Hindawi Publishing Corporation
Journal of Robotics
Volume 2010, Article ID 745834, 8 pages
doi:10.1155/2010/745834

*Research Article*

# Massive Sensor Array Fault Tolerance: Tolerance Mechanism and Fault Injection for Validation

**Dugan Um**

*Mechanical Engineering, Texas A&M University-Corpus Christi, 6300 Ocean Drive Unit 5797, Corpus Christi, TX 78412, USA*

Correspondence should be addressed to Dugan Um, dugan.um@tamucc.edu

Received 30 March 2010; Accepted 14 July 2010

Academic Editor: Tarek M. Sobh



As today's machines become increasingly complex in order to handle intricate tasks, the number of sensors must increase for intelligent operations. Given the large number of sensors, detecting, isolating, and then tolerating faulty sensors is especially important. In this paper, we propose fault tolerance architecture suitable for a massive sensor array often found in highly advanced systems such as autonomous robots. One example is the sensitive skin, a type of massive sensor array. The objective of the sensitive skin is autonomous guidance of machines in unknown environments, requiring elongated operations in a remote site. The entirety of such a system needs to be able to work remotely without human attendance for an extended period of time. To that end, we propose a fault-tolerant architecture whereby component and analytical redundancies are integrated cohesively for effective failure tolerance of a massive array type sensor or sensor system. In addition, we discuss the evaluation results of the proposed tolerance scheme by means of fault injection and validation analysis as a measure of system reliability and performance.

## 1. Introduction

As manipulators and actuators become more sophisticated, various sensory devices are added to assist actuators and studied for fault tolerance for autonomous operation for an extended period [1–3]. Furthermore, the increasing complexity of today's machines requires large number of sensors or an array of sensors for intricate operations. Sensor fault tolerance has extended its scope of study to sensor network or distributed sensor hardware and software [4].

One outstanding example of a study for sensor network fault tolerance is in [5], in which a distributed solution for a canonical task in wireless sensor networks is proposed via the binary detection of interesting environmental events. The work takes into account the possibility of sensor measurement faults and develops a distributed Bayesian algorithm for detecting and correcting such faults.

The state machine approach is a popular approach for implementing fault-tolerant services in distributed systems. A study on state machine approach [6] presents protocols for two different failure models: Byzantine and fail stop. A reconfigurable fault-tolerant sensor has been addressed in the paper as a means to remove faulty components and to integrate repaired components. As the demands for multiple nodes or data networks arise recently, a fault-tolerant system for a network of data has been studied with the intention toward a self-administering, fault-tolerant, and resilient system for chaotic networks under loads [7].

However, except for several studies in NASA [8, 9], multisensor fault tolerance or sensor array fault tolerance for robotic system has been largely unexplored. Given the dependency on sensors for decision making processes, a single faulty sensor in a sensor array may cause catastrophic failure of the entire system. For instance, an articulated robot in literatures [10, 11] can be disabled by one faulty sensor of the sensor array. Therefore, a fault tolerance system for multiple sensors or array of sensors for highly autonomous operations is of utmost importance for robotics applications.

We are specifically interested in fault tolerance for two-dimensional sensor arrays designed for use in robotic systems with the number of sensors on the order of hundreds or thousands or more [11] (see Figure 1). In a typical fault tolerance system, unexpected faults are accommodated by comparing the behavior of redundant components. In practice, the integrity of a complex system is achieved by replicating critical components [12]. In addition, voting

Figure 1: Close-up section of sensitive skin.

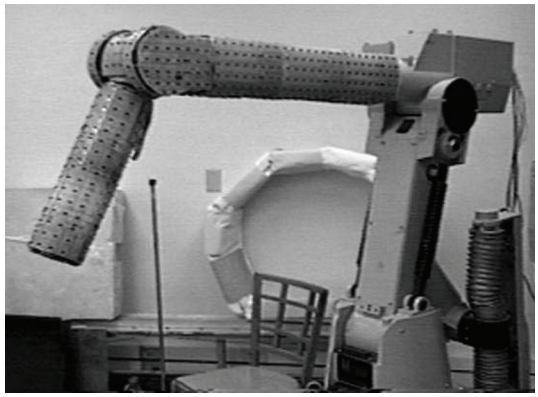

Figure 2: Implementation of the sensitive skin.

schemes with *component redundancy* (CR) are common for decision making processes in a typical fault tolerance system. For example, aircrafts often have more than one navigation controller, with a voting scheme used for critical decisions.

