Failure Recognition and Fault Tolerance of an Autonomous Robot

Cynthia Ferrell *
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
545 Technology Square, Room 741
Cambridge, MA 02139
(617)253-0997
fax: (617)253-0039
ferrell@ai.mit.edu

*Support for this research was provided in part by a NASA Graduate Student Researcher Program Fellowship administered through the Jet Propulsion Laboratory, by Jet Propulsion Laboratory grant 959333, and by the Advanced Research Projects Agency under Office of Naval Research contract N00014–91–J–4038.

Abstract

The purpose of this paper is twofold. The first purpose is to present important issues in designing fault tolerant systems for autonomous robots. The second, is to present the fault tolerance capabilities we implemented on our autonomous robot. Our approach is characterized by a distributed network of concurrently running processes. To tolerate hardware failures, a set of fault tolerance processes are written for each component. These processes are responsible for detecting faults in their respective component, and minimizing the impact of the failure on the robot's performance. By exploiting concurrency and distributedness, the system monitors, detects, and compensates for component failures silmultaneously. The capabilities of this system have been tested by physically disabling and enabling the robot's sensors and actuators. The system quickly recognizes and compensates for both minor and severe sensor and actuator failures. It tolerates a variety of sensor failures such as decalibration, erroneous readings, and permanent failures. It also tolerates various combinations of failures such as individual failures, concurrent failures, and accumulative failures. We hope this work will inspire further research in fault tolerant autonomy.

keywords: fault tolerance, failure recognition, autonomous robot, distributed control, legged locomotion, adaptive sensing.

shortened title: Fault Tolerant Autonomy

1 Introduction

Fault tolerant behavior is important for autonomous mobile robots whether their task is grand or mundane. Building autonomous mobile robots capable of performing tasks in environments that are either too dangerous or unsuitable for humans has been a long term goal of the field. For example, research is underway to develop autonomous systems capable of exploring planets (Angle & Brooks 1990), (Bares & Whittaker 1990) or oceans (Payton, Keirsey, Kimple, Krozel & Rosenblatt 1992), (Stuart 1988). These tasks require that the robot perform its task for long periods of time without the luxury of repair when components fail. It is vital to the success of the mission that the robot continue to function effectively despite component failures. For day to day goals, anyone who works with real autonomous robots is familiar with how frequently their hardware fails. Sensor and actuator failures are not surprising given how often the robots bump into things, rattle components, snag wires, and stress connectors as they move through their environment. It would be nice if we did not have to stop our research to repair the robot every time something fails.

2 Hannibal

Hannibal, as shown in figure 1, is a small autonomous robot. It was designed and built under the supervision of Prof. Rodney Brooks in the Artificial Intelligence lab at MIT (Angle & Brooks 1990). The robot is perhaps the most sophisticated and complex robot for its size. It measures 35 cm long, stands 15 cm high and weighs 2.8 kg. It has six 3 degree of freedom (DOF) legs. Despite its small size, Hannibal currently has 19 actuators and over 60 sensors of 5 different types all connected via a local network to 8 on-board computers. The robot's

control organization is behavior-based (Brooks 1986). It is programmed in the Behavior Language, a front end for the Subsumption Architecture (Brooks 1990).

The following types of sensors are mounted on Hannibal:

- Leg mounted force sensors: these are foil strain gauges that can be used to measure loads on the leg and to detect leg collisions. There is a set of strain gauges for each DOF of the legs.
- Joint angle sensors: These are potentiometers that measure the joint angle for each DOF of the leg.
- Joint velocity sensors: The joint angle sensors are differentiated in analog for each DOF of the leg.
- Ground contact sensor: This is a linear potentiometer mounted on the ankle that measures the deflection of the foot as it presses against the ground.
- Inclinometer: This sensing unit is made up of a +/- 45 degree roll sensor and a 360 degree pitch sensor.

3 The Issues

For Hannibal, having many sensors and actuators is a double edged sword. Multiple sensors provide more reliable sensing and a richer view of the world. More actuators provides more degrees of freedom. However, more components also means there is more that can fail and subsequently degrade performance. Physical failures can be attributed to either mechanical failure, electronic failure, or sensor failure. Subtle changes in the state of the robot such

as sensor signal drift also degrades performance. Hannibal's task is to locomote in rough and hazardous environments. As it scrambles along, it can be subjected to repeated bumps, snags, and stresses. This places significant wear and tear on Hannibal's hardware. Not surprisingly, components fail or decalibrate over time. These problems require that the fault tolerant network address the following issues:

- *Fast response time to failures*: Hannibal operates in a hazardous environment. Consequently, Hannibal must detect and remedy failures quickly or else its safety may be jeopardized. This means failures must be detected and compensated for *before* system performance degrades to an unacceptable level.

- *Graceful degradation of performance*: Hannibal's performance must degrade gracefully as failures accumulate. This requires Hannibal to maintain the highest level of performance possible given the functional state of the hardware.

- *Access to all reliable resources*: More sensors and actuators enhance system performance provided they are functional. Hence, Hannibal should reincorporate the use of repaired components.

- *Fault coverage*: The robot can suffer from a variety of failures. Failures can be permanent or transient. Some failures have a local effect while others have a global effect on performance. Sensors may decalibrate. Furthermore, the robot should be able to recover from different combinations of failures. Failures can occur individually, concurrently with other failures, or accumulate over time.

4 Confinement of Errors

Sensor and actuator faults affect various levels of Hannibal's system hierarchy. The lowest level is the hardware level, the next level is low level control where sensor information is processed, and the top level is high level control which determines the robot's behavior. Clearly sensor or actuator failures affect the hardware level of the system. Sensor failures affect low level control because they affect the reliability of the *virtual sensors* within this level. The virtual sensors are processes that are responsible for characterizing the robot's interaction with its environment using sensor information. Sensor failures may cause the virtual sensor results to be incorrect. If this is indeed the case, then high level control is also affected by sensor failures. The high level control uses the results from the virtual sensors to activate the correct behavior at the appropriate time. Thus, if the results of the virtual sensors are incorrect, then the wrong behaviors will be activated.

It is important to detect and confine errors to the lowest possible level in which they occur. If an error is not confined to the level in which it originated, then higher levels must detect and compensate for the fault. As an error propagates up the levels of the system hierarchy, it affects increasing amounts of system state. Longer response times to error correction means the error manisfestations become more diverse. Hence, detecting and confining errors to the lowest possible level of the system hierarchy maximizes the effectiveness of the recovery procedures and minimizes the impact of the error on system performance.

5 Levels of Fault Tolerance

Given that hardware failures affect various levels of the system, fault tolerance techniques can be implemented at each level. Below, we present possible fault tolerance strategies for each level, and describe the merits and shortcomings of each method.

5.1 Replication of Hardware

We are concerned with sensor and actuator failures, so reliable sensing and actuating hardware is desirable. Replication of hardware is commonly used to enhance hardware reliability. Several commercial systems, such as Gray (1990), Siewiorek & Johnson (1981), use this approach to achieve robustness. Replication of processors has been used to achieve fault tolerance in a few robotics applications ((Lin & Lee 1991) and (Kabuka, Harjadi & Younis 1990)). In Hannibal's case, this approach would correspond to replicating sensors and actuators. For a sensing example, multiple potentiometers could be used to sense a particular joint angle. An aribitor could gather the joint angle values of each potentiometer and accept the most common joint angle value as correct. With this approach, hardware failures would be detected and confined to the hardware level of the system hierarchy.

Ideally we could detect, confine, and correct hardware errors at the hardware level. By doing so, the application software need not be alerted to these failures. However, this approach has some drawbacks. First, replication of sensors and actuators is expensive. Second, it assumes that a majority of replicated components are working. Ideally, Hannibal should perform reliably with a minority of functional sensors. Third, Hannibal's size and weight contraints restrict how may sensors and actuators can be mounted on the robot.

Therefore it is impractical to enhance sensing or actuating capabilities on Hannibal by using this approach. However, Hannibal was designed with multiple sensors and actuators that provide complementary capabilites. For example, Hannibal can use several different sensors to detect ground contact, and it can lose the mechanical function of a leg and still walk. Hannibal's control system must be clever in the way it uses its existing sensors and actuators to compensate for sensor or actuator failures.

5.2 Redundant Control Behaviors

Fault tolerance techniques using redundant sets of control strategies have been investigated by Payton et al. (1992). The redundant strategy approach implements fault tolerance in high level control and addresses high level failures. For example, a high level failure occurs when the robot encounters a situation it was not explicitly programmed to handle. In this approach, the controller is designed with redundant strategies to perform a task. A performance model exists for each stragety, and a failure is detected if the behavior's actual performance is worse than the behavior's expected performance. If the first strategy tried does not suffice, the controller eventually tries another strategy. The controller goes through its repertoire of strategies until it finds a strategy with acceptable performance instead of unsuccessfully trying the same thing over and over.

The redundant strategy approach is not well suited for hardware failures. First, it does not specifically address the cause of the problem, it only addresses the symptoms. It may take several tries before the controller finds a strategy that works. For example, let us say Hannibal has redundant walking behaviors–each behavior implements a different gait. If a leg fails, some of these gaits are unstable. The redundant strategy approach requires that

Hannibal undergo unstable locomotion until the controller finds a gait that is stable with the loss of that leg. It is more desireable if Hannibal could recognize which leg failed and adapt its gait to specifically address the failure. Second, the redundant strategy approach inherently requires that the hardware errors manifest themselves in the robot's behavior before the control system can detect something is wrong. Thus, the performance of the infected behaviors must degenerate to an unacceptable level before the controller takes corrective action. This could be detrimental to a robot that must function in a hazardous environment.

5.3 Robust Virtual Sensors

Hannibal's controller uses robust virtual sensors to confine hardware failures to low level control. Robust virtual sensors are virtual sensors which remain reliable despite sensor failures. Recall that the virtual sensors are responsible for characterizing the robot's interaction with its environment using sensor information, and for activating the robot's behaviors. If the virtual sensors give the correct output despite sensor failures, then the robot will continue to do the right thing at the right time despite these failures.

