SAND2008-3826C

Development of a Software Framework that Provides Flexible Sensor Fusion

Tiffany A. S. Pierce and Douglas G. Adams
Sandia National Laboratories
Albuquerque NM, USA, 87185-0780

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

July 13-17, 2008

ABSTRACT

Since 2006, Sandia National Laboratories has developed a sensor fusion application that combines raw security sensor data in an effort to reduce nuisance alarms for security systems. This Fusion Framework allows various algorithms to be "plugged in" to combine sensor information. The purpose of the software is to reduce alarm data rates as well as nuisance alarms caused by environmental conditions. The software is part of an effort by Sandia to create a security detection system that can operate in uncontrolled exterior environments beyond the perimeter of a standard security system for a facility. User confidence in the detection system depends on the application of sensor fusion algorithms. Too many false or nuisance alarms can overwhelm operators, making a security system unusable. The fusion framework team has developed a modular software framework that handles the input, output, and flow of sensor events through the fusion application. Various algorithms can then be attached to the framework to handle sensor fusion processing. Algorithm developers can concentrate on development of sophisticated algorithms without the need to spend time writing code to handle data input and output. This architecture allows new fusion algorithms to be created, configured, and tested quickly. The design also supports reconfiguration of algorithm parameters in real time to adapt algorithms for various external events such as time of day or changing weather conditions. The fusion framework and a select set of algorithms have been deployed for testing. This paper describes the sensor fusion framework and presents some results of its performance in the field over the last several months.

1.0 INTRODUCTION

The sensor fusion framework is part of an early detection system that has been developed at Sandia National Laboratories. This security system is intended as an extension of existing systems that can provide early intrusion detection beyond a facility's perimeter or in areas with low visibility. It consists of (1) a network of sensors distributed over areas of interest, (2) infrastructure to relay detection information to the user, and (3) a user interface that displays maps and alarm information. The success of the detection system depends in part on the value of the information presented to the operator. Systems that receive too many false alarms place a burden on the operator; he must assess each alarm to determine if it represents a legitimate intrusion. A single legitimate event can be associated with multiple detection alarms, so the operator must also verify that each alarm is related to the same event and does not indicate a separate threat. In both cases, the flood of alarm messages can adversely affect the usability of the security system.

To maintain data integrity, this security system includes a flexible, sophisticated sensor fusion application. Sensor fusion consists of filtering and condensing sensor data so that the operator sees more meaningful information and fewer false alarms. The sensor fusion application has two main components. The first component is the fusion framework, which handles bookkeeping, data

input and output, and configuration. The second part of the application is a set of fusion algorithms that apply customized rules to filter and merge incoming data. Correctly configured fusion algorithms will keep false and nuisance alarms at acceptable levels, ensuring the system is usable and reliable. The fusion framework supports efficient development, integration, testing, and configuration of fusion algorithms so that site-specific solutions can be implemented with relative ease. This emphasis on flexibility yields an application that can be quickly configured to operate in a variety of locations, configurations, and weather conditions. Since the fusion framework architecture supports efficient algorithm development, many different ideas and approaches to sensor fusion can be tested simultaneously within the same application.

Section 2 describes the general structure of the detection system and the design of the fusion framework. It also describes some basic fusion algorithms that are flexible enough to handle most fusion needs. The deployment configuration for testing the system is described in Section 3, and preliminary performance results are presented in Section 4.

2.0 FUSION FRAMEWORK DESIGN

2.1 System Overview

The fusion application is just one component of the detection system. The detection system consists of a hierarchy of sensors and network nodes that pass messages from the deployed sensors to the operator of the security system. The individual sensors at the bottom of the hierarchy are distributed over an area or along a road. Sensor distribution patterns vary according to the particular application and terrain. The sensors pass information up the hierarchy to cluster nodes, which gather data from a subset of sensors or other cluster nodes. Members of the top level of cluster nodes send messages to the command node, which displays information and interacts with the operator. The sensor fusion application can operate both on cluster nodes and on the command node. This enables the application of local fusion algorithms at the cluster level and implementation of global fusion algorithms at the command node. This hierarchical structure can ease the amount of unnecessary network traffic at the command node, since data have been filtered and condensed by the time it is reaches the top level of the hierarchy.

