PROCEEDINGS

IEEE International Conference on Robotics and Automation

Sponsored by
IEEE Robotics and Automation Society

May 2 – 6, 1993
Atlanta, Georgia

Volume 2: May 4, 1993

IEEE Computer Society Press
Los Alamitos, California

Washington • Brussels • Tokyo

Proceedings of the
IEEE International Conference on Robotics and Automation, Volume 2
Atlanta, GA (May 1993).

Layered Dynamic Fault Detection and Tolerance for Robots

M. L. Visinsky, I. D. Walker, and J. R. Cavallaro
Department of Electrical & Computer Engineering
Rice University, Houston, TX 77251-1892

Abstract

Fault tolerance is increasingly important for space and hazardous environment robotics. Robots need to quickly detect and tolerate internal failures in order to continue performing their tasks without the need for immediate human intervention. Using analytical redundancy, this paper derives tests with which the robot can detect failures. The paper also develops a layered intelligent control framework containing these new sensor-based fault detection and tolerance schemes. The servo, interface, and supervisor layers form a hierarchy of fault tolerance which provide different levels of detection and tolerance capabilities for structurally diverse robots.

1 Introduction

Robots are extremely desirable for work in hazardous environments and for the exploration of space. However, the dangerous and long distance nature of such missions can place robots far from any local human intervention. The isolation of these missions can cause long enough delays between the robot action and the operator notification of the action to allow potentially tolerable faults to blossom into system-wide malfunctions, forcing the mission to abort. Failures within the robot cannot be repaired frequently without reinstating the risk to human life and the expense of sending humans to the robot. The robot needs to independently detect and isolate internal failures and attempt to utilize its remaining capabilities to overcome the limitations imposed by the failures. With fault tolerance, the robot can continue its assigned tasks without jeopardizing the mission or damaging the working environment.

We have designed several robot fault tolerant algorithms which detect and tolerate motor and internal sensor failures in the robot by utilizing the existing robot structure [16, 17]. Kinematically redundant robots, such as the RRC or NASA proposed SPDM, or robots with multiple sensors per joint can use these algorithms to obtain more extensive failure detection and tolerance capabilities. The algorithms can also provide basic detection capabilities for robots, such as a 6 DOF Puma, which have little or no redundancy with which to tolerate such failures.

In order to modularize these fault detection and fault tolerance algorithms and enable the algorithms to more easily adapt to a wide variety of robot structures, this paper develops a multilayer intelligent control framework (see Figure 1) similar to the hybrid dynamical system framework proposed in [6]. Fault tolerant control of a robot system can be initially divided into three layers: a continuous servo layer, an interface monitor layer, and a discrete supervisor layer. The servo layer consists of the normal robot controller and the robot. The interface layer contains the planner and provides fault detection and some basic fault tolerance for the system. The supervisor layer provides higher level fault tolerance and general action commands for the robot.

2 Servo Layer

The servo layer of the proposed framework includes the robot and the robot control computer running some common type of control algorithm. A typical PD "computed-torque" controller takes the form:

$$\underline{\tau} = [\hat{M}(\underline{\theta})]\{\underline{\ddot{\theta}}_d + [K_P](\underline{\theta}_d - \underline{\theta}) + [K_D](\underline{\dot{\theta}}_d - \underline{\dot{\theta}})\} + \underline{\hat{N}}(\underline{\theta}, \underline{\dot{\theta}}). \quad (1)$$

where $\underline{\tau}$ is the joint torque vector. The robot controller uses sensed estimates of joint positions, $\underline{\theta}$, to calculate the estimated inertial, via $[\hat{M}]$, and Coriolis and centrifugal, $\underline{\hat{N}}$, effects. $\underline{\theta}_d$ is the desired trajectory and $[K_P]$ and $[K_D]$ are selected diagonal gain matrices. If a sensor is damaged and the failure remains undetected, the controller will receive incorrect information about the joint and the resulting $[\hat{M}]$ and $\underline{\hat{N}}$ will be significantly in error.