Similarly, in the multiclock synchronization problem, if one wishes to know the correct time at some location, several clocks may be used for fault tolerance. It has been known that $3m + 1$ clocks are necessary and sufficient to ensure nonfaulty performance in the presence of $m$ Byzantine faults [13]. Therefore, a component redundancy based approach, not only for hardware but also for software, has been a dominant trend in fault tolerance schemes. It is especially attractive in systems whose components are crucial for system operation and similar to each other in nature. Grouping multiple components employed for synergistic performance enhancement enables us to realize a component redundancy based fault tolerance scheme with no extra hardware cost. Sensor arrays exemplify such systems. The sensitive skin [11], for instance, is composed of thousands of duplications of infrared emitter and receiver pairs for system performance enhancement in unknown environment motion planning.

One alternative to the component redundancy is *analytic redundancy* (*AR*). *AR utilizes* the reference model of a target system. For example, in Bickmore's 1987 report, internal gas dynamics was estimated and used as a reference model for jet engine sensor diagnostics. In 1988, Mulcare wrote that the output of the system dynamics model is used as a third sensor value in a voting scheme that compares three sensor inputs to arrive at an error-free output for aircraft navigation. The advantage of *AR* is in its being free from the growing complexity associated with component redundancy. In this case, the original sensor resolution can be kept intact for full functionality of the sensory array. As is discussed in the previous literature [14], the AR-based tolerance scheme demonstrated useful results for sensitive skin fault tolerance. In this paper, however, we focus on the integration of component and analytic redundancy as a complete embodiment for massive sensor array fault-tolerance architecture.

The target system for the fault-tolerance system is shown in Figure 2. To ensure the performance of the proposed fault tolerant system, fault injection and validation technique is implemented for system performance analysis [15]. Consequently, we were able to measure the performance of the proposed tolerance system in a statistical basis. The test-bed of the proposed fault tolerance system is an articulated type robotic system with sensitive skin developed at the University of Wisconsin-Madison [11]. Sections 2, 3, and 4 describe the proposed fault tolerance system for massive sensor arrays utilizing both component redundancy and analytic redundancy in a cohesive architecture, while Section 5 demonstrates fault injection and evaluation results.

## 2. Our Approach for Massive Sensor Array Fault Tolerance

The sensitive skin developed for robotic motion planning is a massive 2D proximity sensor array covering the entire body of a 6 degrees of freedom (DOF) robot (Figure 2). Over 1000 sensors react to environments simultaneously to detect a potential collision with unknown obstacles. The failure of a single sensor, however, may cause the robot to move out of the work space disabling the robot itself. For unmanned operation for an extended period of time, the robot must be able to operate with faulty sensors, sacrificing system performance to some degree. Due to the excessive number of sensors in a massive sensor array, it is not a trivial matter to check each sensor. Instead, a fault tolerance means can be used on-line and off-line to accommodate faulty sensors. In this paper, we are interested in an on-line fault tolerance system for unmanned operations at a remote site such as unexplored planets or underwater.

In an integrated scheme, fault tolerance is taking place implicitly in the framework of component redundancy, while the analytic redundancy provides a means for faulty sensor detection and isolation. Sensor grouping is what we propose for the component redundancy-based fault tolerance system. In terms of performance measure, statistical framework is a favorable basis for fault tolerance system design and evaluation. Considering the fact that the primary means of fault accommodation of the sensitive skin is a grouping scheme, we defined a new statistical process measure, namely,

Mean Number of Faulty sensors in a Group (MNFG), defined as

$$\text{MNFG}(\%) = \frac{\text{No. of faulty sensors in a group}}{\text{Total no. of sensors}} \times 100. \quad (1)$$

With MNFG, for a given period of operation in a remote site, one can evaluate the average number of faulty sensors in a group. For instance, the MNFG of five faulty sensors for 100 total sensors in a group is 5%. An experiment is performed to project the change of the MNFG with respect to MTBF (Mean Time Between Failure) for 100 sensors as illustrated in Figure 3(a). For the experiment, the number of sensors in a group is adjusted dynamically to $2m + 1$ (see Section 3), when $m$ is equal to the number of faulty sensors.

