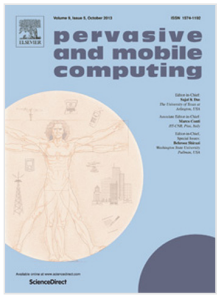
Accepted Manuscript



Detecting abnormal events on binary sensors in smart home environments

Juan Ye, Graeme Stevenson, Simon Dobson

PII: S1574-1192(16)30080-3
DOI: http://dx.doi.org/10.1016/j.pmcj.2016.06.012
Reference: PMCJ 718

To appear in: *Pervasive and Mobile Computing*

Received date: 17 November 2015
Revised date: 5 April 2016
Accepted date: 21 June 2016

Please cite this article as: J. Ye, G. Stevenson, S. Dobson, Detecting abnormal events on binary sensors in smart home environments, *Pervasive and Mobile Computing* (2016), http://dx.doi.org/10.1016/j.pmcj.2016.06.012

Detecting Abnormal Events on Binary Sensors in Smart Home Environments

Juan Ye, Graeme Stevenson, and Simon Dobson

School of Computer Science, University of St Andrews, UK
E-mail: juan.ye@st-andrews.ac.uk

Abstract

With a rising ageing population, smart home technologies have been demonstrated as a promising paradigm to enable technology-driven healthcare delivery. Smart home technologies, composed of advanced sensing, computing, and communication technologies, offer an unprecedented opportunity to keep track of behaviours and activities of the elderly and provide context-aware services that enable the elderly to remain active and independent in their own homes. However, experiments in developed prototypes demonstrate that abnormal sensor events hamper the correct identification of critical (and potentially life-threatening) situations, and that existing learning, estimation, and time-based approaches to situation recognition are inaccurate and inflexible when applied to multiple people sharing a living space. We propose a novel technique, called CLEAN, that integrates the semantics of sensor readings with statistical outlier detection. We evaluate the technique against four real-world datasets across different environments including the datasets with multiple residents. The results have shown that CLEAN can successfully detect sensor anomaly and improve activity recognition accuracies.

Keywords: Ontologies, Smart home, Fault detection, Semantics, Domain knowledge

1. Introduction

As our population ages there will be increasing need for developing cost-effective solutions to reduce burdens on over-stretched healthcare resources. Smart home technologies have been widely recognised as a promising solution to support independent assistive living [4]. A smart home is a residential home setting where sensors are embedded and attached to all sorts of everyday objects such as beds, mugs, appliances, and even our bodies. These sensors perceive the state of the physical environment through the interactions people have with these instrumented objects. By reasoning on these captured states, an intelligent system can infer what tasks residents are carrying out, and therefore automatically provide services to help achieve these goals without the need of direct or explicit guidance from the residents [7]. Beyond this, smart home technologies have a potential to automatically identify early symptoms of illness and help the elderly live independently in their own homes [4].

Research in smart home technologies has been popular in the last decade, and many prototypes have been developed, including the Aware Home [1], MavHome [38], Gator Tech Smart Home [14], and iDorm [8]. However, the possibility of widespread deployment of such systems remains unclear. One obstacle among many (such as privacy and ethical issues) is the high likelihood of sensor anomalies. Researchers from the University of Virginia conclude their experience of deploying sensors with the following observation: "Homes [...] can be hazardous for sensors, particularly when hundreds of sensors are deployed over long time durations" [16]. During their years of experimenting, they report an average of one sensor failure per day. The high rate of failure can result in incorrect activity detection (e.g., unable to detect life-threatening situations), leading to the failure of delivering appropriate services in time.

The high rate of sensor failure results from various types of technical limitation in pervasive sensing technologies [25]. The sensors are subject to hardware failure, disconnection from the network, vulnerability

to environmental interference, and limited battery life. For example, sensors are more likely to produce erratic readings when the battery has reached the end of its life. Or the presence of metallic structures or electronic equipment could lead to interference with the location detection hardware that use ultra-wideband technologies [5].

Another major factor contributing to sensor anomaly in smart home environments is the presence of *humans*; that is, users (including both foreground users who are target subjects and background users who are active in the same environment like visitors, children, or pets) might dislodge or move sensors accidentally [16, 19]. This often leads to *sensor anomaly*, where sensors do not fail, but continue to report values that are technically reasonable (e.g., the reported values are still within a reasonable range) but are unexpected or contradict the events occurring in the world.

Addressing sensor anomalies is more challenging than detecting sensor failure or sensor readings that are out of range. Hnat et al. [16] discuss solutions towards addressing the sensor failure. For example, they set a time interval that is reasonably long for any two consecutive data points. If no data is reported from a sensor within the interval since the last report, then this sensor can be considered broken. In contrast, sensor anomalies can be much more subtle and thus more difficult to detect. Simply setting fixed or variable time intervals or checking the valid range of readings cannot solve the problem, and often we need to validate collected values against either a data model that represents expected sensor values or values collected from neighbouring sensors via a correlation model.

A large number of fault detection techniques have been proposed in sensor network, which mainly deal with homogeneous, periodic, and real-numbered readings. These techniques often target individual sensors by tracking their historic values. However, the most common classes of sensors deployed in a smart home are infrared sensors, RFIDs, motion sensors, accelerometers, camera and microphones [33]. These sensors produce heterogeneous, event-triggered, and binary or featured readings. First of all, compared to the homogeneous, real-numbered sensor readings, it is not straightforward to define an explicit numeric relationship between readings of the smart home sensors, such as correlation between infrared sensor readings and acceleration data. Secondly, it is difficult to define a range limit over RFID readings or a seasonal pattern of sound.

To address these challenges, we propose *CLEAN* — a knowledge-driven technique to detect anomalies in event-driven binary sensors [37]. We focus on two types of anomaly — *random* events that occur sporadically due to short or intermittent environmental interference; and *systematic* events where sensors consistently behave abnormally due to the sensor recalibration or dislodgement. We claim the following novelty and contributions:

- It is *knowledge-driven* in that it does not rely on any training data or annotated data and thus can be used from system initialisation and is not affected by changes in the patterns or routines of users' activities.
- It novelly combines knowledge and statistical models by using well-defined knowledge as part of a clustering-based outlier detection technique. It is equipped with a flexible and dynamic mechanism to configure and adjust thresholds at runtime, which reduces engineering effort in setting parameters for different environments.
- It can detect multiple sensor anomalies simultaneously, and can scale up to a large number of sensors.
- It is not constrained by the number of users cohabiting the same environment.

To demonstrate the wide applicability, we evaluate CLEAN over four third-party real-world datasets with different sensor deployments, user profiles, and collection periods. As dealing with sensor faults for a long-term smart home environments becomes a more and more important topic, there is a need for a standard methodology to assess the effectiveness of fault detection algorithm. In this paper we present such methodology including how to systematically inject random and systematic abnormal events into a dataset, measure the accuracies of a detection algorithm, and study the impact of the fault detection algorithm on activity recognition.

The rest of the paper is organised as follows. Section 2 briefly discusses existing fault detection work in sensor network and identifies their limitations and differences from CLEAN. Section 3 introduces the proposed approach where we discuss a similarity measure between sensor events and elicit a clustering-based outlier detection algorithm. In Section 4 we design a set of evaluation algorithms and assess the performance of CLEAN through a comprehensive set of experiments. We discuss the benefits and limitations of the approach in Section 5 along with directions for future work. Finally, we conclude in Section 6.

2. Related Work

Sensor fault detection is a popular topic in sensor network, and numerous techniques have been proposed. We will introduce the most representative techniques and discuss why they are not applicable to detect anomaly on event-driven, binary sensors. In addition, we will describe the recent anomaly detection in smart home environment in particular and highlight the difference of our work from them.

2.1. Fault detection in Wireless Sensor Networks

Fault detection has gained much attention in wireless sensor networks, as longer-term deployment in real-world settings significantly increases [29]. Fault detection techniques can be categorised into four groups: rule-based methods, estimation methods, time series analysis, and learning-based methods [25, 29].

2.1.1. Rule-based Methods

A rule-based method relies on expert knowledge about sensor readings to develop heuristic rules for identifying and classifying faulty sensor data. This method works best when the types of faults that can occur and the methods to detect them are known *a priori*. An early solution adopted by Mourand et al., defines ranges for valid sensor readings so as to exclude any observations falling out of reasonable domain limits [23]. This approach is mainly used to clean the data and thus reduce the burden from the domain experts before performing any data analysis process. A more recent example is Suelo, an embedded networked sensing system designed for soil monitoring [27]. Suelo centres on human experts in that human experts need to define a feature space transform that are a set of functions to be called on data points to help assess whether they are valid or not. Based on the initial setup, it actively requests the help of a human when it is necessary to validate, calibrate, repair or replace sensors. Also it learns from the human's actions and responses to automatically update its knowledge base.

The rule-based method could apply to detecting binary sensor anomaly; for example in a single-resident environment, positioning sensors in both front door and bedroom door cannot fire at the same time, suggesting that it is impossible that a person is reported present at these two spatially disjoint places. However, this method requires a great deal of knowledge on spatial layout of the environment or resident behaviour pattern, which might incur a large amount of knowledge engineering effort. Also it is less feasible and error-prone if many sensors are deployed in an environment; for example, it is difficult to specify all the valid rules for a dozen of or twenty sensors, not to mention hundreds of sensors [21].

2.1.2. Estimation methods

An estimation method learns sensor correlations to predict normal behaviour of sensor observations. For physical phenomena like temperature or light, there often exist statistical correlations between sensor measurements. For example, correlations between measurements of sensors that monitor the same physical phenomena but are deployed at different locations, correlations of readings across time, or correlations between measurements of sensors that monitor potentially related phenomena (e.g., temperature and humidity) but are deployed spatially together [10]. For example, Sharma et al use previous reported temperature by one sensor to estimate the future reading by another sensor [29]. If the deviation between the predicted and actual reported readings is over a threshold, then they classify the reported reading as erroneous. Linear Least-Squares estimation is a commonly used method, which is data-centric and mainly based on the covariance between the readings of two sensors. In terms of choosing a threshold, there are two heuristics:

either using the maximum estimation error if there are no faulty samples in the training data, or using the confidence limit if the training dataset has faults.