2.2 Framework Design

The fusion framework represents the bulk of the sensor fusion application. It handles network input and output protocols, message sorting and bookkeeping, and algorithm configuration. The fusion algorithms are objects that analyze incoming messages and determine the value of the information, each according to its own rules. Each instance of an algorithm object represents a unique fusion rule, but all fusion algorithms have a standard interface for accessing the data and interacting with the framework. Fusion algorithms are described in more detail in a later section.

The fusion framework architecture can be considered a variant of the blackboard design pattern for software development [1]. In the blackboard pattern, several “agents” collaboratively solve a problem using a shared data repository, or “blackboard.” An “agent” entity is specialized to solve one part of the problem and uses the repository to add, modify, or remove data according to its own expertise or operation rules. Each agent monitors the blackboard for data applicable to its own task and the agents solve the problem iteratively—the contribution of one agent may provide enough information for another agent to contribute, and so on. This event-based architecture naturally supports collaboration between agents since information can be communicated between agents via the blackboard.

In the fusion framework, access to shared information and interaction between fusion algorithm objects is similar to the generic blackboard architecture. The "blackboard" contains messages from sensors or cluster nodes, and the "agents" solve many instances of the same problem: For each message, their task is to determine how important it is. The agents are fusion algorithms, each with specific rules for identification and fusion of relevant data. Each algorithm object watches the repository for new data, and if the conditions of its fusion rules are met, it creates a new fused message, places it in the repository, and assigns it a high-priority value. The abilities of fusion algorithms are more restricted than agents in the generic blackboard pattern. The framework acts as a supervisor, and allows algorithms to operate only in a specified order. Further, only one agent is allowed to erase data, and does so when it has expired or has been identified as low priority by other algorithms. Most of the problem solving involves the agents assigning and modifying priority values of the messages in the repository. If a message is assigned a priority value that exceeds user-defined thresholds, it is forwarded to the next node in the network.

The framework can be notified when a new algorithm must be added or an old one halted via network commands. This blackboard-style architecture provides a general interface for accessing sensor data. Since the algorithms are "plugged in" and supervised by the framework, it is easy to add or remove algorithms, even during runtime. This feature can be useful when the algorithm parameters need to be tuned in response to changing weather, temperature, or time of day. Fusion algorithms all have the same base class that provides the framework interface and several basic methods required by most algorithms. Since the framework itself handles the bookkeeping and data control issues, all the algorithm designer needs to do is implement the new fusion rule and identify the configurable components. In this way, the fusion framework supports quick development and integration by providing a flexible architecture that allows runtime configuration and addition of fusion algorithms.

2.3 Fusion Algorithm Implementation

To ensure that fusion algorithms are as general as possible, we allow the user to specify many algorithm parameters at startup or during runtime. For example, algorithms can be configured to only analyze data from specific sensors or sensor types. The user can also specify a score for certain sensors or sensor types, which affects the influence of the sensors. In this way, a noisy or error-prone sensor can be assigned less weight than a relatively reliable sensor. Sensors can even be assigned negative values. This is useful, for example, if a particular algorithm is designed to ignore any events caused by vehicles. It may assign negative values to magnetic sensors, so that messages from magnetic sensors are ignored by this algorithm. Finally, many algorithms depend on the presence of a number of messages within a time window. This time window can also be configured, so that the same algorithm can be applied to detection of vehicles and pedestrians by adjusting the window to reflect the speed of the object. These configurable parameters help ensure that the installation of a site-specific security system is relatively efficient.

