Available online at www.sciencedirect.com

ScienceDirect

Performance Evaluation 00 (2010) 1–24

Performance Evaluation

Online Anomaly Detection for Sensor Systems: a Simple and Efficient Approach

Yuan Yao[a], Abhishek Sharma[b], Leana Golubchik[a,b], Ramesh Govindan[b]

[a]*Department of Electrical Engineering-Systems, USC, Los Angeles, CA 90089*
[b]*Department of Computer Science, USC, Los Angeles, CA 90089*

Abstract

Wireless sensor systems aid scientific studies by instrumenting the real world and collecting measurements. Given the large volume of measurements collected by sensor systems, one problem arises – an automated approach to identifying the "interesting" parts of these data sets, or *anomaly detection*. A good anomaly detection methodology should be able to accurately identify many types of anomalies, be robust, require relatively little resources, and perform detection in (near) real-time. Thus, in this paper we focus on an approach to *online* anomaly detection in measurements collected by sensor systems, where our evaluation, using real-world datasets, shows that our approach is accurate (it detects over 90% of the anomalies with few false positives), works well over a range of parameter choices, and has a small (CPU, memory) footprint.

Keywords: anomaly detection, sensor systems, real-world deployments.

1. Introduction

Wireless sensor systems have significant potential for aiding scientific studies by instrumenting the real world and collecting measurements, with the aim of observing, detecting, and tracking scientific phenomena that were previous only partially observable or understood. However, one obstacle to achieving the full potential of such systems, is the ability to process, in a timely and meaningful manner, the huge amounts of measurements they collect. Given such large volumes of collected measurements, one natural question might be: *Can we devise an efficient automated approach to identifying the "interesting" parts of these data sets?* For instance, consider a marine biology application collecting fine-grained measurements in near real-time (e.g., temperature, light, micro-organisms concentrations) – one might want to rapidly identify "abnormal" measurements that might lead to algal blooms which can have devasting consequences.

Email addresses: yuanyao@usc.edu (Yuan Yao), absharma@usc.edu (Abhishek Sharma), leana@usc.edu (Leana Golubchik), ramesh@usc.edu (Ramesh Govindan)

We can view identification of such "interesting" or "unexpected" measurements (or events) in collected data as anomaly detection. In the remainder of the paper, we use the generic term "anomaly" for all interesting (typically, other-than-normal) events occurring either on the measured phenomena or the measuring equipment. Automated *online (or near real-time) anomaly detection in measurements collected by sensor systems* is the focus of this paper.

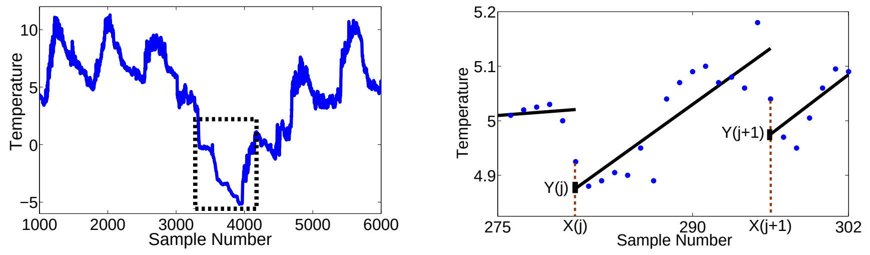
Anomalies can have a variety of lengths, magnitudes, and patterns. For instance, Figure 1(a) depicts a long duration, relatively gradual change in sensor reading, whereas Figure 2(b) includes several short duration, quite abrupt change in sensor readings. Both scenarios correspond to anomalous events and should be accurately detected by an anomaly detection methodology.

Thus, a good anomaly detection methodology should have the following properties. First, it should be able to accurately identify all types of anomalies as well as normal behavior (i.e., it should have low false negative and false positive rates). Second, it should be robust, i.e., the methodology should be relatively insensitive to parameter settings as well as pattern changes in the data sets. Third, it should require relatively small amounts of resources, as these are typically limited in sensor systems. That is, to run on sensor systems, it should ideally have low computational complexity, occupy little memory space, and require little transmission power. Last, it is also desirable for a detection algorithm to be able to detect anomalies in real-time or near real-time. This is particularly important for sensor systems corresponding to temporary deployments (as it might not be as useful to detect anomalies once the deployment is over) and those monitoring hazardous natural phenomena (e.g., spread of contaminants in aquatic ecosystems), where prompt detection (and reaction) can be essential to reducing loss of life and money.

Anomaly detection, in general, has been studied in a number of systems contexts, most notably in networking, where several techniques have been proposed for detecting network traffic anomalies [1, 2, 3, 4]. While one might take the approach of adapting one (or more) of these techniques to sensor systems, we believe that they do not satisfy all the desirable properties described above, at least in their current form. In Section 6, we provide (a) quantitative results from applying network anomaly detection techniques to data collected by real sensor systems deployments, and (b) intuition for why these techniques did not yield good results on such data. Consequently, the properties required of an effective anomaly detection method for sensor data and our experience with applying network traffic anomaly detection techniques to sensor measurements, motivated us to explore methods different from prior work in network anomaly detection.

We also note that little exists in the literature on the topic of anomaly detection in sensor systems data. Most efforts are focused on detection of faulty sensor readings, such as those depicted in Figures 3(a) and 3(b) – these are typically short duration events, with values significantly deviating from the "normal" sensor readings [5]. Often, such sensor data faults are modeled as outliers and can be detected using simple Rule-based approaches or by using statistical models to capture the pattern of normal sensor readings and flagging any significantly different samples as faulty [6]. In this work, we view faulty sensor readings as a special case of anomalies. As illustrated in Section 4, our approach is able to capture such faulty readings, as well as other long duration, "gradual" anomalies such as the one depicted in Figure 1(a).