In CLEAN, we also assume that there exists a correlation between event-driven sensors which is not statistical between their values but semantic in nature, related to the locations in which sensors are placed and objects to which they are attached. We use such semantic relations to spot abnormal sensor events. For example in a single-resident environment, we could regard as abnormal a single firing of a sensor deployed in a bathroom among a collection of kitchen-hosted sensor firings.

2.1.3. Time series analysis

Time series analysis builds a model for data streams collected by the same sensor to exploit their temporal correlations over a long-term period. To detect a fault, a sensor measurement is compared against its predicted value computed using time series forecasting. This method works best if the monitored physical phenomena such as temperature or light exhibits a certain periodic or seasonal pattern. If these phenomena are measured periodically over a long period of time, the resulting time series of measurements can capture the pattern as well as other time scale temporal correlations.

AutoRegressive Integrated Moving Average (ARIMA) is a common time series model for fault detection [11, 29]. The model can account for periodicity in that the predicted measurement at a certain time t depends not only on the measurement at previous time sample $t-1$ but also on measurements made s time samples in the past; that is, at times $t-s$ and $t-s-1$. Similar to the estimation method, the fault detection is done by checking whether the difference between the predicted measurement and the reported measurement at a certain time is greater than a threshold. The prediction is often done at *one-step ahead* – forecast the sensor measurement for the next time $t+1$ and *L-step ahead* – forecast the sensor measurements for time $t+i$ where $1 \leq i \leq L$ [29].

Fang et al. [11] also propose to use ARIMA to reduce errors in data collection while achieving energy efficiency. Each node learns an ARIMA model that will predict if the measurements sampled by the node are within a certain error bound. If the sampled measurement does not agree with the predicted measurement, then it will be further validated by a spatial model to determine whether the model is out of date, or the sampled data is faulty.

The assumption of the time series analysis is that sensors report values in a fixed frequency, which does not hold for event-driven sensors that only report when a triggering condition is satisfied.

2.1.4. Learning-based methods

A learning-based method uses a certain amount of training data to derive a model for normal and faulty sensor readings and then, given an input sensor reading, statistically detects and identifies classes of sensor faults. It is usually integrated with the above estimation and temporal correlation based methods. Bayesian networks, Hidden Markov Models, and neural networks are the most common techniques applied. Dereszynski et al. [6] propose to use Bayesian Network for real-time fault detection and correction on temperature sensor stream collected in an ecological monitoring setting. The approach has two steps: *(i)* inferring a Bayesian network structure for each sensor deployment site, which captures spatial relationships between sensors and then *(ii)* extending the network structures to a Dynamic Bayesian network (DBN) to incorporate temporal correlations. The spatial and temporal correlations captured in different Bayesian networks can help to distinguish sensor failures from valid observations and as well as to predict the true values for the missing or corrupted readings. Similarly, Hill et al. [15] apply a DBN to analyse and diagnose anomalous wind velocity data. They build individual sensor models, on top of which a coupled DBN model is learned to represent the joint distribution of two sensors.

Garneriwal et al. proposes a reputation-based framework for sensor networks [12], where each sensor node maintains a reputation metics about the other nodes in its neighbourhood. The reputation is obtained by making direct observation about these other nodes, and is used to predict their future behaviours. A Bayesian formulation is employed for the algorithm steps of reputation representation, update, integration, and trust evolution.

Paschalidis et al. [26] use Markov models to characterise the normal behaviour of sensor networks; that is, a Markov model at each sensor node is built to estimate anomaly-free probabilities from its past observation

traces, and a tree-indexed Markov model is developed to capture their spatial correlations across the network. Based on derived optimal anomaly detection rules, the approach can assess whether its most recent empirical measure is consistent with the anomaly-free probability model.

The learning-based method is more useful if the phenomena of interest is not spatio-temporally correlated, or the pattern of "normal" sensor readings and the effect of sensor faults on the reported readings are well understood [29]. The need for training data is the main obstacle to the use of learning-based techniques. This is particularly a problem in smart home applications, compared to environmental monitoring applications. First of all, collecting a high quality of training data is difficult [35]. Secondly, the firing of event-driven sensors is subject to the activities being undertaken by human users, whose pattern may vary from time to time. Thirdly, given that the sensors in a smart home environment have a high number of interactions with human users, they are subject to displacement or misuse.

2.2. Detection Techniques in Smart Home Environments

Munir et al. [24] proposes *FailureSense* to detect sensor failures by monitoring the usage of electrical appliances in the home. The assumption behind the algorithm is that there is a correlation between the presence of the users and the activation of the electrical appliances; that is, for the majority of the appliances, they can only be switched on when the user is physically present. Based on this assumption, they learn all the regular intervals between the motion sensors that detect users' presence and the appliance activation, using the Gaussian Mixture Model. The system will report a failure when there is a significant deviation from the regular pattern.

Researchers also propose top down, application-level methods to detect sensor faults [19, 28]. The principle is to look at how sensor failure affects reasoners. Such techniques build a performance profile for a set of classifier instances that are trained with all possible combinations of sensors. Detecting a sensor failure is achieved by comparing the runtime performance with these acquired profiles.

For example, Kapitanova et al. [19] train state-of-the-art classifiers (like Naive Bayes and Hidden Markov Model) to learn high-level human activities (like cooking) from a subset of sensors: excluding one sensor from the whole set of sensors at a time. Fault detection is performed by comparing the performance of these classifiers at recognising real-time activities. There are three main drawbacks to this method: *i)* its principle is to spot single sensor failure at a time, however in reality there could be multiple sensors failing simultaneously, *ii)* while the method might work on a small number of sensors (e.g., dozens) the complexity of constructing the classifier profile suggests that the technique is unlikely to scale to hundreds or thousands of sensors, *iii)* over the long term, sensors never function the same as they do in the training data collection period, and neither do residents behave as expected. During the trial period, the sensors are typically in their best condition (e.g., fully charged and finely tuned) and the residents carefully (sometimes deliberately) interact with sensors. However when the sensors are not under the close watch of professional technicians, they frequently exhibit quite different behaviour. The proposed technique, CLEAN, overcomes these drawbacks.

3. Proposed Approach

We consider sensor anomaly detection as an outlier detection problem, where we assume the majority of sensor events function coherently and we try to detect the minority of sensor events that behave inconsistently from the majority. There are many different outlier detection algorithms [17], and here we choose an unsupervised solution, a clustering-based outlier detection algorithm – *FindCBLOF* [13]. This technique has been successfully applied to detecting abnormal network behaviours, and shares the above assumptions. The basic principles of *FindCBLOF* are as follows:

1. Cluster all the data points into groups, and sort the groups by their size in descending order;
2. To each data point, assign a Cluster-Based Local Outlier Factor (*CBLOF*), which is a product of the size of the cluster that the point belongs to and the similarity between the point and the closest large cluster. The large cluster here means the cluster containing the majority of data points (say 90%). The *CBLOF* suggests the similarity between a data point and a cluster in a statistical way that represents

the probability that the point belongs to the larger cluster. Any data point whose *CBLOF* is below a pre-defined threshold is considered an outlier. That is, the smaller the *CBLOF*, the less similar the point and the larger cluster are, and thus the point is more likely to be an outlier.

To adapt the *FindCBLOF* algorithm to detect anomaly in sensor events, we need to address the following four questions:

- What is the distance measure between two sensor events?
- How do we define a cluster as "large"?
- As sensor events are not static but streaming and continuous data, it is highly likely that abnormal sensor events occur repeatedly. How do we take into account the historic behaviour of sensors and combine it with the *CBLOF*?
- How do we set a threshold on the *CBLOFs* to decide which data points are outliers?

In the following we propose strategies to address these questions.

3.1. Distance Measures between Sensor Events

Event-driven sensor readings are binary, and the distance between these binary numbers alone has little meaning. However, each sensor can be characterised by its implicit semantics such as where the sensor is placed, which object the sensor is attached to, and to whom the room or object belongs. These semantics provide more information than the binary readings alone. In this section we follow the approach we proposed in [35] to characterise the semantics of a sensor event and quantify their distance measure. More technical details can be found in that paper.

We characterise a sensor event into semantic features `[t, l, o, u]`, describing that at the timestamp textttt, a sensor that is installed on an object texttto at a location texttl reports a reading about a user textttu. For example, a sensor event can be represented as `[2008-02-25T00:20:14Z, bedroom, door, main_user]`, indicating that the sensor installed at the door of the bedroom that belongs to the main user fires at the give timestamp. We adopt an ontological approach where we organise concepts in each feature space into a hierarchy based on their granularity level [34]. In the above example, `bedroom`, `door`, and `main_user` are concepts or instances in the Location, Object, and User feature space, and their relationships with the other peer concepts can be: `bedroom` $\sqsubseteq$ `sleeping_area` $\sqsubseteq$ `living_environment`, `door` $\sqsubseteq$ `movable_structure` (from WordNet [22]), and `main_user` $\sqsubseteq$ `any_resident`.

We can use the hierarchy to quantify the similarity of any two of its concepts. Wu et al. [31] propose a conceptual similarity function that works by finding the Least Common Subsumer (*LCS*) of the two input concepts and computing the path length from the LCS up to the root node. The LCS is the most specific concept that both concepts share as an ancestor. This is given by:

$$sim(c_1, c_2) = \frac{2 \times N_3}{N_1 + N_2 + 2 \times N_3}$$

where c_1 and c_2 are concepts in a feature space, N_1 (N_2) is the path length between c_1 (c_2) and the LCS node of c_1 and c_2, and N_3 is the path length between the LCS and the root.

