[4] J. Canny, "Finding edges and lines in images," *Rep. AI-TR-720, AI lab, MIT*, 1983.
[5] ——, "A computational approach to edge detectio,." *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 8, no. 6, pp. 679–698, 1986.
[6] G. Chen and Y. H. Yang, *Edge Detection by Regularized Cubic B-Spline Fitting*, Research Report, #91-8, Computer Vision Lab, Dept. of Computational Science, University of Saskatchewan, 1991.
[7] R. O. Duta and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons, 1973.
[8] P. H. Eichel and E. J. Delp, "Quantitative analysis of a moment-based edge operator," *IEEE Trans. Syst. Man Cyber.*, vol. 20, no. 1, pp. 59–66, 1990.
[9] R. M. Haralick, "Edge and region analysis for digital image data," *Computer Graphics and Image Processing*, vol. 12, pp. 60–73, 1980.
[10] ——, "Digital step edges from zero crossing of second directional derivatives," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 6, no. 1, pp. 58–68, 1984.
[11] R. M. Haralick and L. Watson, "A facet model for image data," *Computer Graphics and Image Processing*, vol. 15, pp. 113–129, 1981.
[12] L. J. Kitchen and A. Rosenfeld, "Edge evaluation using local edge coherence," *IEEE Trans. Syst. Man Cyber.*, vol. 11, no. 9, pp. 597–605, 1981.
[13] Y. Li and Y. Huang. *Numerical Approximation*, People's Education Press, China, 1978.
[14] D. Marr and E. Hildreth, "Theory of edge detection," *Proc. Royal Society London, ser. B*, vol. 207, pp. 187–217, 1980.
[15] V. S. Nalway and T. O. Binford, "On detecting edges," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 8, no. 6, pp. 699–714, 1986.
[16] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature*, vol. 317, pp. 314–319, 1985.
[17] J. M. S. Prewitt, *Object Enhancement and Extraction, in Picture Processing and Psychopictorics*, B. S. Kipkin and A. Rosenfeld, Eds, Academic Press, 1970.
[18] L. G. Roberts, *Machine Perception of Three-Dimensional Solids, in Optical and Electro-Optical Information Processing*, J. T. Tippett *et al.*, Eds, MIT Press, 1965.
[19] S. Sarkar and K. L. Boyer, "On optimal infinite impulse response edge detection filters," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, no. 11, pp. 1154–1171, 1991.
[20] ——, "On the localization performance measure and optimal edge detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 16, no. 1, pp. 106–108, 1994.
[21] H. D. Tagare and R. J. P. deFigueiredo, "On the localization performance measure and optimal edge detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, no. 12, pp. 1186–1190, 1990.
[22] ——, Reply to "on the localization performance measure and optimal edge detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 16, no. 1, pp. 108–110, 1994.
[23] H. L. Tan, S. B. Gelfand, and E. J. Delp, "A comparative cost function approach to edge detection," *IEEE Trans.Systems, Man, and Cybernetics*, vol. 19, no. 6, pp. 1337–1349, 1989.
[24] ——, "A cost minimization approach to edge detection using simulated annealing," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, no. 1, pp. 3–18, 1991.
[25] A. N. Tikhonov and V. Y. Arsenin. *Solution of Ill-Posed Problems*, Winston and Wiley, 1977.
[26] V. Torre and T. A. Poggio, "On edge detection." *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 8, no. 2, pp. 147–163, 1986.
[27] S. Venkatesh and L. J. Kitchen, "Edge evaluation using necessary components," *CVGIP: Graphical Models and Image Processing*, vol. 54, no. 1, pp. 23–30, 1992.
[28] Z. Wu and Y. H. Yang. *An Experimental Study in the Use of Robust Statistics in Edge Detection and Performance Evaluation*, Research Report, #93–4, Computer Vision Lab, Dept. of Computational Science, University of Saskatchewan, 1993.

A General Computational Framework for Distributed Sensing and Fault-Tolerant Sensor Integration

S. S. Iyengar and L. Prasad

***Abstract*—This paper proposes an abstract framework to address the problem of fault-tolerant integration of information provided by multiple sensors. Extending our earlier results[10], this paper presents a formal description of spatially distributed sensor networks, where i) clusters of sensors monitor (possible overlapping) regions of the environment; ii) sensors return measured values of a multi-dimensional parameter of interest; and iii) uncertainities associated with a sensor output are represented by a connected subset in the parameter space. A method to obtain interval estimates of components of the actual parameter vector is developed, wherein information from faulty sensors (whose uncertainty subspaces do not contain the true value of the parameter) are filtered out.**

The problem addressed involves combining interval estimates of sensor outputs into a best intersection estimate of outputs. The sensor fault model used assumes most faults are "tame." That is, faults cluster in the neighborhood of the correct values. The procedure of this paper, assuming this fault model, is superior to earlier work by Marzullo.

To test the theoretical analysis of the computational framework proposed in our paper, we have developed a modular parameter-driven simulator SIMDSN for the fault-tolerant integration of abstract sensor interval estimates. The simulator uses the well-known Monte-Carlo technique to generate random correct and tamely faulty intervals. The results of our algorithm and Marzullo's are compared. We conclude based on the simulation that the union of the disjoint intervals (if any are disjoint) determined by our algorithm and the gaps between these intervals together can never exceed the width of the final output of Marzullo's method. Thus, while Marzullo's method specifies a rather large interval as the final output of the integration process, our method subdivides this interval into (possibly) disjoint intervals each with an associated *reliability measure*.

I. Introduction

A distributed sensor network (Distributed Sensor Network) consists of several sensors distributed spatially, which collect data from their surroundings. The data from the sensors are received by processors which put together the information obtained from the sensors into a desirable form and send the integrated information for further processing and utilization by other processors and actuators. The distributed sensor integration problem has been shown to have wide ranging applications in many areas such as surveillance and tracking of moving objects in military applications. medical imaging, space application, etc.

