

Determining Object Safety using a Multiagent, Collaborative System *

Brian Quanz
Information and Telecommunication
Technology Center
Univ. of Kansas, Lawrence, KS
bquanz@ittc.ku.edu

Costas Tsatsoulis
Dept. of Computer Science and Engineering
University of North Texas
Denton, TX 76203
tsatsoul@unt.edu

Abstract

We consider the problem of Object Safety: how objects endowed with processing, communication, and sensing capabilities can determine their safety. We assign an agent to each object capable of looking out for its own self interests, while concurrently collaborating with its neighbors and learning/reinforcing its beliefs from them. After considering related work, we propose a general framework consisting of agents with case-bases of threat detection systems and a mechanism for sharing and confirming beliefs with other agents. While our approach is designed to be applicable to Object Safety domains in general, we particularly consider its application to transport chain security. We further propose a testing framework that uses real sensor data and Object Safety scenario simulations to evaluate our approach.

1. Introduction

As sensing and computing technologies advance, it becomes feasible to incorporate sensors and processors into physical objects themselves. Potential applications then arise in what we term "Object Safety" domains, domains in which objects must determine their own safety. Potential applications include transport-chain security, hazardous chemical storage and handling, and valuable goods protection. What specifically constitutes safety depends on the domain and the objects themselves. For example, an object in a valuable goods store may be safe so long as it is not dropped, mishandled, or carried from the display area. Transport chain security involves ensuring packages are safely transported without being stolen, tampered with, or otherwise mishandled. Furthermore, what constitutes safety may be different for different objects.

*This work was supported in part by the Office of Naval Research under award number N00014-07-1-1042

For objects endowed with sensors and processors, determining Object Safety also represents a task of fully-distributed security, since even the security of a single object is important. This naturally lends itself to the concept of objects as agents, essentially allowing objects to be agents in their own safety, in cases where the safety of the objects is a primary concern. The challenge lies in creating the agents' intelligence through algorithms and software systems. To address this challenge, we propose a general multiagent framework for determining Object Safety. This system is composed of a number of components which interact with each other. For handling situated knowledge and incorporating past experience a *case-based reasoning system* component is included. To allow automatic and adaptive learning of what constitutes a threat for a given situation, a *Threat Detection System* component is included. To allow agents to collaborate and learn from each other, a *belief sharing* component is included. Additionally, we develop a testing framework to evaluate our approach which involves testing the overall design and its components by simulating Object Safety scenarios using collected real sensor data.

2. Related Work

Recently there has been some directly related work dealing with objects monitoring their own safety, e.g. [22, 21, 7], in which objects are endowed with sensors and micro-controllers and use rules that test for threshold conditions, with the particular application of enabling chemical barrels to set off alarms when they detect themselves to be in hazardous situations. There are several limitations to these current rule-based system approaches. One is the inability to detect complex events through only combining threshold rules [25, 26]. Current systems would also have problems adapting to changing and new threats, e.g. unforeseen threats which are not covered under the existing rule base. Also in many object safety scenarios knowledge is situated, and changes based on the environment, for instance a rail package may progress through several different situations

during which what constitutes a threat changes. In existing work the knowledge is static. Furthermore, since agents will often be in the same situation (e.g. in the same truck bed together) some form of collaboration could be desirable.

Similar to the concept of an object determining its safety, is the idea of context awareness [6, 2], particularly as applied to sensor networks. Context awareness can be described as the ability of a computing device to sense, react, and adapt to its changing environment. For ubiquitous computing this means providing useful information and services to an entity/user using any information relative to that entity/user's situation. Such work includes augmenting everyday objects such as a lamp with sensors to allow them to react to a user's context [14], and a Context Aware SensorNet [11] in which sensor nodes keep track of context and adjust their behavior accordingly, e.g. sleeping when possible.