When c_1 is equal to c_2, their LCS node is itself and the similarity is 1.0. When c_1 is semantically far from c_2, their LCS node might be close to the root in the hierarchy, which makes N_1 and N_2 large and N_3 small, so the similarity is close to 0. Therefore, the higher the similarity measure, the closer the two concepts. There exist other measures to quantify the distances between categorical values [2]; most are based on frequencies of the values occurring in a certain dataset, which is not applicable in our approach.

Finally the distance between any two sensor events s_1 and s_2 can be defined as

$$1 - \sum_{i \in \{l,o,u\}} sim(s_1^i, s_2^i) \times \omega_i, (\sum_{i \in \{l,o,u\}} \omega_i = 1)$$

where *sim* is the above similarity function of the hierarchical concepts, and ω_i is the weight of each feature, which reflects the importance of each feature on capturing the similarity of two sensors. For example, if the activities of interest are location-specific, a higher weight can be placed on the location feature. For the purposes of the evaluation in this paper, we uniformly assign the same weight to these three features across all the datasets; that is, 0.33. We use this calculation of similarity between sensors to spot the abnormal sensor events that are distant from dense clusters.

3.2. Ordering of Clusters

Once we have defined the distance measure between any two sensor events, we can cluster them. Let a sensor sequence contain a temporally-ordered list of sensor events. Clustering this sensor sequence will lead to multiple groups, some of which may correspond to different activities from different users while the others could contain the abnormal events. Therefore, it is not as likely that one cluster takes the absolute majority of data points as the original algorithm assumes. To solve this problem, we use the *shoulder-locating* method; we order the clusters by their size, and an abrupt change in their sizes suggests a threshold for distinguishing large and small clusters. For example in Figure 1, we cluster a given sensor sequence into six clusters, and the percentages of their size to the whole number of data points in a descending order are 40%, 38%, 12%, 5%, 3%, and 2%. As we can see, the shoulder point is at the cluster whose percentage is 12%, where we observe the maximal difference between the size percentages. We then consider any cluster whose size percentage is above 12% to be large; that is, any cluster to the left of the dotted red line is considered as large and thus normal; i.e., Cluster 1 and 2. If the clustering results in one group or groups with identical

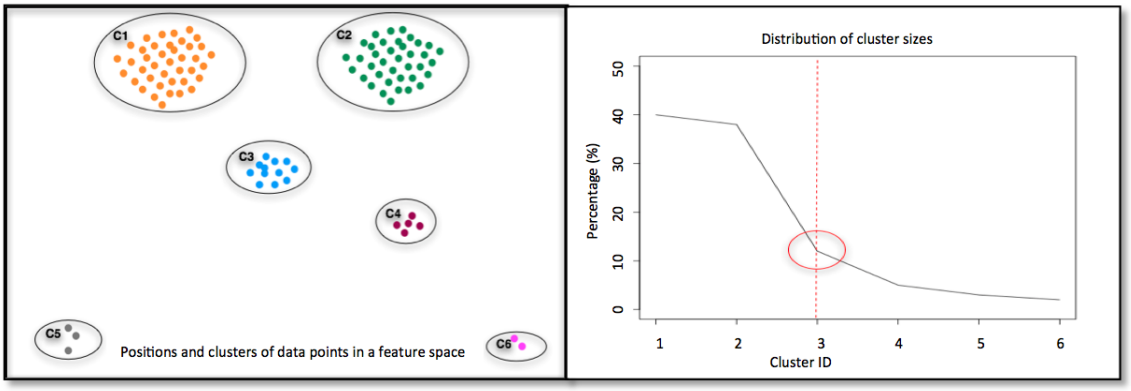


Figure 1: Distribution of Cluster Sizes and the corresponding shoulder locating

sizes, the shoulder cannot be located; i.e., the percentage on the shoulder point will be zero. If so, we cannot find the minority of sparse points and thus we conclude there is no anomaly in this sensor sequence. If the gaps of sizes between groups are the same; e.g., a slope line, then the shoulder point will move downward to the smallest group. In this case, all groups with the exception of the smallest are considered large.

The shoulder-locating method provides a flexible alternative to using a predefined fixed threshold (say 50% or 30%) to determine a "large" group. The reason is that the threshold always depends on the number of sensors deployed, the number of currently fired events, and activities being conducted at the moment. This saves effort, both in terms of knowledge engineering and training normally required to configure or adjust the best threshold setting. In Section 4 we will demonstrate that we run CLEAN over four datasets without the need to re-configure any of these thresholds.

3.3. Considering Historic Sensor Behaviours

As mentioned earlier, abnormal sensor events may be persistent rather than one-off, especially for the systematic type of anomaly caused by technical degradation or dislodgement. Here we consider two extra factors: *frequency* and *temporality*. We assume that the more often and the more recent a sensor behaves

abnormally, the more likely it is that a fault occurs again. We apply these two factors as a weight to *CBLOF* in an exponential function:

$$fw = e^{-f/N}$$

where N is the total number of sensor events being monitored and f is the number of times that abnormal events among the last N events are reported by a certain sensor. The choice of N depends on the intensity of sensing. In the following experiments, we set it universally to be 100; that is, we will look at how many times a sensor will function abnormally in the last 100 events.

$$tw = e^{min(1,td/T)-1}$$

where td is the temporal distance between the current time and the reported event, and T is the range of time of interest (for example, one day is used in our experiment).

Finally for each data point, the extended *CBLOF* will be

$$size\_of\_cluster * sim\_to\_large\_cluster * tw * fw$$

To decide the threshold below which the data point is considered as an outlier, it is unrealistic to set a fixed threshold because the number of sensor events varies with the activities being conducted by the users; for example, some activities like cooking tend to fire more sensors, while the other activities like sleeping only fires a limited number of sensors (e.g., the bedroom door) at its beginning and ending states. It also matters with the number of sensors being installed and the number of users living in the environment. Also different parameters in both temporal and frequency weight functions will make it difficult to fix the thresholds. To solve the problem, we re-use the shoulder-locating method; that is, we order all the *CBLOFs* in a descending order, and find the shoulder point where the maximum change is found. Then we use the *CBLOF* at this shoulder as the threshold. Any data point whose *CBLOF* is below this threshold is treated as an outlier.

4. Experiment and Evaluation

The main objective of the evaluation is to assess the effectiveness of the CLEAN algorithm in detecting *random* and *systematic* abnormal events. Expanding from this objective, we evaluate whether CLEAN can help improve activity recognition accuracies, and uncover insights on the features of abnormal events that are more likely to be detected by CLEAN. In the following, we introduce the datasets to be evaluated, experiment methodologies, and evaluation results.

4.1. Datasets

CLEAN is evaluated on four third-party, publicly-available, real-world datasets that are collected from different smart home environments with different human users and sensor configurations. These datasets capture typical activities and more importantly they represent common types of smart home datasets in terms of the number of sensors, the number of users cohabiting, and the degree of inherent noise. We believe that experimenting on these datasets gives us a comprehensive view of the effectiveness of the proposed technique.

The first two datasets are collected by the University of Amsterdam (named TVK A and TVK B respectively in the following) from two real-world, single-resident houses which were instrumented with wireless sensor networks. The sensor network in the first house is composed of 14 state-change sensors attached to household objects like doors, cupboards, and toilet flushes, while the network in the second house contains reed switches to measure whether doors and cupboards are open or closed; pressure mats to measure sitting on a couch or lying in bed; mercury contacts to detect the movement of objects (e.g., drawers); passive infrared to detect motion in a specific area; float sensors to measure the flush of toilet. All these sensors output binary readings (0 or 1), indicating whether or not a sensor fires. These two datasets

aim to support the recognition of a similar set of activities (e.g., TVK B contains one more activity), while the TVK B sensory data contains more noise than the TVK A data [20].

The third dataset is the PlaceLab Couple dataset [21]. To the best of our knowledge, this dataset is by far the most complicated and largest dataset collected in a real-world environment that is publicly available. The PlaceLab dataset contains over nine hundred sensor inputs, among which 707 are object sensors, including wireless infra-red motion sensors, stick-on object motion sensors, switch sensors, and RFIDs. The dataset was gathered over a period of 15 days during which a married couple (who were unaffiliated with the PlaceLab research) lived in the PlaceLab, generating 455,852 object sensor events in total. This dataset is not only composed of the highest variety of sensors but also contains many noisy events, which is due to the following three reasons: (1) the majority of the sensors (except RFID sensors) are not identity-specific, (2) they are very sensitive to environmental interference (e.g., motion detection sensors), and (3) this couple often perform interleaved activities and only one subject's activities have been annotated [21].

The fourth dataset is the interleaved activities of daily living (labelled as IAA) dataset from the CASAS smart home project [3]. This dataset was collected in a smart apartment testbed hosted at Washington State University during the 2009-2010 academic year. The apartment was instrumented with various types of sensors to detect user movements, interaction with selected items, the states of doors and lights, consumption of water and electrical energy, and temperature, resulting in $2,804,812$ sensor events. The apartment housed two people, `R1` and `R2`, who performed their normal daily activities during the collection period. This dataset will demonstrate CLEAN's performance in detecting abnormal sensor events in a multi-user environment.

4.2. Experiment and Evaluation Methodology

In this section, we will describe the measurements that we use to evaluate the effectiveness of CLEAN, the techniques and parameters to configure CLEAN, and the specific evaluation goals of each of the experiments we conduct.

4.2.1. Measurements

CLEAN is designed as a knowledge-driven anomaly detection algorithm, so we do not need any training data to build the model and use the whole data set for testing. The effectiveness is measured in *precision* – the percentage of the times of detected abnormal events are actually noise being injected into the data, and *recall* – the percentage of the times of injected abnormal events are detected.

4.2.2. Technique and Parameter Selection

In terms of the clustering algorithm, we use DBSCAN [9], which does not require pre-defined cluster sizes and is amenable to our needs – grouping events by their distance and neighbourhood density, which are set as 0.5 and 2 respectively. That is, we cluster events if their distance is close enough (i.e., within 0.5) and have enough close neighbours (i.e., 2 is the minimum number for being a group). This is the only place that we need to set thresholds for the clustering algorithm to work. As mentioned in Section 3, we do not need to configure the thresholds to determine whether a cluster is large or whether an event is an outlier.