Robust virtual sensors are appealing because they confine the effect of faults to low level control and prevent errors from infecting high level control. This approach effectively compensates for *local failures*. Local failures (also called non-catastrophic failures) are failures whose effect is confined to the leg on which it occurs. For example, the failure of a leg's ankle sensor is a local failure because it affects that leg's ability to sense ground contact, but it does not affect the ability of the other legs to sense ground contact.

Unfortunately, it is not possible to confine the effects of all sensor-actuator faults to low level control. Some failures affect the behavior of the overall system - we call these failures

global failures. Global failures (also called catastrophic failures) must be compensated for within high level control. For example, if a leg's shoulder actuator fails, then the leg cannot support the body. Consequently, this failure affects the global stability of the robot. The high level control must compensate for this failure by changing the robot's gait so the robot can walk in a stable manner with one less leg.

6 Adaptivity vs Redundancy

Hannibal's fault tolerance capabilities are implemented using *adaptive* agents. The use of adaptive agents distinguishes Hannibal's implementation of robustness from other implementations. One adaptive virtual sensor (instead of several redundant virtual sensors) exists per leg for each leg-terrain interaction of interest to the controller. As sensors fail, the adaptive virtual sensors maintain reliable performance by changing how they use their sensor information. For example, when a failure is detected, the appropriate virtual sensors are alerted of the failure and respond by reconfiguring the way they use their sensory information. This entails ignoring the input from the broken sensor and changing the way the virtual sensors use information from the reliable sensors. In this manner, the virtual sensors use their most reliable sensors to produce the most reliable result. If the failed sensor starts working again, the virtual sensor reintegrates the previously failed component.

The approach Hannibal uses to tolerate catastrophic failures also exploits adaptivity instead of redundancy. When a leg suffers a catastrophic failure it cannot be used. High level control must change the gait so locomotion remains stable with one less leg. A redundant approach might involve implementing redundant walking behaviors where each behavior

exhibits a different gait. This is undesirable because of the extra code space required to implement each gait behavior plus the gait switching mechanism. In contrast, the adaptive approach implements one walking behavior which can alter its gait by changing a parameter. Low level control is responsible for detecting catastrophic failures and alerting high level control. High level control is responsible for adapting the robot's behavior so that locomotion remains stable (see (Ferrell 1993) for details).

7 Fault Tolerance Network

The following sections describe the distributed network that implements fault tolerance on Hannibal. As with the rest of Hannibal's control system, fault tolerance is implemented with concurrently running processes. Fault tolerance consists of four phases: error detection, masking, recovery, and reintegration. Non-catastrophic faults affect local control. They are detected within the low level network and compensated for within the virtual sensors. In this way, these faults do not affect the high level performance of the system. Catastrophic faults unavoidably affect global system performance. They are detected within the low level control and compensated for within the high level control.

In the following sections we illustrate Hannibal's fault tolerance processes through an example. The example we use to illustrate tolerance to local failures is a robust *ground contact virtual sensor*. Keep in mind that this example is of only one virtual sensor on one leg which uses only a few of the sensors on that leg. Similar processes run concurrently on the same leg to make its other virtual sensors robust as well. The example we use to illustrate tolerance to global failures is a broken shoulder actuator. Other processes run concurrently

on the same leg to tolerate a variety global failures as well. The processes mentioned above are replicated for each leg and run simutaneously.

We refer the interested reader to (Ferrell 1993) which presents our fault tolerance implementation in more detail as well as the overall control of our legged robot.

7.1 Detection

The detection processes are responsible for recognizing sensor and acutator failures. Detection is the hard part of the fault tolerance problem because the robot does not know a priori the correct behavior of the sensor. Sensor failure recognition is performed using two methods. The first method exploits the context provided by the time history of the leg motion. The robot does not know what the correct sensor behavior is for a given step cycle. However, the robot does know the *plausible* leg motions because the plausible leg motions are the behaviors that have been programmed for the leg. We call the set of plausible leg motions the *model*. If the leg sensors reflect a plausible leg motion, i.e. they agree with the model, then the robot has some confidence that the sensor is working. However, we could still have the case where the robot's sensors do not reflect reality although they reflect a plausible reality. For instance, a sensor could say the robot is stepping on the ground when it is really stepping in a hole. To overcome this problem, the second method exploits the context provided by complementary sensors. If reliable complementary sensors agree with the sensor in question, i.e. they confirm the robot is stepping on the ground, then the system has more confidence in that sensor. The confidence level in a sensor is reflected by a *pain parameter* affiliated with that sensor. The pain level is the inverse of the confidence level.

7.1.1 Sensor Model

It is possible to model sensor behavior if the behavior of the leg is known. Rotational potentiometers measure leg joint angle, strain gauges measure leg loading and linear potentiometers measure foot loading. Hence the motion of the leg and the leg's interaction with the environment directly affect sensor output. To model plausible sensor behavior, we first classify the leg behavior in terms of states. Each phase of the step cycle is divided into four possible leg states: S_{start}, S_{middle}, S_{end}, and $S_{exception}$. The S_{start}, S_{middle}, and S_{end} states account for the typical behavior of the leg (see figure 2). The $S_{exception}$ state accounts for leg-terrain interactions that may occur during that phase but typically do not. For example, during the step phase S_{end} corresponds to ground contact and $S_{exception}$ corresponds to stepping in a hole. We assume the leg behaves as programmed (unless a catastrophic failure occurs), so the constraints on the transitions between leg states for each phase of the step cycle are known. A set of sensor values corresponds to each leg state. From this sensor-state relationship, we derive a model for plausible sensor behavior. The sensor model consists of the expected transitions between sets of sensor values given the plausible transitions between leg states. The transformation from leg states to sensor values were derived experimentally by observing sensor values as the robot walked through its environment. Processes (called *sensor-state processes*) are written for each sensor and classify the sensor's values into leg states.

7.1.2 Monitoring Individual Sensors

Each sensor has a process that monitors sensor performance. These processes are called *sensor monitor processes* and they exploit the context provided by the time history of the

leg motion. Each sensor's monitoring process uses the corresponding model of acceptible sensor-state transitions to detect sensor failures. Essentially, if a sensor's behavior does not reflect plausible behavior, then the confidence in that sensor decreases. Each process monitors the transitions between states by recording when each state occurs. The sensor monitor processes check that a timely and sequential transition of leg states is upheld. If the sequential and time contraints of state transitions are upheld, the process inhibits that sensor's pain parameter, otherwise it excites it. Figure 3 presents run-time data for the sensor monitor processes of the vertical force sensor.

7.1.3 Consensus Monitoring of Sensors

Each step cycle phase has a *consensus monitor process* that monitors agreement between complementory sensors. The sensor-state processes categorize sensor values into leg states; this provides a common measure for comparing sensor behavior. The consensus monitor processes use discrepancies between sensor-state values of complementary sensors to detect sensor failures. To do this, each sensor casts votes for the state it thinks the leg is in based on its sensor value. The state with the majority vote is elected as the leg state. Each time a leg state is elected, each sensor's votes are compared with the newly elected leg state. If a sensor voted for the elected state, the consensus monitor process inhibits that sensor's pain parameter, otherwise it excites it. Figure 4 presents run-time data for the consensus monitor processes for the ground-contact virtual sensor.

7.1.4 Injury Agents

An *injury agent* for each sensor determines whether the sensor is working or broken by monitoring the sensor's pain level. As described above, each sensor's sensor monitor process and consensus monitor process excite its pain parameter when a discrepancy occurs, and inhibit the pain parameter when no discrepancy occurs. The level of the pain parameter increases upon excitation and decreases upon inhibition. The injury agent compares the sensor's pain level with a threshold value. If the pain level exceeds the threshold, the injury agent declares the sensor is broken, and when the pain level is below threshold, the injury agent declares the sensor is working. See figure 5.

7.2 Masking

The masking processes are responsible for removing the affects of local faults so that these faults do not effect high level behaviors. Masking is performed by the virtual sensors. A minor form of masking is also performed by the consensus monitoring processes. The injury agents continually inform the virtual sensors and consensus monitor processes about which sensors are functional or broken. Once informed of a broken sensor, the masking processes within the virtual sensors and consensus monitor processes remove the effects of the broken sensor.

To uphold the integrity of the elected leg state, the consensus monitoring processes disregard information from broken sensors. Provided the elected leg state is correct, the consensus monitor processes can detect valid disagreements between a sensor's leg state vote and the actual leg state. However, the wrong leg state could be elected if votes from broken sensors

are honored. Therefore, if a sensor is faulty, the sensor consensus process removes the broken sensor's votes in the leg state election process. In this way, the leg state is determined only by functional sensors, and the result remains reliable despite failures.

The virtual sensors mask sensor failures by disregarding information from failed sensors. To show how this is implemented, we use the ground-contact virtual sensor as an example (see figure 6). The ground-contact virtual sensor uses information from the shoulder potentiometer, the ankle displacement potentiometer, and the vertical load strain gauge. Each sensor value is passed through a filter. The filter outputs `true` if the value satisfies its condition for ground contact, otherwise it outputs `false`. The filtered results are sent to a decision process that combines these results to determine whether the leg is contacting the ground or not. This process also receives inputs from the injury agent of each sensor. If an injury agent declares its sensor is broken, the decision process ignores the broken sensor in computing the output. Consequently the final ground-contact decision is made only by functional sensors. Figure 7 presents run-time data of the masking process performed by a ground contact virtual sensor.

7.3 Recovery

The purpose of recovery is to return the system to an operational state once components fail. The new operational state should have as many of the original resources available as possible, and the transition to this new state should have minimal impact on normal system operation. We want the system to recover from transient errors as well as permenant errors. Transient errors result from occasional erroneous sensor values or sensor drift. Permenent errors result from sensor failure. Recovery takes three forms: retry addresses transient errors, dynamic

recalibration addresses sensor drift, and reconfiguration addresses permanent failures.

7.3.1 Retry

Transient sensor errors are filtered out by the pain mechanism. In essence, the pain mechanism provides a means for "retrying" a sensor if it produces a bad value. Instead of calling a sensor broken after it produces an erroneous sensor reading, the pain mechanism continually adjusts the pain level of the sensor. The pain threshold of each sensor is set such that a series of errors must occur before the pain level rises above threshold. However, since the error is transient, the sensor displays normal behavior during subsequent cycles, and the pain level diminishes. In effect, occasional errors are averaged out and carry little weight for determining whether a sensor is functional or broken.