The problem of objects determining their safety can also be thought of as anomaly or event detection in sensor networks, if we consider a threat to an object's safety within a group of similar objects as the event or the anomaly. A query-based interface to sensors (e.g. [17]) is typically used, where events are detected through queries, often as set thresholds for different sensor readings, or as more complex spatial and temporal patterns using, e.g., contour map matching [25]. There also exists work on anomaly detection in sensor networks using machine learning, e.g. using a one-class support vector machine [18, 5] and using a combination of context awareness and event detection with time series sensor data transformed into strings [26]. Another work focuses on reducing false positives by using group-based event detection [23], where each sensor node has the ability to detect some event, but the accuracy of detection may not be high, so a confidence in the detection is constructed using multiple nodes.

This same idea has recently been the focus of a group of research in the area of intrusion detection in computer networks, using end-host computers with weak detection systems to detect worm attacks, combining local detections [3, 4, 20]. The ideas of collaborating feelings of safety can be applied to groups of objects with communication abilities for influencing their feelings of safety, and is thus related to the goal of our work.

Also a part of intrusion detection is work on artificial immune systems [10, 8, 13]. Artificial Immune Systems (AIS) model the human immune system to achieve engineering goals that are similar to its goals, e.g. anomaly detection. Similarly, object safety is related to another field related to biological systems, that of *self-healing systems*, in which computer systems detect faults in their components and attempt to repair them. Object safety work could also be thought of as work on the detection element of self-healing systems [9].

For the related work that does not specifically address the problem of object safety, it is not their own safety the sensing components are concerned with. In ubiquitous computing, objects are trying to improve conditions for the user. In sensor networks, the sensor nodes are concerned with detecting events of interest to some specific entity. For intrusion detection, the agents care about the security of the network over their own security. In these cases individual sensors are typically treated as expendable; they are not individually necessary. However, in an Object Safety framework an object must be able to look out for itself as well.

3. An Object Safety Framework

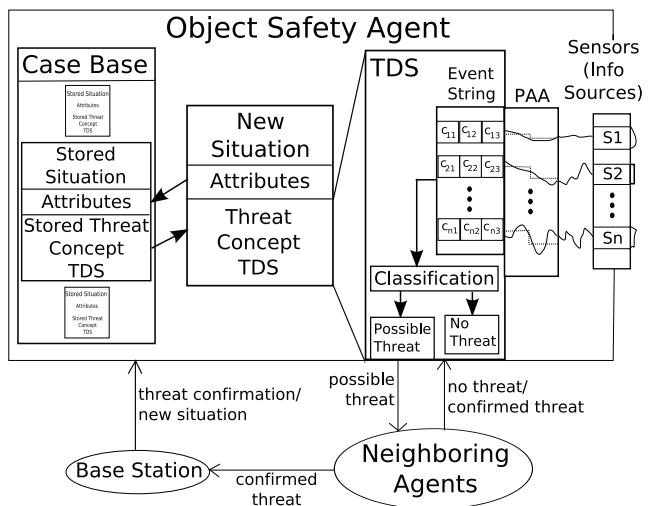


Figure 1. Overview of Approach

From considering limitations of previous work and the goals of an object safety agent, we can identify some key components of an ideal agent. An object safety agent should be able to:

- Adapt to new and changing threats
- Detect previously unknown threats
- Utilize prior learned or provided knowledge
- Change its perception of threats based on its current situation
- Utilize communication with other agents for sharing and enforcing beliefs in threats
- Be easily applied to different object safety scenarios
- Handle inconsistent information across objects (such as from sensor bias)

To address these capabilities, we propose a system consisting of a case-base of threat concepts, which would essentially combine Case-Based Reasoning (CBR) [24, 16] and a Threat Detection System (TDS). Figure 1 depicts the general framework. Each object is represented by an Object