4.2.3. Experiments

We conduct two types of experiments relating to the detection of both *random* and *systematic* anomalies. To evaluate CLEAN, we inject different types of noisy events into these datasets and assess how accurately CLEAN can detect them. As the four datasets we evaluate against were collected in real-world environments over a long period of time, it is reasonable to assume that they contain noise; we will discuss the impact of this inherent noise in our evaluation. In the following, we introduce the evaluation methodology for each type of experiments in terms of specific goals and experiment setup.

Random Anomaly Detection Experiment – RADE

Specific Goals. We assess the anomaly detection accuracies of CLEAN. Beyond this, we also assess the

impact of CLEAN on activity recognition. We hypothesise that after running CLEAN, higher activity recognition accuracies will be achieved relative to the accuracies obtained on noisy data. We investigate this using off-the-shelf widely adopted classifiers from the Weka software toolkit [13], including the J48 Decision Tree, Naive Bayes, Bayes Network, and Random Forest. We present the accuracies of recognising activities, and employ Welch's t-test to test the statistical significance of the difference in accuracies. This comparison will demonstrate the effectiveness of CLEAN in terms of helping recognise activities more accurately in the presence of noise, and further demonstrate the resilience of the different classifiers to varying degrees of noise.

Noise Injection and Detection Process. We prepare the datasets by segmenting the sensor events into one-minute time slots—the most common segmentation technique in use [19, 30]. Algorithm 1 illustrates the random anomaly injection and evaluation process. If we want to inject a percentage, P, of abnormal events into the dataset, the total number of abnormal events is given by $P * N$, where N is the total number of sensor events in the original data. For each injection, we randomly *(i)* select a time slot, *(ii)* generate a timestamp within the interval of the time slot, and *(iii)* select a sensor id distinct from all the sensor ids contained in the time slot, which indicates this injected sensor event as a noise. Then we create a new sensor event with the timestamp and sensor id and inject it into the time slot. We repeat the experiment with values of P chosen from 10% to 90%. For each percentage, we run I iterations (in our experiment, I = 100). The results are presented in a box plot, showing the precision and recall distribution of detection, including the minimum, maximum, and mean. This gives a more detailed and complete view than the averaged precision and recall.

Algorithm 1: Evaluation of Random Abnormal Events

Data: L: a list of one-minute segments of sensor events
I: the number of iterations
N: the number of sensor events in total
P: the injection rate of abnormal events
S: the number of sensors
Result: A: the detection precision and recalls

```
for i ← 1 to I do
    IL = L
    for n ← 1 to P * N do
        seg_id = rand_gen(1, size(L))
        ts = rand_gen(L.get(seg_id).start_time, L.get(seg_id).end_time)
        found = False
        while !found do
            sid = rand_gen(1, S)
            if !L.get(seg_id).contains(sid) then
                found = True
        IL.get(seg_id).inject(create_event(timestamp, sid))
    // run CLEAN over CL and put the evaluation accuracy into results
    CL = clean(IL)
    A.append(eval(CL, L))
```

Activity Recognition Process. We take the whole dataset and randomly shuffle it into training and testing sets. For example, we use 50% of the data for training a classifier, and then use the remaining 50% to test the classifier's accuracy. We execute the classifier over three "versions" of the test data:

1. the original, unmodified test data, where we expect the best recognition accuracies.
2. the noise injected data; that is, we inject random noisy events into the original testing data, and run the classifier. We expect recognition accuracies to drop.

3. the noise removed data; that is, we run the CLEAN algorithm on the noise injected data, and run the classifier on the output of the CLEAN algorithm. We hypothesise that the recognition accuracies should improve upon the accuracies attained on the noise injected data.

Algorithm 2 summarises the above process.

Algorithm 2: Impact study experiment setup

Data: L: a list of one-minute segments of sensor events annotated with activities
I: the number of iterations
TP: the training data percentage
T: the number of sensor events in total
P: the injection rate of abnormal events
C: the classifier
Result: A: the detection accuracies on the original, noise injected, and noise removed data
for $i \leftarrow 1$ **to** I **do**
  {train_data, test_data } = shuffle(L, TP)
  train(C, train_data)
  A_O = test(C, test_data)
  test_injected_data = random_injection(test_data, $P * T$)
  A_I = test(C, test_injected_data)
  test_removed_data = clean(test_injected_data)
  A_R = test(C, test_removed_data)
  A.append(A_O, A_I, A_R)

Systematic Anomaly Detection Experiment – SADE

Specific Goals. Beyond evaluating the anomaly detection accuracies, we assess the features of abnormal sensor events that affect the effectiveness of CLEAN; that is, what types of abnormal sensor events are more likely to be detected by CLEAN. To do this, we extract features of injected systematic noisy events and run a linear regression algorithm to build relationships between the features and the detection accuracies, and to identify which features contribute the most.

Noise Injection and Detection Process. To simulate systematic anomalies, we randomly select a number of sensors, and for each randomly selected sensor we create a sensor event and inject it into each time slot, indicating this sensor has been constantly faulty. The number of sensors is chosen from 1 to half of the total number of sensors. For the PlaceLab dataset we only chose 10% of the sensors (i.e.,71). For each number of selected sensors, we run 100 iterations and present the precision and recall over these iterations in a box plot. For example, if the number is 20, then the 100 iterations generates 100 combinations of sensors, each 20 in size. Algorithm 3 illustrates the process.

Correlation study. We attempt to understand whether the following features of sensors have an impact on detection if they suffer systematic faults:

- the number of sensor faults occurring simultaneously: `N_s`; we expect that the more faults there are, the more difficult detection becomes;
- the occurrence and contribution to activities of the sensors before they suffer systematic faults. We consider:
  - the occurrence distribution: the max `Max_o`, mean `Mean_o`, and standard deviation `Std_o`. We want to understand whether faults with sensors that report less frequently will be easier to detect if they suddenly report events constantly.

Algorithm 3: Evaluation of Systematic Abnormal Events

```
Data: L: a list of one-minute segments of sensor events
I: the number of iterations
N: the number of sensors to be injected
S: the set of sensors
Result: A: the detection precision and recalls
for i ← 1 to I do
    // randomly select N number of sensors from S
    NS = random_generate(N, S)
    foreach l in L do
        foreach ns in NS do
            ts = rand_gen(l.start_time, l.end_time)
            // create a sensor event using the randomly generated timestamp with each sensor in NS
            and inject it into each segment.
            l.inject(create_event(ts, ns))
// run CLEAN over CL and put the evaluation accuracy into results
CL = clean(IL)
A.append(eval(CL, L))
```

– the contribution degree of each faulty sensor to the activities; that is, the probabilities of a sensor reporting when an activity is being performed: the max `Max_P(S|A)`, mean `Mean_P(S|A)`, and standard deviation `Std_P(S|A)`. We want to understand whether a sensor fault is easier to detect if the sensor is a significant indicator of a certain activity.

We record the above features along with the precision and recall, when running Algorithm 3. We apply a regression model, where the sensor features are input as predictor variables, and the precision and recall as response variables. The coefficients (e.g., t values) on the predictor variables tell us whether they have a positive or negative impact on predicting the response variables, and their probability values ($\Pr(>|t|)$) indicate whether we should reject the null hypothesis; that is, the coefficients have values of zero. For example, if the probability on a predictor variable is less than 0.05, then there is a strong evidence that the coefficient on this predictor is significantly different than zero, implying this predictor has a significant impact on predicting the response variables.

We also adopt a method called *relative weights*, described in [18], to rank the importance of each predictor variable to predicting the response variables. This method approximates the average increase in R-Square obtained by adding a predictor variable across all possible regression submodels; that is, the higher contribution to R-Square a predictor accounts for, the greater relative importance it has.

4.3. RADE Results

The majority of detected events are abnormal. Figure 2 presents the precisions of detecting random anomalies on these four datasets. The precision is consistently high across all the datasets, indicating that the most detected events are abnormal. Note that the assumption of the algorithm is that the majority of sensor events are normal, which suggests that the majority of sensor events are more likely to form into a cluster whose density is greater than the clusters that contain abnormal events. So, although we inject noisy events that are over 50% percent of the total number of sensor events, it is less likely that these events form a high-density group. For example in the IAA dataset, in one sensor segment that originally contains the events from the three sensors 46, 47, and 48, which are all installed in the front hall, we randomly inject eight sensors which are spread across different bedrooms, the hallway, kitchen, and toilet. Some of these eight sensors form into groups, however, these groups are less cohesive than the group formed by the original three sensors.

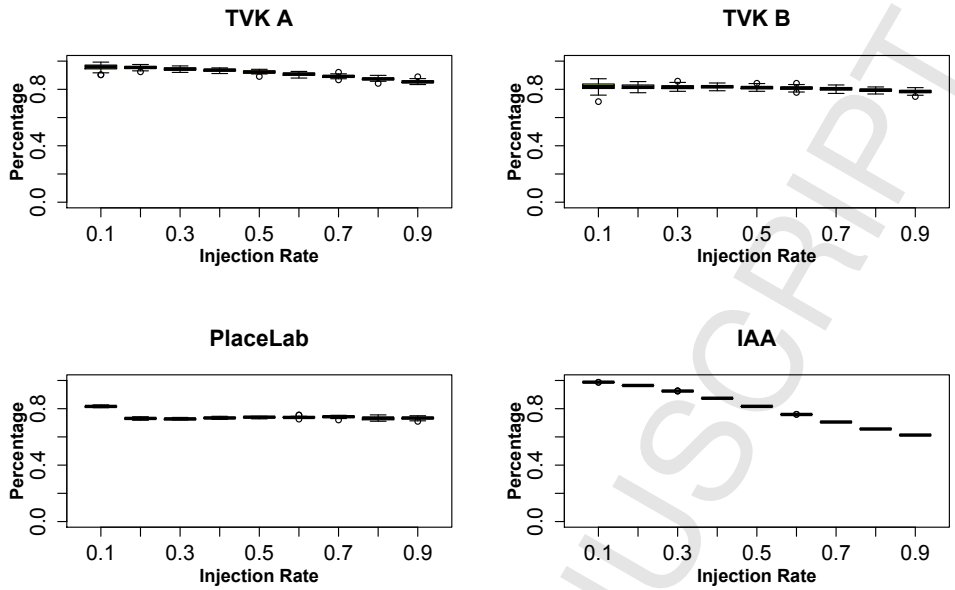


Figure 2: Precisions of detecting random noise on the original datasets. The consistently high precision indicates that detected noise has a high chance of being the injected random noise.

