Real-time distributed sensor fusion for time-critical sensor readings

Richard R. Brooks
California State University Monterey Bay
Institute of Communications Science
and Technology
100 Campus Center, Building 18
Seaside, California 93955-8001
E-mail: richard_brooks@otter.monterey.edu

S. Sitharamar Iyengar
Louisiana State University
Department of Computer Science
Baton Rouge, Louisiana 70803-4020

Abstract. The problem of dynamic distributed sensor fusion, where time critical sensor readings are fused concurrently by physically isolated modules, is considered. A large number of critical applications including target tracking, remote sensing, and autonomous vehicle navigation can depend on interpreting sensor readings in real time. Independent sensor modules receive partially contradictory data, but their answers must agree within computed accuracy bounds. Real-time constraints cause additional uncertainty factors. The dynamic nature of this problem requires that the problem be viewed in four dimensions and processed efficiently. We present a new method that robustly tolerates a limited number of component failures. This method has complexity $O(f^4 N \log N)$, where N is the total number of modules and f is the number of faulty modules tolerated. Experimental results are given showing the merits of the proposed method.

Subject terms: sensor fusion; fault masking; real time; distributed agreement.

Paper SFN-14 received June 5, 1996; revised manuscript received Sep. 13, 1996; accepted for publication Oct. 15, 1996.

1 Introduction

Sensor integration is concerned with the synergistic use of multiple sources of information.[1] A major component of sensor integration is sensor fusion, merging multiple inputs into a common representation.[2] Sensor fusion can be divided into the following classes[3]:

1. Complementary sensors do not depend on each other directly but can be merged to form a more complete picture of the environment, for example, a set of radar stations covering nonoverlapping geographic regions. Complementary fusion is easily implemented since no conflicting information is present.
2. Competitive sensors each provide equivalent information about the environment. A typical competitive sensing configuration is a form of N-modular redundancy. For example, a configuration with three identical radar units can tolerate the failure of one unit. This is a general problem that is challenging since it involves interpreting conflicting readings.
3. Cooperative sensors work together to derive information that neither sensor alone could provide. An example of cooperative sensing would be using two video cameras in stereo for 3-D vision. This type of fusion is dependent on details of the physical devices involved and can not be approached as a general problem.

The configurations discussed in this paper are competitive fusion problems. This is a large class of challenging problems.

This paper considers distributed multisensor systems which are dynamic in that they are used for competitive sensor fusion in real time. They are distributed in that each sensor module moves independently and has local intelligence. To work in this environment, a dynamic distributed fusion algorithm must tolerate both sensor noise and temporal uncertainties. The sensors are independent processing elements (PEs) that must coordinate their activities in the presence of partially contradictory data. We propose a methodology, consisting of two real-time processing steps, that guarantees robust agreement among the PEs.

Our method accepts readings from potentially heterogeneous modules and is tolerant of a large number of error sources. These sources of error include clock skewing, transmission delays, and arbitrary failure of a limited number of modules. This method is independent of the hardware and methodology used for sensing and data association. Each module should use the methods most appropriate for its components. The overall system is then able to compensate for errors made by any subset of modules, as long as the cardinality of the subset is within well-defined bounds. This may in fact be the best way of handling a number of difficult problems such as data association, where no single method has been found to handle all the sources of error treated by our method.

The main assumption made regarding the overall system dynamics is similar to the assumption made for application of the extended Kalman filter. A linear equation must present an adequate estimate of the dynamic system. Our interest is in finding a computationally efficient fault-tolerant method of fusing noise corrupted sensor data. In contrast to the extended Kalman filter, it is not assumed that the noise is Gaussian. The only assumption made is that the noise is within predefined bounds for most of the sensors, a given number of sensors may return readings

with an arbitrarily large amount of noise without adversely affecting the results of the method.

The paper is organized as follows. Section 2 formally describes the problem under consideration. Section 3 details the approach we propose for distributed dynamic sensor fusion. Section 4 demonstrates an application of this method. The paper concludes with a brief discussion in Sec. 5 that contrasts this approach with established methods.

2 Problem Description

We consider a broad class of sensor fusion problems and classify them as distributed dynamic sensor fusion. These problems are characterized by the use of multiple redundant sensors that collect time-sensitive data. The problems inherent in distributed data fusion are discussed in Ref. 4, where an algorithm is given for robust agreement. This paper extends that approach to real-time problems in three dimensions by combining it with the multidimensional sensor fusion method described in Ref. 5. This paper differs from the previous papers in that multidimensional data are fused, and that an explicit framework is derived for dealing with data that varies over time. The time-critical nature of dynamic distributed fusion requires the use of computationally efficient algorithms, and introduces additional uncertainties.

The problem to be solved can be phrased as follows. An object O is observed by N independent modules $m_1, m_2, \ldots, m_n$. We use the term module instead of sensor to signify that each module has local intelligence, memory, and sensors. The modules may be heterogeneous or homogeneous. Every module communicates with all other modules. We do not consider network topology directly, since network faults are included in the model as failures of communications components of individual modules. Similarly, data association errors are included in the model as sensor component failures.

Each m_i determines the instantaneous location and velocity of O in three dimensions, and broadcasts this data d_i to all other modules. Assuming every sensor has limited accuracy and a limited number of readings may be arbitrarily faulty, each m_i uses identical logic to deduce the position of O. Figure 1 illustrates this problem. Note that, in spite of the problem being presented as an object-tracking problem, any problem involving changing values in multiple dimensions can be put in this framework. A number of remote sensing applications such as charting sea currents and weather systems can easily be modeled in this manner. We present this approach phrased as a target-tracking problem since it is the most intuitive abstraction.

This method is situated at a medium level of sensor fusion, the feature level as defined by Luo and Kay.[2] It fuses information regarding image features that have already been extracted from the raw sensor data. Broida[6] refers to this type of approach as "track-level" fusion, he notes that this type of approach has advantages in fault-tolerance and bandwidth requirements over more centralized approaches. In this approach, a number of tasks, such as data association, can be performed as a part of the sensing task by the local sensor modules. This level of fusion is appropriate for the problem since it does not require the bandwidth that would be needed by a lower level approach

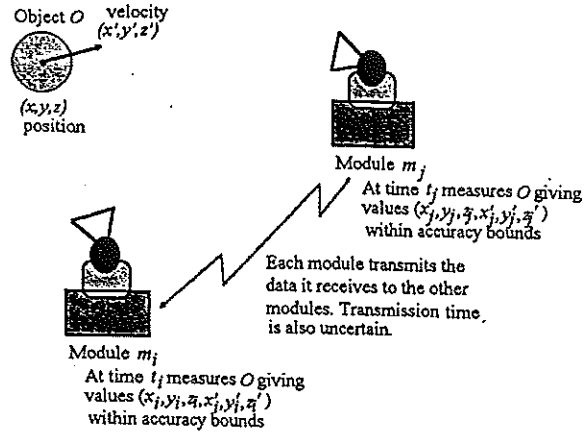

Fig. 1 Modules m_i and m_j measure the position and velocity of object O.

fusing raw data, but still provides information about the environment at a level below decision fusion. This approach enables a fault tolerant system to infer information about its changing environment by using a number of heterogeneous methods, such as extended Kalman filters, at lower levels of abstraction.

Each module measures object O relative to its own position and velocity. For each element of the sextuplet $d_i = (x_i, y_i, z_i, x_i', y_i', z_i')$ measured by a correctly functioning module m_i there are two sources of uncertainty: measurement inaccuracy (measurement noise) and uncertainty in the frame of reference of the measurement (position noise). Similar uncertainties exist for the data module m_i receives from all other modules.

Our discussion uses Cartesian coordinates. The method is equally valid for a problem phrased in spherical coordinates. A system relying on range and angle information can be implemented with only trivial modifications. Note that to adequately handle a value that changes over time, some measure of the rate of change of the value must be available.