Safety Agent. A *base station* is a domain-dependent entity which keeps track of objects' situations and supplies an agent with situation descriptions, including its initial situation if necessary. For instance, in transport chain security, the base station could correspond to a system responsible for holding trip information for packages, or it could be as simple as a fixed system at each portion of the journey that broadcasts the new situation to objects arriving at its location. When presented with a new situation via a situation description, the agent finds its closest matching stored situation, and loads the associated Threat Detection System (TDS). The TDS processes the output obtained from the agent's sensors (denoted S_1, S_2, \dots, S_n in Figure 1), which continuously receive (sense) information from the environment. The sensor outputs are translated into event strings (in Figure 1 c_{ij} corresponds to the j^{th} character of the event string for sensor i), which are classified by the TDS as *threat* events or *normal* events. When the agent detects a threat, it seeks confirmation from its neighbors. Upon confirmation, a base station is notified, and either confirms the threat and takes further action, or sends the agent a new situation description.

When an object is exposed to a new situation, it should not have to completely re-learn what constitutes a threat if it has been in a similar situation before. Thus, we store cases based on the attributes of a situation and use an existing case and solution (threat concept) when a new situation arises.

For detecting threats, an Artificial Immune System (AIS) framework contains many of the components of an ideal agent, so it makes sense to incorporate AIS components. An alternative approach to threat detection is a One-Class Support Vector Machine, which has potential benefits of smaller resource usage and the ability to handle high-dimensional data. The idea here is to estimate the support of the distribution of normal events, essentially defining a boundary around what is normal.

Our approach is described in detail in the following sections.

3.1. Case-Based Reasoning Layer

With this layer, an object is informed of what situation it is currently in. This situation is described by attributes and matched to the closest stored situation in the case-base using the number of matching attributes. The threat concept of that case is then loaded into the new case if a tunable matching threshold is surpassed, or used directly if an exact matching threshold criterion is met.

What constitutes a threat in one situation may constitute normal behavior in another so it is necessary to inform an object of situation changes. One option, *on-demand situation switching*, is depicted in Figure 1 as feedback from the base station to the object safety agent; if the central loca-

tion knows that there has recently been a situation switch an agent is uninformed of, instead of treating a distress signal as a threat, it updates the agent with its new situation, allowing smooth operation.

Situations are represented by attributes that describe the environment of the situation and expected normal behavior for an object in the situation, for instance, "light intensity should be less than or equal to 0". Attributes of a situation are represented by basic conditions and allow operations such as comparison and logical operations on elements represented by strings. This allows situations to be logically described with whatever symbolical terms are applicable for a particular domain. Figure 2 shows an example situation for a rail safety object, demonstrating the format of attributes. Each parentheses block represents a single condition (attribute) for the situation.

Situation: Transit Station A to B
Attributes: (in-transit) (< light-intensity 2)
(< speed 100) (expected-speed 75) (stop-frequency 1)
(expected-temperature 72) (< temperature 102)
(> temperature -10) (county Douglas)

Figure 2. Example Situation Description

A matching condition consists of an exact match of elements. Partial matching of conditions is possible for conditions containing numerical values, in this case percent error is used to estimate the match value. If a condition contains multiple numerical values the average percent error is used. The *match value* for two conditions is given as:

$$MatchValue = \begin{cases} 1 & \text{if exact match} \\ 0 & \text{if mismatch} \\ 1 - \frac{|v_1 - v_2|}{|v_1| + |v_2|} & \text{if numerical } v_1 \text{ and } v_2 \end{cases}$$

The *situation matching score* S between two situations is then calculated as $S = \frac{S}{\min(|\{conditions_{s_1}\}|, |\{conditions_{s_2}\}|)}$ where $MatchValue_c$ is the match value for condition c , and $\{conditions\}$ represents the set of shared conditions. A higher situation score signifies a closer match. The *situation matching threshold criteria* between two situations, Situation 1 and Situation 2, is $\frac{S}{\min(|\{conditions_{s_1}\}|, |\{conditions_{s_2}\}|)} > \tau$ and the *situation exact matching threshold criteria* between Situation 1 and Situation 2 is $\frac{S}{\max(|\{conditions_{s_1}\}|, |\{conditions_{s_2}\}|)} > \tau_{exact}$ where $conditions_{s_1}$ is the set of conditions for Situation 1 and $conditions_{s_2}$ is the set of conditions for Situation 2. For equal matching cases preference is given to situations having the closest number of conditions to that of the new situation, after that the selection is arbitrarily chosen.