Precision drops more quickly on multi-resident dataset. We observe that the precision on the IAA dataset, which involves two residents, decreases more rapidly than the precision on the other datasets. The reason for this is that the injected noise might form a cluster which is not significantly different from the cluster representing a less active behaviour being conducted by a background user. Taking an example in the IAA dataset, one segment originally contains the events from the five sensors: three in one of the bedrooms, and two in the bathroom. We randomly inject another nine sensor events, among which four of them are from the same sensor. In this case, these four sensors form the biggest group, while the original two events from the bathroom are detected as noise.

Not all abnormal events can be detected by CLEAN, especially when the number of the abnormal events is very small. From Figure 2 and Figure 3, we can see that the recall is much lower than precision, indicating that CLEAN is not effective at detecting all the injected noise events. Also Figure 3 illustrates that the recalls on these four datasets increase with the injection rate; that is, the more noise injected, the more visible they are and thus the higher chance of detecting them. When there are a small number of noisy events being injected, there is a chance that they are grouped with existing sensor events, which cannot be picked up by our algorithm. As more events are injected, the chance of them being separated in different groups becomes higher. The recalls on the TVK B and PlaceLab datasets are worse than the other two datasets, which is partly due to the inherent noise in these datasets [20, 21, 32]; e.g., we detect abnormal events that are not injected but already exist in the data.

Our second experiment is to remove noisy sensor events inherent to the original datasets and repeat the above random noise evaluation process. Although we do not have ground truth about which sensor events correspond to noise, we assume that running the CLEAN algorithm on the raw datasets will remove the majority of inconsistent sensor events from the data. We note that not all the removed sensor events will necessarily be noise, however, this process will give us a cleaner testbed on which to assess the algorithm's accuracy. From this starting point we inject the random noise and see if we can achieve better detection rates.

After cleaning, recall improves. Figure 4 presents the improved recalls on the cleaned datasets. The reason behind this improvement is that after cleaning, the sensor events reported in one time slots are more

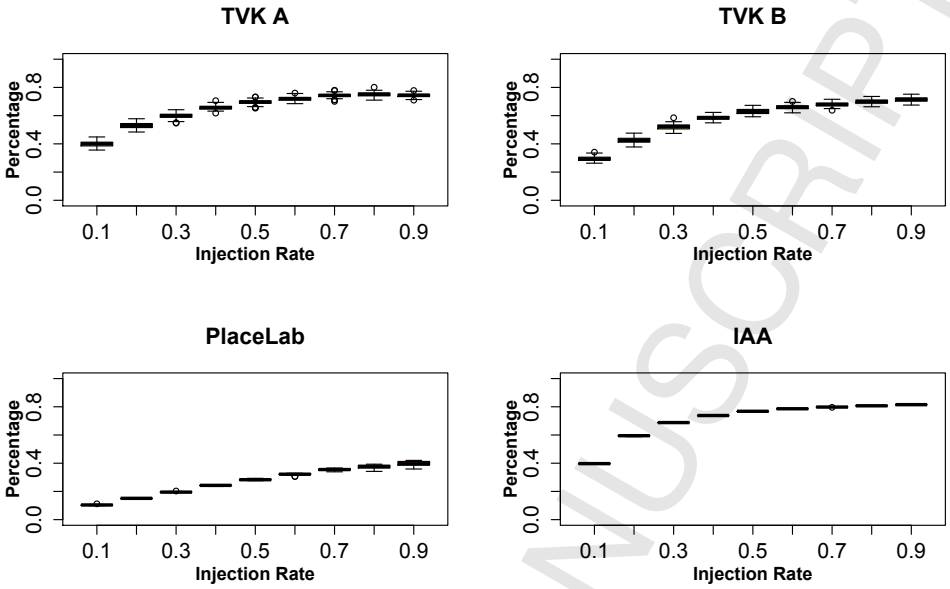


Figure 3: Recalls of detecting random noise on the original datasets. The various recalls over different datasets are due to the potential noise in the original datasets.

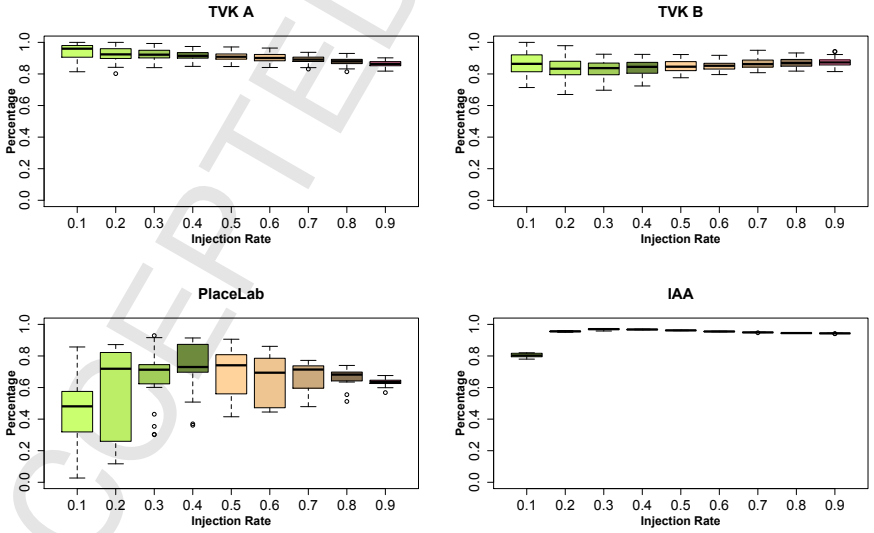


Figure 4: Recalls of detecting random noise in the cleaned datasets.

consistent, and thus any injected abnormal event is more standout and easier to be detected. We also run Welch's t-test to compare the precision and recall on the cleaned datasets to those on the original datasets. The majority of the p-values reported in Table 1 are less than 0.05, demonstrating that the improvement of both the precisions and recalls are statistically significant, especially for the precision on the TVK B and PlaceLab datasets. The improvement on the other two datasets is less significant, as the cleaning process on them is not as effective. The TVK A dataset is relatively clean. The IAA dataset contains two residents, so each segment is often composed of multiple groups of sensor traces to represent activities from each resident. Due to the nature of these activities, the groups of sensor traces may have unbalanced sizes, which makes it difficult to distinguish between noisy events and sensor events representing less active activities.

Table 1: p-values on improved precision and recall of detecting random noisy events on the original and cleaned datasets

p-value	TVK A	TVK B	PlaceLab	IAA
Precision	0.05	1.9e-09	7.93e-08	1
Recall	0.0002955	5.15e-05	5.209e-07	6.689e-05

To characterise the potential noise in the original dataset, we present the frequency of sensors that have reported abnormal events on a daily basis from the original TVK B dataset in Figure 5. It is clear that sensor 28 consistently reports abnormal readings. We manually examine the raw data and find that this sensor is a passive infra-red sensor installed in the kitchen, which reports all the time, especially on the last day of data collection. CLEAN does not detect any abnormal events from this sensor because as the user was not at home on that day, no other sensor fired. Without peer comparison, CLEAN always forms one cluster that contains only this sensor, thus the conclusion that there is no anomaly is drawn. This is one drawback of CLEAN, which can be complemented by a rule based check; that is, if one sensor reports continuously over an long period (e.g., one day), then this sensor should be considered abnormal.

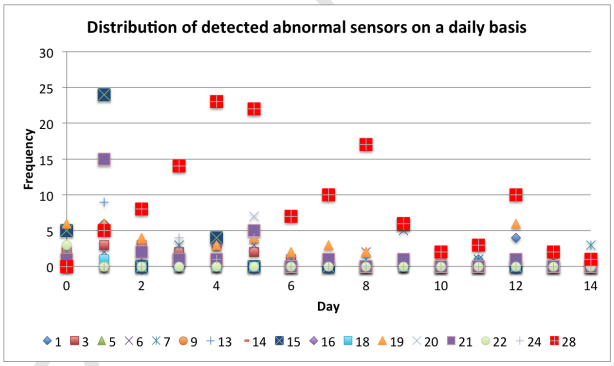


Figure 5: Frequency of sensors that have been detected to report abnormal events on a daily basis in TVK B dataset

4.4. Impact Study Results

Running CLEAN can improve activity recognition accuracies. Figure 6 shows the activity recognition accuracies across three classifiers – J48, Bayes Network, and Random Forest on the TVK A dataset when the classifiers are trained on 50% of the data. We can observe that the accuracies achieved on noise removed data lie between those attained on the original and the noise injected data. The accuracies on the noise removed data are very close to those on the original data, and there is an observable increase compared to the the accuracies on the noise injected data as the error injection rate increases. The higher the error injection rate, the larger the gap between the accuracies on the noise injected data and the noise removed data, indicating the better improvement is achieved.