Notice that the number of faulty sensors increases dramatically as it passes the MTBF. This is due to the fact that the MTBF is a statistically averaged time of fault occurrence. Notice also that the MNFG is close to a logarithmic function over time converging toward 50% because $\lim_{m \to \infty}(m/(2m+1)) = 0$, meaning that the MNFG prediction curve is only valid up to a certain number of faulty sensors (425 from Figure 3(b)), otherwise the value of $m$ will be a negative integer. This is because, according to the definition, MNFG is less than 0.5, and from the fitting curve, y(425) is equal to 50%.

In light of the configuration of the sensitive skin (Figure 4), more than 4 faulty sensors in a group turned out to be impractical for implementation since a sensor in the same group may result in a different side of the robotic arm. For the dynamic regrouping process, one can use the following equation to predict the number of faulty sensors and calculate the number of sensors per group:

$$m = \frac{\text{MNFG}}{1 - 2 \cdot \text{MNFG}}, \qquad N_{\text{sensor}} = \frac{m}{\text{MNFG}}, \quad (2)$$

where $N_{\text{sensor}}$ stands for the number of sensors in a group. Notice that the value of $m$ must be the rounded-up value of the above equation. The equation above is inspired by replacing the "Total no. of sensors in a group" in (1) by $2m + 1$, which will be discussed at Section 4 in more detail. That said, we studied the fault tolerance strategy for a massive sensor array as outlined below.

*Assumption 1.* Faulty sensors are evenly distributed around the sensor array (optimistic assumption).

(1) Predict MNFG($t$) for current time of operation since the inception.

(2) If $m/(2m + 1) < MNFG(t)$, then dynamically regroup sensors,

where $t$ is the elapsed time and $2m + 1$ is the desirable total number of sensors to tolerate $m$ faulty sensors in a group (see Section 4). The assumption made above is optimistic, but legitimate in that faults are taking place rather in a distributed fashion. The proposed tolerance scheme may not work effectively otherwise due to the nature of grouping scheme. The 2nd step in the strategy is to assure that the number of sensors in a group is always sufficient to tolerate estimated faulty sensors by MNFG in a group. If one can detect the number of faulty sensors in a group, then the dynamic regrouping (Section 3) can take place in practical manner, preventing any unnecessary regrouping, or can minimize performance degradation due to inopportune or delayed regrouping. In Section 4; we discuss how to utilize the analytic redundancy for faulty sensor detection.

## 3. Dynamic Regrouping Strategy and Voting Scheme

In order to handle faulty sensors in a remote site, we propose a novel sensor grouping method, namely, the "closest neighborhood grouping (CNG) scheme." The CNG scheme selects sensors in the neighborhood by checking closeness in distance. Once the number of sensors necessary for a group is determined by MNFG projection, then the algorithm regroups the sensor array. Following is the grouping algorithm outlined for the closest neighborhood grouping scheme.

*Definition 1.* $N_{\text{sensor}}$ is the number of sensors necessary for a group (projection by MNFG). $N_{\text{group}}$ is the number of sensor groups. $N$th degree neighborhood sensors are sensors forming a square lattice around a reference sensor at the center (see Figure 4(a)). Available sensor is a sensor not included in any group.

*Algorithm 1.* (1) Find an available sensor at the upper left corner of the sensor matrix (array) and mark it as a reference sensor.

(2) With the reference sensor at the center, check $N$th degree neighborhood sensors clockwise starting from the upper left corner to find an available sensor. ($N$ starts from 1).

(3) If all the sensors belonging to $N$th degree neighborhood are checked, then continue on to the next degree neighborhood sensor group.

(4) Add an available sensor found in the previous step and add it to the current group.

(5) If the number of new sensors is equal to $N_{\text{sensor}}$, then go to step (1) to form a new sensor group; otherwise go to step (2) to continue.

Figure 4(b) illustrates an example of the closest neighborhood grouping scheme when the number of sensors is five. Black dots are reference sensors in each group from which $N$th degree neighborhood sensors are identified. Given each group of the sensor array, we installed a virtual sensor to represent each group. The location of a virtual sensor was placed at the area center of each group. Some sensors may have been left behind without being included in a group at the end of the grouping operation. They were later redistributed into neighboring sensors heuristically (the left over sensor was included in the last group, thus $N_{\text{group}}$ remains 11 in Figure 4(b)).