The robot controller computes the necessary torque to apply to each motor in order to move the robot from the given current position to the next desired position. Information about the current position or velocity of the robot joints is relayed to the controller from the

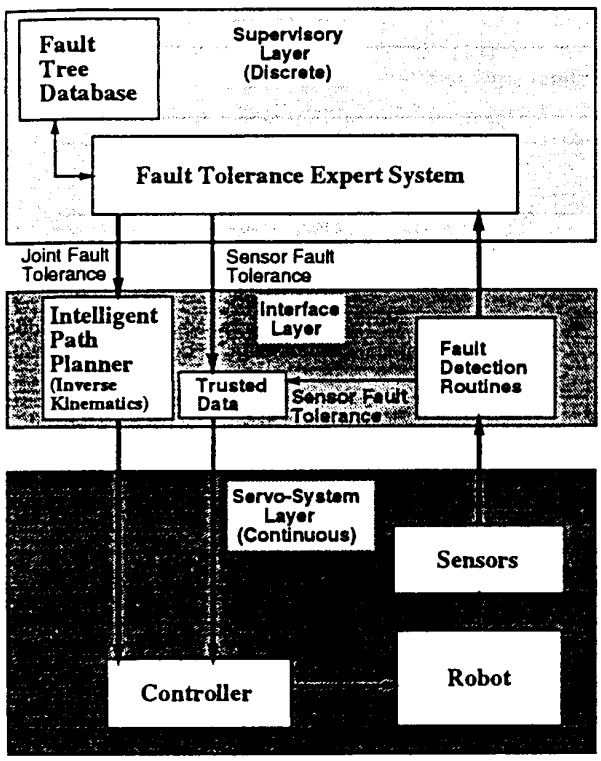

Figure 1: Fault Tolerant Robot System.

robot's internal sensors. The controller also uses the sensors to determine the deviation from desired joint positions, velocities, or forces in order to compensate for these errors using feedback control.

The well-being of the sensors is, therefore, critical for the robot in completing its task. A sensor failure will immediately jeopardize the control of the robot. 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 [15]. We concentrate on the frozen failure mode for this paper with suggestions on appropriate modifications for the remaining modes.

The ability to lock a motor in the event of a failure is also important in supporting the fault tolerance schemes of Section 3. A failed joint which swings freely pulls at the other joints and draws them off course. 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, locking may be implemented as a direct command from the interface to the robot. The paths of surviving joints are modified to move the end effector to its goal (Section 3.3).

3 Interface Layer

The interface layer contains the robot planner and fault detection routines. Based on the end effector target point passed down from the supervisor, the planner computes the next configuration for each joint in order to reach the desired goal in the given amount of time. The new configuration is then passed on to the robot controller in the servo layer. Fault detection routines continually monitor the state of the robot in order to catch any failures in the system. The routines are unable to differentiate all failures because only the effect of the failures can be gleaned through the robot's sensors [16]. The typically small number of sensors further limits the ability of the detection routines to quickly isolate failures within the system.

3.1 Analytical Redundancy

To provide the interface with detection/isolation abilities, we have developed tests which use available data to monitor for critical sensor or motor failures. It is important to identify as many useful independent tests as possible while eliminating redundant checks in order to improve response accuracy and speed. There are established results for the generation of such detection tests or residuals for general dynamic systems using the concept of analytical redundancy [14]. This section develops the results for robotics along the lines of Chow and Willsky's [1] mathematical characterization of analytical redundancy for dynamic systems modeled as:

$$\underline{x}(k+1) = [A]\underline{x}(k) + \sum_{i=1}^{q} b_i u_i(k), \qquad (2)$$

$$y_j(k) = c_j \underline{x}(k), \quad j = 1, \cdots, m. \qquad (3)$$

The vector $\underline{x} \in \Re^N$ is the state vector. Of the constants, $[A] \in \Re^{N \times N}$, $b_i \in \Re^N$, and $c_j^T \in \Re^N$. The scalar u_i is the known input to the i^{th} actuator and y_j is the scalar output of the j^{th} sensor.

In the following analysis, the number of sensors per joint is assumed to be two ($m = 2$) and there are two states (N = 2, position and velocity) per joint. This assumption will enable us to show the improved isolation capabilities over the intuitive base case while still maintaining a realistic structure for the robot. The results can easily be extended to the case of m sensors in general [16]. The sensors are chosen to be an encoder and a tachometer which enables us to examine the structure in [1] for temporal redundancy between two different types of sensors. For each joint, the number of actuators, q, is 1. To simplify the derivation, we assume that the sensors read the exact values for the position and speed of the robot and

that the linearization performed by the robot computed torque controller to eliminate the time varying and non-linear aspects of robot dynamics is reasonably accurate. We thus work with the linearized system whose input is given by the modified acceleration $u(k) = \ddot{\theta}_d(k) + [K_P](\theta_d(k) - \theta(k)) + [K_D](\dot{\theta}_d(k) - \dot{\theta}(k))$ where θ is the angular position of each joint and Δt is the length of time between each iteration k. We note that $\underline{x}(k) = [\theta(k)\dot{\theta}(k)]^T$ and thus move $\theta(k)$ and $\dot{\theta}(k)$ out of $u(k)$ and combine them with $\underline{x}(k)$ in equation (2) above. The actuator input $u(k)$ now becomes $u_d(k) = \ddot{\theta}_d(k) + [K_P]\theta_d(k) + [K_D]\dot{\theta}_d(k)$ which is only a function of desired values. This modifies the results we obtained in the more detailed original derivation found in [16]. The resulting coefficient matrices in equations (2) and (3) are:

$$A = \begin{bmatrix} 1 & (\Delta t) \\ -K_p(\Delta t) & 1 - K_d(\Delta t) \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ (\Delta t) \end{bmatrix},$$

$$c_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}, \text{ and } c_2 = \begin{bmatrix} 0 & 1 \end{bmatrix},$$

where K_p and K_d are the diagonal entries of $[K_P]$ and $[K_D]$ corresponding to the current joint.

With $C_j(k) = [c_j, c_jA, \cdots, c_jA^k]^T$, $(j = 1, 2; k = 0, 1, \ldots)$, the observable subspaces [1] for the encoder $(j = 1)$ and the tachometer $(j = 2)$ are spanned by the rows of:

$$C_1(n_1) = \begin{bmatrix} 1 & 0 \\ 1 & \Delta t \\ 1 - K_p\Delta t^2 & 2\Delta t - K_d\Delta t^2 \end{bmatrix}, \quad (4)$$

$$\text{and } C_2(n_2) = \begin{bmatrix} 0 & 1 \\ -K_p\Delta t & 1 - K_d\Delta t \\ c_{31} & c_{32} \end{bmatrix}, \quad (5)$$

where

$c_{31} = K_p(K_d\Delta t^2 - 2\Delta t)$ and
$c_{32} = 1 + (K_d{}^2 - K_p)\Delta t^2 - 2K_d\Delta t.$

The ranks of the arrays are $n_1 = 2$ and $n_2 = 2$, respectively. These ranks imply that no new information can be gained after observing either the encoder or the tachometer for two time-steps [1]. The feedback data provides additional information about the tachometer. We thus get four linearly independent ω's (rows in Ω) which combine the observable data and provide tests for the fault-free case by satisfying:

$$[\Omega]_{4\times 6} \begin{bmatrix} C_1(n_1) \\ C_2(n_2) \end{bmatrix} \underline{x}(k) = 0. \quad (6)$$

One representation of the 4×6 Ω matrix is:

$$\Omega = \begin{bmatrix} 1 & -1 & 0 & (\Delta t) & 0 & 0 \\ K_p\Delta t^2 - 1 & 0 & 1 & K_d\Delta t^2 - 2\Delta t & 0 & 0 \\ K_p\Delta t & 0 & 0 & K_d\Delta t - 1 & 1 & 0 \\ -c_{31} & 0 & 0 & -c_{32} & 0 & 1 \end{bmatrix} \quad (7)$$

Each row of Ω physically represents various comparisons among the encoder and tachometer readings. When Ω is multiplied by the appropriate histories of $\underline{x}$ $([\theta(k)\theta(k+1)\theta(k+2)\dot{\theta}(k)\dot{\theta}(k+1)\theta(k+2)]^T)$ in equation (6), the result should be zero for the fault free case. The sensor inputs $(\underline{x})$ or actual robot positions, however, are not observable by fault detection routines. The detection routines only see the sensor outputs $\underline{y}$ (see Figure 1). We can obtain $\underline{x}$ in terms of the known $\underline{y}$, b, and $u(k)$. The result, multiplied by Ω, is the parity vector (equation (8)) which now defines the observable relationships for fault detection in terms of obtainable values. The general definition of the parity vector, including the modified actuator inputs u_d, is:

$$P(k) = \Omega\left\{\begin{bmatrix} Y_1(k, n_1) \\ \vdots \\ Y_m(k, n_m) \end{bmatrix} - \begin{bmatrix} B_1(n_1) \\ \vdots \\ B_m(n_m) \end{bmatrix} U_d(k, n_0)\right\}, \quad (8)$$

where $n_0 = \max(n_1, n_2)$ and $Y_i(k, n_i)$ and $B_i(n_i)$ are as defined in [1, 16]. In the fault-free case, $P(k) = 0$. With simple noise (due to inexact linearization in the controller, for example), $P(k)$ becomes a random vector with zero mean [1]. With noise and failures, $P(k)$ will be biased away from zero indicating a failure. For this analysis, $P(k)$ is defined by the following matrices.