Each module has a local clock and takes readings at known intervals. The time needed to transmit readings between modules varies slightly. This causes a small variation between modules regarding the exact time the measurement was made (temporal noise). This temporal noise Δt is due to both the skewing of local clocks and variations in transmission times. Figure 2 shows these uncertainties in relation to module m_i.

The final source of error consists of arbitrary errors caused by equipment malfunctions (failure). A discussion of methods used for tolerating arbitrary failures in systems can be found in Barborak[7] and its application to sensor fusion in Brooks and Iyengar.[4] We denote a method that tolerates all the sources of error listed in this section as being robust.

Certain assumptions must be true for this problem to be well posed. From basic principles it is clear that the trajectory of O in 3-D space can be described by a piece wise continuous function $f(t)$. The following theorem concerning the function $f(t)$ explains which dynamic distributed sensor fusion problems are computationally tractable.

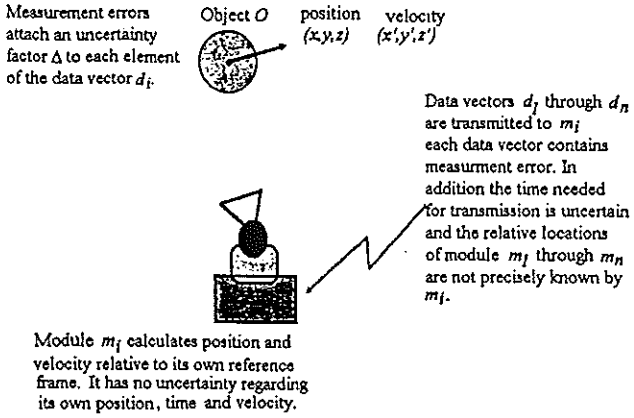

Fig. 2 Data uncertainties for module m_i.

Theorem 1. Module m_j receives data from module m_i, which is identified as having been taken at time t_i. The tracking problem is well posed if and only if the temporal noise Δt between any two modules i and j is small enough that the first two factors of a Taylor series expansion of $f(t_i)$ is an adequate approximation of $f(t)$ for $t_i-\Delta t \leq t \leq t_i+\Delta t$.

Proof. The Taylor expansion of $f(t_i)$ at time t is

$$f(t_i)+f'(t_i)(t-t_i)+f''(t_i)(t-t_i)^2+f'''(t_i)(t-t_i)^3\cdots +f^{(n)}(t-t_i)^n+\cdots .$$

If Δt is large the magnitude of Δt^n grows as $n\rightarrow\infty$ and the value of $f(t)$ will depend predominately on the higher order terms of the Taylor series expansion, when these terms are nonzero. The data sextuplets only contain information regarding the first two terms, so that the value of $f(t)$ would be impossible to infer from the data available if higher order terms are significant. This shows that Theorem 1 is necessary.

Conversely, as $\Delta t\rightarrow 0$ the values of Δt^n approach zero as $n\rightarrow\infty$. Therefore there exists a sufficiently small Δt so that for any given $f(t)$, only the first two factors of the Taylor expansion are non-negligible. This shows that Theorem 1 is sufficient.

From Theorem 1 we see that the location of O at time t is approximated by a linear function of the measured velocity and position of O at time t_i. Another assumption that must be made is that the number of failures (arbitrary errors) is limited. This is intuitively reasonable since the correct information must outweigh the incorrect information in some manner. The exact limit needed for this problem and a reference to the proof of this limit is given in Sec. 3.

Note that the restriction given by Theorem 1 is roughly equivalent to the restriction required for using the extended Kalman filter. The system must be approximated by a linear system. In addition, since we are considering physical systems, the instantaneous velocity is always well defined for objects of non-negligible mass where Newtonian mechanics adequately describe the system.

To solve the dynamic distributed sensor fusion problem an algorithm must compute both a fused data vector $\mathbf{d}f$ containing a good estimate of the position of object O in the neighborhood of time t, and a measure of the uncertainty of $\mathbf{d}f$ ($\Delta\mathbf{d}f$). The values of $\mathbf{d}f$ may not necessarily be identical at each m_j, but all $\mathbf{d}f_j$ must be contained within the uncertainty range $\Delta\mathbf{d}f_j$ for all m_j.

The modules described are also reasonable. A large number of inexpensive sensors are commercially available for measuring the location of an object in space and for measuring an object's velocity. Many of these velocity sensors use the Doppler effect in combination with some form of electromagnetic radiation: microwave, laser, radar, etc.

Theorem 2. Within the context given, it is impossible to derive bounds for fused values of the velocity of O.

Proof. The only assumption made regarding the second derivative of $f(t)$ is that it is negligible when multiplied by the temporal noise squared. Since the temporal noise may be extremely small, the velocity of O can vary greatly within this short period of time. In fact, as $\Delta t\rightarrow 0$ the second derivative of $f(t_j)$ may be arbitrarily large. The information available, therefore, gives no criteria for judging the correctness of values of $f'(t_i)$, and no criteria for bounding the possible values of fused velocity.

Theorem 2 shows that to reliably fuse velocity data, more strict assumptions are needed. These assumptions may be justified in many practical instances, to treat the largest possible set of problems we do not make any assumptions regarding the second (or higher) order derivative(s) of $f(t)$. For this reason, we restrict ourselves to fusing the position data and making an estimate of the velocity data that has no accuracy bounds.

3 New Paradigm for Distributed Dynamic Sensor Fusion

The distributed real-time sensor fusion methodology presented in this section assumes known limits to position, measurement, and temporal noise. These accuracy limitations can either be known *a priori* or calculated using sample variance and an appropriate probability distribution for each sensor module.

Our method tolerates failure of up to f out of N modules by using the readings from the correctly functioning modules to mask out the readings of the faulty modules. A module functions correctly as long as none of the noise factors exceed the accuracy limits for the module. If N is the total number of modules in the system, f must be less than $N/2D$, where D is the total number of dimensions in the problem. For the distributed dynamic data fusion problem, D is equal to 4, three spatial and one temporal dimensions, so $f<N/8$. This limitation has been derived by considering intersections of readings from continuous valued sensors, the derivation of this limit can be found in Chew and Marzullo.[8]

The method we use is known as fault masking,[9] which is closely related to N-modular redundancy[10] and the Byzantine generals problem.[7] From the problem definition, the correct answer must be contained in a range defined by the intersection of $N-f$ or more sensor readings. By comparing all values returned by the modules we find the region containing all ranges where groups of $N-f$ or more readings concur. The limits of this region form a new virtual

sensor reading.[8] The number of possible intersections is limited by f (Ref. 11) and their distribution is limited by the accuracy of the individual modules. Chew and Marzullo proved that the fused data is at least as accurate as the data from the least accurate module.[8]

The first step in our distributed dynamic sensor fusion methodology finds the smallest four dimensional region containing all points where $N'-f$ or more sensors agree. This region defines the accuracy limits of the fused data, and can be found using the algorithm described in Sec. 3.1. This algorithm[5] has complexity $O(f^D N \log N)$.

The second step in our method uses the results of step 1 to find an answer that robustly approximates the answer computed by all other modules in the system. Finding a robust approximation is nontrivial because the data received by each PE may differ radically due to the influence of the faulty modules. The robust approximation algorithm is given in Sec. 3.2, and has complexity[4] $O(N \log N)$.

Used in combination, these steps provide a fused answer that satisfies our robustness criteria and provides reliable accuracy bounds for the fused answer. The accuracy of the fused answer is always at least as accurate as the least accurate module in the system when temporal disturbances are taken into account. The overall complexity of our approach for sequential machines is $O(f^4 N \log N)$. This paper differs from our previous work in three important respects: (1) it combines the two methods into a unifying sensor fusion framework, (2) it is the first presentation of a fault-tolerant approximate agreement algorithm for multidimensional problems, and (3) it puts sensor fusion in a dynamic context.