A Distributed sensor network consists of several clusters of sensors, each cluster feeding into a processor. Each cluster is assigned to a region in space from which it collects data. All the sensors in a cluster observe the same phenomenon, i.e. read the same value in the region to which they are assigned. This redundancy is introduced to ensure fault-tolerance since some of the sensors in a cluster may be faulty. The set of all the regions, to each of which a sensor cluster is assigned, constitutes the space under observation by the Distributed Sensor Network. This must include the cooperative solution of problems by a decentralized and loosely coupled collection of processors, each of which integrates information received from a cluster of spatially distributed sensors into a manageable and reliable output for further

Manuscript received December 16, 1992; revised March 1, 1994, April 27, 1994 and June 3, 1994. This research in part is partially supported by Board of Regents (LEQSF-RD-A-04, 1990) grant and by ONR N000014-91-J1306.

The authors are with the Robotics Research Laboratory, Department of Computer Science, Louisana State University, Baton Rouge, LA 70803 USA.

IEEE Log Number 9406641.

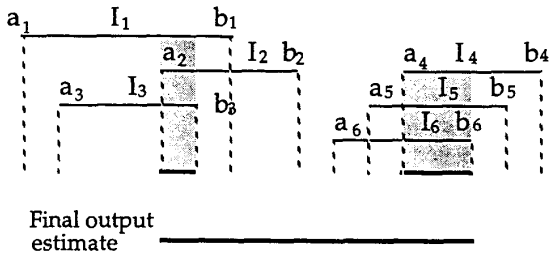


Fig. 1. Integration of interval estimates.

integration at a higher level. Integration of information at the sensor level requires techniques to be developed to abstractly represent and integrate sensor information. Further these techniques have to be robust in the sense that even if some of the sensors are faulty, the integrated output should still be reliable. One of the features that distinguishes distributed sensor processing is its demand for computational framework for fault-tolerant sensor integration.

The work described in this paper is allied to the methodology provided by Marzullo in masking failure of multidimensional sensors. Our method coincides with Marzullo's [6] as far as considering intersections of the geometric sets representing sensor outputs is concerned. Marzullo encloses these intersections by a connected minimal set, and obtains bounds for the size of this set in terms of bounds for the number of faulty sensors.

Our approach to Distributed Sensor Network problem differs from the previous work [6] in that we regard sensors as giving point-set outputs, with no probability or weight function associated with them. We then look at the problem of integrating these geometric sets to obtain a set which contains with high reliability the point corresponding to the actual value observed by the sensors, even though some of the sensors may be inaccurate or faulty.

The sensors in general have uncertainties in their readings and may not yield a definite value but 'hover' over a range of values. Thus it is appropriate to look upon a sensor's output as a contiguous set of values, rather than as a point value. The parameter being observed may in general be a vector (eg. velocity, position, etc. The processors fed by the various sensor clusters integrate the information to obtain a reliable estimate of the parameter(s) being measured in the respective regions. This information is local to the region and may be used to effect an action in that region or be integrated at higher processors to obtain a global picture or to make a global decision.

The organization of this paper is as follows. Section II discusses related work in this area. In Section III, we describe an abstract representation of Distributed Sensor Network. A scheme for patching together local processor information to obtain a global picture of the parameter measured is given in Section IV. Finally, in Section V, we compare the performance of our model to that of Marzullo's.

II. Related Work

Refer to our earlier papers for preliminary definitions on sensors [4], [5]. Also, please refer Marzullo's [6] paper about his work. Fig. 1 describes integration of interval estimates using [6].

A. Motivation and Summary of This Work:

While the problem of distributed sensing has been addressed often with reference to specific kinds of tasks and sensors, there has been no general formulation of the problem in a formal sense. Following the work in [4], [5] and [6], we propose a formal description of an abstract Distributed Sensor Network that subsumes the description suggested and employed in the above mentioned papers and includes

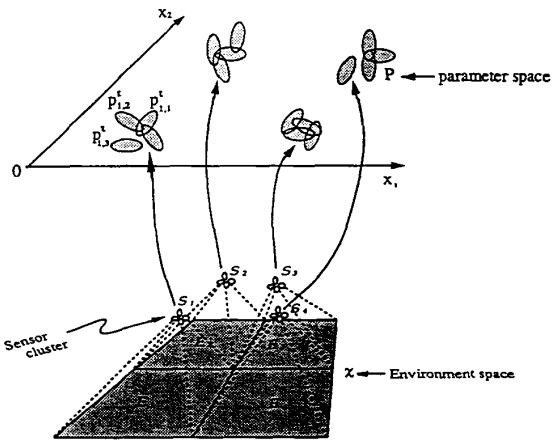


Fig. 2. Schematic representation of an abstract DSN.

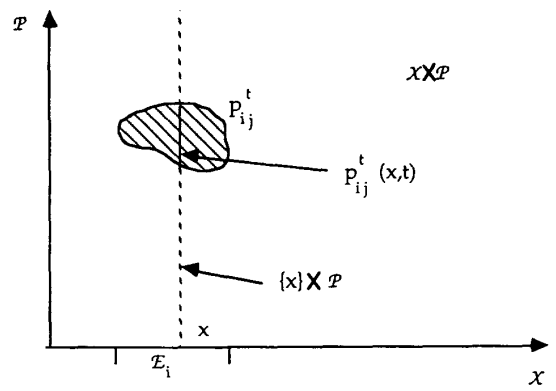


Fig. 3. The subsets $p^t_{i,j}$ corresponding to the sensor $\sigma^t_{i,j}$

a wide class of Distributed Sensor Networks. It is to be noted that this formal description is a very theoretical framework for characterizing a general Distributed Sensor Network and addressing the general computational features and problem of a Distributed Sensor Network and looks upon sensor information as geometric point-sets rather than as distributions. It is a symbolic description capturing the features of a large class of Distributed Sensor Networks and not an implementation of the Distributed Sensor Networks in the class implicit in the characterization. Indeed, the exact nature of the implementation would depend upon the specific Distributed Sensor Network and the nature of the phenomenon being sensed and the sensors. Yet its abstract structure would be concordant with the characterization to be given.

Keeping the above remarks in mind, we present an abstract representation of a general Distributed Sensor Network. We first introduce the formal definitions with notation. The notation and definitions will be followed by explanations and examples.