A key benefit of this approach to handling different situations is that the human description of a certain situation is separated from the actual concept of the threats with respect

to the sensor values, since it may be the case that human ideas of object safety are not easily translated into actual temporal sensor patterns. The human description is used to find situations with similar expected normal operations, and determining what constitutes a threat is left to the threat detection layer.

Another important benefit of using cases is that it allows the sharing of beliefs with regard to events. For objects in the same or similar situations it is possible for them to share opinions and experience within our TDS framework, as described in Section 3.4, so that objects can learn from other more experienced or better informed objects, or collaboratively enforce their beliefs in a threat.

3.2. Threat Detection Layer

Coming up with rules for threats can be particularly difficult in Object Safety domains. Experts don't usually exist for objects. When generating a rule, designers may not be able to accurately predict all threats and scenarios. Furthermore, there is not always an easy translation from the logical human perspective to the object sensor (information source) perspective. Additionally, the process of developing rules would need to be repeated for objects with different sensors or in different situations. For these reasons it makes sense to incorporate automatic learning of threat concepts, using what is generally available: normal (safe) behavior for the objects.

One way of doing this is through negative selection, a key component of Artificial Immune Systems (AIS). We simply allow the system to generate a coverage of abnormal events around what is defined as normal behavior for an object. This concept can be visualized with Figure 3, a two-dimensional example is given in which t_{1a} , t_{1b} and t_{2a} , t_{2b} correspond to thresholds of normal behavior in the event space - any values outside of those thresholds were never encountered as a part of normal behavior. Within the region of event space containing normal events, a memory-based approach is used; events are generated and those that do not match normal behavior are kept, to create a coverage around safe events. The (-) symbols correspond to possible threats and the circles around them correspond to points in the space that are within the threshold matching distance. The two regions labeled *safe* represent the portions of the event space that constitute no threat.

The overall AIS framework is depicted in Figure 4. The first part is the training stage (tolerization) in which the set of candidate detector strings are randomly generated and eliminated if matched to any self string. The detector consists of the feature string (described in the next section) and a radius, and a match occurs if the Euclidean distance to the matching string is within the radius. This can occur in an online setting that happens continuously, during which

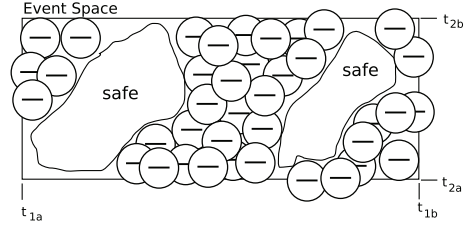


Figure 3. Negative Selection Concept

the detectors are immature for a certain amount of time (tolerization period). In parallel to the running system, as resources allow, an existing detector is randomly picked for merging with nearest neighbor. A copy is made of the new detector which undergoes tolerization and replaces the old detectors if it survives, and if the detector set is no longer at maximum capacity, new detectors are generated. If a false-positive occurs, if the location of the matching self string is within a threshold of the detector's radius, the detector is deleted, otherwise it is shifted so that the self string is just outside the radius, to be later merged if possible. This approach combines aspects of real-valued negative selection [12], with the online components of an AIS framework [10].

