Nov. 1992. SPIE, Bellingham, WA, 1992, pp. 251–61.
73. Hamilton, D. L., Visinsky, M. L., Bennett, J. K., Cavallaro, J. R. & Walker, I. D., Fault-tolerant algorithms and architectures for robotics. In *1994 IEEE Mediterranean Electrotechnical Conference,* Antalya, Turkey, Apr. 1994. IEEE, Piscataway, NJ, 1994, pp. 1034–6.
74. Cavallaro, J. R., Near, C. D. & Uyar, M. Ü., Fault-tolerant VLSI processor array for the SVD. In *IEEE Int. Conf. Computer Design,* Cambridge, MA,

Oct. 1989. IEEE Computer Society Press, Los Alamitos, CA, 1989, pp. 176–80.
75. Kota, K. & Cavallaro, J. R., CMOS processor element for a fault-tolerant SVD array. In *1993 SPIE Conf. Advanced Signal Processing Algorithms, Architectures and Implementations IV,* vol. 2027, San Diego, CA, July 1993. SPIE, Bellingham, WA, 1993, pp. 483–94.
76. Graham, J. H. (ed.) *Safety, Reliability, and Human Factors in Robotic Systems.* Van Nostrand Reinhold, NY, 1991.
77. Donald, B. R., *Error Detection and Recovery in Robotics.* Springer-Verlag, NY, 1989.
78. Han, J. Y., Fault-tolerant computing for robot kinematics using linear arithmetic codes. In *1990 IEEE Int. Conf. Robotics and Automation,* Cincinnati, OH, May 1990, pp. 285–90.
79. Chow, E. Y. & Willsky, A. S., Analytical redundancy and the design of robot failure detection systems. *IEEE Transactions on Automatic Control,* **AC-29**(7) (1984) 603–14.
80. Horak, D. T., Failure detection in dynamic systems with modeling errors. *AIAA J. Guidance, Control, and Dynamics,* **11**(6) (1988) 508–16.
81. Visinsky, M. L., Walker, I. D. & Cavallaro, J. R., New dynamic model-based fault detection thresholds for robot manipulators. In *1994 IEEE Int. Conf. Robotics and Automation,* San Diego, CA, May 1994. IEEE Computer Society Press, Los Alamitos, CA, 1994, 1388–95.
82. Willsky, A. S., A survey of design methods for failure detection in dynamic systems. *Automatica,* **12** (1976) 601–11.
83. Moya, M. M. & Davidson, W. M., Sensor-driven, fault-tolerant control of a maintenance robot. In *1986 IEEE Int. Conf. Robotics and Automation,* San Francisco, CA, Feb. 1986. IEEE Computer Society Press, Los Alamitos, CA, 1986, pp. 428–34.
84. Ding, X. & Frank, P. M., Frequency domain approach and threshold selector for robust model-based fault detection and isolation. In *Proc. IFAC Fault Detection, Supervision and Safety for Technical Processes,* Baden-Baden, Germany, 1991. Pergamon Press, Oxford, UK, 1992, pp. 271–6.
85. Emami-Naeini, A., Akhter, M. M. & Rock, S. M., Effect of model uncertainty on failure detection: The threshold selector. *IEEE Transactions on Automatic Control,* **33**(12) (1988) 1106–15.
86. Isermann, R., Appel, W., Freyermuth, B., Fuchs, A., Janik, W., Neumann, D., Reiss, Th. & Wanke, P., Model based fault diagnosis and supervision of machines and drives. In *Proc. IFAC 11th Triennial World Congr.,* Tallinn, Estonia, Aug. 1990. Pergamon Press, Oxford, UK, 1991, pp. 1–12.
87. Rückwald, R., A model-based fault detection concept for mechanical systems. In *Proc. IFAC Fault Detection, Supervision and Safety for Technical Processes,* Baden-Baden, Germany, 1991. Pergamon Press, Oxford, UK, 1992, pp. 205–10.
88. Wünnenberg, J. & Frank, P. M., Dynamic model based incipient fault detection concept for robots. In *Proc. IFAC 11th Triennial World Congr.,* Tallinn, Estonia, Aug. 1990. Pergamon Press, Oxford, UK, 1991, pp. 61–6.
89. Lumelsky, V. J. & Cheung, E., Real-time collision avoidance in teleoperated whole-sensitive robot arm manipulators. *IEEE Transactions on Systems, Man, and Cybernetics,* **23**(1) (1993) 194–203.