$$Y_1(k, n_1) = \begin{bmatrix} \theta(k) \\ \theta(k+1) \\ \theta(k+2) \end{bmatrix}, \; Y_2(k, n_2) = \begin{bmatrix} \dot{\theta}(k) \\ \dot{\theta}(k+1) \\ \dot{\theta}(k+2) \end{bmatrix},$$

$$B_1(n_1) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \Delta t^2 & 0 \end{bmatrix}, \; B_2(n_2) = \begin{bmatrix} 0 & 0 \\ \Delta t & 0 \\ \Delta t(1 - K_d) & \Delta t \end{bmatrix},$$

$$\text{and } U_d(k, n_0) = \begin{bmatrix} u_d(k) \\ u_d(k+1) \end{bmatrix}.$$

Substituting these matrices into (8) with $P(k) = 0$ and performing various algebraic manipulations and substitutions, the following tests are found:

1. Encoder Reading vs. Tachometer Reading: $\frac{\theta(k+1) - \theta(k)}{\Delta t} = \dot{\theta}(k)$.
2. Encoder Reading vs. Computed Acceleration: $\frac{(\theta(k+2) - \theta(k+1)) - (\theta(k+1) - \theta(k))}{\Delta t^2} = u(k)$,
3. Tachometer Reading vs. Computed Acceleration: $\frac{\dot{\theta}(k+1) - \dot{\theta}}{\Delta t} = u(k)$,
4. Tachometer Reading vs. Computed Jerk: $\dddot{\theta} = \frac{(u(k+1) - u(k))}{\Delta t}$,

where

$u(k) = u_d(k) - K_p\theta(k) - K_d\dot{\theta}(k)$,
$u(k+1) = u_d(k+1) - K_p\theta(k+1) - K_d\dot{\theta}(k+1)$,
and $\dddot{\theta} = \frac{(\dot{\theta}(k+2) - \dot{\theta}(k+1)) - (\dot{\theta}(k+1) - \dot{\theta}(k))}{\Delta t^2}$.

$\dddot{\theta}$ (second derivative of the tachometer value) is compared with the jerk or first derivative of acceleration.

The mathematical analysis of [1] focuses on sensors to provide useful detection checks for failures. However, the amount of uncertainty about the system dynamics present in most robotic control systems makes several of the proposed tests for general applications difficult to use directly in robotics [9]. Numerical differentiation of sensor readings which contain noise is problematic, making the tests impractical in their pure form. An acceptable bound or threshold for the difference between the desired value and the sensor reading must be chosen [11, 16] to mask out the inaccuracies of the sensors and the robot models. In our work, we have successfully utilized threshold-modified versions of tests 1 and 3 above. Model-based fault diagnosis schemes [5] which monitor the model parameters as physical characteristics of the system assume that the robot model is well known.

To allow for different failure modes such as runaway or free spinning, an additional check can be included which relies on the sensor with a reading closer to the desired value if a sensor failure is indicated by another test. This check chooses correctly if the failed sensor spins away from the desired value while the working sensor continues to track the desired path. If the failed sensor spins closer to the desired value, however, the routine could select it as the survivor and the system would start relying on incorrect data, spawning false alarms in other joints. The routines shut down more of the system than necessary but prevent the robot from damaging itself or the environment.

3.2 Interface States

The state diagram in Figure 2 illustrates the flow of fault detection in the interface for one joint as failures are detected in that joint. Signals prefaced by an "S_" originate in the supervisor layer. Signals which are sent by the interface state machine described below are prefaced with an "I_". Transition conditions without a preface are local to the interface algorithms. Many of the signals, such as S_ReadiFromSnsr or I_LogSnsrFail, are also accompanied by information to indicate the appropriate component or variable to modify. In the initial state, the interface is idle and awaiting a start command from the supervisor. When an interface state machine receives the S_ReadiFromSnsr command for its specific joint from the supervisor, it moves into the PROCESS state and scans the data structure accompanying the signal to see which of the controller data variables (θ, $\dot{\theta}$, $\ddot{\theta}$) are to be read from which joint sensors for joint i. The interface also determines which detection checks to use for each type of sensor indicated.

Once this initialization is complete, the state moves into the appropriate JOINT-WELL state and monitors the health of the m sensors currently in use. When a detection test is violated, the interface removes that sensor from the current working set for the joint and reorganizes the sources of information to obtain the required controller data from the surviving sensors. The interface also signals the supervisor that a sensor has failed so that the supervisor can log the failure. The interface then moves into the next JOINT-WELL state in the chain and monitors the remaining $m-1$ sensors.