7.3.2 Dynamic Recalibration

Dynamic recalibration processes are written for each sensor. These processes update the reference values used by the sensor model processes. These values are used to compute the state transition values for the sensors. It is important to deal with sensor decalibration because the sensor models become less accurate as the reference values become less accurate. The dynamic recalibration behaviors assume sensor gain remains constant; hence any decalibration is attributed to DC offset only. This is a reasonable assumption given Hannibal's leg sensors.

7.3.3 Reconfiguration

As sensors fail we want system performance to degrade gracefully. In fact, we want to maintain the highest level of performance given the functional state of the robot. To accomplish this, the virtual sensors specifically tailor their use of sensor information to minimize the impact of sensor failures on virtual sensor performance. Virtual sensor performance consists of reliability and response time. We have found, by experimental means, that virtual sensors perform better as more sensors are used.

The virtual sensors can achieve faster response times without compromising reliabiltiy, provided sufficient quantity and type of sensors are used. To illustrate this, we look at the ground contact virtual sensor as an example. The ground contact virtual sensor uses information from three types of sensors to ascertain leg loading: shoulder position, ankle displacement, and vertical force. Velocity and position are the only types of information the position sensor provides to determine loading. Loading can be inferred from the position sensor when the leg is prematurely stopped above its lowest possible position. The force and ankle sensors directly measure loading and do it faster than the position sensor–the leg does not have to stop moving before they signal loading. Thus, if only the position sensor is working, the leg must come to a complete stop in the vertical axis to satisfy the condition for ground contact. However, if a force or ankle sensor is working, the condition for the position sensor can be relaxed so that it is satisfied sooner. In this case, the position sensor satisfies the condition for ground contact if either the downward velocity is sufficiently small or zero. Similarly, if all sensors are working, the conditions for the force and ankle sensors can be relaxed as well. Hence, the response time of the ground contact virtual sensor can be sped up

if more sensors are used without compromising reliability. Figure 8 presents our results for how performance relates to the number of sensors used for a ground contact virtual sensor.

7.4 Reintegration

The robot's performance is enhanced if more sensors are used. The purpose of reintegration is to reincorporate repaired sensors so the robot uses the maximum number of reliabile sensors. Repaired sensors are sensors that were previously faulty but behave well again. Reintegration is useful if a sensor is broken and then fixed, or if the sensor was incorrectly classified as broken. In either case, we call a sensor "broken" if its pain level is above threshold, and we call it "repaired" if the sensor's pain level rises above threshold and then lowers below threshold.

Both the sensor monitor processes and sensor consensus processes induce reintegration of repaired sensors (figure 9). They accomplish this by inhibiting the pain parameter. If a failed sensor exhibits normal behavior, the sensor's behavior agrees with the modeled behavior again. Consequently, the sensor monitor process inhibits the pain parameter. Similarly, if a failed sensor exhibits normal behavior, the sensor behaves in consensus with other functional sensors again. Consequently, the sensor consensus process also inhibits the pain parameter. Eventually, the sensor monitor process and sensor consensus process lower the sensor's pain level below threshold. Once this occurs, the sensor's injury process tells the virtual sensors that the sensor is working. The virtual sensors respond by reincorporating the repaired sensor in computing their output. Hence, the influence of the repaired sensor is reintegrated into the control system. Figure 10 presents run-time results for reintegration of the ankle sensor.

7.5 Catastrophic Failures

Global failures are detected in low level control, but must be compensated for in the high level control. Hip actuator failures, hip potentiometer failures, shoulder actuator failures, and shoulder potentiometer failures are global failures. These failures effectively prevent the leg from behaving as programmed. This is obviously the case if an actuator fails. If a joint angle potentiometer fails, the servo processors have no way of knowing the positional error, so they cannot servo the actuator. In the event of a global failure, the leg is rendered not usable, so the robot must modify its behavior to function with fewer legs.

7.5.1 Detection

Global failures are detected by the same processes used for detecting local failures. Potentiometer failures are found using their respective sensor monitor process and consensus monitor process. Actuator failures are inferred through concurrent failure of sensors whose behavior depends on that actuator working. Once an actuator fails, all dependent sensor models are invalid. Consequently, the corresponding sensors look as if they have failed even though this may not be the case. If the shoulder actuator fails, for example, the ankle sensor, vertical loading sensor, and shoulder position sensor all appear broken to the monitoring processes (figure 11). Once a global failure occurs, it is irrelevant whether the local sensors appear broken because the leg is not usable anyway. The detection of global failures can be reduced to detecting potentiometer failures only because the monitoring processes detect joint angle potentiometer failures when either type of global failure occurs.

7.5.2 Masking

Once a leg fails, the output of its ground-contact virtual sensor is not valid. The ground-contact virtual sensors of the stepping legs influence when the next step cycle occurs. Each recovering leg inhibits the supporting legs from proceeding to the next step cycle until it attains ground contact. Thus, it is important to mask the effect of non-valid ground-contact virtual sensors or else they may adversely affect the robot's gait. To prevent this from happening, the output of the ground-contact virtual sensor defaults to `ground-contact = true` for all broken legs (figure 12). By doing so, the effect of renegade ground-contact virtual sensors on the robot's gait is removed.

7.5.3 Recovery

Given a global failure, high level control agents compensate by lesioning the broken leg. Each leg has a lesion mechanism which is responsible for lesioning the leg once it suffers a global failure. Within high level control, each critical potentiometer has a leg-pain parameter and a leg-injury agent associated with it (figure 13). Each leg-injury agent receives messages from its corresponding low level potentiometer injury agent every 0.1 second. If a message indicates the potentiometer is broken, the corresponding leg-injury agent excites the appropriate leg-pain level. The leg-pain level automatically decays every three seconds. If a leg-pain energy level rises above the `lesion` threshold, the corresponding leg-injury agent activates the lesion behavior. The lesion behavior disables the leg and adopts a gait that is stable without the use of the damaged leg (figure 14). The lesion behavior is described in (Ferrell 1993).

7.5.4 Reintegration

If the broken leg is repaired or the leg was wrongly determined to be broken, high level agents reintegrate the leg. This is accomplished by occasionally testing the leg to see if it is functional again. Leg reintegration is performed by the lesion mechanism. After the leg-pain level rises above the `lesion` threshold, the leg-injury agent is prevented from exciting the leg-pain level. Consequently, the leg-pain level decays slowly back towards zero. When the leg-pain level lowers to the `retry` threshold value, the system tries to use the leg and the leg-injury agent is allowed to increase the leg-pain level (figure 15). If the leg is still broken the leg-pain level raises above threshold, and the process repeats. If the leg works the next time it is tested, the leg-pain level decays to zero. Once this happens, the system acknowledges the leg is functional again, de-activates the lesion behavior, and resumes using the leg.

8 Performance

8.1 Tests

Several tests were conducted to determine the system's response to various types and combinations of failures. The tests involved inflicting the desired fault and observing its effect on the robot's behavior while the robot walked through its environment. We tested sensors with a local effect and sensors with a global effect. Permanent sensor failures were inflicted by disconnecting the power, ground, or output wires to the sensors. Broken wires are the most common cause of permanent sensor failure on Hannibal. Sensor decalibration was

induced by adjusting the reference voltage offset of the sensor signals; decalibration also occured naturally over time. Transient errors are difficult to force but arose during the course of the tests. Different combinations of permanent errors were tested as well. We disabled sensors individually, concurrently, and sequentially over time. We repeated the concurrent and sequential tests by disabling the sensors in various permutations. Actuator failures were evoked by disconnecting the wires to the motor. On one occasion the shoulder actuator shaft sheared off, so we had the opportunity to test for motor shaft breakage as well.

8.2 Results

The results of the tests described above are presented in figure 16. As shown, the system is quite flexible and responds to a wide variety of types and combinations of failures. We conducted the tests by evoking various types and combinations of sensor faults on the left front leg (the same processes run on the rest of the legs). The tests were performed while Hannibal walked over flat terrain with holes and cliffs. Within this environment, the interesting virtual sensors are the ground-contact virtual sensor and the step-in-hole virtual sensor. As a result, we focused our tests on the sensors used by these virtual sensors and the acutators that affect the behavior of these sensors. Hence, the sensors and actuators involved in the tests are the shoulder potentiometer, ankle displacement sensor, vertical force sensor, and shoulder actuator. It is interesting to note that the system responds to the failure of a local leg processor as well. Once a local leg processor fails it cannot send the leg sensor signals to the main processor. Consequently all sensors look as if they have failed, and the system responds by lesioning the leg.

Figure 17 presents the response time of the system to recover from failures and to reine-

grate repaired components. The system recovers from failures quickly enough such that the robot's behavior does not have to degrade to an unacceptable level before the failure is compensated for. It also reintegrates repaired components quickly so the system readily has access to its reliable resources.

8.3 Evaluation

To evaluate the fault tolerant aspects of Hannibal's system, we discuss them in relation to the following topics:

8.3.1 Completeness of fault detection

The system successfully detects a wide assortment of common failures. It distinguishes between non-catastrophic failures and catastrophic failures. Regarding local failures, it recognizes which type of sensor has failed. Regarding catastrophic failures, the system recognizes potentiometer failures. Actuator failures appear as the massive failure of all sensors whose behavior depends on that actuator. Local processor failures appear as the massive failure of all sensors whose values are communicated to the main processor by that processor. When these types of catastrophic failures occur, the corresponding potentiometer(s) appears broken. Furthermore, all catastrophic failures evoke the same recovery procedure. Hence, the system only looks for potentiometer failures to detect catastrophic failures.

8.3.2 Fault coverage

The robot successfully recovers from a wide assortment of failures. The system recovers from sensor failures, actuator failures, and local processor failures. It addresses failures with

a local effect as well as failures with a global effect. It handles transient errors, sensor decalibration, and permanent failures. It compensates for various combinations of failures: failures that occur individually, concurrently with other failures, or accumulate over time. It tolerates these combinations of failures in various permutations.