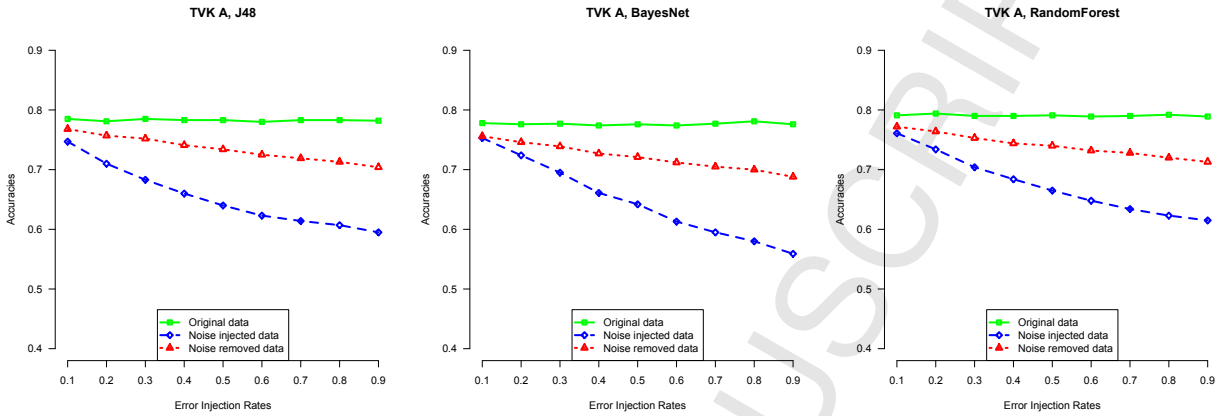


Figure 6: Comparison of activity recognition accuracies across different classifiers on TVK A with 50% of training data

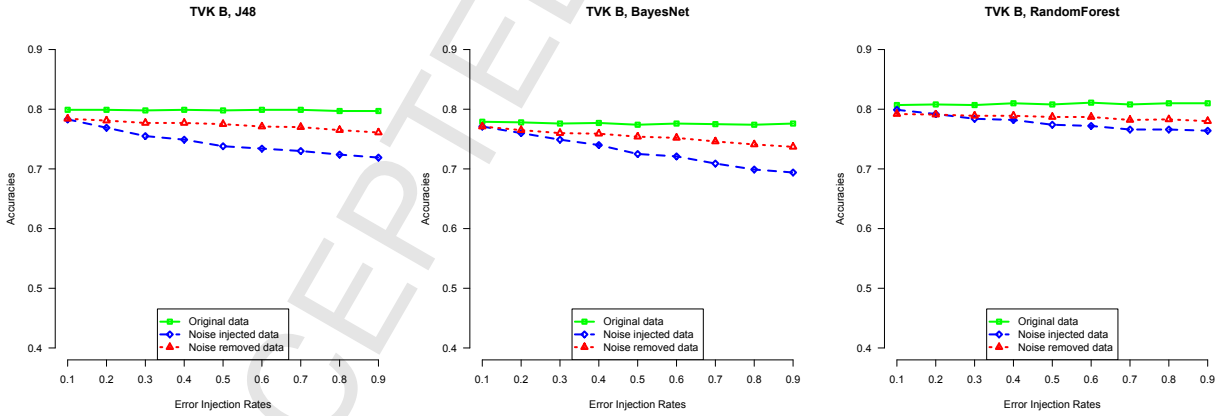


Figure 7: Comparison of activity recognition accuracies across different classifiers on TVK B with 50% of training data

The improvement on activity recognition accuracies is less observable on the originally noisy datasets. Figure 7 presents the impact study results on the TVK B dataset. Compared to the results on the TVK A dataset, the improvement of the recognition accuracies from the noise removed data to those from the noise injected data across all the four classifiers is much less significant. This phenomena is consistent with the relatively lower anomaly detection precision and recall on the TVK B dataset in Figure 2 and 3. The detection recall in Figure 3 is approximately running from 25% to 60%, indicating that the injected noisy events have not been sufficiently detected. Thus, the noise removed data are still heavily combined with noisy events, so the recognition accuracies on the noise injected and removed data are not significantly different.

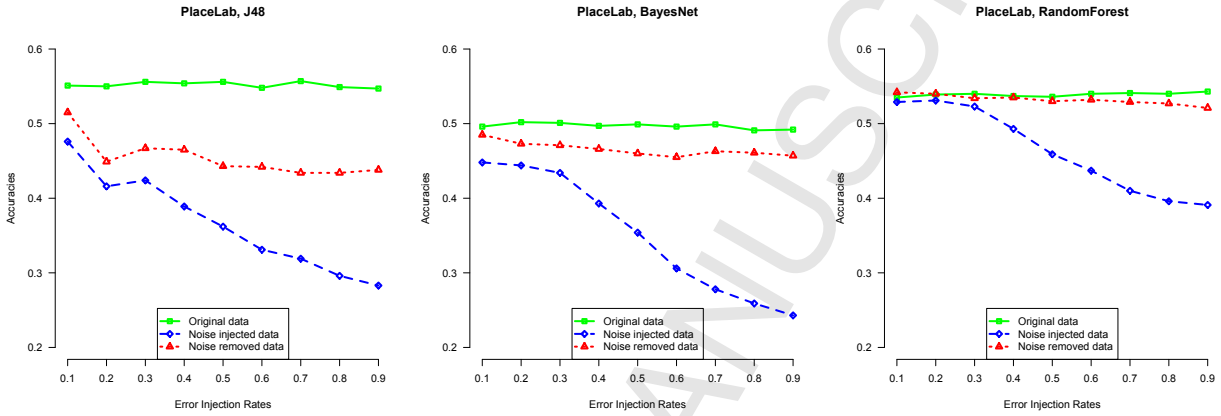


Figure 8: Comparison of activity recognition accuracies across different classifiers on PlaceLab with 50% of training data

Figure 8 presents the impact study results on the PlaceLab dataset. Compared to the previous two, the recognition accuracies on the PlaceLab dataset are much poorer. With the exception of the J48 decision tree, the recognition accuracies on the original data and noise removed data are very close on the other two classifiers, which have exhibited visible improvement from the accuracies on the noise injected data. As mentioned earlier, both the PlaceLab and TVK B datasets contain a significant number of noisy events, consequently we would expect similar results on both datasets. However, we note that the PlaceLab dataset contains far more sensors (i.e., 700 vs 22) and its total number of sensor events is also much higher—injecting noise at higher injection rates has a larger impact on the original data, leading to faster degradation of recognition accuracies.

Ensemble classifiers are good at dealing with noisy data. In summary, the above results demonstrate the effectiveness of the CLEAN algorithm by showing the improved accuracies from the noise removed data compared to the noise injected data. The results also show the noise resistance degrees of different classifiers. The Random Forest achieves the best recognition accuracies overall, as it is an ensemble learning method that makes a prediction on multiple decision trees constructed at the training time. The recognition accuracies on the noise injected data from the J48 decision tree and Bayes Network degrade faster as the error injection rate increases. Both phenomena can also be explained by the large number of sensors in the PlaceLab dataset and the feature selection in these two classifiers. For example, from manual inspection of the J48 decision tree model, we find that the tree is only built on around 70 sensors, which is 10% of the total input sensors on the PlaceLab dataset. With the higher error injection rate, this subset of sensors is more likely to be contaminated, leading to a different decision path and thus incorrect inference result.

4.5. SADE Results

Systematic detection accuracies vary with individual sensors being faulty. Figure 9 and 10 present the precisions and recalls of detecting systematic sensor anomalies. We can see that there are

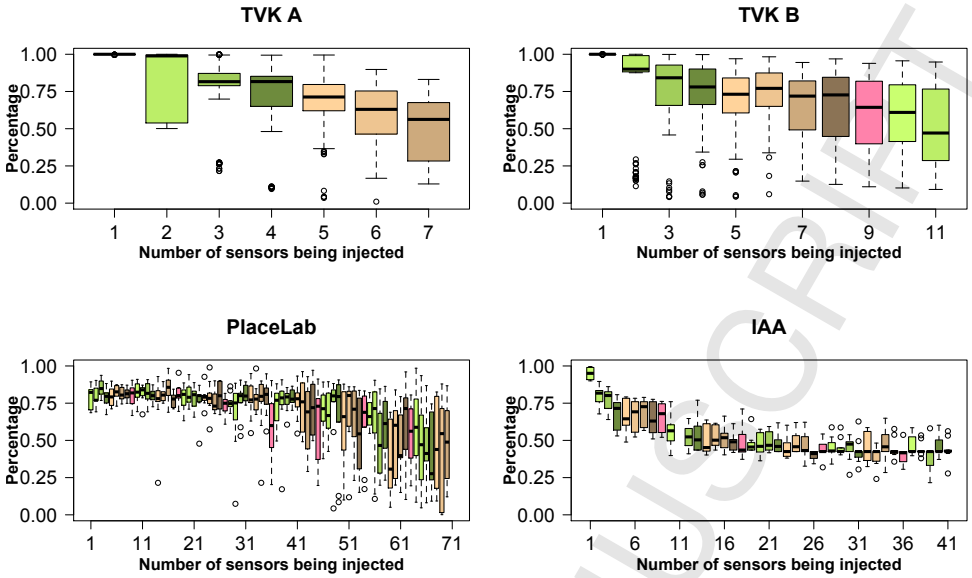


Figure 9: Precisions of detecting systematic noise on the original datasets. Precision of detecting systematic sensor noise is highly influenced by the chosen sensors.

fluctuations in both precision and recall, especially on the PlaceLab dataset. It shows that different sensors being consistently faulty will have different impact on the detection accuracies. For example, for a sensor that fires already frequently throughout the dataset, if we set this sensor to be constantly faulty and report readings all the time, then it being faulty will be less likely to be detected. On the contrary, our algorithm will be more likely to detect the other co-occurring normal events as abnormal, and thus update their temporal and frequency weights to make them more likely to be detected as an outlier.

On the other hand, if a generated event is associated with a sensor that is critical to a less frequent activity, then it is more likely that this sensor event will be detected as being abnormal. For example, when we deliberately set the sensor on the front door – which only contributes to identifying the user's entry and departure from the house – in the TVK A dataset to report readings in every time slot, the detection precision and recall are 94.3% and 80% respectively. Because of these frequency and contribution factors of sensors, the precisions and recalls do not follow a more observable trend as that of detecting random anomaly.