III. Abstract Representation of a Distributed Sensor Network

A. Notation and Definitions

We Formally introduce here the abstract set up for a general distributed sensor network:

Definition 1: An abstract distributed sensor network is a 4-tuple $(\mathcal{X}, \{\mathcal{E}_i\}_{i=1}^m, \{\mathcal{S}_i\}_{i=1}^m, \mathcal{P})$, where:

1) $\mathcal{X}$ is the space under observation by the sensors of the Distributed Sensor Network, called the *environment space*. Clusters of sensors are allocated to various regions of $\mathcal{X}$, and these clusters pick up data from their respective regions and transmit them to the processors to which they are connected. The collection of regions to which the various sensor clusters are allocated constitute the space $\mathcal{X}$.
2) $\mathcal{P}$ is the space of all possible values of the parameter being measured, called the *parameter space*. If the parameter being measured is a k-dimensional vector, then $\mathcal{P}$ is the Euclidean Space of dimension k. The value of each sensor is represented by a subset of $\mathcal{P}$. The exact nature of these subsets will be discussed later.
3) $\{\mathcal{E}_i\}_{i=1}^m$ is a collection of subsets of the environment space $\mathcal{X}$ such that $\cup_{i=1}^m \mathcal{E}_i = \mathcal{X}$. The collection $\{\mathcal{E}_i\}_{i=1}^m$ is called a *chart* on $\mathcal{X}$ (Fig. 2). Each subset $\mathcal{E}_i$ is allocated a cluster of sensors which obtain data from $\mathcal{E}_i$ and report to a processor.
4) $\{\mathcal{S}_i\}_{i=1}^m$ is a collection of m sensor-clusters, each $\mathcal{S}_i$ being assigned to an $\mathcal{E}_i$. Each $\mathcal{S}_i$ is a cluster of n_i abstract sensors $\mathcal{S}_i = \{\sigma_{i,j}^t\}_{j=1}^{n_i}$, where each abstract sensor $\sigma_{i,j}^t$ is a time-varying map which maps the set $\mathcal{E}_i$ onto a subset $p_{i,j}^t$ of $\mathcal{X} \times \mathcal{P}$. The values of the subsets $p_{i,j}^t$ will be described soon. Thus at any instant of time t, we have the collection of $\{p_{i,j}^t\}_{j=1}^{n_i}$ of $\mathcal{X} \times \mathcal{P}$ as the abstract sensor estimates of the sensors $\{\sigma_{i,j}^t\}_{j=1}^{n_i}$ of the parameter values observed in $\mathcal{E}_i$ at time t.

   The sensor $\sigma_{i,j}^t$ senses data about a parameter at each point $x \times \mathcal{E}_i$ since a sensor in general has some uncertainty in its reading, it is realistic to expect it to give a range of values for the parameters' value at the point x instead of a single value. Thus we assume that for each $x \in \mathcal{E}_i, \sigma_{i,j}^t$ gives a connected subset of values representing the value of the parameter at x. If the actual value of the parameter at x belongs to this set of values, then $\sigma_{i,j}^t$ is correct at the point x, else it is faulty. Thus $\sigma_{i,j}^t$ maps $x \in \mathcal{E}_i$ onto the subset $p_{i,j}^t$ of $\mathcal{X} \times \mathcal{P}$ given by $p_{i,j}^t(x) = \{(x, z) | z$ is in the range of values given out by $\sigma_{i,j}^t$ at $x\}$. We assume $p_{i,j}^t(x)$ is connected for each $x \in \mathcal{E}_i$. We designate $p_{i,j}^t = \cup_{x \in \mathcal{E}_i} p_{i,j}^t(x)$. If the parameter manifests itself only at finitely many discrete points $x_1, x_2, \cdots, x_k$ in $\mathcal{E}_i$ then $p_{i,j}^t = p_{i,j}^t(x_1) \cup p_{i,j}^t(x_2) \cup, \cdots, \cup p_{i,j}^t(x_k)$. If the parameter manifests itself on finitely many connected subsets of $\mathcal{E}_i$, then $p_{i,j}^t$ itself is a union of finitely many connected subsets of $\mathcal{X} \times \mathcal{P}$, or more specifically of $\mathcal{E}_i \times \mathcal{P}$. Thus in any case, we see that $p_{i,j}^t$ is a set of finitely many disjoint connected subsets of $\mathcal{X} \times \mathcal{P}$. The dimension of the sets $p_{i,j}^t(x)$ is in general the same as the dimension of the parameter measured by the sensor $\sigma_{i,j}^t$ (Fig. 3).

Schematic anatomy of the set $p_{i,j}^t$: The set $p_{i,j}^t$ represents readings of the sensor j in the cluster i. Here, it is assumed that it reads the parameter values at all points of the 1-dimensional interval set $\mathcal{E}_i$. $\sigma_{i,j}^t$ returns an interval of values $(p_{i,j}^t(x))$ as the abstract sensor reading at the point x. Here, the parameter is assumed to be 1-dimensional. Each sensor $\sigma_{i,j}^t$ in the cluster $\mathcal{S}_i$ collects data from the same region in the environment space $\mathcal{X}$, namely $\mathcal{E}_i$. Under ideal fault-free conditions, all the $\sigma_{i,j}^t$ in $\mathcal{S}_i$ have identical outputs.

B. Summary of Notations

We can say that the $p_{i,j}^t(x)$ corresponding to the abstract sensor $\sigma_{i,j}^t$'s readings at $x \in \mathcal{E}_i$ be connected, the subset $p_{i,j}^t$ may itself be either totally connected or be a disjoint union of connected subsets. It is also clear that the abstract sensors modeling sensors measuring multi-dimensional parameters defined in [6] are included in our definition since the $p_{i,j}^t$(x) could very well be d-dimensional rectangles or d-dimensional circles representing abstract sensor readings of a d-dimensional parameter measured at the point x. Depending upon the specific problem, we could model the abstract sensors to yield connected convex sets for the $p_{i,j}^t$(x), and in particular, d-dimensional rectangles or circles.