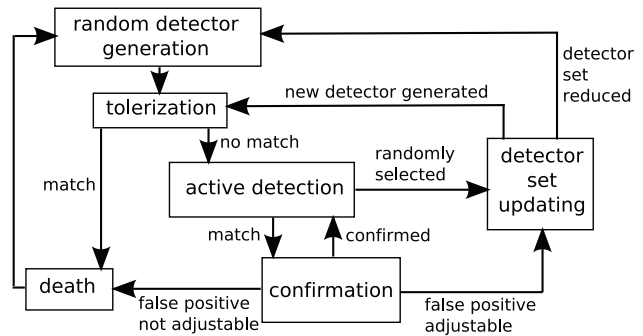


Figure 4. Overview of AIS Framework

An alternative approach to threat detection we also consider is an online one-class Support Vector Machine (SVM) [5]. One-class SVM works by mapping the data (event vectors) into a high-dimensional *kernel space* (through the use of a *kernel function* which calculates the inner product of two input vectors in the kernel space), and finding a hyperplane that separates the data from the origin with maximal margin and minimal error by solving an optimization problem [19]. We include both methods in our framework to compare their ability to detect threats and to illustrate the benefit of our approach in allowing different objects to use different threat detection systems for different situations. For instance a method may be particularly suited to a given situation, or an object may be resource-limited and only able to use certain TDSs.

3.3. String Representation of Streaming Sensor Data

The basis of the threat detection layer is a representation of feature strings from the sensor data, essentially mapping the data to strings to allow matching between string representations, or classification. In order to allow threats corresponding to a specific pattern over time to be captured, and to avoid a high-volume event space that would suffer from the curse of dimensionality, it is desirable to create an approximation of the data capable of retaining its important features. We propose to accomplish this by using an existing simple method of approximation for string generation, Piecewise Aggregate Approximation (PAA) which was shown to lower-bound the Euclidean distance between time series curves and to perform competitively with more complex methods [15]. In addition, PAA is simple, fast, and can still be applied in an online setting. Figure 5 shows an example of PAA for a single sensor, S_c represents the current event string (of max length 3), t_c the current time, and 0.96 the newly added character.

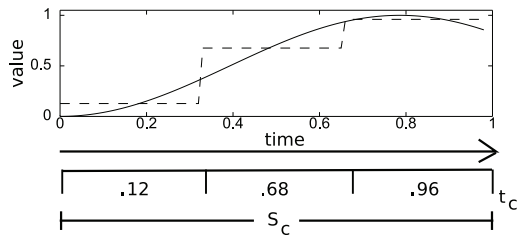


Figure 5. PAA Example

Feature strings are obtained by segmenting the original time series data from a sensor into equal-sized windows, and taking the average value in each window as the character value. With a fixed window size, as data is received, when it fills up the next window, the resulting character can be added to the end of the current string for that sensor (after a max string length is reached, the first character of the string is dropped). In this way matching to the current event string can occur online and different detector string lengths could be used. The complete event string is made from concatenating each sensor string.

3.4. Sharing Beliefs

Sharing beliefs about the nature of events among agents in the same situation can be useful for learning from neighbors in the same situation for an agent that is uncertain of its situation (no good match in the case base) or that contains an under-developed or non-existent threat concept. Also, different agents may have developed different representations of the threat concept for a situation, so sharing their

beliefs could help to overcome false positives, particularly for weak detectors, which may be desirable in resource-constrained object safety domains.

In order to share beliefs we propose a modified version of the sequential hypothesis testing as described in [3]. This is basically a voting process; when an agent detects a threat, it propagates the suspected threat event string to its neighbors who test the string with their own TDSs and continue to propagate it, and confirm the threat when a specified ratio of positive detections to total tests is met. The key difference here is passing the threat event string as opposed to an indication of a threat, since an object may be in the same situation as the other objects but experience a threat locally (e.g., in the rail safety domain, packages may be together in a container but only one targeted for theft). In addition if an event is confirmed as a threat, the last neighbor can inform the base station (in case the target agent can no longer communicate). Besides potentially reducing false alarms, this approach allows untrained/underdeveloped agents to learn from the experience of their peers by using them to confirm possible threats and adjust their own threat concepts as necessary until a training period is completed (e.g. tolerization in AIS).