It is important to realize that many applications for sensor data involve dynamic processes and sensor fusion involving distributed sensors must therefore account for temporal differences between modules.

3.1 *Finding the Optimal 4-D Region*

This step is an extension of the method presented in Brooks et al.[5] and Brooks and Iyengar.[12] We describe the algorithm as performed by module m_i. The identical algorithm is performed concurrently on all modules $m_0, m_1, \ldots, m_N$. As explained in Sec. 2, each module m_i measures one data sextuplet d_i and receives N sextuplets transmitted from the other modules. The time of arrival t_j of d_j is recorded. Module m_i has *a priori* knowledge of the acceptable temporal noise (Δt) for each module and the average transmission time (T_{ji}) from m_j to m_i. Module m_i's measurement was taken within the time range $t_i \pm \Delta t_i$. For all $m_j (j \neq i)$ the measurement was taken sometime within the range $t_j - T_{ji} \pm \Delta t_j$.

Analogous to the limit of acceptable temporal noise, there are known acceptable accuracy limits for every element of the sextuple d_j from module m_j. These are represented by Δx_j, Δy_j, Δz_j, $\Delta x_j'$, $\Delta y_i'$, and $\Delta z_i'$.

Theorem 3. The lower limit for x from reading d_j is $x_j - [\Delta x_j + (x_j' + \Delta x_j') * \Delta t_j]$, and the upper limit $x_j + \Delta x_j + (x_j' + \Delta x_j') * \Delta t_j$.

Proof. The velocity of O measured by m_j in direction x is x_j'. It is given that if m_j functions correctly the actual velocity x' must be in the range $x_j' - \Delta x_j' \leq x' \leq x_j' + \Delta x_j'$. So that the maximum possible velocity for O at the time of the measurement is $x_j' + \Delta x_j'$.

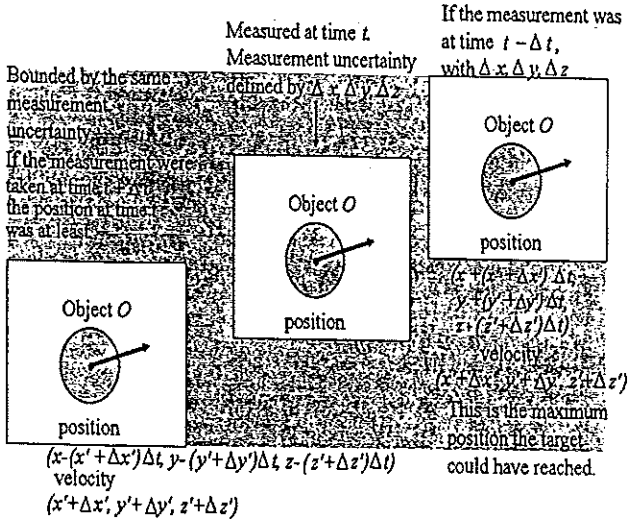

Fig. 3 Temporal uncertainty increasing the spatial uncertainty of the location of O.

The lower limit is found by assuming that the reading was actually taken at time $t_j + \Delta t_j$ and that the object was moving at the maximum velocity. This means that O moved the distance $(x_j' + \Delta x_j') * \Delta t_j$ from its actual position at time t_j before the reading was taken. In addition to this, the measurement inaccuracy for x_j is taken into account. This establishes the lower bound for the correct position of O at time t_j in dimension x as $x_j - [\Delta x_j + (x_j' + \Delta x_j') * \Delta t_j]$.

Similarly the upper bound is found by assuming that the reading was actually taken at time $t_j - \Delta t_j$. Which means that after the reading object O moved distance $(x_j' + \Delta x_j') * \Delta t_j$ before arriving at its true position at time t_j. Taking the uncertainty for x into account, an upper bound of $x_j + \Delta x_j + (x_j' + \Delta x_j') * \Delta t_j$ is established.

This shows that with temporal uncertainty Δt_j, position uncertainty Δx_j, and velocity uncertainty $\Delta x_j'$, the x coordinate of O at time t_j is bounded by $x_j - [\Delta x_j + (x_j' + \Delta x_j') * \Delta t_j]$ and $x_j + \Delta x_j + (x_j' + \Delta x_j') * \Delta t_j$.

Identical logic is used to prove corresponding limits on the position of O in the y and z spatial dimensions. Again, an identical approach can be taken using spherical coordinates when appropriate. This relationship is illustrated by Fig. 3. Note, unless a specific distribution is assumed within the uncertainties, the likelihood of O being at any specific position within this range is unknown.

The reading from every module m_j now consists of upper and lower bounds in each of four dimensions. This form may be referred to as either a 4-D hyper-rectangle, or a 4-rectangle. We denote this new form of the input data vector $\mathbf{d}_j$ as $\mathbf{d}_j'$. The set of $\mathbf{d}_j'$ is used to find the smallest possible 4-rectangle where we can be certain that O was contained within the spatial dimensions of the 4-rectangle

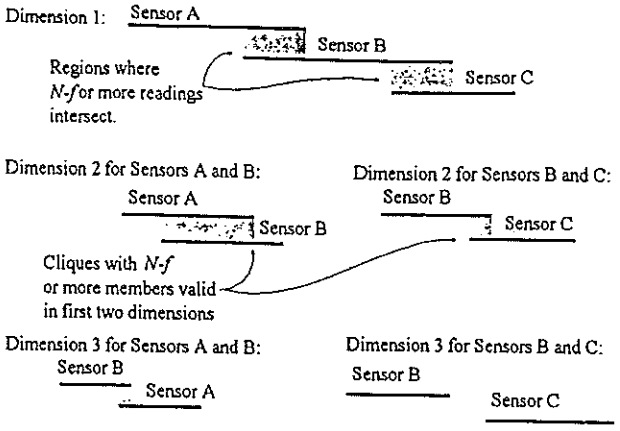

Fig. 4 The 3-D problem with $N=3$ and $f=1$. Final set of cliques consist only of clique AB.

at some time within the time dimension of the 4-rectangle. This 4-rectangle is the optimal region.

To calculate the optimal region, m_i converts the readings from all m_j into a set of $\mathbf{d}'_j$. The x ranges of the $\mathbf{d}'_j$ are sorted by their lower bound. The ranges are traversed in sorted order, keeping track of how many modules agree in any region. Any intersection of $N-f$ or more modules represents a range of possibly correct values for x. When an intersection of cardinality $N-f$ or greater is found, this process is performed recursively for the y, z, and t dimensions using exclusively the $\mathbf{d}'_j$ of modules whose values are contained in the intersection.

Any intersection of cardinality $N-f$ or greater in the t dimension is 4-rectangle representing an intersection of readings from $N-f$ or more modules in all four dimensions. A list is kept of all such 4-rectangles. When the procedure terminates, the list contains all combinations of $N-f$ or more modules whose readings agree in all 4 dimensions. To help visualize this process, Fig. 4 shows a simplified example of this logic with three dimensions, three modules, and one fault tolerated. Remember that the actual problem has four dimensions and the system must contain at least nine modules to tolerate one fault.

Any combination of $N-f$ or more sensors on the list could be the set of correctly functioning modules, and one of their ranges must therefore contain the correct reading. We call a set of modules a clique and define C as the set of cliques with $N-f$ or more elements whose readings intersect in all four dimensions. We define R as the set of associated ranges.

The elements of R are sorted in each dimension. The accuracy limitations of our procedure in any dimension are defined by the smallest lower bound and the largest upper bound of any element of R in that dimension.

The overall complexity comes from the recursion in step 5. Using Jayasimha's finding that there are at most $2f+1$ cliques of size $N-f$ or greater at each dimension,[11] we construct a recursion relation for step 5. The relation reduces to $N \log N * [(2f+1)^5]/2f$. The worst case complexity is therefore in the order of $O(f^4 N \log N)$.