Several algorithms for basic nuisance alarm filtering and data fusion are already implemented. These algorithms help ensure that the messages received at the command node are useful and informative. Since an algorithm must have raised the priority value of data for the message to be seen at the command node, each of these algorithms implicitly filter nuisance alarms merely by ignoring them.

2.3.1 Boosting Algorithms

Since the default priority of all incoming sensor messages is below the fusion threshold, the boosting algorithm is available to detect important sensor messages and immediately boost their priorities. This algorithm is commonly used to boost the priority of messages that are not detection alarms. For example, the system may need state of health messages and tamper messages to

pass straight through the fusion framework without delay. Also, there may be situations in which the message from a particular sensor is always relevant, and no filtration is necessary. In this case, a boosting algorithm can be configured to check for a certain type of event, or an event from a specific sensor. These algorithms typically operate first to minimize the delay of important messages.

2.3.2 Time-Gated Algorithms

Some sensors are notorious for sending a single false alarm at seemingly random times. Often the presence of an intruder is accompanied by more than one alarm, so we can ignore these single false alarms with time-gated algorithms that watch for a number of alarms within a specified time window. Time-gated algorithms can be configured to watch for messages from a single sensor or a group of sensors, and fuse them together if they occur within the specified time window. With appropriate fusion configurations, multiple sensor alarms related to a single event can be represented as a single, more informative message for the operator. Instead of several alarms from different sensors assailing the operator at once, the time-gated algorithm can produce a single message that says, for example, that five detections from the same location have been received in the last five seconds. This time-gated algorithm is versatile enough that it is the most often used algorithm. A set of cleverly configured time-gated fusion algorithms is generally sufficient to reduce system-wide nuisance alarms to acceptable levels.

2.3.3 Tracking Algorithms

Another kind of fusion algorithm that adds meaning to the raw sensor messages is the tracking algorithm. A tracking algorithm attempts to identify a pattern of sensor alarms that indicates an intruder is traveling in a certain direction or along a path. These algorithms can report approximate direction of travel or approximate rate of travel. They can also avoid sending fused data when an alarm pattern seems to be a meandering path or a set of random detection alarms. Tracking algorithms are best used when there are many sensors distributed in a grid pattern over a large area. The sensors need to have relatively small, uniform detection areas. If the detection area of the sensors is too large, or if they have large overlaps, pinpointing the location of a possible intruder becomes more difficult. For sets of sensors with overlapping detection areas or detection areas of varied size, an algorithm that learns alarm patterns may be more useful. The hardware and configurations used at the test site were not ideal for tracking algorithms, so only time-gated algorithms are used to evaluate the sensor fusion application's performance.

3.0 APPROACH

3.1 Test Layouts

The test deployment of our extended detection system is composed of two general sensor configurations. In two locations, sensors are distributed over a heavily wooded area, and the goal is to detect intruders moving through the area on foot. In other areas, sensors line small sections of a road, and the goal is to detect both vehicles and walkers traveling along the road. For a forked road the goal is also to determine which fork the vehicle or person takes. Each part of the sensor network can contain several different kinds of sensors including seismic, infrared, and magnetic (metal) sensors.

Configuration of the fusion algorithms was accomplished using test vehicles and personnel to determine the alarm patterns and detection areas of the sensors. For identification of walkers in the wooded area, the time-gated algorithm is used to fuse sensor alarms from groups of sensors that are close to each other. The size of the time window was tuned experimentally. Similarly, sensor groups were determined by location and time windows were determined experimentally.

For vehicle detection, the magnetic sensors are weighted heavily and the algorithms are configured such that without a magnetic sensor alarm, the data will not be fused. The detection requirements for the extended detection system state that foot traffic must have a probability of detection of at least 75% and vehicle traffic must have a probability of detection of at least 90%. The goal for the nuisance alarm rate is less than one nuisance alarm per hour per detection zone. Data from four detection zones are presented here. During installation, the probability of detection was tested and verified to be above the required values. The data presented in this paper show the alarm reduction that the sensor fusion application provides.