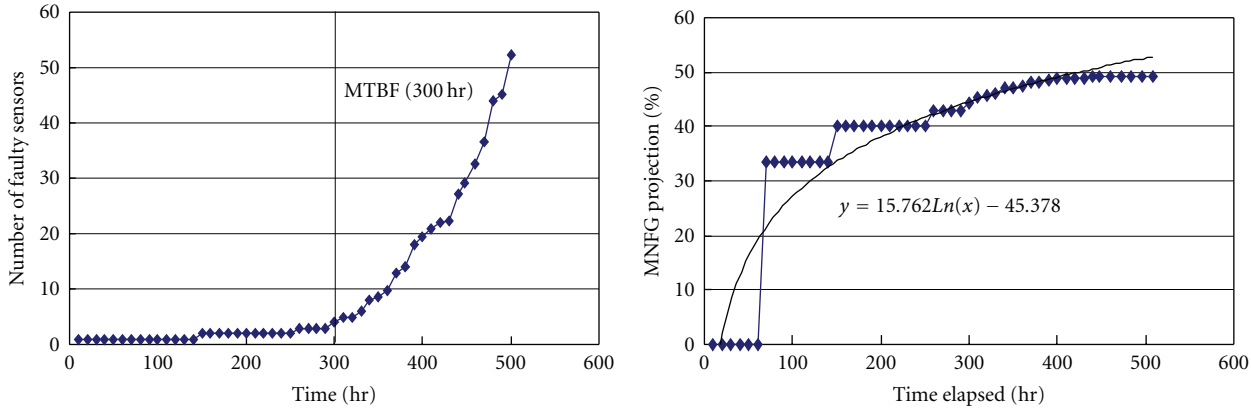


Figure 3: (a) Trend of no. of faulty sensors and (b) experiment result of MNFG projection.

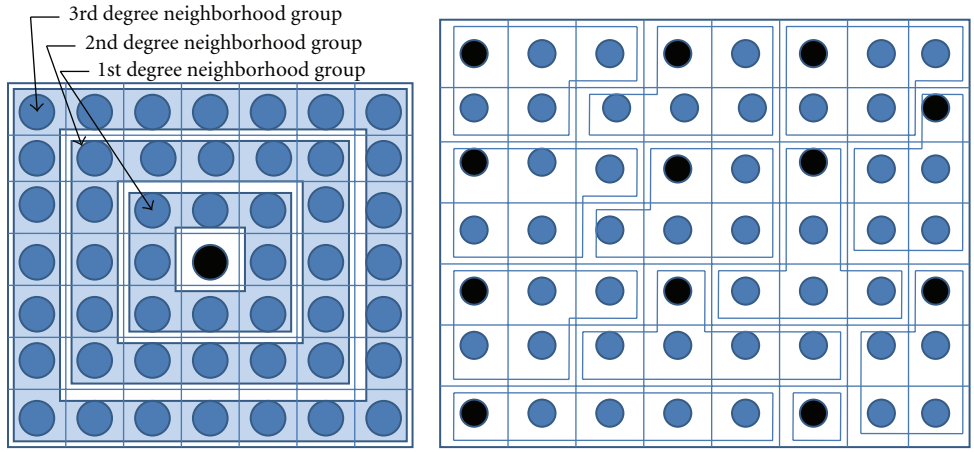


Figure 4: (a) Neighborhood definition and (b) an example of closest neighborhood sequence grouping scheme with $N_{\text{sensor}} = 5$, $N_{\text{group}} = 11$.

## 4. Voting Scheme

Given the dynamic grouping scheme discussed so far, we will continue to study the voting scheme for fault tolerance. The idea behind the voting scheme is that the value of the virtual sensor should always be within the range of the readings of good sensors. If this is accomplished, we will say that the reading of a sensor, even a faulty sensor in the group, has no significant effect on the system performance. The voting scheme we propose for the massive sensor array is what is known as median value scheme. That is, the virtual sensor that represents the group's estimated sensor value, denoted MID, is as

$$V_k^{\text{virtual}} = \underset{i,j \in k}{\text{MID}}\left(v_{ij}\right), \quad i, j \in k, \tag{3}$$

where $V_k^{\text{virtual}}$ is the virtual sensor value for a group $k$ and $v_{ij}$ is the reading from a real sensor located at row $i$ and column $j$ on the (two-dimensional) skin. Apparently, the resolution of the sensitive skin decreases as the number of sensors in a group grows; more sensors take a larger area, which, in turn, makes the gap between virtual sensors larger. This creates an incentive to minimize the number of sensors per group to maximize the overall skin resolution.