The algorithm is

Algorithm: Optimal region
Input: The number of dimensions D, and a set of sensor readings S
Output: A set of 4-rectangles describing regions which may be correct readings

Step 1. Initialize a list of cliques, which we will call C, to NULL. Remember, a clique is a set of intersecting sensor readings.
Step 2. Calculate the uncertainty ranges using the temporal uncertainty. Put S in the form of N 4-rectangles.
Step 3. Sort all elements of S into ascending order along the x dimension. A reading is considered active if its lower bound has been traversed and its upper bound has yet to be traversed. Traverse the list in order, keeping track of the active readings. When $N-F$ or more readings are active, perform step 4 on the clique A of active readings using data from the y dimension.
Step 4. Sort all data for the dimension (y, z, or t) in clique A into ascending order. Traverse the list of readings in ascending order. When a region is found with $N-f$ or more readings active perform step 5 on this clique of active readings A'.
Step 5. There are three possible cases:
Case 1. Step 4 treated dimension y. Perform step 3 recursively using A' for A and data from the z dimension.
Case 2. Step 4 treated dimension z. Perform step 3 recursively using A' for A and data from the t dimension.
Case 3. Step 4 treated dimension t. The 4-rectangle of the intersection of A' is inserted into the list C.
Step 6. Steps 3 through 5 have treated all the sensor data. List C now contains 4-rectangles describing all cliques that correspond to intersections of $(N-F)$ or more sensor readings in all four dimensions; C is used as input data for the agreement algorithm.
The maximum upper-bound and minimum lower-bound of all elements of C are found for each dimension. This is the accuracy bound for the algorithm.

3.2 *Robust Agreement Using the Optimal Region*

Step 2 of our methodology uses the output of Sec. 3.1 to find a numeric value that converges with the values determined by the other modules. More details regarding the distributed agreement problem can be found in Brooks and Iyengar.[4] Agreement in this problem increases the precision of the readings and thus aids coordination among the modules. This is a nontrivial problem since the algorithm must tolerate a limited number of faulty inputs that may be arbitrarily different for each module. Note that even the correct inputs are inexact. The solution to this problem is based on research done on the Byzantine generals problem. More information on the Byzantine Generals Problem can be found in Barborak et al.[7]

The algorithm is

Algorithm: Robust agreement
Input: A list C of 4-rectangles

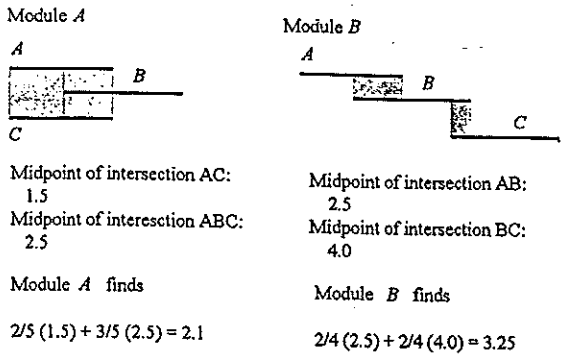

Fig. 5 Robust agreement by modules A and B when faulty module C sends different information to both.

Output: A value in each dimension that is within the accuracy limits found by all other modules

Step 1. Sum the midpoints of the x values of each 4-rectangle in C multiplied by the number of sensors whose readings intersect in that range, and divide by the total number of factors. Output this value as the answer in the x dimension.

Step 2. Sum the midpoints of the y values of each 4-rectangle in C multiplied by the number of sensors whose readings intersect in that range, and divide by the total number of factors. Output this value as the answer in the y dimension.

Step 3. Sum the midpoints of the z values of each 4-rectangle in C multiplied by the number of sensors whose readings intersect in that range, and divide by the total number of factors. Output this value as the answer in the z dimension.

Step 4. Sum the midpoints of the t values of each 4-rectangle in C multiplied by the number of sensors whose readings intersect in that range, and divide by the total number of factors. Output this value as the time t the measurement was made.

Each dimension is handled identically. The value in dimension x is found by summing the midpoint of all x regions contained in R multiplied by the cardinality of the clique whose intersection defines the region. This sum is normalized by dividing it by the sum of the cardinalities of all the cliques in R. Figure 5 shows a simplified example of the calculations performed on two modules that receive different information from the faulty module C. The values in the y, z, and t dimensions are found by identical calculations using regions defined in the corresponding dimension.

As long as our assumptions hold, this method provides a robust approximation of the position of O at the approximate time calculated by the procedure. The values found by any module will be within the bounds calculated in Sec. 3.1 by any other module. These bounds are at most as large as the uncertainty of the least accurate individual module as demonstrated in Chew and Marzullo.[8] Note that this uncertainty in the spatial dimensions is dilated by the movement of O during temporal uncertainty as proved by Theorem 3.

A proof of convergence and strict convergence bounds of $2f/N$ for each iteration of this algorithm are given for the case where all modules have the same accuracy in Mahaney and Schneider.[13] In the case where modules have differing accuracies the same proof can be used to give pessimistic bounds by assuming each module has the accuracy of the least accurate module.

Fusing values of the velocity components is more problematic. No assumption is made directly regarding the acceleration of O, two modules could correctly measure the velocity of O within Δt and have significantly different results. A reasonable value for x' can be obtained, however, by finding the average value of x' in each clique in C and then computing the weighted average of these averages in a manner similar to the robust agreement algorithm. Identical calculations provide estimates for the value of y' and z'.

This estimate is reasonable since only modules whose spatial readings intersect with readings from $N-f$ or more other modules will be contained in the average. These modules have a high probability of being correct. Also, modules that are included in many intersecting regions will be weighted more heavily than modules included in only a few intersections. This reduces the variation in the fused velocity value from module to module.

4 Experimental Results

This section illustrates our method by presenting a sample problem, followed by results obtained from a simulation based on a target tracking problem. All results given in this section use the same modules to observe object O. Each module has known accuracy limits, which are given in Table 1. As stated previously, all modules that function correctly broadcast the same correct reading to all the other modules. These readings consist of a sextuple $d_j = (x_j, y_j, z_j, x_j', y_j', z_j')$. A value for the time the reading was taken t_j is computed by the module m_i that receives the reading. The simulation uses a pseudo-random number generator to ensure that the actual value is within the given tolerance limits of the reported reading, and all readings that fulfill this requirement are equally likely.

Out of nine modules, one faulty module can be tolerated. The faulty module f is chosen at random with each iteration of the simulation. Sensor f sends each m_i a different reading. The faulty reading is generated within accuracy constraints that are larger than those given in Table 1. These errors are more difficult for the algorithm to process correctly than wildly incorrect readings which will be filtered out completely in step 1.

The time the reading is taken t_f is generated using a uniform distribution, to be within the temporal noise limit Δt_f of the correct t but the reading is chosen at random from within $t_f \pm \Delta t_f$. It is not certain that the correct time t is contained within the temporal noise limits of the reading. Remember that all modules take readings at what they consider to be regular intervals. The system cannot be entirely synchronous, however, since there is no common clock and since data transmission may inject additional delays. The temporal noise models both clock skewing and variance in the transfer times between modules, as explained in Sec. 2. If temporal noise cannot be limited, fusion of dynamic data from distributed sensors is not feasible.

The velocity readings are chosen within five times the allowed noise limits, and the readings within the spatial

Table 1 Known accuracy limitations for modules 1 to 9.