For agents having the same event string representation, sharing some set of common sensors, the process of neighbors checking an event string can be applied by directly testing the event string with their own TDS. Handling different sensor representations and incorporating a degree of belief associated with each decision about an event string are areas of future work.

4. Testing and Evaluation

Some specific aspects of our approach to test include: the ability of the overall system to detect threats (i.e. measuring such criteria as false positives, true positives, etc.), the performance for different string representations, the performance for different threat detection systems (e.g., AIS vs. online one-class SVM), the benefit of the belief sharing component, the ability of untrained agents to learn from other agents, and the usefulness of the CBR system for varying situations.

In order to evaluate these ideas and make these comparisons, we have developed a testing framework. To test the main application, sensor data is needed for Object Safety situations. We propose to generate a data set of Object Safety sensor data, which can then be used to evaluate the components of our approach through simulation. We will generate the data using real sensors, modeled after three Object Safety scenarios. One scenario is designed to be similar to our desired application, that of rail transport security. To gather data, we consider a connected group of sensors (in the form of Sun SPOTS [1]) as an "object" and group ob-

jects in the same situation into a box. Normal sensor data is collected for different situations, e.g. while the objects in the box travel around a fixed course and have predefined stops. Then threat data is generated for different threats along the course, such as opening the box while it is meant to be in transit, capsizing of the box, and moving objects away from a loading station. The normal behavior runs can be used for initial training, then runs with threats can be simulated to test response. Additionally, since it is done through simulation and stored sensor data, different agents can have varying amounts of different training. The second scenario is meant to be similar to the first, so that agents trained using the normal behavior of the first scenario can perform in the second scenario. This can be used to test the effectiveness of the CBR system. A third scenario will be used to test the system under a different Object Safety domain, that of object in a valuable goods store.

The sensors we use to collect data are built into Sun SPOTs which represent a feasible platform for Object Safety agents given current technology. The wireless Sun SPOTs contain light, temperature, and 3-axis accelerometer sensors, a processor, and radio communication abilities.

5. Conclusion

Granting objects the ability to determine their own safety could prove beneficial to many application domains, allowing, for example, a fully distributed approach to security. For physical objects, this level of distribution is becoming viable as technology improves. More generally, the idea of Object Safety could be applicable to any problem in which an entity (not necessarily a physical object) with online sources of information (sensors), some form of prior knowledge, and communication capabilities, has a need to determine if it is "safe" or "threatened." To address the Object Safety problem, we have proposed a framework that considers, incorporates, and expands upon ideas from a variety of similar research areas. Our framework is meant to be versatile enough to be applied to numerous Object Safety domains. Our framework consists of three main components which interact and complement each other. Through the case-based reasoning component, capabilities for handling situated knowledge are incorporated. The threat-detection component, stored within the cases of the case-based reasoning system, allows adaptive learning of safety concepts. Updating and learning for the threat detection system is additionally connected to the belief sharing component which allows a level of confirmation and the ability to learn and enforce beliefs from other agents to improve detection of threats and minimize false alarms. Additionally, we have developed a testing framework to test our approach to Object Safety that involves the construction of an Object Safety data set.