8.3.3 Confinement of errors

The system effectively prevents sensor and actuator failures from adversely influencing robot behavior. To accomplish this, the detection processes monitor the sensor outputs and alert the system of failures as they arise. This nips the failure problem in the bud because the system can effectively compensate for failures once it knows when and what type of failures have occurred. For example, robust virtual sensors filter out the effects of non-catastrophic failures so the failures do not influence the robot's behavior. Potentiometer injury processes evoke the lesion behaviors when catastrophic failures occur, so the loss of the leg does not cause the robot to become unstable.

8.3.4 Response time to failures

The system successfully detects and recovers from failures before the robot's performance degrades to an unacceptable level. A fast response time is important for successfully implementing error confinement. After all, it does not do the system much good to implement the dection and recovery procedures in the low level control if it takes them a long time to respond to failures.

8.3.5 Extent of graceful degradation of performance

The recovery processes maintain the robot's performance at the highest level given the functional state of the hardware. Because the system purposefully recognizes failures, the recovery processes can specifically tailor the robot's use of sensors and actuators to minimize the effects of failures on the robot's behavior. At the virtual sensor level, the recovery processes tradeoff speed for reliability as the number of functional sensors decreases. At the locomotion level, the recovery processes trade off speed for stability as the number of useable legs decreases.

8.3.6 Availability of reliable resources

The reintegration processes reincorporate repaired components so the robot has access to all its reliable resources. This is important because the more sensors and actuators the system can use, the better its performance will be. The reintegration response time is relatively fast so the system does not have to wait long before it can reuse repaired components.

9 Conclusion

There are several contributions this work makes towards achieving fault tolerant autonomous robot systems. First, this work presents an autonomous robot which can purposefully recognize sensor failures quickly and reliably. Second, the robot specifically and dynamically tailors its use of sensors and actuators to minimize the impact of failures on its performance. Third, the robot dynamically reintegrates repaired components to enhance its performance. I have tested the capabilities of this system by physically disabling and enabling the robot's

sensors and actuators. I have found the system recognizes and compensates for sensor and actuator failures with a fast response time. It tolerates a variety of sensor failures such as decalibration, erroneous readings, and permanent failures. It also tolerates various combinations of failures such as individual failures, concurrent failures, and accumulative failures. It handles minor failures such as a broken ankle sensor as well as catastrophic failures such as a broken leg. This is the first autonomous robot we know of with this level of fault tolerant capabilities.

10 Acknowledgements

This work was done under the supervision of Professor Rodney Brooks at the MIT Artificial Intelligence Laboratory. Special thanks to Rod Brooks, Anita Flynn, and Gill Pratt for reading earlier versions of this paper and providing useful comments.

References

Angle, C. & Brooks, R. (1990), Small Planetary Rovers, *in* 'Proceedings of IEEE International Workshop on Intelligent Robots and Systems', Ibaraki, Japan, pp. 383–388.

Bares, J. & Whittaker, W. (1990), Walking Robot with a Circulating Gait, *in* 'Proceedings of IEEE International Workshop on Intelligent Robots and Systems', Ibaraki, Japan, pp. 809–815.

Brooks, R. (1986), 'A Robust Layered Control System for a Mobile Robot', *IEEE Journal of Robotics and Automation* **RA-2**, 14–23.

Brooks, R. (1990), 'The Behavior Language; User's Guide', *MIT A. I. Memo 1227*.

Ferrell, C. (1993), 'Robust Agent Control of an Autonomous Robot with Many Sensors and Actuators', *MIT Artificial Intelligence Lab Technical Report 1443*.

Gray, J. (1990), 'A Census of Tandem System Availability Between 1985 and 1990', *IEEE Transactions on Reliability* **39**, 409–417.

Kabuka, M., Harjadi, S. & Younis, A. (1990), 'A Fault-Tolerant Architecture for an Automatic Vision-Guided Vehicle', *IEEE Transactions on systems, man, and cybernetics* **20**, 381–393.

Lin, C. & Lee, C. (1991), 'Fault-Tolerant Reconfigurable Architecture for Robot Kinematics and Dynamics Computations', *IEEE Transactions on Systems, Man, and Cybernetics* **21**, 983–999.

Payton, D., Keirsey, D., Kimple, D., Krozel, J. & Rosenblatt, K. (1992), 'Do Whatever Works: A Robust Approach to Fault-Tolerant Autonomous Control', *Journal of Applied Intelligence* **2**, 225–250.

Siewiorek, D. P. & Johnson, D. (1981), 'A Design Methodology for High Reliability Systems: The Intel 432', *The Practice of Reliable System Design* pp. 621–636.

Stuart, W. K. (1988), 'Multisensor Modeling Underwater with Uncertain Information', *A.I. Lab Technical Report 1143, MIT.*

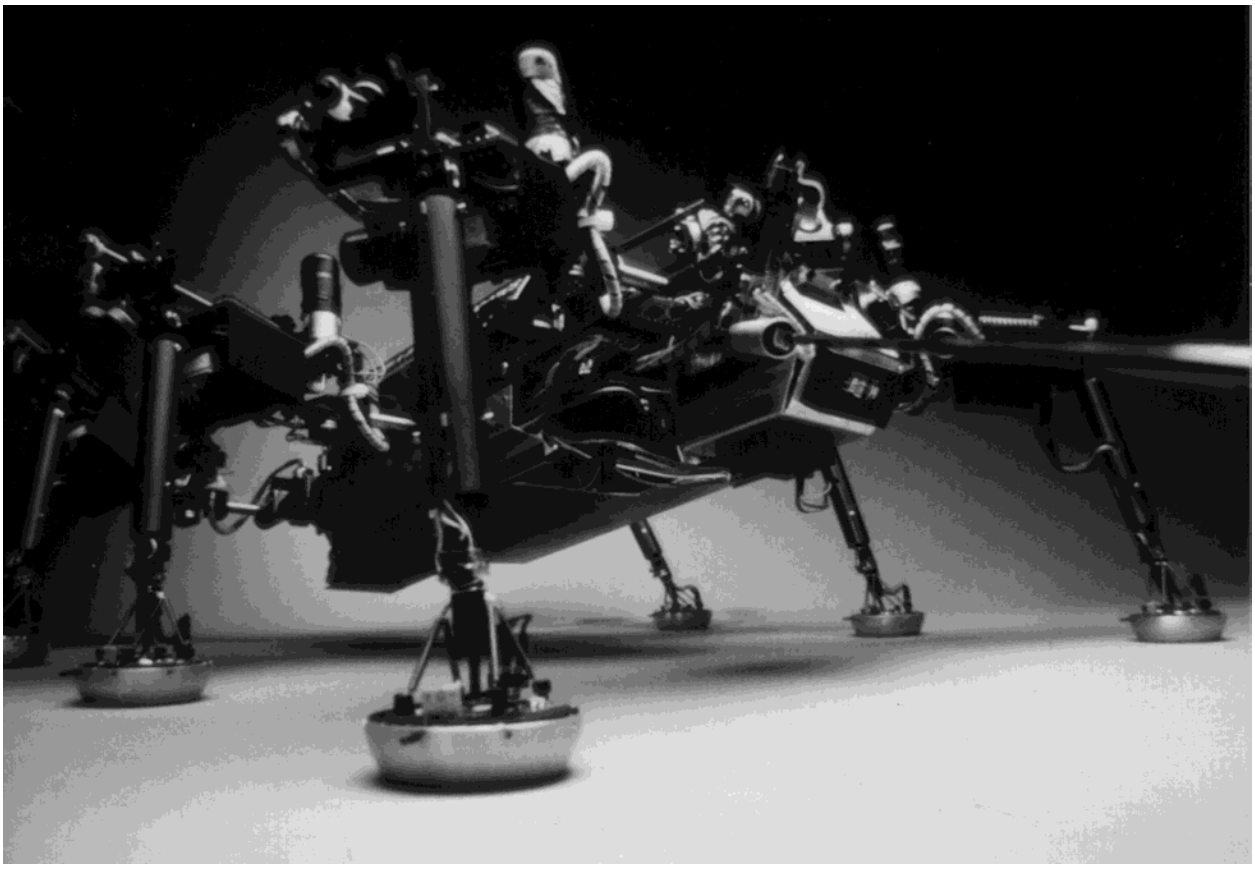

Figure 1: Hannibal is quite complex for its size. It is approximately the size of a bread box and is equipped with 19 degrees of freedom, over 60 sensors, and 8 computers. This robot was used as the platform for our fault tolerance experiments.

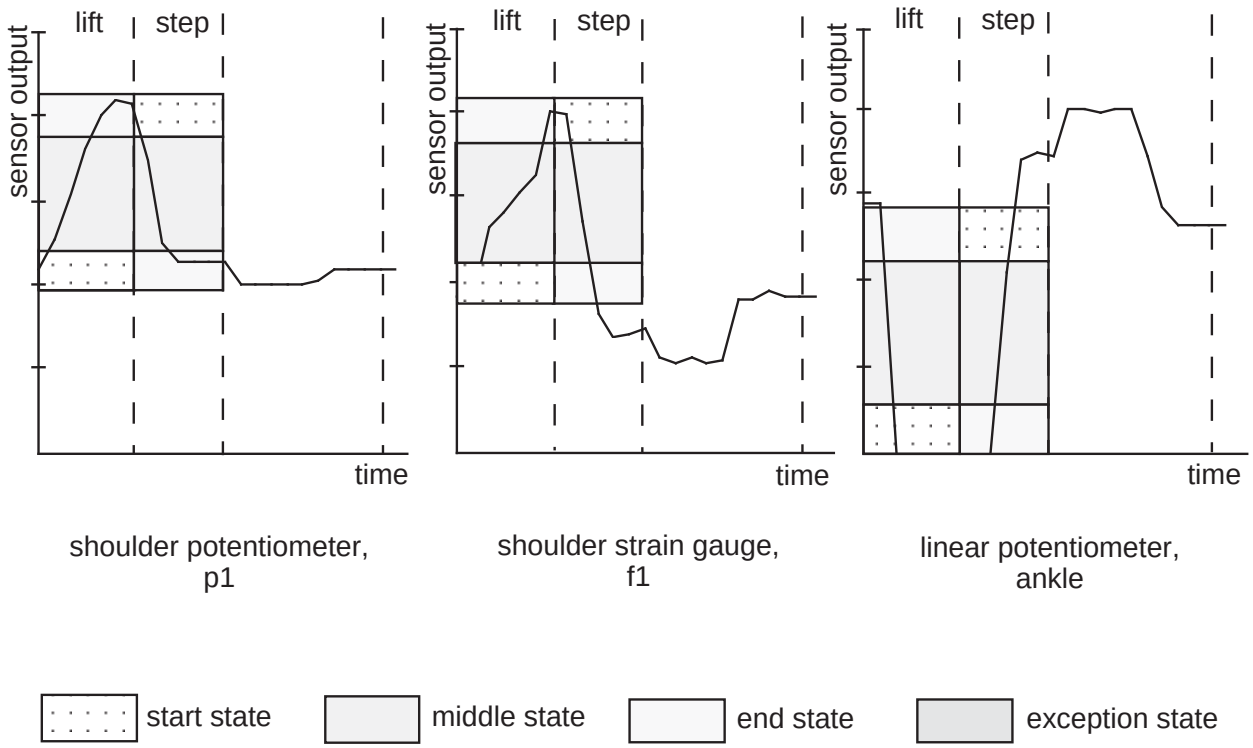

Figure 2: The relationship of sensor output to step cycle phase and leg state for the sensors used by the ground-contact virtual sensor.