The sensors tend to be inaccurate or faulty. The technique of integration of the sensors employed by the processor should be able to tolerate faults to a reasonable extent and provide reliable estimates of the parameters value. A sensor $\sigma_{i,j}^t$ could be inaccurate at a point $x \in \mathcal{E}_i$ by having a $p_{i,j}^t(x)$ of a 'very large' size. More precisely , we may assume that the geometric sets $p_{i,j}^t$ are measurable with respect to some measure μ defined on $\mathcal{X} \in \mathcal{P}$ (say, the Lebesgue measure on n-dimensional Euclidean space). We say that $\sigma_{i,j}^t$ is inaccurate at $x \in \mathcal{E}_i$ if $\mu(p_{i,j}^t(x))$ is greater than a preassigned number M which is decided depending upon the specifications and performance requirements of the specific Distributed Sensor Network. A sensor may be faulty in that its output may not contain the value of the parameter measured by it. A technique of integration that yields in most cases an output which contains the value of the parameter and is fairly accurate, is required at each processor for the Distributed Sensor Network to function reliably. Since it is impossible in general to know which sensor are faulty at any given instant of time, one has to derive a method of integration that indirectly isolates a neighborhood of the correct value of the parameter from among the various sensor outputs.

C. Limitation of Marzullo's Technique

Marzullo's technique of integration gives an effective way of computing a fairly accurate output when the number of faulty sensors are bounded above by a function of n ($f \leq \lfloor n/2 \rfloor$ for 1-interval estimates and $f \leq \lfloor n/2d \rfloor$ for d-dimensional rectangular estimates). However, when one cannot ensure this boundedness of the faults (failure is never often controllable). One may have to reject an entire cluster's data when $f > \lfloor n/2d \rfloor$. It is however possible that in most situations, a cluster of n sensors is still useful even when the number of faults is not less than $\lfloor n/2d \rfloor$. In our earlier work [5], we have provided a polling technique which does not assume any strong bounds on the number of faulty sensor, but which produces an output that is reliable and has measure smaller than that of the output of Marzullo's method in general. We shall show now that this technique carries over easily to the sensor outputs of the abstract Distributed Sensor Network defined above.

D. Fault-tolerant Integration of Abstract Sensor Outputs

We now describe a computational characterization of sensor integration for general distributed sensor networks. We define correctness and faultiness in abstract sensors as follows:

An abstract sensor $\sigma_{i,j}^t \in \mathcal{S}_i$ is correct at $x \in \mathcal{E}_i$ at time t if $(x, a) \in p_{i,j}^t(x)$ where a is the actual value of the parameter being measured at x at time t ($a \in \mathcal{P}$), else it is faulty.

The method of integration of our earlier paper [5] can be used here to obtain a fault-tolerant estimate of the actual physical value being measured in this general case also as shown blow:

Consider the characteristic function of the set $p_{i,j}^t(x)$:

$$\mathcal{X}_{x,t}^{i,j}(y) = \begin{cases} 1 & \text{if } y \in p_{i,j}^t(x) \\ 0 & \text{otherwise} \end{cases}$$

Note that the characteristic function of the set $p_{i,j}^t = \sum_{x \in \mathcal{E}_i} p_{i,j}^t(x)$ is given by

$$\mathcal{X}_t^{i,j} = \sum_{x \in \mathcal{E}_i} \mathcal{X}_{x,t}^{i,j}(x), \qquad (x \neq y \Rightarrow p_{i,j}^t(x) \cap p_{i,j}^t(x) = \phi)$$

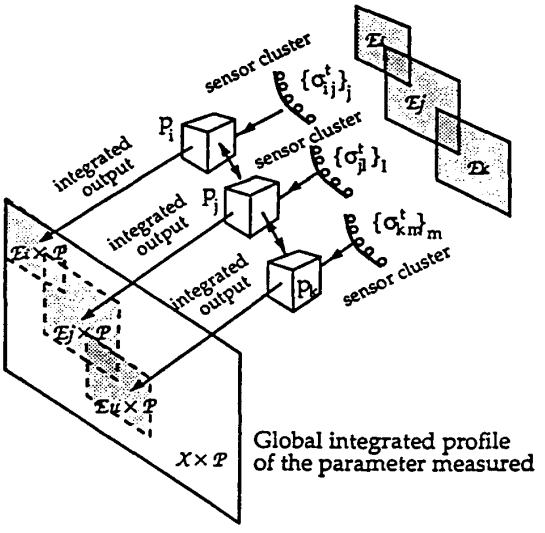


Fig. 4. Schematic representation of information patching to obtain a global picture.

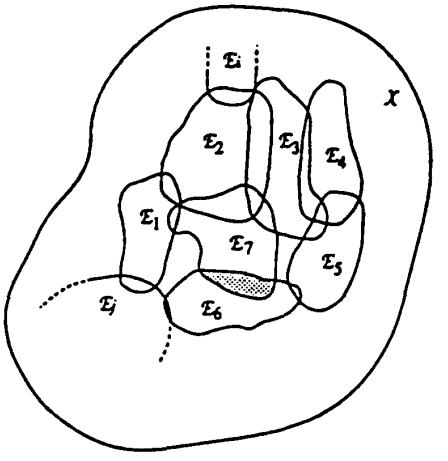


Fig. 5. A general coordinate chart $\{\mathcal{E}_i\}_{i=1}^n$ on an environment space $\mathcal{X}$.

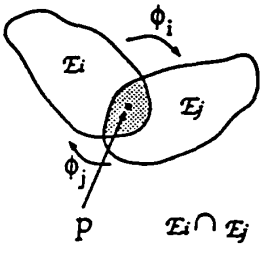


Fig. 6 The transition function o_i and o_j defined on the intersection $\mathcal{E}_i \cap \mathcal{E}_j$.

If atmost f_x of the n_i sensors in the cluster $\mathcal{S}_i$ are known to be faulty at x, then following the method of integration employed in our earlier paper [5], the correct value of the parameter being measured by the sensor cluster s_i at $x \in \mathcal{E}_i$ and at time t is contained in the subset corresponding to the characteristic function $\mathcal{X}^i_{x,t}(y)$ given by

$$\mathcal{X}^i_{x,t}(y) = \mathcal{X}_{[n_i - f_x, \infty)}(O^i_{x,t}(y))$$

where

$$\mathcal{X}^{[a,\infty)}(x) = \begin{cases} 1 & \text{if } x \in [a, \infty) \\ 0 & \text{otherwise} \end{cases}$$

and $O^i_{x,t}(y) = \Sigma_{j=1}^{n_i} \mathcal{X}^{i,j}_{x,t}(y)$ is the overlap function for the cluster $\mathcal{S}_i$ at the point $x \in \mathcal{E}_i$ and at time t i.e. $O^i_{x,t}(y)$ gives the number of sensors $\sigma^t_{i,j}$ whose estimates $p^t_{i,j}$(x) at the point $x \in \mathcal{E}_i$ and at the time t contain the point $(x, y) \in \{x\} \times \mathcal{P}, y \in \mathcal{P}$.