Module	Allowable Uncertainty						
	Temporal	x position	y position	z position	x velocity	y velocity	z velocity
1	0.01	0.5	0.4	0.5	0.1	0.1	0.1
2	0.02	0.3	0.25	0.2	0.08	0.08	0.2
3	0.4	0.1	0.1	0.3	0.05	0.1	0.05
4	0.01	0.4	0.5	0.3	0.04	0.04	0.04
5	0.1	0.05	0.05	0.05	0.01	0.01	0.01
6	0.08	0.2	0.25	0.3	0.12	0.12	0.13
7	0.03	0.35	0.44	0.35	0.09	0.09	0.09
8	0.15	0.09	0.08	0.12	0.09	0.05	0.05
9	0.18	0.35	0.45	0.2	0.1	0.1	0.1

dimensions are chosen within twice the allowed noise limits. Since the uncertainty allowed is a function of the velocity, these readings will be likely to overlap correct readings and will have a larger uncertainty range than normally allowed. The incorrect readings can not be trivially dismissed and illustrate the need of a robust agreement protocol.

The errors allowed for the faulty module were chosen deliberately to simulate the types of errors that would be most difficult for the approximate agreement protocol to handle correctly.

4.1 *Detailed Example of Distributed Dynamic Sensor Fusion*

For this example, module 2 is faulty. Table 2 gives the upper and lower bounds returned by the correctly functioning modules, and the values found by expanding the uncertainty to account for temporal noise. Note that O is located at the origin (0,0,0) and traveling with a velocity of (0.1, 0.1, 0.1). We position O at the origin to make the results easier for the reader to evaluate. This choice of coordinates is arbitrary and has no influence on the algorithm. Table 3

Table 2 Readings from the eight correctly functioning sensors.

Original	Mod. 1	Mod. 3	Mod. 4	Mod. 5	Mod. 6	Mod. 7	Mod. 8	Mod. 9
Min. t	−0.0088	−0.0138	−0.0142	−0.1449	−0.1199	−0.0150	−0.1629	−0.2448
Max. t	0.0112	0.0262	0.0058	0.0551	0.0401	0.0450	0.1371	0.1152
Min. x	−0.4309	−0.1350	−0.0460	−0.0968	−0.3519	−0.5173	−0.0847	−0.2274
Max. x	0.5691	0.4650	0.7540	0.0032	0.0481	0.1827	0.0953	0.4726
Min. x'	0.0239	0.0938	0.0782	0.0957	−0.0114	0.0630	0.0571	−0.0705
Max. x'	0.2239	0.2538	0.1582	0.1157	0.2286	0.2430	0.2371	0.1295
Min. y	0.7147	−0.3861	−0.7269	−0.0408	−0.3886	−0.0297	−0.1265	−0.2143
Max. y	0.0853	0.1139	0.2731	0.0592	0.1114	0.8503	0.0335	0.6857
Min. y'	0.0016	−0.0507	0.0572	0.0956	−0.0077	−0.0358	0.0273	0.0298
Max y'	0.2016	0.1093	0.1372	0.1156	0.2323	0.1442	0.1273	0.2298
Min. z	−0.8813	−0.2835	−0.0739	−0.0333	−0.0365	−0.1279	−0.0213	−0.2256
Max. z	0.1187	0.1165	0.5261	0.0667	0.5635	0.5721	0.2187	0.1744
Min. z'	0.0175	0.0560	0.0481	0.0977	−0.0212	−0.0407	0.0650	−0.0310
Max. z'	0.2175	0.4560	0.1281	0.1177	0.2388	0.1393	0.1650	0.1690
Expanded								
Min. x	−0.4331	−0.1401	−0.0476	−0.1084	−0.3702	−0.5246	−0.1202	−0.2507
Max. x	0.5713	0.4701	0.7556	0.0147	0.0664	0.1899	0.1309	0.4959
Min. y	−0.7168	−0.3883	−0.7283	−0.0524	−0.4072	−0.0341	−0.1456	−0.2557
Max. y	0.0873	0.1161	0.2744	0.0707	0.1300	0.8546	0.0526	0.7270
Min. z	−0.8835	−0.2926	−0.0752	−0.0451	−0.0556	−0.1320	−0.0460	−0.2560
Max. z	0.1209	0.1256	0.5274	0.0785	0.5826	0.5763	0.2435	0.2048

Table 3 Faulty readings from module 2.

	Module Receiving Faulty Reading From Module 2								
	Mod. 1	Mod. 2	Mod. 3	Mod. 4	Mod. 5	Mod. 6	Mod. 7	Mod. 8	Mod. 9
Min. t	−0.3594	−0.4429	−0.7731	−0.3672	−0.0447	−0.3577	−0.5228	0.0481	−0.7837
Max. t	0.4406	0.3571	0.0269	0.4328	0.7553	0.4423	0.2772	0.8481	0.0163
Min. x	−0.0005	−0.1907	−0.2450	0.0273	−0.1299	−0.2862	−0.2715	−0.2433	−0.0932
Max. x	0.1995	0.0093	−0.0450	0.2273	0.0701	−0.0862	−0.0715	−0.0433	0.1068
Min. x'	0.0026	−0.1523	−0.0127	0.1045	−0.1442	−0.1969	−0.0921	0.0877	−0.1648
Max. x'	0.1026	−0.0523	0.0873	0.2045	−0.0442	−0.0969	0.0079	0.1877	−0.0648
Min. y	−0.2030	−0.1022	−0.2240	−0.1797	0.0498	−0.0591	0.0744	−0.2382	−0.0671
Max. y	−0.0030	0.0978	−0.0240	0.0203	0.2498	0.1409	0.2744	−0.0382	0.1329
Min. y'	−0.2157	0.3910	0.0180	0.2128	−0.1949	0.3180	0.0025	0.4981	0.1177
Max. y'	−0.0157	0.5910	0.2180	0.4128	0.0051	0.5180	0.2025	0.6981	0.3177
Min. z	−0.3465	−0.3882	−0.0435	−0.3580	−0.7502	0.3118	−0.0459	0.2213	−0.6407
Max. z	0.2535	0.2118	0.5565	0.2420	−0.1502	0.9118	0.5541	0.8213	−0.0407
Min. z'	0.1824	0.0951	−0.1428	0.1813	0.1681	0.1913	−0.1550	−0.0466	0.1428
Max. z'	0.2824	0.1951	−0.0428	0.2813	0.2681	0.2913	−0.0550	0.0534	0.2428
Expanded									
Min. x	−0.0416	−0.1698	−0.2800	−0.0545	−0.1122	−0.2474	−0.2747	−0.3184	−0.0672
Max. x	0.2405	−0.0117	−0.0101	0.3091	0.0524	−0.1249	−0.0683	0.0318	0.0809
Min y	−0.1967	−0.3386	−0.3112	−0.3448	0.0477	−0.2663	−0.0066	−0.5175	−0.1942
Max. y	−0.0092	0.3342	0.0632	0.1854	0.2518	0.3481	0.3554	0.2410	0.2600
Min. z	−0.4594	−0.4662	−0.0264	−0.4705	−0.8574	0.1953	−0.0239	0.1999	−0.7378
Max. z	0.3665	0.2898	0.5394	0.3546	−0.0430	1.0283	0.5321	0.8426	0.0564

shows the original upper and lower bounds transmitted by the faulty module 2 to all modules and the corresponding expanded uncertainty ranges.

Initial inspection of the data confirms that the actual values of $d=(0, 0, 0, 0.1, 0.1, 0.1)$ are contained within the accuracy limits of all readings given in Table 2. The accuracy limits given in Table 3, however, do not always enclose the correct values. On the other hand, the values in Table 3 sometimes enclose the correct values and are often close to the actual values. These readings can only be partially filtered by the matching procedure in step 1 of the algorithm.

Step 1 is run concurrently on all modules and produces a list of regions where $N-f$ or more readings intersect in all four dimensions. Table 4 lists the nine valid regions that are found by module 1. Note that the regions farthest from the actual value usually appear only in one or two 4-rectangles, while regions containing the actual value occur several times. The most extreme example of this is given by the time variable.

The data from Table 4 is processed in two ways. First the lower and upper bounds are sorted. In this way, the minimum lower bound and the maximum upper bound provide limits for the accuracy of the sensor readings.