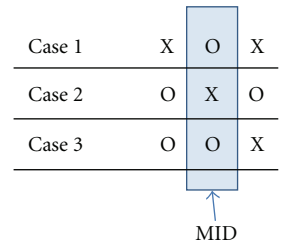


Figure 5: The choice of MID value for the case of three sensors per group.

Consider a grouping example with three sensors per group (Figure 5). Once the sensor readings are sorted in descending order, there are three possible combinations that may appear. Each row in the figure represents one case; the right column represents the highest sensor readings, the left

column, the lowest reading; symbol "X" indicates a faulty sensor, symbol "O"-a good sensor. The value of the virtual sensor, MID, will be then the one in the central column. Notice that, in Cases 1 and 3, MID will correspond to an actual reading of a good sensor. In Case 2, though the virtual sensor value is determined by a faulty sensor reading, it is still acceptable as it is within the range of the good sensor readings, and thus represents the fault-free value of a sensor group. This suggests that three sensors per group are sufficient to tolerate one Byzantine fault (a fault type that would cause a malicious behavior to the tolerance system) per group.

In light of the nonlinear property of the median filter, we are interested in a query as to whether the median statistics guarantees nonfaulty behavior of the proposed closest neighborhood grouping scheme to tolerate a given number of Byzantine faults. To that end, we first define the following variables to establish a theoretical basis of the proposed median filter:

$v_k$ is the value of a group $k$ ($G_k$);

$v_{\text{MID}}$ is the median value of the sensor in a group $k$;

$v_{\text{good}}$ is the value of a good sensor, while $v_{\text{bad}}$ is the value of a bad sensor;

$V_{\text{left}} = \{v \mid v \le v_{\text{MID}},\ v \in G_k\}$, $V_{\text{right}} = \{v \mid v \ge v_{\text{MID}},\ v \in G_k\}$.

With the definition of variables above, two acceptable cases for MID scheme are identified as below.

(1) If $v_k = v_{\text{good}}$, then the value of the virtual sensor is acceptable.

(2) If $v_k = v_{\text{bad}}$, $\exists v_{\text{good}} \in V_{\text{left}}$, and $\exists v_{\text{good}} \in V_{\text{right}}$, then the value of the virtual sensor is acceptable.

That said, the following lemma is true.

**Lemma 1.** *It takes minimum $2m + 1$ sensors per group to tolerate $m$ Byzantine faults by closest neighborhood grouping scheme.*

*Proof.* We have the following cases.

*Case 1.* For $2m + 1$ sensors, total number of possible combinations of $2m + 1$ sensors with $m$ Byzantine faults is

$$ {}_{2m+1}\mathrm{C}_m = \frac{(2m+1)!}{m!(m+1)!}, \tag{4} $$

while $m!$ is the number of combination for bad sensors and $(m + 1)!$ is the number of combination for good sensors. If $v_{\text{MID}} = v_{\text{good}}$, then the number of possible combination becomes

$$ {}_{2m+1}\mathrm{C}_m \big|_{v_{\text{MID}} = v_{\text{good}}} = \frac{(2m+1-1)!}{m!(m+1-1)!} = \frac{(2m)!}{m! \cdot m!}. \tag{5} $$

*Case 2.* If $v_{\text{MID}} = v_{\text{bad}}$, then the number of possible combination is

$$ {}_{2m+1}\mathrm{C}_m \big|_{v_{\text{MID}} = v_{\text{bad}}} = \frac{(2m+1-1)!}{(m-1)!(m+1)!} = \frac{(2m)!}{(m-1)!(m+1)!}, \tag{6} $$

while $(m - 1)!$ is the number of combination for bad sensors and $(m + 1)!$ is the number of combination for good sensors.

Now, if we add (5) and (6),

$$ {}_{2m+1}\mathrm{C}_m \big|_{v_{\text{MID}} = v_{\text{good}}} + {}_{2m+1}\mathrm{C}_m \big|_{v_{\text{MID}} = v_{\text{bad}}} = \frac{(2m)!}{m!m!} + \frac{(2m)!}{(m-1)!(m+1)!} = \frac{(2m+1)!}{m!(m+1)!}. \tag{7} $$

Thus;

$$ {}_{2m+1}\mathrm{C}_m \big|_{v_{\text{MID}} = v_{\text{good}}} + {}_{2m+1}\mathrm{C}_m \big|_{v_{\text{MID}} = v_{\text{bad}}} = {}_{2m+1}\mathrm{C}_m. \tag{8} $$

□

Therefore, the number of two acceptable cases is equal to the total combinations of possible cases with $2m + 1$ sensors. This proves Lemma 1.