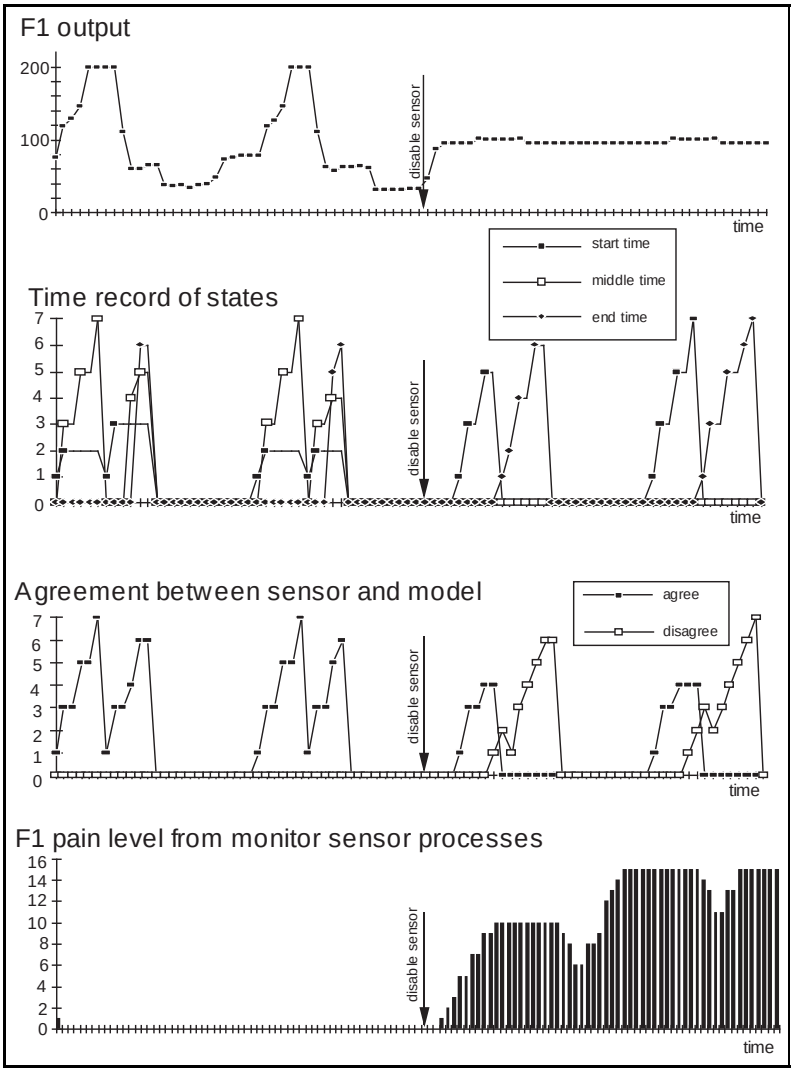

Figure 3: Response of the sensor monitor processes to vertical force sensor failure. Once the sensor fails, the actual sensor behavior frequently disagrees with the modeled sensor behavior. Consequently, the sensor's pain level increases.

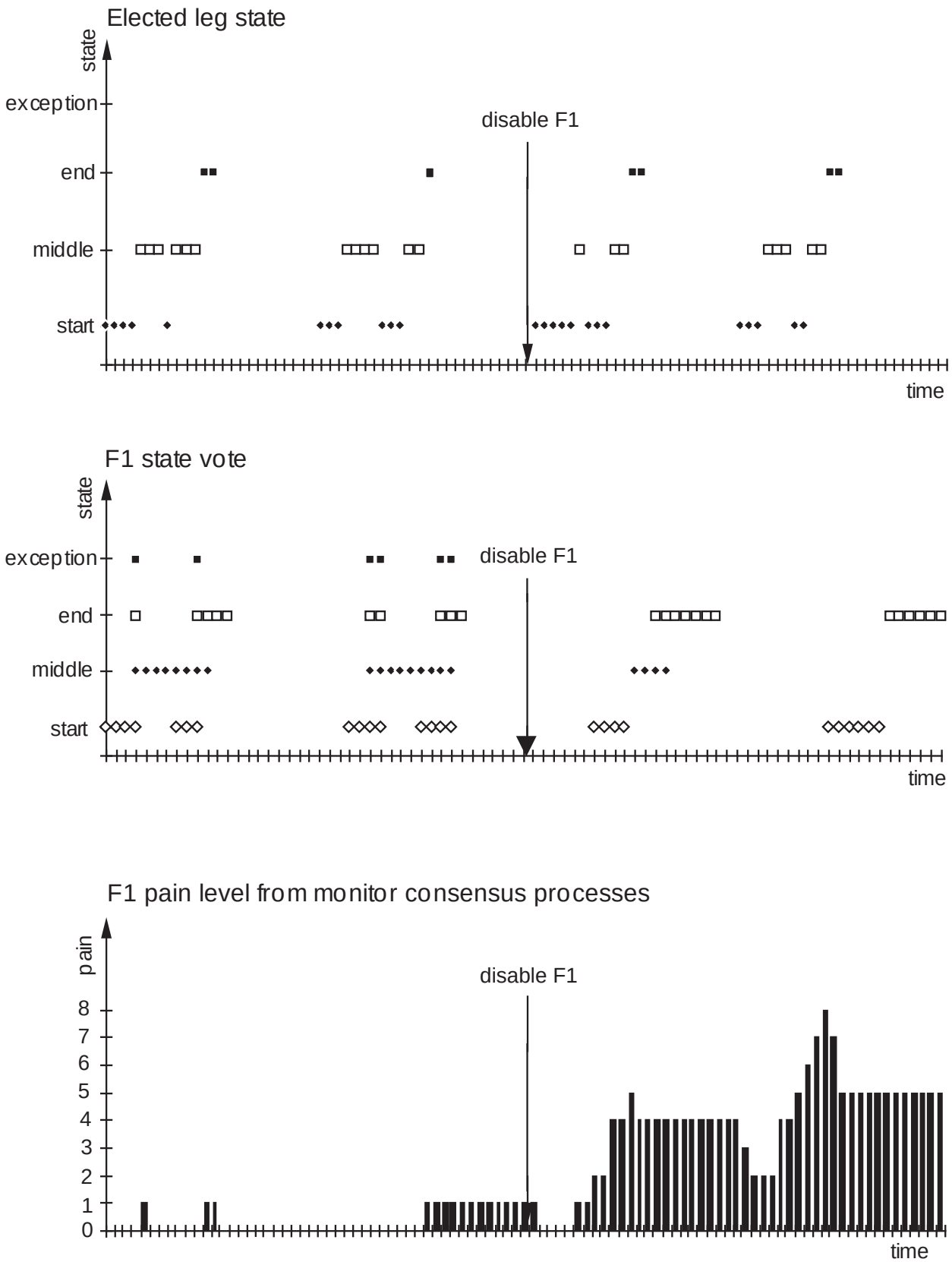

Figure 4: Response of the monitor consensus processes to vertical force sensor failure. Once the sensor fails, it disagrees frequently with the complementary sensors. Consequently, the sensor's pain level increases .

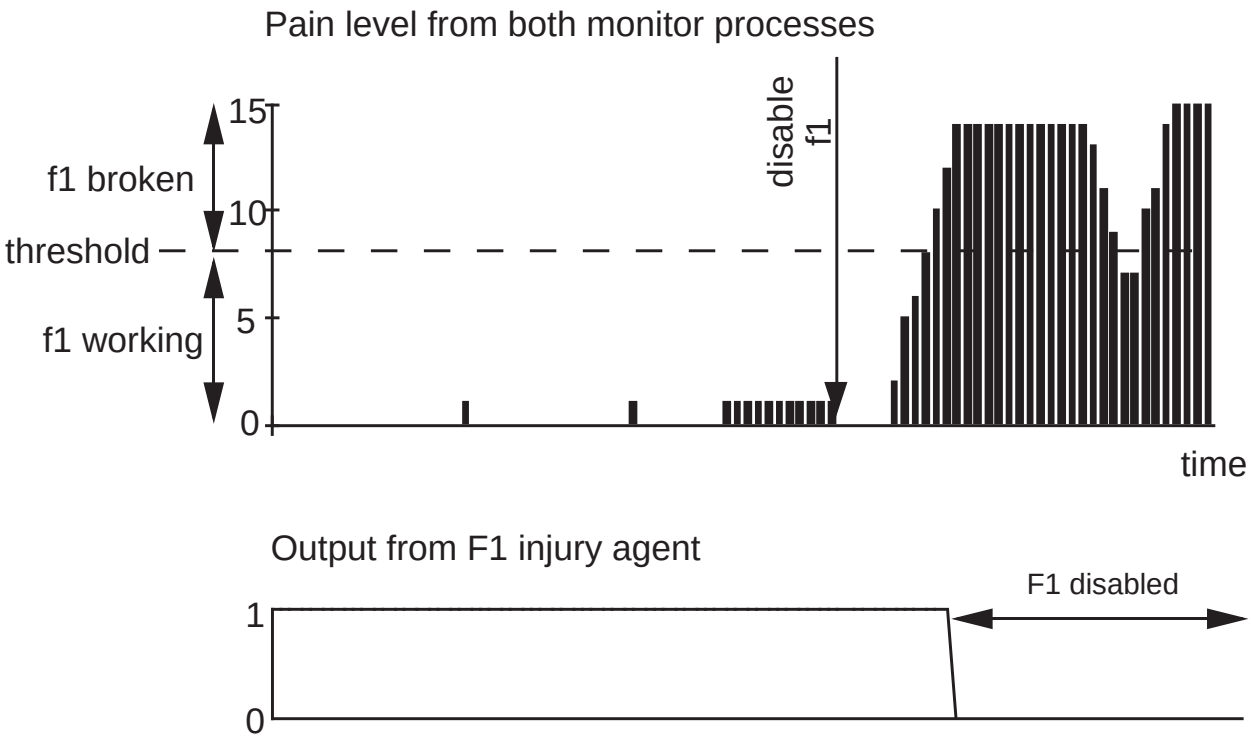

Figure 5: The injury agent of the vertical force sensor determines when the sensor has failed. The output of the injury agent is "1" when the sensor is working and "0" when the sensor is broken. It classifies the sensor as broken once the pain level exceeds the threshold value (the maximum pain level is 15).