When all of the joint's sensors have failed, the interface signals the supervisor that all sensors are faulty with the I_AllSnsrFail signal and transitions into a WAIT state. The interface may also transition into the WAIT state if all the sensors have drifted far enough off course to indicate a possible motor failure. In this situation, the interface signals the supervisor with an I_MtrFail signal and moves from whichever JOINT-WELL state it is currently in directly to the WAIT state. The interface now waits for direction from the supervisor. If there are any external sensors or backup internal sensors which can be used to obtain the position and velocity information for this joint, the supervisor signals the interface using the S_ReadiFromSnsr command which has the appropriate directions within the accompanying structure. If there are no more available sensors, the interface receives an S_ShutDowniMtr command which tells it to shut down the motor in joint i using the I_ShutDowniMtr signal. The state now becomes JOINT-FAILED.

The algorithm remains in the JOINT-FAILED state until the task is completed or aborted or the motor status is reset, at which point it moves back to the IDLE state. Permanent failures can only be reset by the operator. If temporary failures are supported, the algorithm resets the status when a component passes an appropriate reset test.

3.3 Tolerance of Joint Failures

To keep the robot on the desired path in the presence of joint failures, the interface-level planner must be alerted when a joint has failed. A kinematically redundant robot can withstand joint failures up to the degree of kinematic redundancy without limiting the end effector range of motion, although the dexterity may decrease [12]. In a failure situation and in order to maintain the desired trajectory $\underline{v}_n = [J_n(\underline{\theta}_n)]\dot{\underline{\theta}}_n$ ($J_n \in \Re^{l \times n}, \dot{\underline{\theta}}_n \in \Re^n$), the interface planner replaces the n

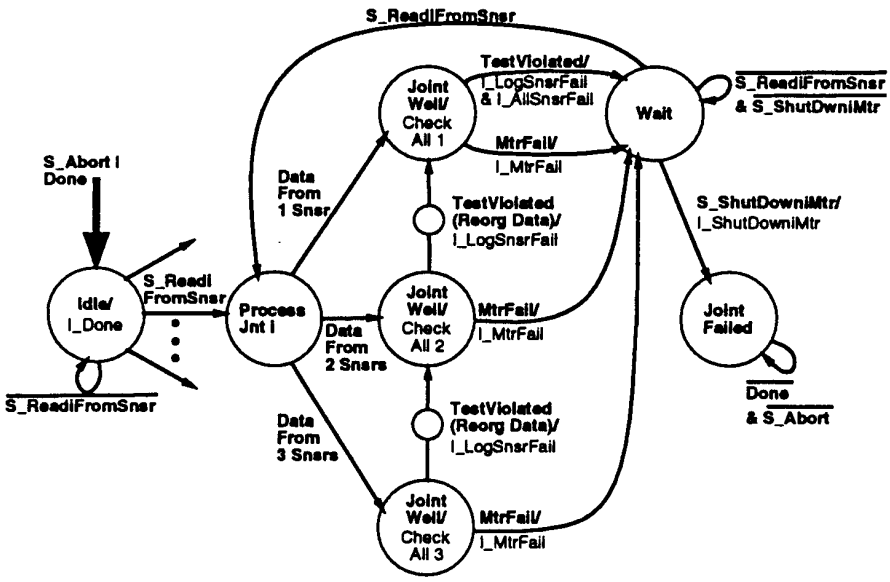

Figure 2: State Diagram for Interface.

joint solution $\dot{\underline{\theta}}_n = [J_n(\underline{\theta}_n)]^+ \underline{v}_n$ $(J^+ = J^T(JJ^T)^{-1})$ by $\dot{\underline{\theta}}_{n-k} = [J_{n-k}(\underline{\theta}_{n-k})]^+ \underline{v}_{n-k}$, with the appropriate k column(s) of $[J_n]$ removed and the locked position(s) of the k failed joint(s) used in the Jacobian calculations. This approach plans end effector motion $\underline{v}$ using the available (reduced) joint set $\underline{\theta}_{n-k}$, $(n-k) \geq l$ [10].

4 Supervisor Layer

The supervisor layer receives signals from each interface layer state machine and works to monitor and consolidate the joint sensor and motor failure information to provide smooth fault tolerance for the whole robot system. Figure 3 shows the generalized state diagram for the supervisor layer of the framework. All states but the IDLE state represent meta-states with several procedures being performed within each state. Again, signals prefaced by an "S_" or an "I_" originate in the supervisor or interface, respectively. The interface signals contain data on which joint state machine has originated the signal. Signals prefaced by an "O_" are from the operator. Transition conditions without a preface are local to the supervisor.