Notice though that, since the effect of the faulty sensor will be neutralized in this case, the faulty sensor itself will not be identified. Another drawback of this scheme is that the sensor resolution will be degraded, since the virtual sensors whose readings replace those of real sensors are spaced less densely. In order to provide logical basis of the closest neighborhood grouping algorithm, the procedure to determine $N_{\text{sensor}}$ (the number of sensors in a group for the dynamic regrouping scheme) needs to be discussed. For a given MNFG and total number of sensors, say $N_T$, the following two assumptions are made for further discussion.

*Assumption 2.* Errors predicted by MNFG for the entire sensor array are concentrated in one area of the sensor matrix.

*Assumption 3.* Errors predicted by MNFG for the entire sensor array are spread evenly throughout different groups.

The two assumptions described above are two extreme cases of faulty sensor distribution. Due to the unpredictability of the location of the faulty sensors in a massive sensor array, the fault tolerance system must be able to accommodate two extreme cases in the statistical basis to guarantee the system performance under the appropriate assumption. To that end, we illustrate how to determine the number of sensors in a group for the closest neighborhood grouping scheme.

(a) For Assumption 2, since all the faulty sensors are in one group, the number of sensors in a group becomes $2(\text{MNFG} \times N_T) + 1$.

(b) For Assumption 3, if the number of sensor group is less than MNFG $\times$ $N_T$, then regroup the sensor array by Algorithm 1.

If $N_{\text{group}}$ is larger than the projected total faults by MNFG$\times N_T$ then Assumption 3 can not be satisfied, since initially $N_{\text{group}}$ with $m = 1$ stands for the total number of faulty sensors, thus regrouping is necessary. Therefore, the algorithm introduced provides the new value of $N_{\text{group}}$ and $N_{\text{sensor}}$ for the dynamic regrouping operation of the closest neighborhood grouping scheme.

Table 1: Fault injection and validation test (30 run per each fault type).

| Fault types injected | Percent of being detected and isolated by analytic redundancy | Percent of being tolerated by voting scheme |
|---|---|---|
| Byzantine | 17% | 90% |
| Omission | 63% | 97% |
| Stuck-at | 0% | 97% |
| High frequency noise | 93% | 93% |
| Delay | 63% | 90% |

$N_{\text{sensor}} = 3$ (can accommodate 1 Byzantine fault in each group by Lemma 1)
If $N_{\text{group}} > \text{MNFG} \times N_T$ then no change is required
Else
  While $N_{\text{group}} < \text{MNFG} \times N_T$
    $m = m + 1$
    $N_{\text{sensor}} = 2m + 1$
    $N_{\text{group}} = N_T / N_{\text{sensor}}$
  End While
End If

Algorithm 1

## 5. Fault Detection by Analytic Redundancy

The aforementioned voting scheme for the massive sensor array by component redundancy allows an extended period of operation until the next maintenance schedule. The voting scheme, however, does not isolate a faulty sensor since the proposed voting scheme implicitly neutralizes faulty sensors. In the previous literature [14], a fault isolation technique by analytic redundancy is demonstrated to show how to isolate a faulty sensor in a massive sensor array for maximum resolution or for immediate repair during the maintenance period. In brief, since the skin sensors are attached on the robot, sensors move when the robot is in motion. This suggests that one can infer the motion of a robot by observing sensor signals as sensors detect stationary objects. Inversely, one may also be able to predict sensor readings from knowing how the robot moves in space.

In principle, the change in the sensor reading cannot exceed the change in motion vector in a static environment unless it is faulty, which establishes the basis of the on-line error detection method. That is,

$$V_c + \delta < V_e \text{ implies "sensor is faulty,"} \tag{9}$$

where $V_c$ stands for a calculated sensor velocity from robot dynamics, $V_e$ stands for an estimated sensor velocity from sensor reading and $\delta$ is the small number determined impartially. The reason for the "less than" inequality in the detection algorithm is to prevent false alarm. For instance, if there is no obstacles in the surrounding environment at a moment, $V_e$ can be zero even for a good sensor.