Then step 2 finds a robust value that converges with the values found by the other modules. For module 1 this is done by computing the midpoint of each range in Table 4, multiplying these midpoints by the cardinality of the region, summing these values, and dividing the sum by the total number of factors (in this case 72). Table 5 presents the results of this process for all nine modules in this example. Note that the values found by each module are much closer to the real values ($t=0$, $x=0$, $y=0$, $z=0$, $x'=0.1$, $y'=0.1$, $z'=0.1$) than any of the initial readings. Similarly, the accuracy bounds are tighter than the bounds of the original data.

Summary statistics of this example are given in Table 6. The results found are very good approximations of the actual values. Even the extreme values found by the modules are well within the original accuracy bounds. It is also interesting to note that the amount of disagreement between the modules is minimal. Note that the average position and average error differ in that the error is the absolute value of the position when the actual target position is the origin.

4.2 *Simulation Results*

The simulation has been tested using a large number of scenarios with different seeds for the pseudo-random number generator and a number of different combinations of modules. The results always differ slightly but are qualitatively very similar. For the sake of comparison in this paper, we use only the modules defined in Table 1. Remember that a new faulty module is chosen at random with each iteration of the algorithm, so that every module is defective for a significant number of iterations. As the number of

Table 4 The 4-rectangles where eight or more readings intersect in all four dimensions.

Region Cardinality	*A* 8	*B* 8	*C* 8	*D* 8	*E* 8	*F* 8	*G* 8	*H* 8	*I* 8
Min. *t*	−0.0138	−0.0088	−0.0088	−0.0088	−0.0088	−0.0088	−0.0088	−0.0088	0.0058
Max. *t*	−0.0088	0.0058	0.0058	0.0058	0.0058	0.0058	0.0058	0.0058	0.0112
Min. *x*	−0.0416	−0.0476	−0.0416	−0.0416	−0.0416	−0.0416	−0.0416	0.0147	−0.0416
Max. *x*	0.0147	−0.0416	0.0147	0.0147	0.0147	0.0147	0.0147	0.0664	0.0147
Min. *x'*	0.0386	0.0412	0.0337	0.0296	0.0369	0.0296	0.0412	0.0296	0.0318
Max. *x'*	0.1836	0.1987	0.1812	0.1971	0.1881	0.1971	0.1987	0.1971	0.1918
Min. *y*	−0.0341	−0.0341	−0.0524	−0.0341	−0.0341	−0.0341	−0.0092	−0.0341	−0.0341
Max. *y*	−0.0092	0.0526	−0.0341	−0.0092	−0.0092	−0.0092	0.0526	−0.0092	−0.0092
Min. *y'*	−0.0125	0.0147	−0.0078	−0.0243	−0.0109	−0.0243	0.0147	−0.0243	−0.0195
Max. *y'*	0.1350	0.1622	0.1422	0.1457	0.1424	0.1457	0.1622	0.1457	0.1430
Min. *z*	−0.0451	−0.0451	−0.0451	−0.0460	−0.0451	0.0785	−0.0451	−0.0460	−0.0451
Max. *z*	0.0785	0.0785	0.0785	−0.0451	0.0785	0.1209	0.0785	0.1209	0.0785
Min. *z'*	0.0445	0.0239	0.0518	0.0345	0.0415	0.0345	0.0239	0.0345	0.0407
Max. *z'*	0.2120	0.2039	0.2218	0.2245	0.2126	0.2245	0.2039	0.2245	0.2232

Table 5 Results of distributed dynamic sensor fusion algorithm for each module.

Robust Values	Mod. 1	Mod. 2	Mod. 3	Mod. 4	Mod. 5	Mod. 6	Mod. 7	Mod. 8	Mod. 9
t	−0.0015	−0.0015	−0.0015	−0.0015	−0.0015	−0.0015	0.0018	−0.0015	−0.0015
x	−0.0109	−0.0348	−0.0341	−0.0143	−0.0210	−0.0164	−0.0644	−0.0164	−0.0157
x'	0.1136	0.0952	0.1114	0.1240	0.0995	0.1200	0.1066	0.1200	0.0938
y	−0.0159	0.0093	0.0089	0.0093	0.0467	0.0093	0.0184	0.0093	0.0093
y'	0.0683	0.1322	0.0912	0.1156	0.0701	0.0884	0.0888	0.0884	0.1023
z	0.0212	0.0190	0.0263	0.0212	−0.0308	0.0167	0.0238	0.0167	0.0068
z'	0.1267	0.1190	0.0957	0.1302	0.1247	0.1139	0.0978	0.1139	0.1243
Accuracy Bounds									
Min. *t*	−0.0138	−0.0138	−0.0138	−0.0138	−0.0138	−0.0088	−0.0088	−0.0088	−0.0138
Max. *t*	0.0112	0.0112	0.0112	0.0112	0.0112	0.0058	0.0112	0.0058	0.0112
Min. *x*	−0.0476	−0.1084	−0.1084	−0.0545	−0.1084	−0.0476	−0.1084	−0.0476	−0.0672
Max. *x*	0.0664	0.0147	0.0147	0.0664	0.0524	0.0147	0.0147	0.0147	0.0664
Min. *x'*	0.0296	0.0102	0.0277	0.0423	0.0112	0.0412	0.0199	0.0412	0.0087
Max. *x'*	0.1987	0.1987	0.1987	0.2098	0.1987	0.1987	0.1987	0.1987	0.1987
Min. *y*	−0.0524	−0.0524	−0.0524	−0.0524	−0.0341	−0.0341	−0.0341	−0.0341	−0.0524
Max. *y*	0.0526	0.0707	0.0632	0.0707	0.0707	0.0526	0.0526	0.0526	0.0707
Min. *y'*	−0.0243	0.0147	0.0050	0.0293	−0.0217	0.0147	0.0078	0.0147	0.0147
Max. *y'*	0.1622	0.2216	0.1750	0.1993	0.1622	0.1622	0.1703	0.1622	0.1874
Min. *z*	−0.0460	−0.0460	−0.0451	−0.0460	−0.0460	−0.0451	−0.0451	−0.0451	−0.0460
Max. *z*	0.1209	0.1209	0.1209	0.1209	0.0785	0.0785	0.0785	0.0785	0.0785
Min. *z'*	0.0239	0.0236	−0.0061	0.0344	0.0239	0.0239	−0.0015	0.0239	0.0239
Max. *z'*	0.2245	0.2136	0.2039	0.2244	0.2227	0.2039	0.2039	0.2039	0.2196

Table 6 Summary of final results from example.

Average position	$t=-0.0011$	$x=-0.0253$	$y=0.0116$	$z=0.0134$
Average position error	$t=0.0015$	$x=0.0253$	$y=0.0151$	$z=0.0203$
Average velocity		$x=0.1093$	$y=0.0939$	$z=0.1163$
Average velocity error		$x=0.0119$	$y=0.0172$	$z=0.0177$
Maximum position	$t=0.0018$	$x=-0.0109$	$y=0.0467$	$z=0.0263$
Minimum position	$t=-0.0015$	$x=-0.0644$	$y=-0.0159$	$z=-0.0308$
Maximum velocity		$x=0.1240$	$y=0.1322$	$z=0.1302$
Minimum velocity		$x=0.0938$	$y=0.0683$	$z=0.0957$
Size of maximum disagreement	$t=0.0033$	$x=0.0535$	$y=0.0626$	$z=0.0571$
Size of maximum velocity disagreement		$x=0.0302$	$y=0.0640$	$z=0.0345$
Average upper bounds	$t=0.0100$	$x=0.0361$	$y=0.0618$	$z=0.0973$
Average lower bounds	$t=-0.0121$	$x=-0.0776$	$y=-0.0442$	$z=-0.0456$
Maximum upper bound	$t=0.0112$	$x=0.0664$	$y=0.0707$	$z=0.1209$
Minimum lower bound	$t=-0.0138$	$x=-0.1084$	$y=-0.0524$	$z=-0.0460$

iterations increase the average values found approach the correct values asymptotically. On the other hand, the average error remains relatively constant. The maximum and minimum values found quickly reach values that are close to their extremes and then remain almost constant. These results match our expectations as to how the method should function. Table 7 summarizes the results of testing the simulation with increasing numbers of iterations. It is interesting to note the small magnitude of change between 10 and 2000 iterations. The magnitude of the extreme values found during 2000 iterations are still extremely small, which indicates that the method is indeed robust.