The supervisor initially receives from the operator a target position for the end effector and a list of which internal sensors to use for monitoring each joint as well as the O_Start signal. The supervisor transitions on the signal from the IDLE state to the PROCESS state where it digests the command data, sends the starting velocities to the interface-level planner, and signals the interface with the appropriate set of S_ReadiFromSnsr commands to start monitoring the task. The supervisor then waits in this state until a signal is received from an interface state machine or the operator signals an abort. The supervisor handles the signals from each joint state machine on a first-come, first-serve basis. The signals are recognized only when the supervisor is in the PROCESS state and are cued up if the supervisor is in another active state. The supervisor moves back into the IDLE state from the PROCESS state if the interface signals that the task has been completed with the I_Done signal.

When the supervisor receives an I_LogSnsrFail signal from a specific joint, it accesses the accompanying data structure and records the failure in a database [18]. The supervisor alerts the operator to the failure. The operator may signal an abort at any time if it is deemed that the mission has been compromised. The supervisor does not determine an alternate component for the joint until an I_AllSnsrFail signal is also received from the interface. The supervisor will then transition to the CHECK-FOR-SENSORS state and use information learned by traversing the database to develop new tolerance strategies based on the existing structure of the robot. If there are alternate sensors available or external sensors which can be used to monitor a joint, the supervisor signals the interface with the S_ReadiFromSnsr command and corresponding data. The supervisor then transitions back into the PROCESS state. If there are no available re-

placements, the supervisor transitions to the CHECK-FOR-MOTORS state.

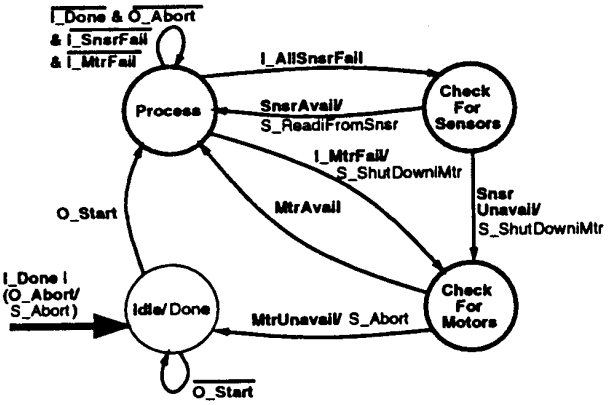

Figure 3: State Diagram for Supervisor.

The supervisor also transitions to the CHECK-FOR-MOTORS state in response to an I_MtrFail signal. The supervisor again modifies its records and alerts the operator of the failure and now checks to ensure that the robot can continue working with one less joint. If the robot is kinematically redundant and has not yet dropped down to the base number of joints for the robot, the supervisor moves back to the PROCESS state and sends an S_ShutDowniMtr command to the interface. In the presence of failed joints, the supervisor command S_ShutDowniMtr signals the interface-level planner to keep the robot on the desired course using the techniques of Section 3.3. If the robot cannot complete its mission because the number of joints is no longer kinematically redundant, the supervisor signals the interface to shut down the entire robot with an S_Abort command. The supervisor also alerts the operator and transitions into the IDLE state to await further instruction from the operator. At this stage the operator would need to repair the robot or alter the task required before restarting the robot. The component failure condition tables would need to be reset when a component is repaired or replaced.

The supervisor provides higher levels of intelligence to the basic tolerance of sensor and motor failures provided by the interface fault detection algorithms. To provide wider ranges of survivor choices for the supervisor or interface, the redundancy of the system can be increased using work such as that of Sreevijayan, et al [13] and Wu, et al [20] on redundantly actuated drives. The supervisor can be further enhanced to monitor the reachable workspace of the robot throughout its operation [12, 13]. When motor failures occur and a joint is shut down, the workspace shrinks due to the immobility of the failed joint. While it waits for a signal in the PROCESS state, the supervisor can spawn a process to determine the area reachable by the end effector with the robot in its current state. Using the reported changes to the reachable workspace [12], the supervisor 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 in the IDLE state for operator directions.

Fault trees [16, 19], which show the flow of failures through a robot system, provide the supervisor with a map of failure interactions. Given a failure, the supervisor traverses the appropriate subtree to accumulate a list of components which may have caused the failure. The list is sent to an operator for future inspections and repairs. When pruning the database trees of failed components, the supervisor also searches for interactions which may indicate that another component is likely to fail because it is linked either structurally or functionally to the failed component.