4.6. Results on Correlation Study

As the correlation study is independent of datasets, we combine the collected features and precision and recalls on all the TVK A, B, and PlaceLab datasets. As we do not have ground truth on how each sensor event contributes to the co-occurrence of activities, we exclude the IAA dataset from this correlation study. We fit the data to a linear regression model, which achieves a p-value <2.2e-16, indicating that the data fits the model well and is better than random noise. Table 2 presents the coefficients (i.e., t value) and probability values (i.e., $\Pr(>|t|)$) on each predictor variable (i.e., sensor features) to the two response variables (i.e., precision and recall).

Since the linear regression model is data-driven and the systematic anomaly detection experiments are conducted on randomly shuffled and generated data, the actual t values for each predictor variable are irrelevant and will be more likely to change if we run Algorithm 3 another 100 times. Therefore, we focus on their positive and negative impact on predicting the precision and recall, and the relative importance between these variables. As presented in Table 2, the number of faulty sensors has a negative impact on both precision and recall as expected. In terms of sensor occurrence profile, from the negative impact of

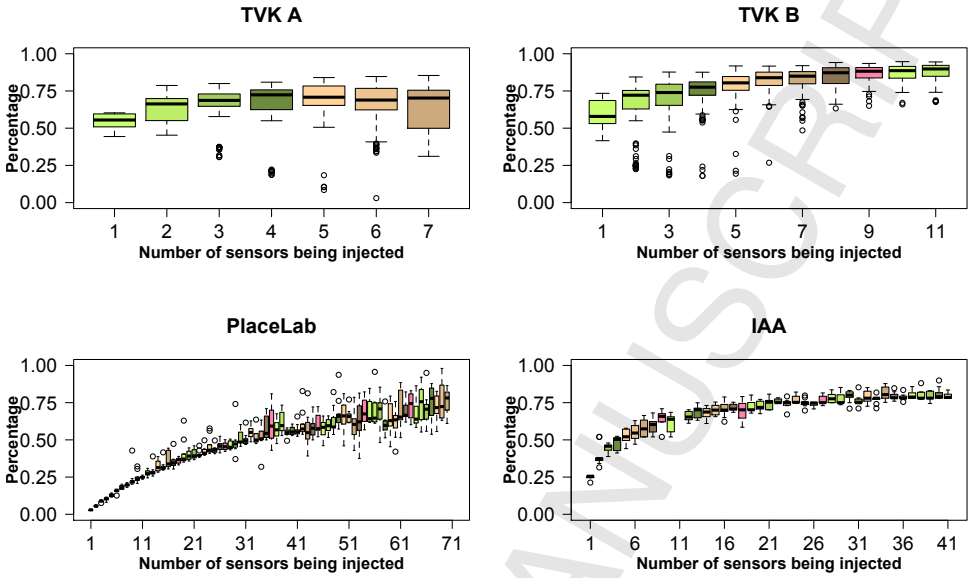
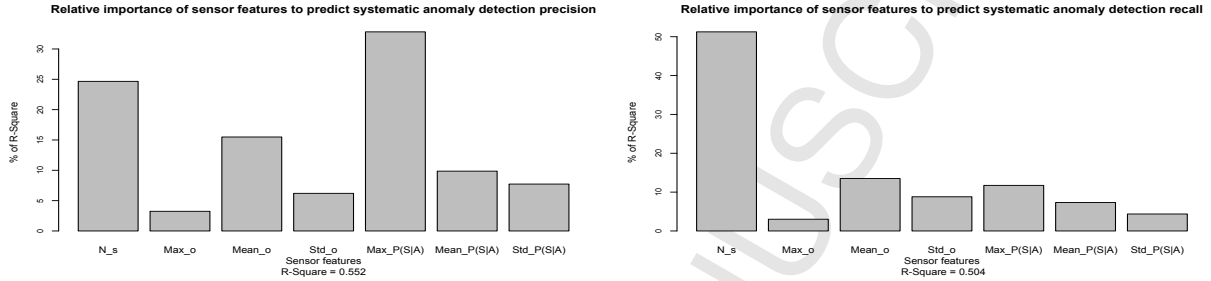


Figure 10: Recalls of detecting systematic noise on the original datasets

Table 2: Coefficients and probabilities of a linear regression model on correlating sensor features to systematic anomaly detection precision and recall on the TVK A, B, and PlaceLab datasets

Predictor	Precision		Recall	
	t value	Pr(> |t|)	t value	Pr(> |t|)
N_s	-20.85	<2e-16	-51.82	<2e-16
Max_o	-15.49	<2e-16	-10.24	<2e-16
Mean_o	24.39	<2e-16	14.8	<2e-16
Std_o	0.51	0.612	7.14	9.9e-13
Max_P(S|A)	-48.11	<2e-16	-16.63	<2e-16
Mean_P(S|A)	5.6	<2e-16	-4.13	3.74e-5
Std_P(S|A)	10.25	<2e-16	8.79	<2e-16

Max_o we can derive that if a sensor with high occurrence ratio becomes faulty, then it will be more difficult to detect, given that this sensor is expected to report frequently. However, if all the sensors with high occurrence ratio become faulty, then it is less likely they will be detected. The reason is that it is less likely that all these sensors are semantically consistent, and thus it would be easier to detect contradiction by CLEAN. In terms of the correlation between sensors and activities, the higher the contribution of a sensor to an activity, the less likely an anomaly is to be detected. This shares the same reason as above: that is, the sensor is expected to report whenever an activity is being performed. However, the average contribution of faulty sensors to all the activities has less impact on either precision or recall, given that the coefficients on this predictor are much smaller than those on Max_P(S|A).



(a) Precision (b) Recall

Figure 11: Relative importance of sensor features to predict systematic anomaly detection accuracies on the TVK A, B, and PlaceLab datasets

Figure 11 ranks these features according to their relative importance to predicting the precision and recall. The ranking is based on their contribution to the R-Square value, indicating how well the data fits the model. In terms of precision, the most important indicator is the maximum contribution degree of a sensor to an activity. It is consistent with what we have observed during the diagnosis process. As discovered, if the front door sensor becomes faulty, then it is highly likely to be detected, because it contributes 100% to identifying the leaving house activity. In terms of recall, the most important indicator is the number of faulty sensors. Comparing Figure 10 with Figure 9, we can see that there is an observable trend of recall increasing with the number of sensors being injected across all the datasets.

5. Discussion

In this section we discuss the utility of the CLEAN technique, and the practical issues of using it and improving it in the future.

5.1. Novel Integration of Knowledge and Statistical Techniques

CLEAN novelly integrates knowledge within a clustering-centred technique. It is built on top of a general ontological model that formally represents the semantic concepts of a sensor event, such as its location and originating object. The hierarchy of such conceptual models allows us to quantify the distance between any two sensor events, so that we may cluster a sequence of sensor events into groups and further identify the minority of outlier events inconsistent with the majority.

5.2. Independence of Training Data

The novel integration of knowledge with statistical techniques makes CLEAN independent of training data; that is, we do not need the training data to build a model describing what "properly functioning" sensor events are; for example, which sensors should fire together, or which sensors should fire with what activities. Such information is often difficult to specify, especially when there are many sensors deployed (e.g., hundreds or thousands), or sensors are unreliable and subject to change in the environment along with

residents' behaviour patterns; that is, users might change the way that they perform certain activities, leading to different associations between sensor events and activities. CLEAN does not rely on such information but on a more reliable knowledge model. This makes it possible to use CLEAN for a long-term period without the need to collect training data or to update a model.

In Section 4.4, we evaluated the impact of CLEAN on improving the activity recognition on four classic supervised classifiers that make use of training data. However, we consider that CLEAN would better compliment unsupervised activity recognition approaches, such as USMART [36], thus removing the need for training data at any part of the recognition process.

5.3. Scalability and Extensibility

The independence of training data makes CLEAN applicable to a wide range of home environments in that the ontological model is generic enough to be reused without much re-engineering effort [35]. The main change resides in the location model; that is, the location ontology needs to capture the spatial layout particular to each environment.

CLEAN is neither constrained by the number of sensors being deployed nor by the number of sensors being faulty at the same time. Kapitanova et al. [19] build numerous classifier profiles by excluding each single sensor so as to spot single sensor failure. This approach is inefficient when it comes to a large number of sensors, and infeasible as there might be multiple sensors being faulty at the same time. CLEAN detects anomalies by comparing semantic distances between co-reporting sensor events. CLEAN is not constrained by pre-trained models and thus is more flexible.

Recognising co-occurring activities for multiple users cohabiting the same environment is challenging, and detecting abnormal events in such an environment is even more challenging as it is difficult to distinguish which sensors contribute to which user's activities. CLEAN adopts the *FindCBLOF* technique that supports the clustering of multiple groups of events and only identifies the inconsistent minority as an outlier.

5.4. Use of CLEAN

The main functionality of CLEAN is to identify abnormal sensor events and clean the incoming sensor stream. We have demonstrated that it can help improve the activity recognition accuracies across multiple off-the-shelf classifiers. On top of the identified sensor events, we can further define application logic about customised actions to take when anomaly is detected on certain sensors. For example, the system might just log abnormal events from most sensors without taking any action, but it may want to keep an eye on critical sensors such as a fire detection sensor. Even if a single abnormal event is detected from such sensor, the system should raise an alarm on either checking whether the fire is actually happening or the sensor is dislodged.

5.5. Limitations and Future Work

5.5.1. Improving Recall

The results presented in Section 4 demonstrate that CLEAN achieves high precision, indicating that it is good at detecting injected noisy events; however lower recall implies that not all the detected events are injected noise. This is partly due to the noise inherent to in the original datasets, as after cleaning the original datasets, we demonstrate significant improvements in recall. However, on the PlaceLab dataset, which contains far more sensors than the other datasets, recall is relatively low. The clustering on a sequence of sensor events often results in a number of small groups, which all are regarded as outliers. Further investigation of a more nuanced approach to quantifying the distance between sensor events, or setting of the distance threshold may achieve better clustering and outlier detection.