After system installation and configuration, log files track network activity and provide enough information for an estimate of the effect of sensor fusion with respect to nuisance alarms. The number of raw messages from sensors, the number of messages contributing to a fused message, and the number of fused messages sent to the command node can all be recorded. A sampling of several days of the fusion system's performance provides a good estimate for the average alarm rates of the system. Section 4 presents a summary of these findings.

4.0 RESULTS

4.1 Performance of Algorithms

The primary goal of the sensor fusion application is to reduce nuisance alarm rates and combine valid alarms that from the same event into a single message. In this way, the system avoids overwhelming operators or losing the operator's trust. Each fused alarm that the command node receives is composed of at least one raw message from the sensor. The raw messages that can be combined into a fused message are referred to as *legitimate raw data.* The amount of legitimate raw data that makes up a single fused message depends on the specific configurations of the fusion algorithms. Generally, an algorithm will continue to collect legitimate raw data after sending a fused message to the command node, but these additional messages are not recorded. Therefore, the legitimate raw data represent a lower bound on the number of messages related to a single event and the actual numbers may be unknown. In situations where the total number of raw messages from sensors is unknown, legitimate raw data also provide a lower bound for the total number of messages. The amount of raw sensor messages per day is at least the amount of legitimate raw data represented by fused messages at the command node.

The bar chart in Figure 1 depicts typical daily alarm activity at the command node. The heights of the black bars indicate the number of fused alarms, which are the alarms received at the command node; the gray bars indicate the amount of legitimate alarms represented by the fused alarms. The number of fused alarms generally remains under 100.

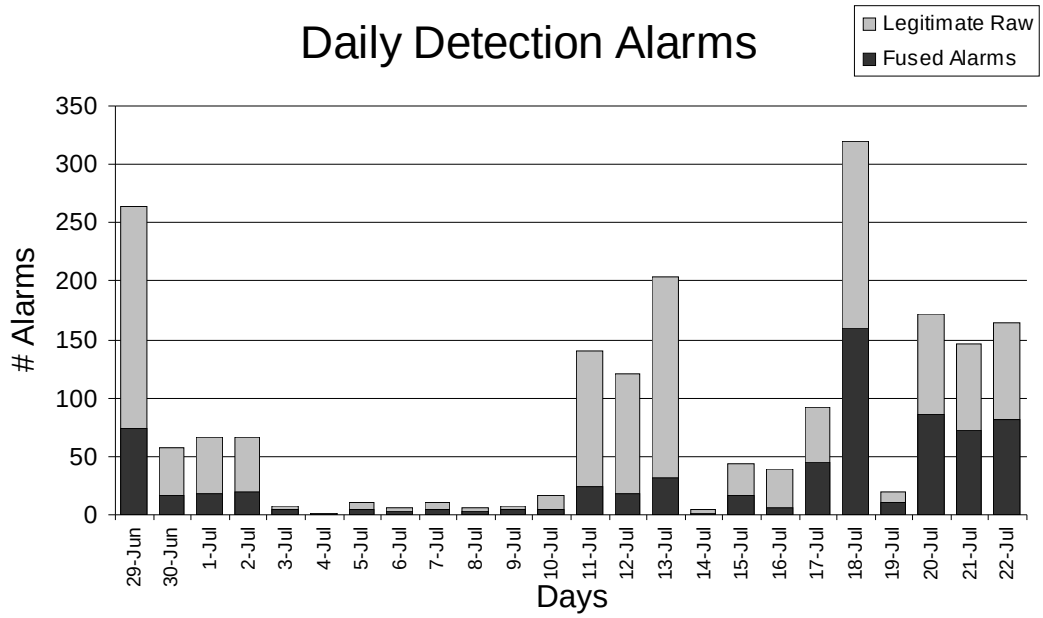

Figure 1 - Example of daily alarm activity across the entire test site.