Consequently, $\mathcal{X}^i_{x,t}(y)$ is the characteristic function of the set of all those points of $\{x\} \times \mathcal{P}$ which lie in the intersection of at least $n - f_x$ intersections of the sets $p^t_{i,j}(x)$.

The correct value of the parameter being measured at $x \in \mathcal{E}_i$ at time t must lie in the subset $\Sigma^i(x,t) = \{y \in \mathcal{P} | \mathcal{X}^i_{x,t}(y) = 1\}$, since the assumption that atmost f_x sensors are faulty at x implies at least $n_i - f_x$ sensors are correct and hence must overlap since their estimates $p^t_{i,j}(x)$ must all contain the point (x, a) where a is the actual value of the parameter being measured.

As there is no reason to assume that f_x is fixed or bounded quantity, we may as well treat it as a parameter which we may change at will to obtain larger or smaller intersection as desired. For instance, we may look upon f_x as

$$\min\{f | \mathcal{X}^i_{x,t}(y) = \mathcal{X}_{[n_i - f, \infty)}(O^i_{x,t}(y)) \neq 0\},$$

and thus obtain those intersections which are a result of the largest number of sets intersecting to give a nonempty set.

Faulty sensors could have random or "wild" faults, in which event there is little correlation between the sensor's estimate and the correct physical value of the parameter being measured. On the other hand, the fault may be "tame," in which case the faulty sensor's estimate although does not contain the actual physical value of the parameter being measured, lies sufficiently close to it.

In order to obtain a reliable estimate of any of the above mentioned parameters, it is important to have redundancy in reading the parameter value. The parameter value is sampled in a local neighborhood of the point in question, the value of the parameter being assumed to be stable in a small neighborhood . The values these obtained are compared with each other to yield a consensus value, this is achieved by a process of sensor integration.

Thus sensor failure can result in the failure of the electronic component or the physical interface or due to local variations in the environment.

Electronic failure, normally spells permanent failure. The nature of electronic failure is uncorrelated and hence is "wild." The failure resulting from a defective physical interface or a locally fluctuating environment (e.g. turbulence in fluids, statistical variations in the sample space observed etc.) results in values which are close to correct values and hence are "tame."

It is reasonable to assume that sensors more often fault tamely due to physical perturbation than wildly due to random electronic failures, In which case faulty sensors tend to "cluster" in the neighborhood of the correct value of the parameter being measured.

We may make use of this clustering to predict with reliability the subset with highest chance of containing the actual value of the parameter among those subsets which belong to $\Sigma^i(x,t)$. In order to do this, we need to make the notion of a tame fault more rigorous. The following definition is derived from above characteristics of tame faults.

Definition 2: A faulty sensor is *tamely* faulty at x at time t if it intersects with a sensor that is correct at x at time t.

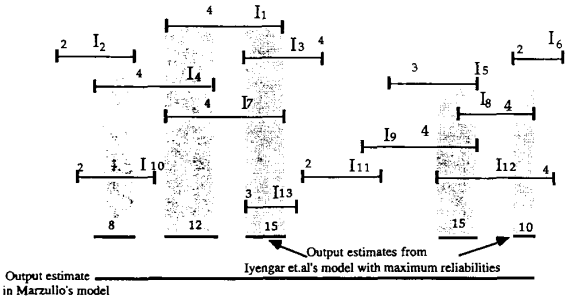


Fig. 7. A comparison of performance.

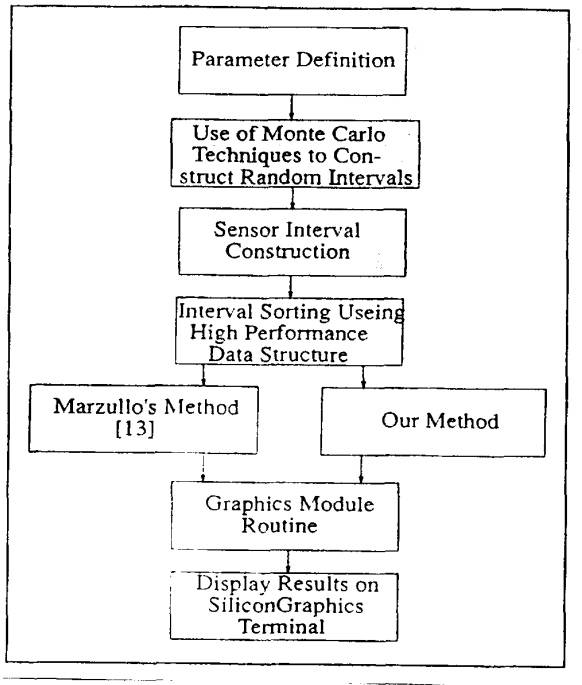
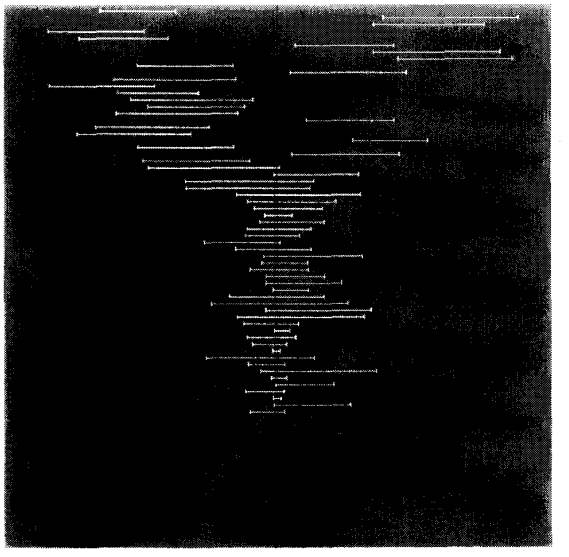
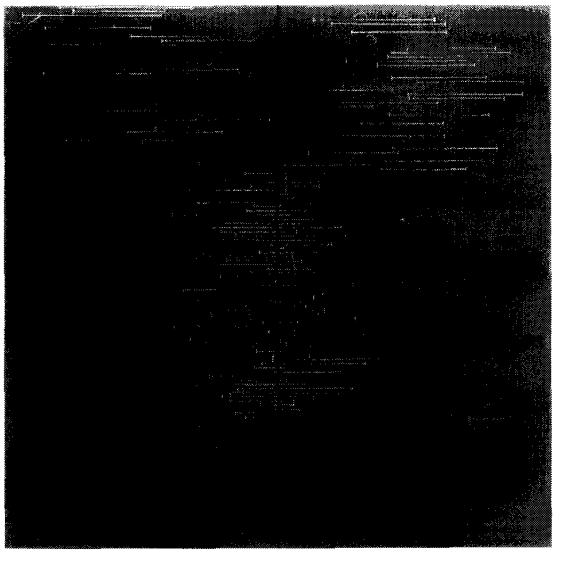