5.5.2. Isolated Single Sensor Failure

As shown in Section 4.3, CLEAN does not work well when the fault sensor is the only sensor reporting, because only a single cluster is created. However, this situation may be easily resolved by combining CLEAN with a rule: Any sensor that reports continuously over an unreasonably long period should be considered as faulty. The time threshold could be customised to different types of sensors, based on how often each sensor normally fires.

5.6. Missing Data Detection

Currently, CLEAN is designed to detect "excessive" data, and cannot yet detect "missing" data. Missing sensor data is subtle in that not reporting a value does not imply a sensor is broken, and could simply mean that the user has not interacted with the object that the sensor is attached to. For example, in the activity of making coffee, we cannot conclude that the sensor on the sugar jar is not working just because it did not fire as the activity was carried out—the user could simply have chosen not to add sugar to the coffee. That is, we need to distinguish the causes of the missing sensor data: whether the sensor becomes faulty or the users change their behaviour pattern, thus not firing certain sensors. Another missing data example would be that a temperature sensor originally on a kettle that reports the boiling of the water has been dislodged and thus has not reported any reading. This case can also be classified as a missing data problem. One way to address this is to look for the correlations between sensors. In this particular example, we could check whether the electricity current monitoring sensor (if exists) reports valid readings for the usage of the kettle. If so and the temperature sensor has not reported as expected, then we can conclude that the temperature sensor has becomes faulty or been dislodged. However, the correlation between these two sensors will need to be either trained from historical data or specified by the domain experts. Our future work will look into correlations between sensors, and between sensors and activities, and integrate such knowledge with CLEAN to try to detect missing events.

6. Conclusion and Future Work

This paper presents CLEAN, a technique that leverages sensor semantics in a statistics-driven outlier detection method to detect abnormal events, which does not rely on any training data nor requires ground truth annotation. It is generic in that it scales well with the number of sensors, can be deployed with single- or multi-resident environments, and can be integrated with existing activity recognition techniques. By taking the streaming sensor events gathered from various sensors as input, CLEAN detects and removes abnormal events, which can be used to filter data fed as input to any activity recognition algorithms. We demonstrate its performance on four real-world datasets with various environments, sensor deployments, the number of users living in an environment, and the extent of noise underlying in the dataset. Using four different activity recognition classifiers, we demonstrate that CLEAN improves activity recognition rates in the presence of noise. We discuss the benefits and limitations of CLEAN, and identify areas of future work.

References

[1] G. D. Abowd and E. D. Mynatt. *Designing for the Human Experience in Smart Environments*, pages 151–174. John Wiley & Sons, Inc., 2005.

[2] S. Boriah, V. Chandola, and V. Kumar. Similarity measures for categorical data: A comparative evaluation. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2008, April 24-26, 2008, Atlanta, Georgia, USA*, pages 243–254. SIAM, 2008.

[3] D. Cook and M. Schmitter-Edgecombe. Assessing the quality of activities in a smart environment. *Methods of Information in Medicine*, 48:480–485, 2009.

[4] D. Cook, M. Schmitter-Edgecombe, and P. Dawadi. Analyzing activity behavior and movement in a naturalistic environment using smart home techniques. *IEEE Journal of Biomedical and Health Informatics*, 19(6):1882–1892, Nov 2015.

[5] L. Coyle, J. Ye, E. Loureiro, S. Knox, S. Dobson, and P. Nixon. A proposed approach to evaluate the accuracy of tag-based location systems. In *Ubicomp 2007 Workshops Proceedings: the First Workshop on Ubiquitous Systems Evaluation*, pages 292 – 296, Innsbruck Austria, September 2007.

[6] E. W. Dereszynski and T. G. Dietterich. Spatiotemporal models for data-anomaly detection in dynamic environmental monitoring campaigns. *ACM Transactions on Sensor Network*, 8(1):3:1–3:36, Aug. 2011.

[7] D.J.Cook. How smart is your home? *Science*, pages 1579–1581, 2012.

[8] F. Doctor, H. Hagras, and V. Callaghan. A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments. *IEEE Transactions on Systems, Man, and Cybernetics (A)*, 35(1):55–65, 2005.

[9] M. Ester, H. peter Kriegel, J. S, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of Knowledge Discovery and Data Mining: KDD'96*, pages 226–231. AAAI Press, 1996.

[10] L. Fang and S. Dobson. In-network sensor data modelling methods for fault detection. In *Evolving Ambient Intelligence*, volume 413 of *Communications in Computer and Information Science*, pages 176–189. Springer International Publishing, 2013.

[11] L. Fang and S. Dobson. Unifying sensor fault detection with energy conservation. In *Proceedings of International Workshop on Self-Organizing Systems IWSOS'13*, 2013.

[12] S. Ganeriwal, L. K. Balzano, and M. B. Srivastava. Reputation-based framework for high integrity sensor networks. *ACM Transaction on Sensor Network*, 4(3):15:1–15:37, June 2008.

[13] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques, 3rd Edition*. Morgan Kaufmann, 2011.

[14] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen. The gator tech smart house: a programmable pervasive space. *Computer*, 38(3):50–60, March 2005.

[15] D. Hill, B. Minsker, and E. Amir. Real-time bayesian anomaly detection for environmental sensor data. In *Proceedings of IAHR' 07*, Marid, Spain, 2007.

[16] T. W. Hnat, V. Srinivasan, J. Lu, T. I. Sookoor, R. Dawson, J. Stankovic, and K. Whitehouse. The hitchhiker's guide to successful residential sensing deployments. In *Proceedings of SenSys '11*, pages 232–245. ACM, 2011.

[17] V. J. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22:2004, 2004.

[18] R. Kabacoff. *R in Action*. Manning Publications, 2011.

[19] K. Kapitanova, E. Hoque, J. A. Stankovic, K. Whitehouse, and S. H. Son. Being smart about failures: Assessing repairs in smart homes. In *Proceedings of UbiComp '12*, pages 51–60, 2012.

[20] T. L. M. Kasteren, G. Englebienne, and B. J. A. Krose. Human activity recognition from wireless sensor network data: Benchmark and software. In *Activity Recognition in Pervasive Intelligent Environments*, volume 4 of *Atlantis Ambient and Pervasive Intelligence*, pages 165–186. 2011.

[21] B. Logan, J. Healey, M. Philipose, E. M. Tapia, and S. Intille. A long-term evaluation of sensing modalities for activity recognition. In *Proceedings of UbiComp '07*, pages 483–500, 2007.

[22] G. A. Miller. Wordnet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, Nov. 1995.

[23] M. Mourad and J. Bertrand-Krajewski. A method for automatic validation of long time series of data in urban hydrology. *Water Science Technologies*, 45:263–270, 2002.

[24] S. Munir and J. Stankovic. Failuresense: Detecting sensor failure using electrical appliances in the home. In *Mobile Ad Hoc and Sensor Systems (MASS), 2014 IEEE 11th International Conference on*, pages 73–81, Oct 2014.

[25] K. Ni, N. Ramantahan, M. Nabil, H. Chehade, L. Balzano, S. Niar, E. Zahedi, S. Kohler, G. Pottie, M. Hansen, and M. Srivastava. Sensor network data fault types. *ACM Transaction on Sensor Network*, 5, 2009.

[26] I. C. Paschalidis and Y. Chen. Statistical anomaly detection with sensor networks. *ACM Transaction on Sensor Network*, 7, 2010.

[27] N. Ramanathan, T. Schoellhammer, E. Kohler, K. Whitehouse, T. Harmon, and D. Estrin. Suelo: Human-assisted sensing for exploratory soil monitoring studies. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 197–210, New York, NY, USA, 2009. ACM.

[28] H. Sagha, J. Millan, and R. Chavarriaga. Detecting and rectifying anomalies in body sensor networks. In *Proceedings of the International Conference on Body Sensor Networks*, 2011.

[29] A. B. Sharma, L. Golubchik, and R. Govindan. Sensor faults: Detection methods and prevalence in real-world datasets. *ACM Transaction on Sensor Network*, 6(3):23:1–23:39, June 2010.

[30] T. van Kasteren, A. Noulas, G. Englebienne, and B. Kröse. Accurate activity recognition in a home setting. In *Proceedings of UbiComp '08*, pages 1–9, 2008.

[31] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, ACL '94, pages 133–138, Stroudsburg, PA, USA, 1994.

[32] J. Ye, L. Coyle, S. Dobson, and P. Nixon. Using situation lattices in sensor analysis. In *Proceedings of PERCOM '09*, pages 1–11, 2009.

[33] J. Ye, S. Dobson, and S. McKeever. Situation identification techniques in pervasive computing: a review. *Pervasive and mobile computing*, 8:36–66, Feb. 2012.

[34] J. Ye, G. Stevenson, and S. Dobson. A top-level ontology for smart environments. *Pervasive and Mobile Computing*, 7:359–378, June 2011.

[35] J. Ye, G. Stevenson, and S. Dobson. KCAR: A knowledge-driven approach for concurrent activity recognition. *Pervasive and Mobile Computing*, (0), 2014.

[36] J. Ye, G. Stevenson, and S. Dobson. Usmart: An unsupervised semantic mining activity recognition technique. *ACM Transactions on Interactive and Intelligent System*, 4(4):16:1–16:27, Nov. 2014.

[37] J. Ye, G. Stevenson, and S. Dobson. Fault detection for binary sensors in smart home environments. In *Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on*, pages 20–28, March 2015.

[38] G. Youngblood and D. Cook. Data mining for hierarchical model creation. *IEEE Transactions on Systems, Man, and Cybernetics (C)*, 37(4):561–572, July 2007.