To the contrary, when the relative motion between the sensor and the environment is known in the sensing range, then, sensor reading should be proportional to the motion vector of the sensor, which establishes the basis of the off-line error detection method. That is,

$$V_e + \delta < Vc \text{ implies "sensor is faulty."} \tag{10}$$

When *CR* and *AR* are combined in a cohesive manner, the *AR*-based detection scheme can provide inputs to the *CR*-based tolerance scheme. That is, instead of assuming the number of faulty sensors in a group ($m$) by MNFG, the *AR*-based detection scheme can provide the exact number of faulty sensors and their location, thereby; the *CR*-based tolerance scheme can immediately regroup the sensor array accordingly by the CNG method (Section 3). This allows a synthetic integration of the *AR*- and *CR*-based fault tolerance as a complete embodiment for the sensitive skin fault tolerance. However, since the types of faulty behaviors detectable by *AR* are limited, it needs to be combined into the *CR*-based tolerance system for some degree of performance advantage.

## 6. Fault Injection and Fault Tolerance System Validation

The fault tolerance experiment setup is shown in Figure 6, with fault injection point and behavior monitoring means indicated in the figure. With the setup illustrated, various fault types of sensitive skin were examined to test the tolerance mechanism. Tolerance algorithms are embedded in the motion planner in the figure. During each experiment, a set of faults ($F$ set) is injected in conjunction with the activity of the target systems ($A$ set). That is, $A$ set specifies the target system behavior determining the error patterns corresponding to the fault types in the $F$ set, thus forming input/output sets to the fault tolerance system.

The output domain for a given input domain includes readouts ($R$ set), which is the collection of output data, and

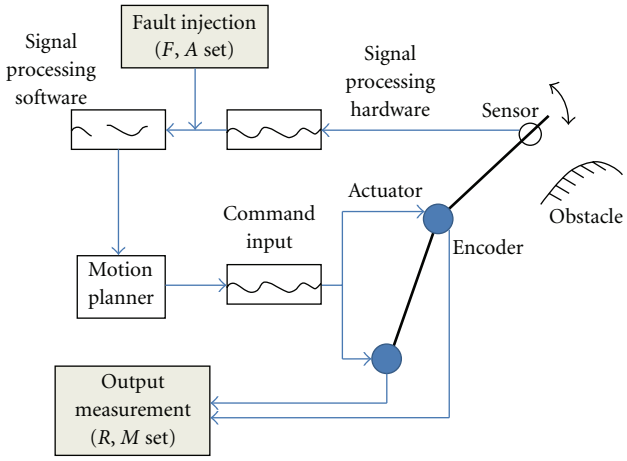


FIGURE 6: Fault injection test experiment setup.

measurement data ($M$ set), the result of the measured data in $R$ set. All together, they thus form FARM sets (see [15] for more detail) constituting the major attributes to fully characterize a fault injection test sequence.

For simplicity and to facilitate the relationship with the proposed tolerance model, we perform tests in a finite observation domain rather than using a cumulative distribution function. The result of the experiment, therefore, is the probability of fault detection and fault tolerance with respect to the identified fault models.

As for the $F$ sets, the five common fault models (Byzantine, Omission, Stuck-at, High frequency noise, Delay) are taken into consideration for the experiments. Due to the difficulty of modeling the Byzantine fault, it is modeled as a random combination of the rest of the faults. Although in some studies, a high-frequency noise fault is regarded as a stuck-at fault, and we discriminated it from the stuck-at fault for diverse analytic results. The random variable associated with each predicate in the injection sequence affected the fault tolerance process and the consequential faulty behavior were observed assuming finite reaction time for failure. In the experiments, the system behaviors ($R$, $M$ set) were observed by the consequences of the tolerance behavior.

The test result of the experiments is shown in Table 1 demonstrating correlated probability of fault detection and fault tolerance depending on each fault type. Different fault types are injected 30 times for higher statistical confidence. Notice in the table that the probability of being tolerated by the voting scheme is equal to or greater than 90% for most of the fault types considered for the tolerance system. According to Lemma 1, the component redundancy-based voting scheme should be able to tolerate any type of fault. The probability, however, is not 100% because of the fact that each transition predicate in the fault injection sequence is driven by a random variable with assigned probability.