Fig. 8. Block diagram of simulation.



(a)



(b)

Fig. 9(a). Sensors: 60, tame faults: 23. (b) Sensors: 100, tame faults: 40.

E. Reduction of the Output Measure When Most Faults are Tame

It is unlikely that sensors with wild or random faults cluster since by their very nature, they are uncorrelated and hence distributed more or less evenly. In the case when most sensor faults are tame, we may resort to the polling technique introduced in an earlier paper [5] to reduce the measure of the subset containing the correct physical value with high reliability:

Let $L_1^i(x,t),\cdots,L_{k_i}^i(x,t)$ be the disjoint maximal connected subsets of $\mathcal{P}$ whose union is the subset $\Sigma^i(x,t)$ containing the correct physical value of the parameter measured at x at time $t\,(x \in \mathcal{E}_i)$.

Define the popularity of the output of the kth sensor $(1 \le k \le n_i)$ in the cluster $\mathcal{S}_i$ to be the nonnegative integer $\pi^{ik}(x,t)$ given by:

$$\pi^{ik}(x,t) = \left(\sum_{j=1}^{n_i} \|\mathcal{X}_{x,t}^{ij}, \mathcal{X}_{x,t}^{ik}\|\right) - 1$$

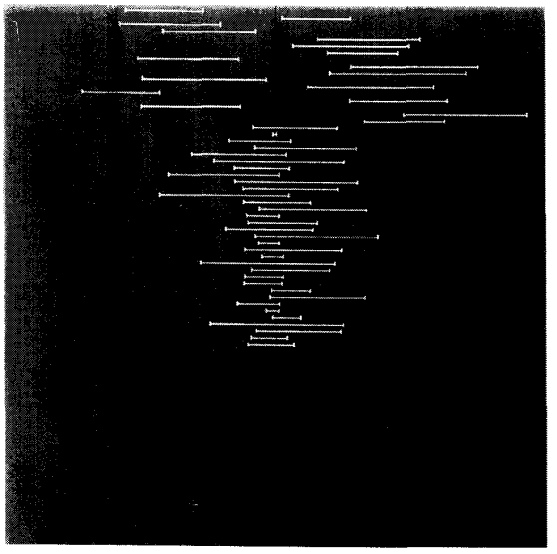
where $\|f\|$ is the maximum value of the function f. $\pi^{ik}(x,t)$ gives the number of sensor outputs having nonempty intersection with the output of the sensor $\sigma_{i,k}^t$ at x and time t.

If $\Lambda_{x,t}^{ij}$ is the characteristic function of the set $L_j^i(x,t).(1 \le j \le k_i)$ then define the *reliability* $R^{ik}(x,t)$ of the set $L_j^i(x,t)$ to be the nonnegative integer given by:

$$R^{ik}(x,t) = \sum_{k=1}^{n_i} \|\Lambda_{x,t}^{ij}, \mathcal{V}_{x,t}^{ik}\| \pi^{ik}(x,t)$$

$R^{ik}(x,t)$ is the sum of the popularities of all those sensors' outputs which contain the set $L_j^i(x,t)$.

It is clear that the larger the reliability of the connected set $L_j^i(x,t)$. the greater the clustering of sensors about $L_j^i(x,t)$. and hence the greater the likelihood of $L_j^i(x,t)$. containing the correct value of the parameter. Thus $R^{ij}(x,t)$ is a good measure of the connected subsets $L_j^i(x,t)$ of $\Sigma^i(x,t)$ containing the correct value of the parameter.



(c)

(d)

Fig. 9(c). Sensors: 50, tame faults: 17. (d) Sensors: 60, tame faults: 23.

Thus, it is seen that the techniques employed by us in the paper [5] though used on 1-dimensional interval estimates, hold for a much more general class of sensor outputs.

IV. A Scheme for Patching Together Local Processor Information to Obtain a Global Picture of the Parameter Measured

We now look at the problem of addressing the global behavior of a parameter over the space $\mathcal{X}$. This involves the interaction of the processors allocated to the subsets of the chart and comparison of common information for smooth patching of local information to obtain a global picture of the phenomenon being observed over $\mathcal{X}$(Fig. 4).

Each of the subsets $\mathcal{E}_i$ $(1 \leq i \leq m. \cup_{i=1}^{m} \mathcal{E}_i = \mathcal{X})$ covers a region of the environment space $\mathcal{X}$. and is equipped with a sensor-cluster $\mathcal{S}_i = \{\sigma_{i,j}^{i}\}_{j=1}^{n_i}$. where each sensor $\sigma_{i,j}^{i}$ monitors all of $\mathcal{E}_i$ and sends data to a common processor allocated to $\mathcal{E}_i$. Each $\mathcal{E}_i$ is equipped with a coordinate system of its own, and all the sensors in the cluster $\mathcal{S}_i$ measure with reference to this coordinate system. Thus all points x in $\mathcal{E}_i$ have local coordinates and if two of the regions $\mathcal{E}_i$ overlap then the coordinates of a point in the intersection will be different in the different regions containing it. The relative arrangement of the subsets $\mathcal{E}_i$ of the chart on X depends upon the specific needs of a distributed sensor network. However, it is desirable to have them overlapping since this helps in patching up the local scenes to form a global picture as well as in increasing fault-tolerance and aiding fault-detection in sensors by comparison of data at common points.