Assessment data are not available, so it is impossible to identify whether fused alarms represent false alarms; but, daily alarm data seem correlated to the level of activity of personnel and vehicles. Figure 1 shows a marked decrease of activity during the week surrounding the 4th of July holiday, which corresponds to a decrease in general activity at the test site.

During a typical weekday, there are hundreds of nuisance and false alarms. The average total number of raw alarms from the network varies greatly from day to day. These variations could be due to a number of environmental factors such as wind, lightning, and changing temperature. Encouragingly, the number of fused alarms remains low even when there are excessive nuisance alarms. The graphs in Figures 2 and 3 show typical daily alarm rates from a single zone for several weeks in October and February, respectively. In these graphs, the number of raw alarm messages is often so high that in comparison, the fused alarm rate appears to be almost zero.

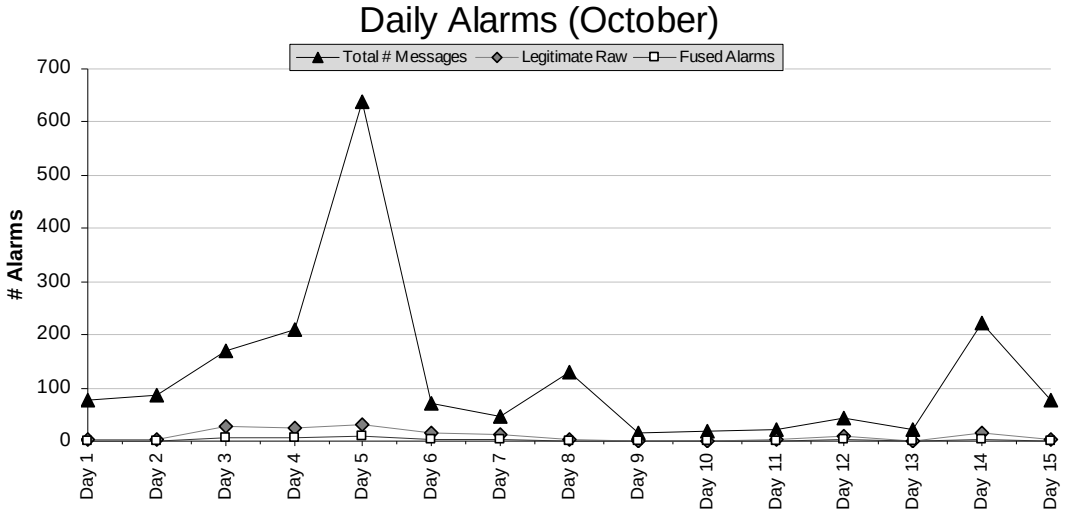

Figure 2 - Example of daily alarm activity for fifteen days in October

Notice that although the total number of messages can be very large, the number of fused messages sent to the command node remains relatively low and stable. In general, the number of fused messages is much less than 24 alarms/day. This means that even if most of the fused messages are actually false alarms, the nuisance alarm rate is usually less than one alarm per hour per zone. Only the Day 2 data point in Figure 3 exceeds 24 fused alarms in a single day. Since it is excessively conservative to assume that all of the fused alarms represent false alarms, it is safe to assume that the average nuisance alarm rate is much less than one alarm per hour.