We have implemented the supervisor structure described above in a CLIPS [7] framework. CLIPS is a NASA-developed expert system software package. Fault tree traversing schemes provide rules for the expert system supervisor layer [18]. Maciejewski's work on kinematic fault tolerance [10] can be integrated into the fault tree database to provide the supervisor with information on new optimal configurations for the robot in the presence of failures. If the fault trees are pruned due to detected failures, a new optimal configuration could be drawn up to provide a more dynamic preventive maintenance measure. The supervisor also keeps track of the changing probability of failure for various subsystems based on how the failure reports from the interface layer modify the database tree structures. The operator can get run-time reports on these dynamic failure rates.

5 Layer Interaction

We have introduced two new layers to add fault detection and tolerance to normal robot control. The supervisor monitors tasks, responds to operator commands, and controls the lower level fault detection and tolerance. The interface monitors the continuous data from the sensors, checks for failure conditions, and returns discrete event or failure information to the supervisor. The interface layer isolates the supervisor layer and the operator from the continuous control algorithms of the servo layer plant.

As an example of the interaction of this three layer framework, let us assume that the supervisor wants

the interface layer to start a task using as controller feedback data only the values derived from the encoder reading at each joint. At some point in the run, the encoder for Joint 2 fails and the test monitoring this sensor is violated. The interface sends an I_AllSnsrFail signal along with the I_LogSnsrFail signal which tells the supervisor that encoder 2 has failed and the interface knows of no more sensors to use. Figure 4 shows the supervisor subsequently removing the encoder for Joint 2 from a fault tree database maintained by the expert system (detailed in [18]). The database reveals that a tachometer is still available to functionally replace the encoder as they are connected by an AND gate and the failure does not move any farther up the tree. The supervisor can therefore command the interface to read the information from this alternate sensor as long as the operator does not request an abort.

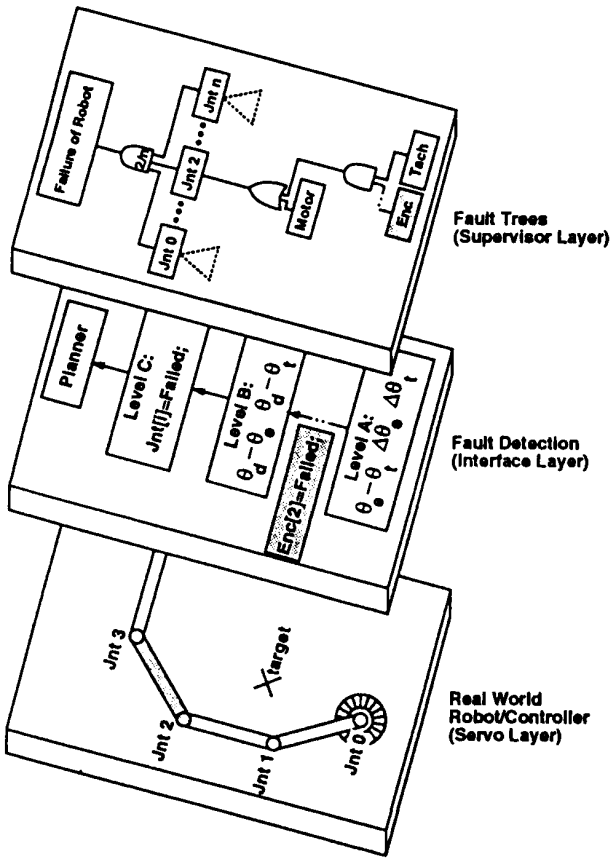

Figure 4: Layer Propagation of a Sensor Failure.

6 Implementation

To demonstrate the fault tolerance framework, we used an extension of a simulated four link robot originally developed by Hamilton [8] and executed on a Silicon Graphics Personal Iris computer. The simulator was enhanced to model two sensors (encoder and tachometer) per joint and to include fault instigation routines to test the detection algorithms [16]. The simulator has recently been expanded to model a sensor-enhanced 6 DOF Puma robot. A modified 7 DOF Puma will also be modeled in future work.

In the framework of Figure 1, the sensor estimates of the angles and velocities are passed to the interface layer (Section 3) which checks for failures using the methods of [16]. This layer either passes to the servo layer controller what it considers to be good estimates of the position, velocity, and acceleration $(\theta, \dot{\theta}, \ddot{\theta})$ or instigates the appropriate restructuring strategy based on the available sensor set or instruction from the supervisor layer (Sections 3 and 4). A planner in the interface produces the desired trajectory through a velocity level inverse kinematics routine [2].