Thus the union of the outputs of the processors over the sets $\mathcal{E}_i$ gives the global profile of the parameter over $\mathcal{X}$. The sets $\mathcal{E}_i$ may be overlapped to corroborate the relative locations of these sets and enhance fault-tolerance and fault-detection by pitting one processor's output against the other, and crosschecking to find out if any cluster has failed (massive failure). One may also conceive of other specification and need based variations of the relative arrangements of the $\mathcal{E}_i$. for instance, when these sets are themselves in motion with relation to one another. A general coordinate chart $\{\mathcal{E}_i\}_{j=1}^{n}$ on an environment space $\mathcal{X}$ is given in Fig. 5.

A. A General Characterization

The following two figures are a schematic representation of overlapping chart sets of an m-dimensional environment space and the transition functions associated with the chart.

The general transformation scheme is represented in Fig. 6 between two sets $\mathcal{E}_i$ and $\mathcal{E}_j$. Let p belongs to $\mathcal{E}_i \cap \mathcal{E}_j$. the

$$p = (x_1^i, \cdots, x_m^i) \text{ in } \mathcal{E}_i$$
$$p = (x_1^j, \cdots, x_m^j) \text{ in } \mathcal{E}_j$$

o_i: the transition function from $\mathcal{E}_i$ to $\mathcal{E}_j$ defined on $\mathcal{E}_i \cap \mathcal{E}_j$.
o_j: the transition function from $\mathcal{E}_j$ to $\mathcal{E}_i$ defined on $\mathcal{E}_i \cap \mathcal{E}_j$.
$o_i \cdot o_j$ = identity on $\mathcal{E}_i \cap \mathcal{E}_j$ w.r.t $\mathcal{E}_j$
$o_j \cdot o_i$ = identity on $\mathcal{E}_i \cap \mathcal{E}_j$ w.r.t $\mathcal{E}_i$.

The coordinated of p in $\mathcal{E}_j$ are obtained from the coordinates of p in $\mathcal{E}_i$ by letting o_i transform p's coordinated from $\mathcal{E}_i$ to $\mathcal{E}_j$:

$$o_i(x_1^i, \cdots, x_m^i) = (x_1^j, \cdots, x_m^j)$$

Similarly

$$o_j(x_1^j, \cdots, x_m^j) = (x_1^i, \cdots, x_m^i)$$

Further analysis in this direction can be done in specific distributed sensor network situations, but the general interprocessor interaction and communication must involve the use of transition function for meaningful comparison of data. The coordinated changes are effected by local processors and only those processors that are connected have transition functions defined. In the case of moving chart sets it is possible that two sets whose processors are not connected directly may overlap for some time. It is possible to perform local coordinate changes by routing data through other intermediate processors since the transition functions are transitive and hence may be composed to obtain transformations between unconnected processors.

V. SIMDSN: Simulation of Fault Tolerant Integration in Distributed Sensor Networks

To test our theoretical analysis of the general computational framework proposed earlier in our paper, we have developed a

modular parameter-driven simulator for the fault tolerant integration of abstract sensor interval estimates. We have extensively tested our framework on test data and compared our approach with that of Marzullo [2]. The simulator uses the well-known Monte Carlo technique to construct random intervals for integration using our algorithm and Marzullo's technique which are compared subsequently.

A block diagram of the simulator is shown in Fig. 8. Broadly speaking, the simulation consists of the following stages: (1) Defining the parameters that drive the simulation. (2) Construction of both correct and faulty intervals. (3) Sorting the intervals on both start and end points. (4) Implementation of our general integration algorithm. (5) Display of the simulation results using a graphics interface module.

A. Simulation Parameters

The simulation is driven by the following parameters. (i) The total number of sensors (n). (ii) The maximum allowable number of faulty sensors. (iii) The maximum width of an interval. (iv) The minimum width of an interval. (v) The correct physical value that the sensors attempt to measure.

B. Interval Construction and Sorting

The simulation starts (stage 1) by constructing intervals corresponding to both faulty and correct sensors. The correct sensor intervals are constructed so as to include the correct physical value somewhere in the interval. The faulty sensor intervals are constructed so that their near ends are located within a distance of at most L. The faulty sensor intervals are randomly scattered on either side of the correct physical value. If the width w of each interval is bounded so that $l \leq w \leq L$, the following equations are used to construct the tamely faulty sensor intervals. $R\#$ represents a random number generated by the simulator. Of course, each instance of $R\#$ corresponds to a different random number.

$$w = l + R\# * (L - l) \tag{1}$$

$$a_i = C + R\# * L; b_i = a_i + w \tag{2a}$$

$$b_i = C - R\# * L; a_i = b_i - w \tag{2b}$$

Eqs. 2(a) and 2(b) locate the faulty sensor interval on different sides of the correct value and during simulation, one of the equations is picked randomly to locate approximately equal numbers of intervals on both sides. The correct intervals are constructed according to the following equations.

$$w = R\# * (L - l) \tag{3}$$

$$a_i = C - R\# * w; b_i = a_i + w \tag{4}$$

This stage is followed by *sorting the randomly generated intervals in ascending order of location of interval start points and interval end points.* Further, the worst case complexity of this sort is limited to $O(n \log n)$ which is the best worst-case complexity for a sorting algorithm on a sequential machine.

C. Algorithm for Integration for the simulator

The algorithm we have used for implementing our scheme for the integration of overlapping intervals is as follows.