One theoretical shortcoming of our method is that the inaccuracy of the position vector is influenced by the velocity of the object O. In theory, a problem could arise when faulty modules return inflated velocity readings for O. Their accuracy bounds then become inflated and could theoretically decrease the accuracy of the entire system. One way to avoid this would be to put limits on the acceleration accepted by the system and discard readings where the velocity of fewer than f modules conflict with the rest of the modules.

To test the magnitude of this problem, we have run simulations with increasing velocity. Table 8 summarizes our results. The results given in Table 8 are encouraging. Although the accuracy of the system is lessened as the speed of O increases, the magnitude of this effect is not very large. Increasing the speed of the object by a factor of 200 has increased the average position error by a factor of about 10. Based on these results and the inaccuracies inherent to the problem it appears that this factor is not very significant. It is therefore not essential to limit the acceleration accepted by the system since the fusion algorithm appears to be capable of compensating for this effect.

6 Conclusion

This paper presents a model of the inaccuracies inherent in real-time multisensor systems. In discussing these inaccuracies, a declaration of the requirements for robust fusion has been made. Many approaches exist for solving similar problems. In this section we briefly compare our method with: weighted average, Kalman filtering, Bayesian inference, and Dempster-Shafer inference.

Weighted average was one of the first approaches used for data fusion. Given N sensor readings $x_1,...,x_N$ parameters $w_1,...,w_N$, where $\Sigma\, w_i=1$ are used to find a fused sensor reading $\Sigma\, w_i x_i$. Judicious choice of w_i can compensate for sensors with different accuracy and reliability. This method was tested with the robot HILARE.[2] It is simple and efficient, with complexity $O(N)$. It is suitable for real time. On the other hand, it ignores the noise factors involved and does not compensate for module failures. The failure of any module with a nonzero weight w can result in the entire system failing. It also fails to provide any measure of the accuracy of the value obtained. These drawbacks make it unsuitable for use in a distributed system.

The problem posed in this paper is not unlike the concept behind distributed Kalman filters.[14] The main differences are that we consider a less restrictive noise model and explicitly consider temporal noise. Kalman filters are a great improvement over the weighted-average approach. The exact algorithm can be found in Bramer and Siffling.[15] Extended Kalman filters could be used for our tracking example since our assumptions enable the problem to be phrased as a set of linear equations.

If the noise factors are Gaussian, the extended Kalman filter provides a minimum variance unbiased estimate of the position of O along with a covariance matrix providing a measure of the quality of this estimate. If the noise is not Gaussian, this can be problematic. A recent attempt to use Kalman filters with non-Gaussian noise for sensor data fusion is Maheshkumar et al.[16] Note that a Kalman filter can handle temporal noise only if it is small enough to be only one component of the over-all Gaussian noise. Also, equipment failure is not tolerated by Kalman filters. Thus the Kalman filter only satisfies part of the robustness criteria we set in this paper.

Table 7 Summary statistics of sensor fusion over 10 and 2000 iterations.

10 iterations				
Average position	t=0.0003	x=0.0030	y=−0.0172	z=−0.0107
Average position error	t=0.0029	x=0.0243	y=0.0254	z=0.0203
Average velocity		x=0.1025	y=0.1092	z=0.1055
Average velocity error		x=0.0155	y=0.0169	z=0.0138
Maximum position	t=0.0086	x=0.0615	y=0.0522	z=0.1080
Minimum position	t=−0.0058	x=−0.0471	y=−0.0714	z=−0.0494
Maximum velocity		x=0.1583	y=0.1648	z=0.15763
Minimum velocity		x=0.0515	y=0.0575	z=0.0094
Size of maximum disagreement	t=0.0089	x=0.0565	y=0.0802	z=0.1167
Size of maximum velocity disagreement		x=0.0930	y=0.0666	z=0.0956
Average upper bounds	t=0.0093	x=0.0639	y=0.0361	z=0.0421
Average lower bounds	t=−0.0091	x=−0.0586	y=−0.0731	z=−0.0606
Maximum upper bound	t=0.0168	x=0.1310	y=0.0947	z=0.1881
Minimum lower bound	t=−0.0148	x=−0.0998	y=−0.1635	z=−0.1213
2000 iterations				
Average position	t=0.0000	x=−0.0001	y=−0.0005	z=0.0000
Average position error	t=0.0023	x=0.0194	y=0.0198	z=0.0202
Average velocity		x=0.1008	y=0.1009	z=0.1012
Average velocity error		x=0.0163	y=0.0163	z=0.0196
Maximum position	t=0.0132	x=0.0975	y=0.1246	z=0.1430
Minimum position	t=−0.0119	x=−0.1055	y=−0.1214	z=−0.1680
Maximum velocity		x=0.2078	y=0.1971	z=0.2306
Minimum velocity		x=−0.0001	y=0.0120	z=−0.0304
Size of maximum disagreement	t=0.0177	x=0.1396	y=0.1350	z=0.1977
Size of maximum velocity disagreement		x=0.1365	y=0.1382	z=0.1975
Average upper bounds	t=0.0094	x=0.0573	y=0.0565	z=0.0581
Average lower bounds	t=−0.0094	x=−0.0571	y=−0.0574	z=−0.0576
Maximum upper bound	t=0.0248	x=0.2191	y=0.2531	z=0.3344
Minimum lower bound	t=−0.0285	x=−0.2317	y=−0.2713	z=−0.3066

Kalman filters are reasonably efficient and can be implemented as a series of matrix manipulations. On sequential machines they are of complexity $O(N^3)$, but they could be implemented as specially designed parallel hardware so as to reduce the complexity dramatically.[17] At the moment Kalman filters are probably the most widely applied approach to this type of problem. The computational complexity of the filters can unfortunately lead to excessive processing times for large systems.

Bayesian inference uses sensor inputs along with an *a priori* model of the environment and known conditional probabilities to find a maximum likelihood value for the fused sensor readings.[3] The use of prior information in Bayesian inference can be problematic and lead to unstable results under certain conditions.[2] This is a drawback since the prior information necessary for reliable results is not always available.

Dempster-Shafer reasoning is an extension of the Bayesian approach which does not have these drawbacks. Unfortunately, the Dempster-Shafer approach has $O(2^N)$ complexity.[2] The exponential complexity of the Dempster-Shafer approach makes it unsuitable for real-time applications. It is also unclear how to combine the possibility of arbitrary errors due to component failure with either approach.

Therefore weighted averaging, Kalman filters, and Bayesian techniques do not fulfill our robustness requirements. In addition to this, Kalman filters and Dempster-Shafer techniques have complexities of $O(N^3)$ and $O(2^N)$,

Table 8 Results with 10 times and 200 times the velocity of the previous problem.