The servo-level robot takes the computed torque from the controller and determines the resulting robot angle accelerations based on the robot dynamics. The robot's position, velocity, and acceleration are then sent to the sensor routines. The robot position is also sent to the NASA developed TDM graphics simulator which displays the motion on the screen [16] as well as color-coded graphical and textual information on the status of fault detection in the robot. A joint (and its associated link) will change color when a failure is detected within the joint. The delay between the instigation of a failure (a flag appears) and the detection of the failure (when the joint changes color) can be readily observed.

7 Conclusions and Ongoing Work

The space program and other hazardous environment programs have increased the need for reliable robots which can withstand system failures without requiring immediate repairs [3, 4, 11]. The main objectives of this paper have been to develop a multi-layered control framework which provides fault tolerance for a variety of robots and to introduce new sensor-based fault tolerance techniques for robotics using the concept of analytical redundancy. The emphasis is on a modular system that will be able to use the existing structure of a robot as the basis for fault detection and tolerance, without requiring specific types or numbers of joints, motors, or other components.

The framework layers work together to tolerate as many sensor and motor failures within the robot as possible without compromising the goal of the assigned task. An operator directs the initial formation and conditions for a given task and expects reports on internal failures. The framework can be embedded in an expert system or used as a simple algorithmic program. This framework can be used to provide fault tolerance to the wide variety of robots which have already been developed to perform tasks for the space program and which may be far from human control for long periods of time.

Acknowledgements

Supported in part by the National Science Foundation under grants MIP-8909498 and MSS-9024391, by DOE Sandia National Laboratory Contract #18-4379A and by DDM 9202639. This work was further supported by a Mitre Corporation Graduate Fellowship and NSF Graduate Fellowship RCD-9154692.

References

[1] E. Y. Chow and A. S. Willsky. Analytical Redundancy and the Design of Robust Failure Detection Systems. *IEEE Transactions on Automatic Control*, AC-29(7):603–614, July 1984.

[2] A. S. Deo and I. D. Walker. Robot Subtask Performance with Singularity Robustness using Optimal Damped Least Squares. In *1992 IEEE International Conference on Robotics and Automation*, pages 434–441, Nice, France, May 1992.

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

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

[5] B. Freyermuth. An Approach to Model Based Fault Diagnosis of Industrial Robots. In *1991 IEEE International Conference on Robotics and Automation*, pages 1350–1356, Sacremento, CA, April 1991.

[6] A. Göllü and P. Varaiya. Hybrid Dynamical Systems. In *IEEE Conference on Decision and Control*, pages 2708–2712, 1989.

[7] J. Giarratano and G. Riley. *Expert Systems: Principles and Programming.* PWS-Kent Publishing Co., Boston, MA, 1989.

[8] D.L. Hamilton, J.K. Bennett, and I.D. Walker. Parallel Fault-Tolerant Robot Control. In *1992 SPIE Conference on Cooperative Intelligent Robotics in Space III*, pages 251–261, Boston, MA, November 1992.

[9] D. T. Horak. Failure Detection in Dynamic Systems with Modeling Errors. *Journal of Guidance, Control, and Dynamics*, 11(6):508–516, November-December 1988.

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

[11] M. M. Moya and W. M. Davidson. Sensor-Driven, Fault-Tolerant Control of a Maintenance Robot. In *1986 IEEE International Conference on Robotics and Automation*, pages 428–434, Feb 1986.

[12] A. K. Pradeep, P. J. Yoder, R. Mukundan, and R. J. Schilling. Crippled Motion in Robots. *IEEE Transactions on Aerospace and Electronic Systems*, 24(1):2–13, January 1988.

[13] D. Sreevijayan and D. Tesar. On The Design of Fault Tolerant Robotic Manipulator Systems. Master's thesis, Mechanical Engineering Department, University of Texas, Austin, TX, June 1992.

[14] R. F. Stengel. Intelligent Failure-Tolerant Control. *IEEE Control Systems Magazine*, 11(4):14–23, June 1991.

[15] F. Tangorra, A. D. Montgomery, and R. F. Grasmeder. Independent Orbiter Assessment Analysis of the Remote Manipulator System. NASA WORKING PAPERS NAS 1.26:185585 and NAS 1.26:185592, NASA STS Orbiter Projects Office, January-February 1987-88.

[16] M. L. Visinsky. Fault Detection and Fault Tolerance Methods for Robotics. Master's thesis, Department of Electrical and Computer Engineering, Rice University, Houston, TX, December 1991.

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

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

[19] M. L. Visinsky, I. D. Walker, and J. R. Cavallaro. Chapter 8, Robotic Fault Tolerance: Algorithms and Architectures. In *Robotics and Remote Systems for Hazardous Environments, Volume 1.* Prentice Hall Publishing Co., 1993. In Press.

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