References

- [1] Project sun spot: Sun small programmable object technologies, 2008.
- [2] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, 2007.
- [3] S. G. Cheetancheri, J. M. Agosta, D. H. Dash, K. N. Levitt, J. Rowe, and E. M. Schooler. A distributed host-based worm detection system. In *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*, New York, NY, USA, 2006. ACM.
- [4] D. H. Dash, B. Kveton, J. M. Agosta, E. M. Schooler, J. Chandrashekar, A. Bachrach, and A. Newman. When gossip is good: Distributed probabilistic inference for detection of slow network intrusions. In *Proceedings of the 21st National Conference on Artificial Intelligence*, 2006.
- [5] M. Davy, F. Desobry, A. Gretton, and C. Doncarli. An online support vector machine for abnormal events detection. *Signal Processing*, 86(8):2009–2025, 2006.
- [6] A. Dey and G. Abowd. Towards a better understanding of context and context-awareness. Technical Report Technical Report GIT-GVU-99-22, College of Computing, Georgia Institute of Technology, 1999.
- [7] D. Dobre and E. Bajic. Smart object design for active security management of hazardous products. In *1st International Workshop on Design and Integration Principles for Smart Objects*, Innsbruck, Austria, 2007.
- [8] S. M. Garrett. How do we evaluate artificial immune systems? *Evolutionary Computation*, 13(2):145–177, 2005.
- [9] D. Ghosh, R. Sharman, H. R. Rao, and S. Upadhyaya. Self-healing systems - survey and synthesis. *Decision Support Systems*, 42(4):2164–2185, 2007.
- [10] S. A. Hofmeyr and S. Forrest. Architecture for an artificial immune system. *Evolutionary Computation*, 8(4):443–473, 2000.
- [11] Q. Huai Feng and Z. Kingshe. Context aware sensor net. In *3rd international workshop on Middleware for pervasive and ad-hoc computing*, Grenoble, France, 2005. ACM Press.
- [12] Z. Ji and D. Dasgupta. Real-valued negative selection algorithm with variable-sized detectors. In *Genetic and Evolutionary Computation Conference (GECCO)*, volume 3102, pages 287–298. Springer, 2004.
- [13] Z. Ji and D. Dasgupta. Revisiting negative selection algorithms. *Evolutionary Computation*, 15(2):223–251, 2007.
- [14] F. Kawsar, K. Fujinami, and T. Nakajima. Augmenting everyday life with sentient artefacts. In *2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, Grenoble, France, 2005. ACM.
- [15] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2001.
- [16] J. Kolodner. *Case-based reasoning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1993.
- [17] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 491–502, 2003.
- [18] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. C. Bezdek. Quarter sphere based distributed anomaly detection in wireless sensor networks. In *IEEE International Conference on Communications*, pages 3864 – 3869, Glasgow, Scotland, 2007.
- [19] B. Scholkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.
- [20] T. Singliar and D. Dash. Cod: Online temporal clustering for outbreak detection. In *Proceedings of the Twenty-second AAAI Conference on Artificial Intelligence*, pages 633–638, Vancouver, British Columbia, Canada, 2007. AAAI Press.
- [21] M. Strohbach, H.-W. Gellersen, G. Kortuem, and C. Kray. Cooperative artefacts: Assessing real world situations with embedded technology. In *Proceedings of Sixth International Conference on Ubiquitous Computing*, 2004.
- [22] M. Strohbach, G. Kortuem, and H.-W. Gellersen. Cooperative artefacts - a framework for embedding knowledge in real world objects. In *'Workshop on Smart Object Systems, co-located with Ubicomp 2005'*, 2005.
- [23] A. Tavakoli, J. Zhang, and S. H. Son. Group-based event detection in undersea sensor networks. In *Proceedings of the 2nd International Workshop for Networked Sensing Systems*, Los Angeles, CA, 2005.
- [24] C. Tsatsoulis and A. Williams. Case-based reasoning. In C. Leondes, editor, *Knowledge-Based Systems - Techniques and Applications (Volume 3: Computer Techniques)*, pages 807–837. Academic Press, 2000.
- [25] X. Wenwei, L. Qiong, C. Lei, and L. Yunhao. Contour map matching for event detection in sensor networks. In *2006 ACM SIGMOD international conference on Management of data*, Chicago, IL, USA, 2006. ACM.
- [26] M. Zoumboulakis and G. Roussos. Escalation: Complex event detection in wireless sensor networks. In *Proceedings of 2nd European Conference on Smart Sensing and Context*, Lake District, UK, 2007.