Algorithm:

Input: Intervals $I_1, I_2, \cdots, I_n, f$ (parameter denoting maximum allowable number of faulty sensors)
Output: Integrated output estimate
begin

1. **Determine** all $(n - f)$-intersections of the intervals to yield intervals $L_1, \cdots, L_k$, each of which is an $(n - f)$ intersection: $\{L_j = [a_j, b_j]\}$
2. **For** each $i\ (1 \leq i \leq k)$
   a. compute the number of intervals intersecting each of the intervals $I_j\ (1 \leq j \leq n)$ having a nonempty intersection with L_i
   b. Add these numbers to obtain a number r_i (the number of intervals intersecting with intervals involved in the formation of L_i—a measure of the reliability of L_i
3. **Choose** the maximum of the $r_i (1 \leq i \leq k)$ and denote it r
4. **Let** $m = \min\{i | r_i = r\}$ and $M = \max\{i | r_i = r\}$
5. Assign $I_p^* = [a_m, b_M]$ to be the integrated output estimate

end

D. Graphics Module and Display details: Language for Implementation

The simulator is developed entirely in the C programming language and uses the powerful *Graphics Library* of the Silicon Graphics 4D/310 GTX Superworkstation to compute and display the results of the simulation. The simulator we have developed is network transparent and allows simulations to be carried out at any compatible host accessible in the Internet domain. Our algorithm has been tested with several sets of random data generated dynamically and figure below shows one set of test data and the result of our method along with the reliability of each disjoint interval. For comparison with our method, the figure also shows, for the same test data, the final output interval determined by Marzullo's approach.

E. Simulation Results

We present here the details of four complete runs of the simulator involving various numbers of correct and faulty sensors. For convenience of graphical visualization of the integration algorithms, only *tame* faults have been considered. A minimal amount of change in the parameters of simulation enables particular runs of the simulation to contain wild sensor faults. Figs. 9(a) through 9(d) show the combined results of the two interval integration techniques compared. The green line down the middle of the display screen indicates the correct physical value being measured by the sensors. Notice that the faulty sensor intervals are located away from this line while the correct intervals include the correct value. The output of Marzullo's technique is colored red and appears at the bottom of the screen. The interval with the highest reliability in our method is shown just above the result of Marzullo's approach and is colored blue. The integer shown above our output is the reliability measure corresponding to the result of our technique. This reliability measure is calculated by summing the popularities of the sensor intervals that overlap to produce the final output. For the first simulation run, involving 60 sensors of which 23 are faulty, Table I summarizes the popularities, the number of overlapping intervals, for each interval.

TABLE I
INTERVALS AND POPULARITIES FOR A PARTICULAR SAMPLE RUN

Interval	Popularity	Interval	Popularity	Interval	Popularity	Interval	Popularity
1	19	16	15	31	40	46	36
2	13	17	15	32	37	47	45
3	38	18	19	33	41	48	45
4	24	19	13	34	37	49	41
5	13	20	24	35	46	50	46
6	21	21	24	36	37	51	37
7	24	22	19	37	37	52	40
8	23	23	6	38	40	53	42
9	9	24	40	39	41	54	43
10	13	25	37	40	38	55	40
11	7	26	37	41	37	56	40
12	9	27	40	42	44	57	50
13	20	28	46	43	42	58	36
14	6	29	46	44	41	59	37
15	21	30	42	45	38	60	43

We conclude this section on the simulation by noting that the union of the disjoint intervals (if any are disjoint) determined by our algorithm and the gaps between these intervals together can never exceed the width of the final output of Marzullo's method. Thus, while Marzullo's method specifies a rather large interval as the final output of the integration process, our method subdivides this interval into (possibly) disjoint intervals each with an associated *reliability measure.* In a real-world situation, a suitable rule may be chosen judiciously depending upon our faith in the tameness of abstract sensor faults. It is clear that the worst-case width for the output interval as determined by our algorithm can be at most as large as that indicated by Marzullo's approach.

VI. CONCLUDING REMARKS

We have proposed an abstract framework to address the problem of fault-tolerant integration of sensor information in a general distributed sensor network. We have also described a general computational scheme for obtaining a reliable integrated output from the inputs of several sensors in a cluster, at its processor. Our method involved a generalization of the polling technique for tamely faulty sensors presented in our earlier work [5]. A comparison of performance is given in Fig. 7.

The time complexity of finding the $(n - f)$—intersection of n sensors (in the case of intervals and higher dimensional rectangles) is $O(n \log n)$ using Bentley's segment-tree. For an implementation of the algorithm see [7]. Also see [6].

Based on our simulation results, we note that the union of the disjoint intervals (if any are disjoint) determined by our algorithm and the gaps between these intervals together can never exceed the width of the final output of Marzullo's method. Thus, while Marzullo's method specifies a rather large interval as the final output of the integration process, our method subdivides this interval into (possibly) disjoint intervals each with an associated *reliability measure.*

ACKNOWLEDGMENT

The authors would like to thank all the referees, especially to Referee A, for their comments, and Professor Marzullo, Cornell University and Professor Brady, Oxford University for their advices.

REFERENCES

[1] S. S. Blackman and T. J. Broida, "Multiple sensor data association and fusion in aerospace applications," *J. Robot. Syst.*, vol. 7, no. 3, pp. 445–485, 1990.

[2] H. F. Durrant-Whyte, "Sensor models and multisensor integration," *Int. J. Robot. Res.*, vol. 7, no. 6, pp. 97–113, 1988.

[3] S. S. Iyengar, "Distributed sensor networks: A computational perspective," *J. Comp. Sci. Informat.*, vol. 21, no. 1, 1991.

[4] S. S. Iyengar, M. B. Mohan, and R. L. Kashyap, "Information routing and reliability issues in distributed sensor networks," to appear in *IEEE Trans. Acoust., Speech, Signal Processing* vol. 40, no. 12, pp. 3012–3021, Dec. 1992.

[5] Lakshman Prasad, S. S. Iyengar, R. L. Kashyap, and R. N. Madan, "Functional characterization of sensor integration in distributed sensor networks." *IEEE Trans. Syst. Man Cyber.*, vol. 22, pp. 1082–1087, Sept./Oct. 1991.

[6] K. Marzullo, "Tolerating failures of continuous-valued sensors," *ACM Trans. Comput. Syst.*, vol. 8, no. 4, pp. 284–304, Nov. 1990.

[7] K. Marzullo and S. Owicki, "Maintaining the time in a distributed system," *Proc. of 3rd Symp. on Principles of Distributed Computing*, pp. 295–305. ACM SIGPLAN/SIGOPS, 1983.