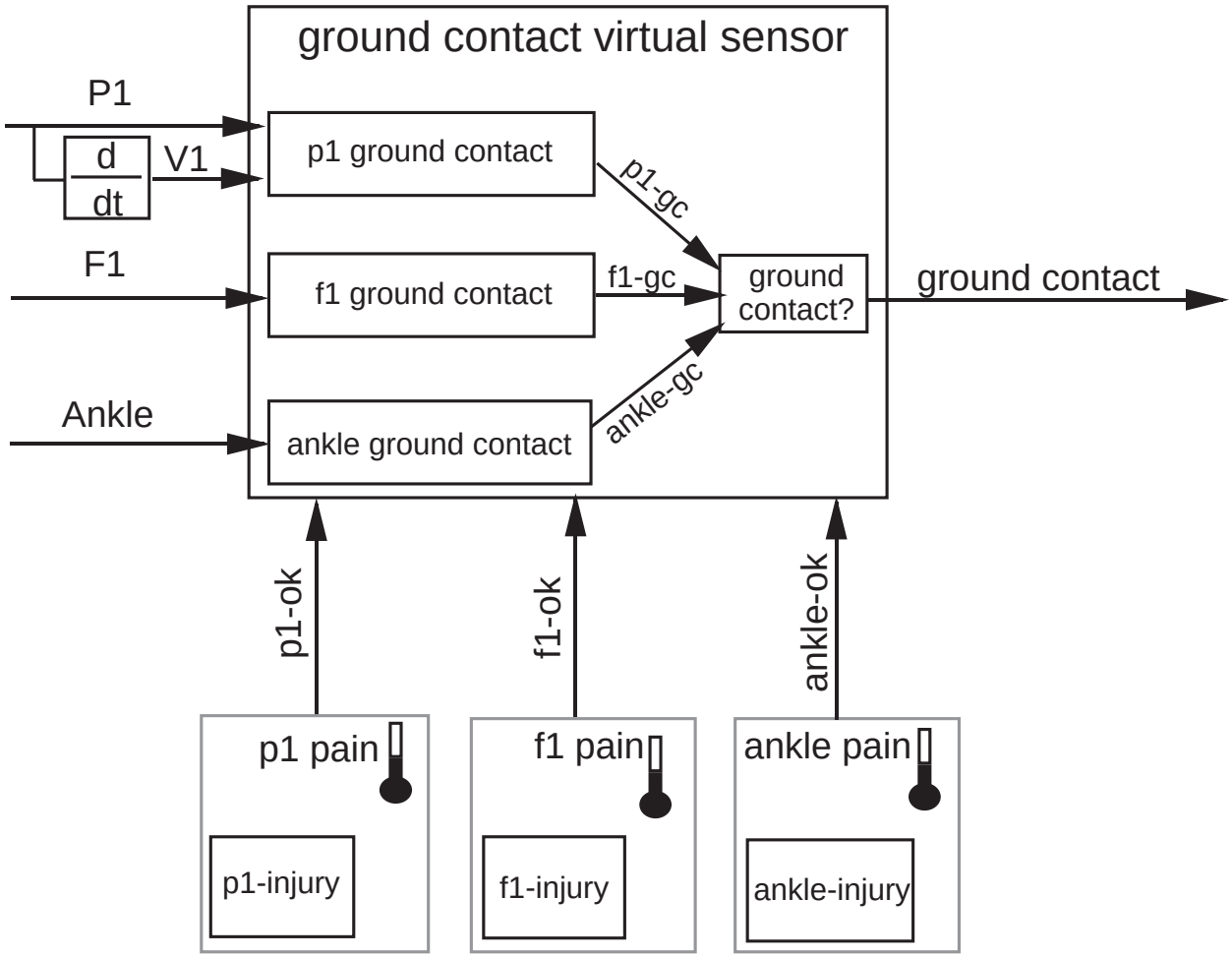

Figure 6: Illustration of a robust ground-contact virtual sensor.

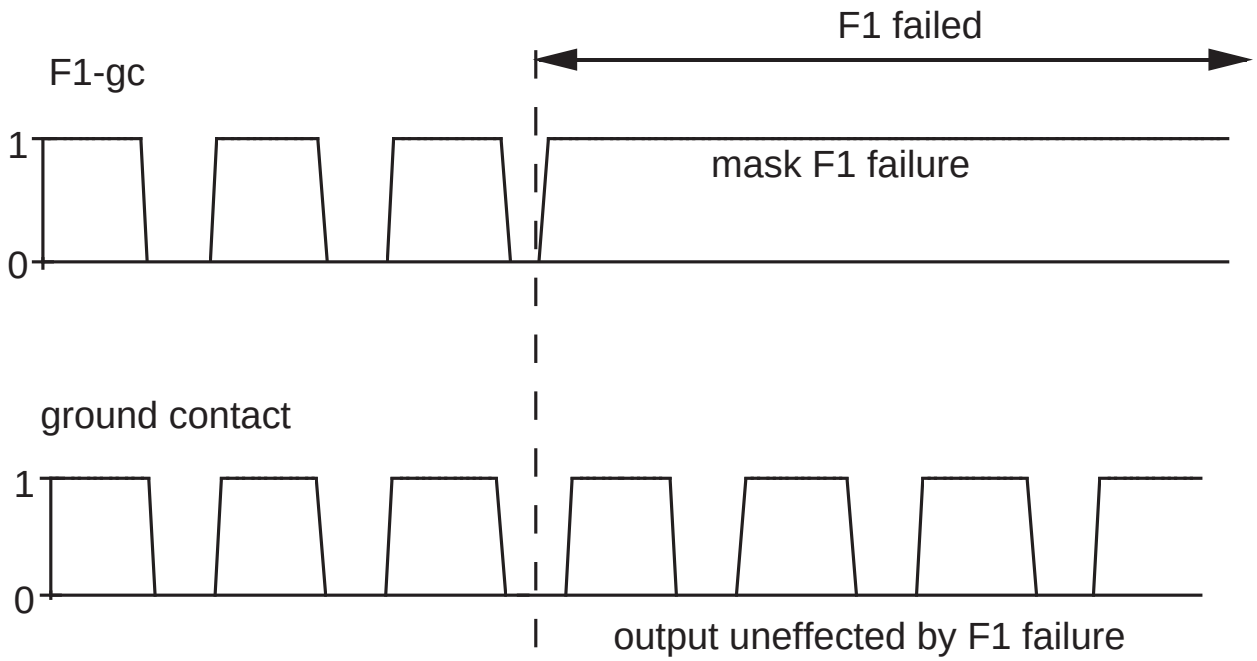

Figure 7: The virtual sensor continues to sense ground contact reliably despite the failure of the vertical force sensor (ground-contact = 1 corresponds to "true" and ground-contact = 0 corresponds to "false").

Reliability and response time of virtual sensor vs functional sensors

virtual sensor	actual sensors	total trials	success	pause
ground contact	position force ankle velocity	200	200	3
	position force velocity	200	200	8
	position ankle velocity	200	200	9
	position velocity	200	200	120

Figure 8: Performance of the ground-contact virtual sensor as a function of sensors used. The reliability is maintained at the expense of response time as fewer sensors are used. The robot does not proceed to the next step cycle unless all stepping legs attain ground contact. Thus, if the ground contact decision takes longer, the robot waits an additional time interval between steps. The number of times an additional time interval is taken is shown in the 'pause' column. This reduces the walking speed of the robot.

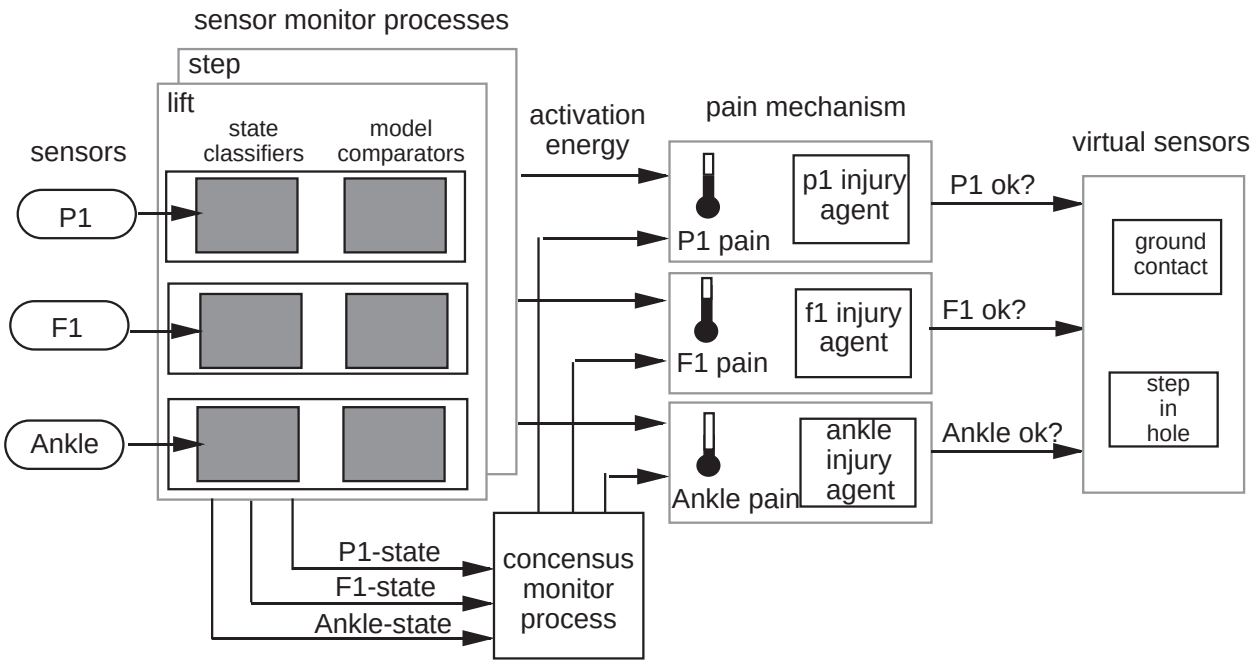

Figure 9: Both the sensor monitor processes and the consensus monitor processes assist in reintegrating repaired sensors.