1000 iterations with d=(0,0,0,1,1,1)				
Average position	t=0.0001	x=0.0003	y=−0.0009	z=−0.0012
Average position error	t=0.0024	x=0.0363	y=0.0375	z=0.0355
Average velocity		x=1.0010	y=1.0005	z=1.0005
Average velocity error		x=0.0189	y=0.0192	z=0.0220
Maximum position	t=0.0132	x=0.1892	y=0.2042	z=0.1895
Minimum position	t=−0.0130	x=−0.2186	y=−0.1987	z=−0.2631
Maximum velocity		x=1.0973	y=1.0949	z=1.1298
Minimum velocity		x=0.9078	y=0.9049	z=0.8738
Size of maximum disagreement	t=0.0177	x=0.2538	y=0.2547	z=0.2854
Size of maximum velocity disagreement		x=0.1364	y=0.1382	z=0.1989
Average upper bounds	t=0.0101	x=0.1283	y=0.1255	z=0.1205
Average lower bounds	t=−0.0100	x=−0.1273	y=−0.1282	z=−0.1225
Maximum upper bound	t=0.0270	x=0.3952	y=0.4133	z=0.3786
Minimum lower bound	t=−0.0285	x=−0.3915	y=−0.4285	z=−0.3776
1000 iterations with d=(0,0,0,20,20,20)				
Average position	t=0.0000	x=0.0003	y=−0.0039	z=−0.0051
Average position error	t=0.0025	x=0.1189	y=0.1201	z=0.1075
Average velocity		x=20.0008	y=20.0001	z=20.0005
Average velocity error		x=0.0225	y=0.0228	z=0.0262
Maximum position	t=0.0132	x=0.5510	y=0.6135	z=0.5482
Minimum position	t=−0.0130	x=−0.6575	y=−0.5218	z=−0.4697
Maximum velocity		x=20.0982	y=20.1016	z=20.1386
Minimum velocity		x=19.8962	y=19.9059	z=19.8638
Size of maximum disagreement	t=0.0177	x=0.7216	y=0.6946	z=0.5954
Size of maximum velocity disagreement		x=0.1448	y=0.1465	z=0.2349
Average upper bounds	t=0.0111	x=0.5177	y=0.5143	z=0.4736
Average lower bounds	t=−0.0110	x=−0.5174	y=−0.5272	z=−0.4808
Maximum upper bound	t=0.0286	x=1.2159	y=1.2525	z=1.1343
Minimum lower bound	t=−0.0285	x=−1.2186	y=−1.3473	z=−1.1265

respectively, which for many practical applications requires significantly more computation than the method presented in this paper.

The method we have presented is capable of dealing with dynamic problems in a deterministic manner with complexity $O(f^4 N \log N)$. This method can be run on each node in a distributed system and provides answers that enable coordination among independent modules even in the presence of a limited number of arbitrary errors.

This method is suitable for real-time applications, tolerates up to $N/8$ faulty modules since it deals with uncertainty in four dimensions, provides accuracy limits for the results obtained, and is general in scope and can be applied to a large number of potential applications.

Acknowledgments

This work was supported in part by the Office of Naval Research grant N 00014-94-1-0343. The authors would like to thank the anonymous reviewers for their suggestions, which have greatly improved the article.

References

1. S. S. Iyengar, L. Prasad, and Hla Min, *Advances in Distributed Sensor Integration: Application and Theory*, Prentice-Hall, Englewood Cliffs, NJ (1995).
2. R. C. Luo and M. G. Kay, "Data fusion and sensor integration: state-of-the-art 1990s," in *Data Fusion in Robotics and Machine Intelligence*, M. A. Abidi and R. C. Gonzalez Eds. pp. 7–135, Academic Press, (1992).
3. H. F. Durrant-Whyte, "Sensor models and multisensor integration," *Int. J. Robot. Res.* 7(6), 97–113 (1988).

4. R. R. Brooks, and S. S. Iyengar, "Robust distributed computing and sensing algorithm," *IEEE Comput.* **29**, 53–60 (June 1996).
5. R. R. Brooks, N. S. V. Rao, and S. S. Iyengar, "Resolution of contradictory sensor data," *J. Intell. Automat. Soft Comput.* (in press) 1997.
6. T. J. Broida "Kinematic and statistical models for data fusion using Kalman filtering," in *Data Fusion in Robotics and Machine Intelligence*, M. A. Abidi and R. C. Gonzalez, Eds., pp. 311–365, Academic Press, San Diego, CA (1992).
7. M. Barborak, M. Malek, and A. Dahbura, "The consensus problem in fault tolerant computing," *ACM Comput. Surv.* **25** (2), 171–220 (1993).
8. P. Chew and K. Marzullo, "Masking failures of multidimensional sensors," in *IEEE Proc. 10th Symp. on Reliable Distributed Systems*, pp. 32–41, IEEE Computer Society Press, Los Alamitos, CA (1991).
9. T. Krol, "(N,K) concept fault tolerance," *IEEE Trans. Comput.* **C-35** (4), 339–349 (1986).
10. D. P. Siewiorek and R. S. Swarz, *The Theory and Practice of Reliable System Design*, Digital Press, Bedford, MA (1982).
11. D. N. Jayasimha, "Fault tolerance in a multisensor environment," Technical Report, Department of Computer Science, Ohio State University (1993).
12. R. R. Brooks and S. S. Iyengar, "Optimal matching algorithm for multi-dimensional sensor readings," in *Sensor Fusion and Networked Robotics*, P. Schenker and G. McKee, Eds. *Proc. SPIE* **2589**, 91–99 (1995).
13. S. Mahaney and F. Schneider "Inexact agreement: accuracy, precision, and graceful degradation," in *4th ACM Symp. on the Principles of Distributed Computing* pp. 237–249 (1985).
14. C. Brown, H. Durrant-Whyte, J. Leonard, B. Rao, and B. Steer, in *Data Fusion in Robotics and Machine Intelligence*, M. A. Abidi and R. C. Gonzalez, Eds., pp. 267–310, Academic Press, San Diego, CA (1992).
15. K. Bramer and G. Siffling, *Kalman-Bucy Filters*, Artech House, Norwood, MA (1989).
16. J. R. Maheshkumar, V. Veeranna, S. S. Iyengar, and R. R. Brooks, "A new computational technique for complementary sensor integration in detection-localization systems," *J. Opt. Eng.* **35**(3), 674–684 (1996).
17. T. Kerr "Decentralized filtering and redundancy management for multisensor navigation," *IEEE Trans. Aerospace Electron. Syst.* **AES-23**(1), 83–119 (1987).

Richard R. Brooks is a professor with the Institute of Communications Science and Technology at California State University, Monterey Bay. He received a PhD in computer science from Louisiana State University in 1996, a BA in mathematical sciences from the Johns Hopkins University in 1979, and studied operations research and computer science at the Conservatoire National des Arts et Metiers (CNAM) in Paris, France. His research interests include robotics, sensor fusion, machine perception, and systems dependability. His work experience includes projects with Goddard Spaceflight Center, Radio Free Europe/Radio Liberty Munich, and the French stock exchange authority. As a consultant for the World Bank he helped implement their network in Africa, Eastern Europe, and Central Asia. Dr. Brooks is a member of ACM and INFORMS.

S. Sitharamar Iyengar chairs the Department of Computer Science at Louisiana State University (LSU). He has been involved with research in high-performance algorithms and data structures since receiving his PhD in engineering from Mississippi State University in 1974 and his MS from the Indian Institute of Science (IIS). He was a principal investigator on research projects supported by the Office of Naval Research, the National Aeronautics and Space Administration, the National Science Foundation, the Jet Propulsion Laboratory (JPL), and the U.S. Army Office. He has published several books and over 200 research papers. He has been a visiting professor at JPL, the Oak Ridge National Laboratory, and IIS. Dr. Iyengar is a series editor for Neuro Computing of Complex Systems, area editor for the Journal of Computer Science and Information, and has served as a guest editor for seven different technical journals. He was awarded the Williams Evans Fellowship from the University of Otago, New Zealand, in 1991, and the LSU Distinguished Faculty Award for excellence in research and the LSU Tiger Athletic Foundation Teaching Award, both in 1996. Dr. Iyengar is a fellow of the IEEE, a distinguished visitor of the IEEE Computer Society (1995 to 1998), an ACM National Lecturer since 1985, a member of the New York Academy of Sciences, is on the NIH-NLM review committee on medical informatics, and has been the program chairman for many national and international conferences, and has been a consultant to several government and industrial organizations.