In summary, the proposed component redundancy-based voting scheme demonstrates good tolerance rates for most of the known faults observed, but the detection scheme worked only with specific fault types.

## 7. Conclusion

In this paper, a fault tolerance scheme for a massive sensor array is introduced. The proposed "closest neighborhood grouping scheme" is based on component redundancy in the context of median statistics and is proved to be capable of tolerating faults in massive sensor array experimentally. In order to develop a tolerance strategy, a statistical measure, namely, Mean Number of Faulty sensors in a Group (MNFG) is introduced. Based on two extreme assumptions for faulty sensor distribution, the dynamic sensor regrouping algorithm calculates the sufficient number of sensors per group utilizing MNFG, so that the closest neighborhood grouping scheme forms new sensor groups to ensure the highest virtual sensor resolution possible.

The fault detection and isolation scheme, developed in the framework of analytic redundancy, demonstrates its capability of detecting and isolating faulty sensors. Together with the voting scheme, it becomes a fault-tolerant architecture accommodating fault detection, isolation, and control (FDIC) system. To be more effective in the fault tolerance experiment, FARM sets are used for fault injection and sequential state transition. The fault injection accelerated fault tolerance validation and allowed fast observation of the system performance in the presence of specific faults. The experimental results validated the performance of the voting scheme. As a complete embodiment with both component and analytic redundancies, the proposed tolerance algorithm helps maintain the health of a massive sensor array used for robotic operations in unknown environments such as under water or unexplored planets.

## References

[1] V. J. Lumelsky, M. S. Shur, and S. Wagner, "Sensitive skin," *IEEE Sensors Journal*, vol. 1, no. 1, pp. 41–51, 2001.

[2] A. De Luca and R. Mattone, "Actuator failure detection and isolation using generalized momenta," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 634–639, September 2003.

[3] M. L. McIntyre, W. E. Dixon, D. M. Dawson, and I. D. Walket, "Fault detection and identification for robot manipulators," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 4981–4986, May 2004.

[4] Y. Liu and S. Y. Nof, "Fault-tolerant sensor integration for micro flow-sensor arrays and networks," *Computers and Industrial Engineering*, vol. 54, no. 3, pp. 634–647, 2008.

[5] B. Krishnamachari and S. Iyengar, "Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," *IEEE Transactions on Computers*, vol. 53, no. 3, pp. 241–250, 2004.

[6] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: a tutorial," *Computing surveys*, vol. 22, no. 4, pp. 299–319, 1990.

[7] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing," Tech. Rep. UCB/CSD-01-1141, Computer Science Division (EECS), University of California at Berkeley, Calif, USA, 2001.

[8] T. W. Bickmore, "Real-time sensor data validation. Data validation final report," NASA contract report 195295, Aerojet propulsion division, Sacramento, Calif, USA, 1987.

[9] D. B. Mulcare, L. E. Downing, and M. K. Smith, "Analytical sensor redundancy assessment: final report," Tech. Rep. N88-22901, National Technical Information Service, NASA/Springfield, Va, USA, 1988.

[10] E. Cheung and V. Lumelsky, "A sensitive skin system for motion control of robot arm manipulators," *Robotics and Autonomous Systems*, vol. 10, no. 1, pp. 9–32, 1992.

[11] D. Um, B. Stankovic, K. Giles, T. Hammond, and V. Lumelsky, "Modularized sensitive skin for motion planning in uncertain environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 7–12, May 1998.

[12] T. Anderson and P. A. Lee, *Fault Tolerance: Principles and Practice*, Prentice Hall, Englewood Cliffs, NJ, USA, 1981.

[13] I.-K. Rhee, J. Lee, J. Kim, E. Serpedin, and Y.-C. Wu, "Clock synchronization in wireless sensor networks: an overview," *Sensors*, vol. 9, no. 1, pp. 56–85, 2009.

[14] D. Um and V. Lumelsky, "Fault tolerance via analytic redundancy for a modularized sensitive skin," *International Journal of Robotics and Automation*, vol. 15, no. 4, pp. 155–165, 2000.

[15] J. Arlat, A. Costes, Y. Crouzet, J. Laprie, and D. Powell, "Fault injection and dependability evaluation of fault-tolerant systems," *IEEE Transactions on Computers*, vol. 42, no. 8, pp. 913–923, 1993.