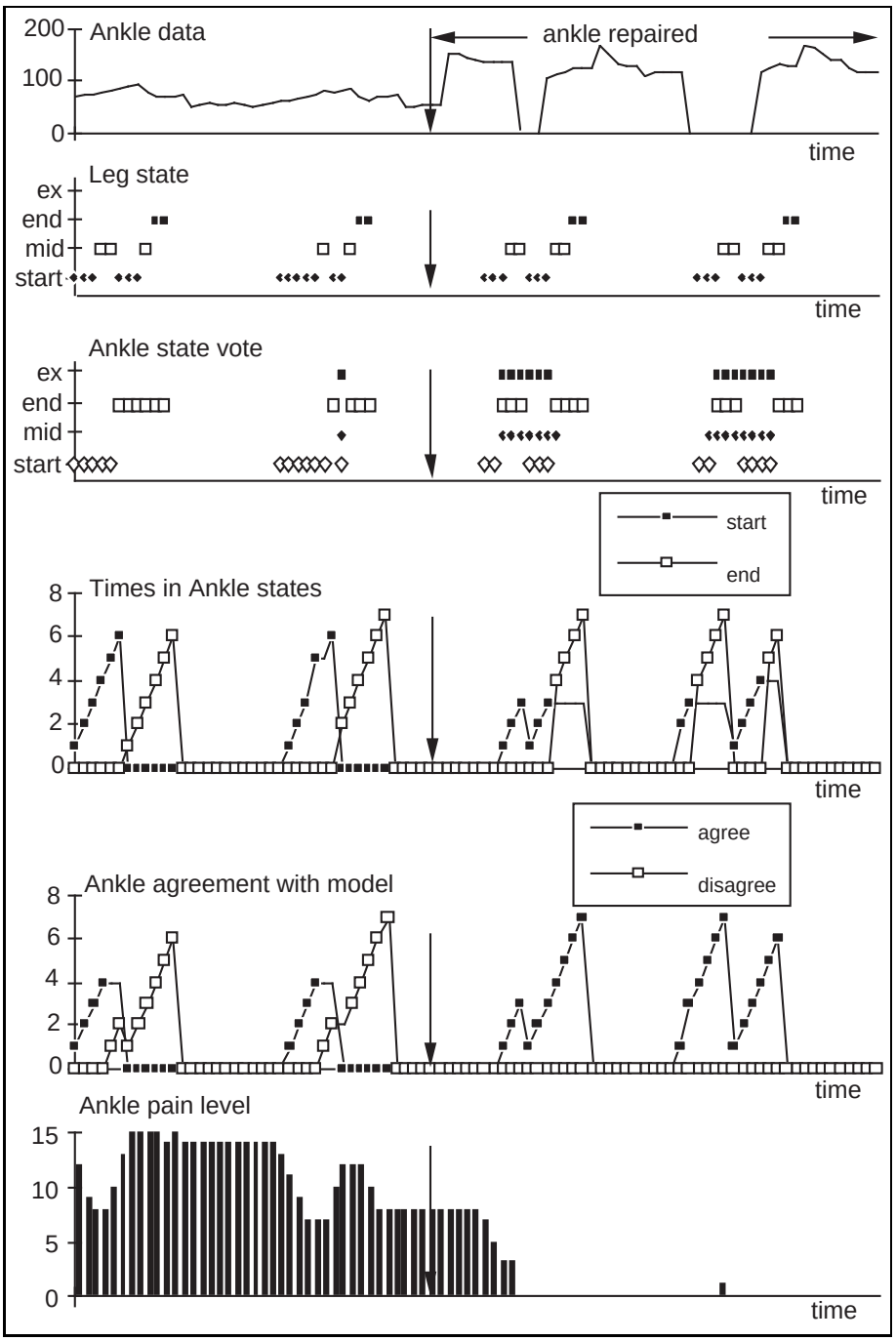

Figure 10: Once the ankle sensor is repaired, the sensor monitor processes and consensus monitor processes reduce the pain level of the sensor.

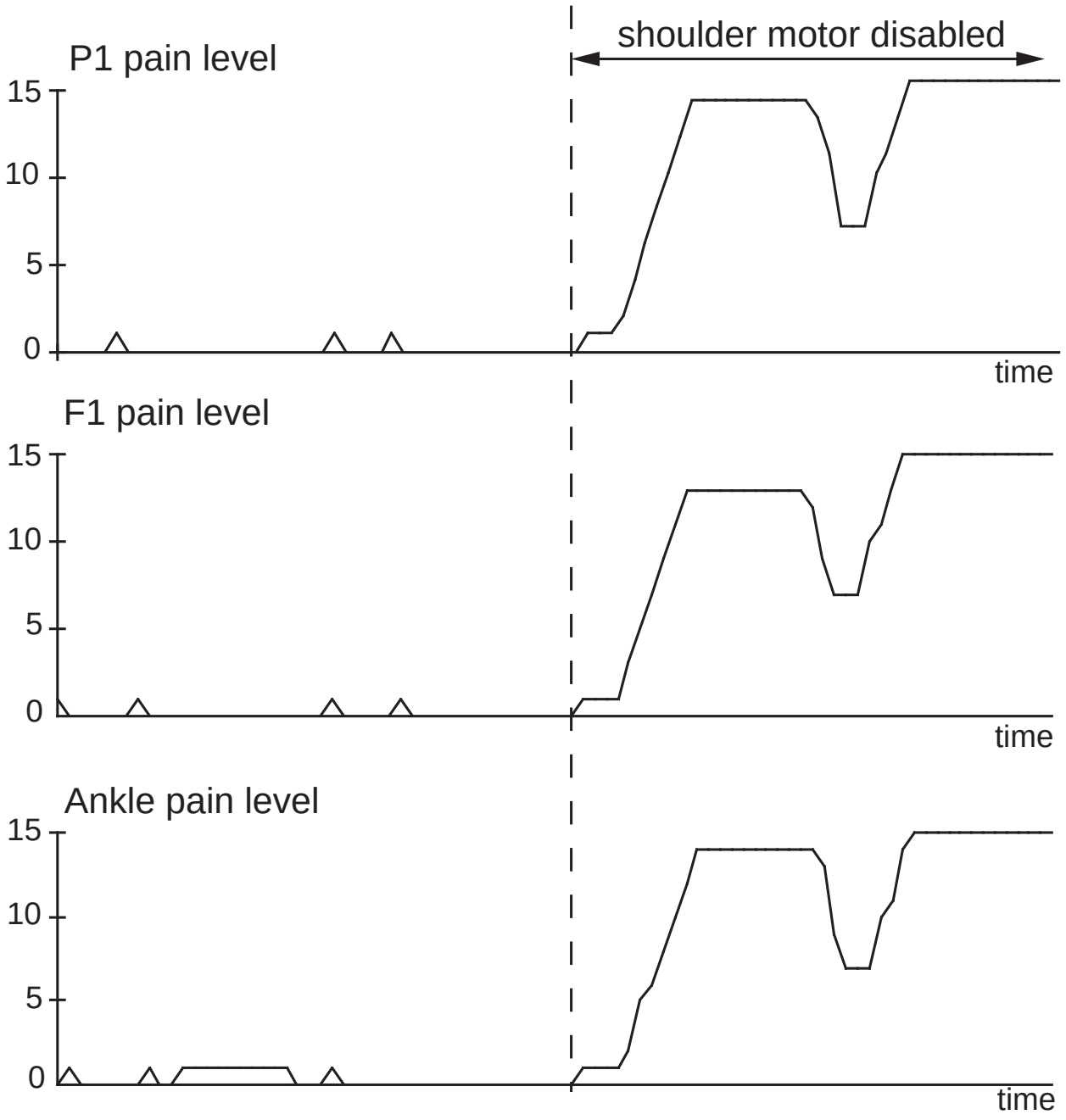

Figure 11: When the shoulder actuator fails, the shoulder position sensor, the ankle sensor, and the vertical force sensor look as if they have failed.