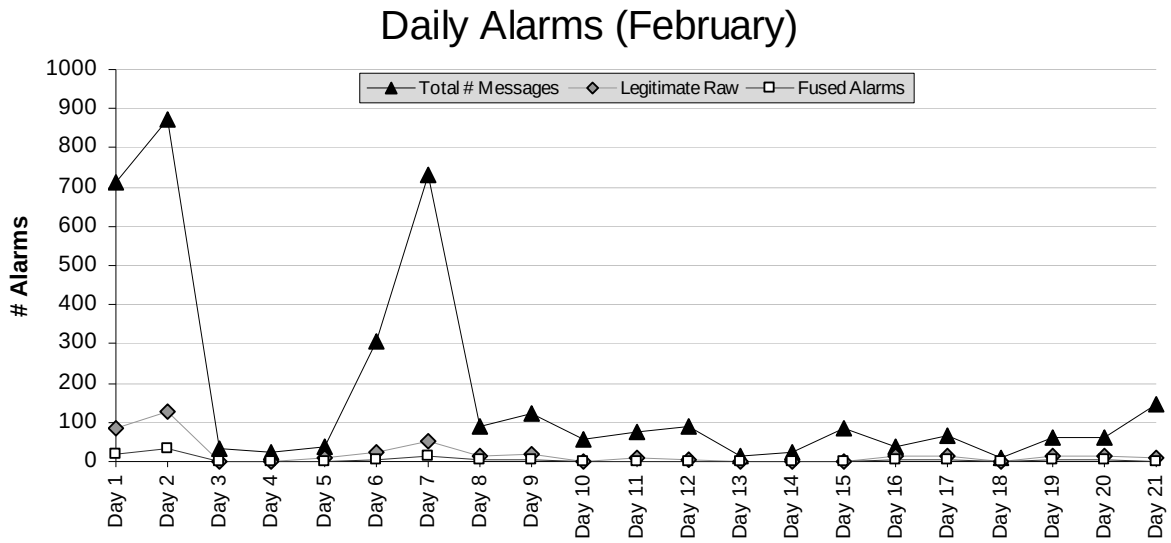

Figure 3 - Example of daily alarm activity for three weeks in February

The values in Table 1 present the average daily alarm rates from 81 sample days distributed over the entire deployment time. Notice that in this table, although the number of raw messages from the sensor network varies significantly, the average daily number of fused alarms remains fairly low. The standard deviation indicates that there is a great deal of variation in the daily level of fused alarms, but this is to be expected given fluctuation in activity levels on weekends and during weekdays. The average number of fused alarms per day is well below the nuisance alarm requirements, and it is safe to assume that not all alarms are false. Further, the number of fused alarms received at the command node on average represents only about 3.4 % of the total number of messages, and less than 26% of the number of legitimate messages. From these data, it is clear that the compression of valid detection information into fused alarm messages is a valuable feature. It is also clear that the nuisance alarms are being successfully filtered: our sensor fusion application, equipped with basic time-gated fusion algorithms, is able to reduce the message load at the command node by an average of approximately 96.5%.

“Total Raw” indicates the average number of alarms generated by the sensor network. The “# Legit” is a lower bound on the number of alarms that are combined into a fused alarm. The “# Fused” is the number of fused alarms received at the command node, and the “# Nuisance” column indicates the daily number of alarms that are not part of a fused alarm. The “% Total” column is the average percentage of total raw alarms that the fused alarms represent. Similarly, the “% Legit” column is the percentage of legitimate raw alarms represented by fused alarms.

Table 1 - Average daily alarm rates and standard deviations.

	Total Raw	# Legit	# Fused	# Nuisance	% Total	% Legit
Daily Average	349	55.5	14.6	294	3.42 %	26.1 %
Std. Deviation	683	114	28.9	604	3.19	2.73

5.0 DISCUSSION

5.1 Algorithm and Framework Performance

The goal of the fusion system is to reduce the nuisance alarm rate to less than one false alarm per hour, or less than 24 false alarms per day. Even assuming that every alarm on a given day is false, the average alarm rate produced by the system is well within the nuisance alarm goals. The sensor fusion application not only reduces false alarms, but also reduces the number of alarms an operator receives that are related to the same event. This approach ensures that the extended detection system will be usable and reliable. Further, the fusion framework is designed with an emphasis on flexibility and efficient algorithm development. This secondary goal ensures that future algorithms will be easy to implement, and site-specific installation of the system will be as efficient as possible.

REFERENCES

[1] Carver, N. and V. Lesser, *The Evolution of Blackboard Control Architectures*. Technical Report. UMI Order Number: UM-CS-1992-071, University of Massachusetts, 1992.