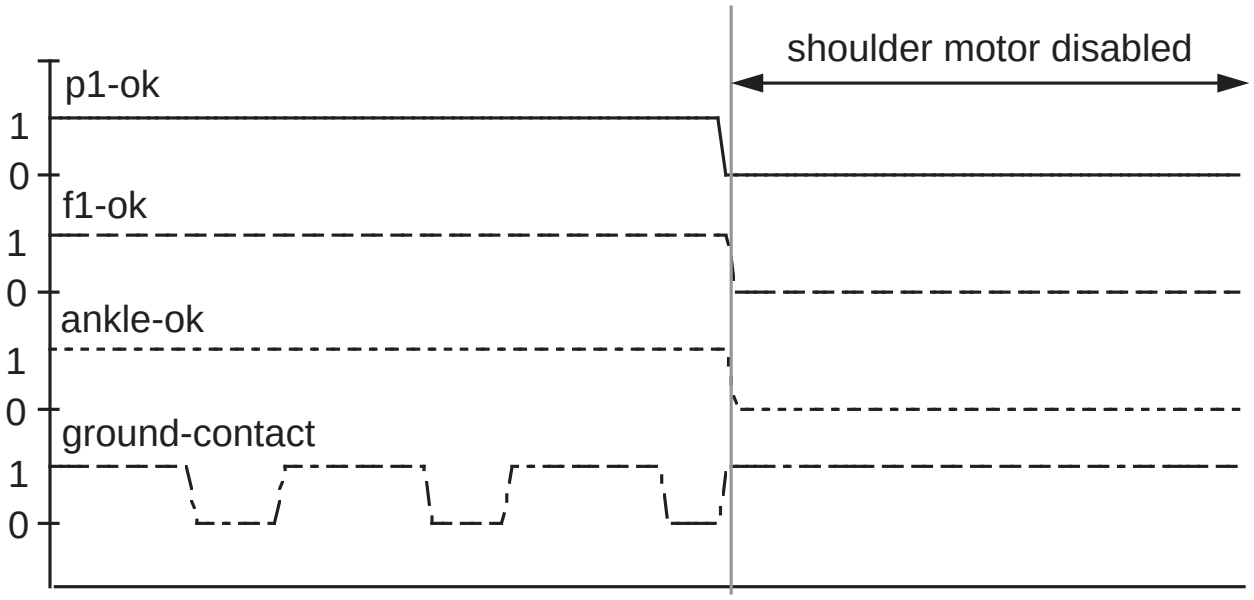

Figure 12: If the shoulder actuator fails, the ground-contact virtual sensor masks its output from the high level control.

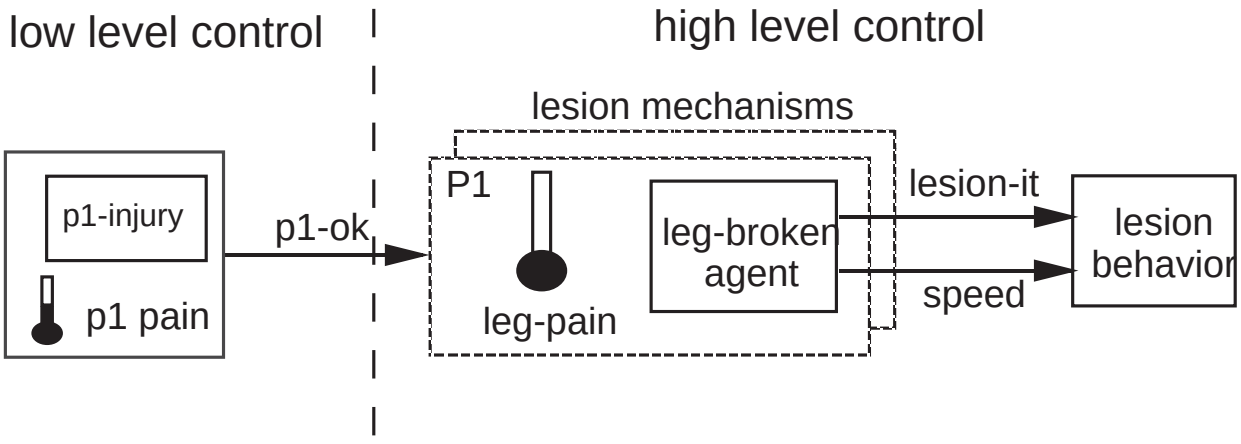

Figure 13: The leg-pain parameter and leg-injury agents are affiliated with high-level control.

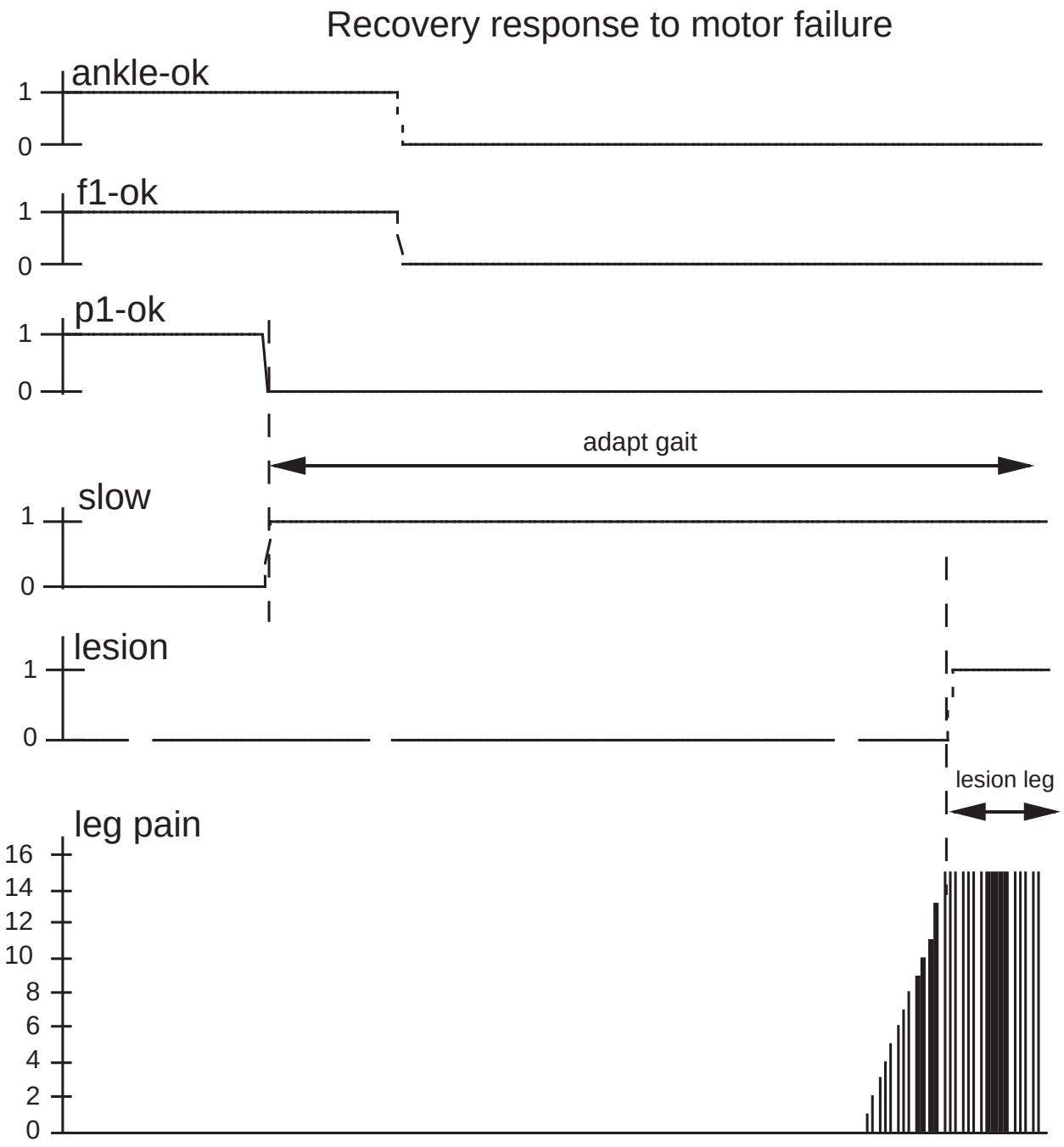

Figure 14: When the shoulder actuator fails, the system responds by lesioning the leg.

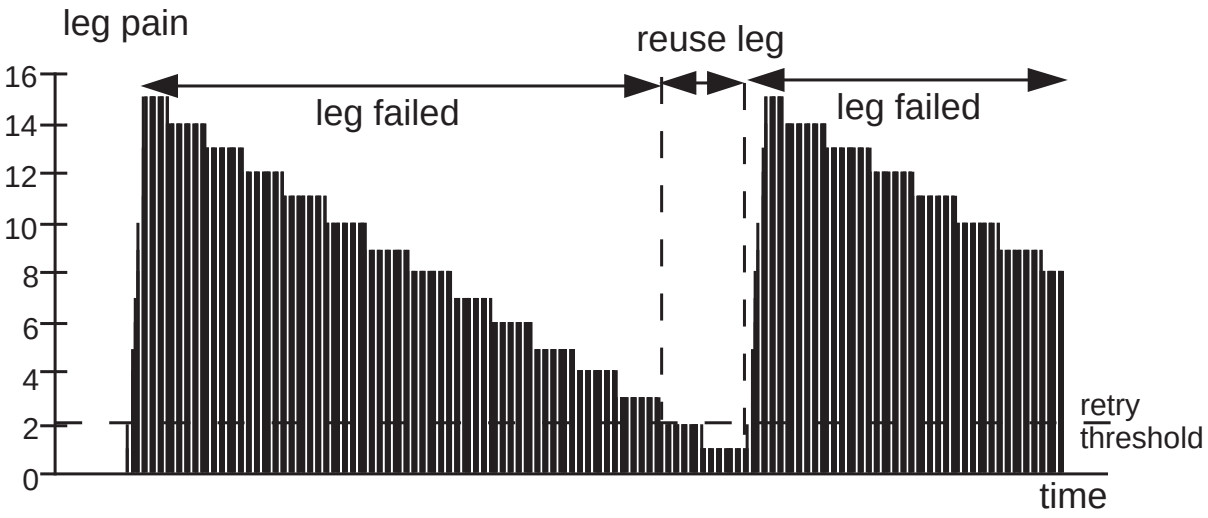

Figure 15: The system occasionally retries to use the leg. If it works again, the system reincorporates the use of that leg.

Failures Addressed

failures tested		types of failures	occurance of failures	fault tolerant technique
global effect	p1 p1,x motor local processor	drift transient permanent	individual concurrent accumulative	high level gait adaptation eg. lesion leg
local effect	f1 ankle f1,ankle			low level robust virtual sensor eg. ground contact step in hole

Figure 16: The system is tolerant of a variety of types and combinations of failures

Response Time of System

failure	response time of detection	response time of recovery	response time of reintegration
global effect	within 2 step cycles	within 2 step cycles	within 2 step cycles
local effect	less than 1 step cycle	less than 1 step cycle	less than 1 step cycle

Figure 17: The fault tolerance processes have fast response times.