To the best of our knowledge, the only efforts focused on anomaly detection in sensor systems data are [7, 8, 9]. Briefly, [7, 8] view measurements collected by a sensor system as coming from the same (unknown) distribution and "pre-defines" anomalies as outliers. The main focus of that effort, which is an *off-line* approach, is on minimizing communication overhead (in transmitting data needed for anomaly detection) and corresponding energy consumption. In contrast, we focus on an *online* approach that, on-the-fly, builds an adaptive model of "normal" data and does *not* a priori define what is an anomaly. For instance, the approach in [7, 8] might only flag the most extreme measurement in Figure 1(a) as an anomaly, whereas our approach would flag the entire event (outlined by the dashed rectangle) as an anomaly We give a more detailed description of [7, 8] and a quantitative comparison in Section 6. In [9] a change point detection based approach is used for detecting distribution changes (e.g., mean, variance, covariances) in sensor measurements. However, (a) this approach assumes knowledge of the (time varying) probability distribution from which sensor measurements are sampled (information often not available in real-world deployments), and (b) such techniques do not meet (at least in their current form) our efficiency criteria (see Section 6).



(a) Long duration anomaly (b) Piecewise linear model for time series data

Figure 1: (a) Data set with long duration anomaly and (b) example of a piecewise linear model.

In this work, we formulate the problem of anomaly detection in sensor systems as an instance of identifying unusual patterns in time series data problem. Of course, one possible direction would then be to construct a timeseries-based approach, e.g., based on [6]. However, we also did not find this direction to be effective as such techniques are (typically) not well-suited for detecting long duration anomalies. So, we do not pursue this direction further here, but in Section 6, we do illustrate quantitative results corresponding to applying a representative timeseries-based approach to data collected by real sensor systems deployments and provide intuition for why such a technique did not yield good results.

In contrast, the basic idea behind our approach is to compare the collected measurements against a reference time series. But, to do this efficiently and robustly, the following challenging problems need to be solved: (1) How do define a reference time series?; (2) How to compare two time series efficiently?; (3) What metric to use in deciding whether two sensor data time series are similar or different?; and (4) How to update the reference time series, to adapt to (normal) changes in sensor data patterns?

We address these challenges by proposing and evaluating an anomaly detection algorithm, termed Segmented Sequence Analysis (SSA) that exhibits the desirable characteristics stated above. Briefly, SSA leverages temporal and spatial correlations in sensor measurements and constructs a piecewise linear model of sensor data time series. This is motivated by [10] which focused on searching for *known patterns* in time series (see Section 6). To detect anomalies, we compare the piecewise linear models of sensor data (collected during a time interval) and a reference model, with significant differences (as determined by a proposed similarity metric) flagged as anomalies. We use data from real-world deployments to evaluate our approach and demonstrate its accuracy, robustness, and efficiency. In summary, our the main contributions are:

- We propose an approach to anomaly detection in sensor systems that is able to detect anomalies accurately and in an *online* manner (Section 2).

- We perform an extensive study using data sets from two real deployments, one consisting of about 30, 000 environmental temperature measurements collected by 23 sensor nodes for around 43 days, and the other consisting of more than 10, 000 soil temperature, moisture, and air humidity measurements collected by 3 sensor nodes for over 5 months. This study illustrates that our approach is accurate (it detects over 90% of the anomalies with few false positives), works well over a range of parameter choices, and has a small (CPU, memory) footprint (Sections 3 and 4).
- We show that our (online) SSA-based based approach is more accurate than potential other (offline) techniques[1], which are more computationally intensive (Section 5 and 6).

2. Methodology

In this section, we first describe a tiered sensor system architecture that is representative of data collection deployments. We then formulate the problem of anomaly detection in sensor readings as an instance of the problem of identifying unusual patterns in time series data. Lastly, we describe our method for detecting anomalous sensor readings.

2.1. Sensor systems for data collection

We consider a typical tiered sensor system [11] consisting of two tiers: a lower-tier of resource-constrained battery-operated wireless *motes* with one or more attached sensors (e.g., temperature, humidity, acceleration), and an upper tier of more capable *master* nodes each of which has significantly higher computation, storage, and communication capabilities than the motes. Here, we are interested in the class of data collection sensor systems, where each mote (usually) collects periodic sensor data, possibly performs some local processing on the data, and then transfers the resulting data over multiple hops. We model the measurements collected by a sensor m as a time series $D_m[t], t = 1, 2, \ldots$. For example, suppose a sensing system had 20 motes, each collecting data from 3 sensors. Then, we would have a total of 60 time series (3 from each of the 20 motes), and we would represent these as a set $\{D_m[t], m = 1, 2, \ldots, 60; t = 1, 2, \ldots\}$.

In many data collection applications, these time series exhibit a high degree of temporal and spatial correlations due to the nature of the physical phenomenon being monitored (e.g., temperature or light conditions). We leverage such correlations to detect anomalies (interesting events) in the sensor data time series. As noted in Section 1, anomalies have various lengths, magnitudes, and patterns, and a good anomaly detection methodology should be robust to such variations.

We first describe the building blocks of our approach, where the basis involves building (and continuously updating) a model of the "normal" and then determining how similar new sensor measurements are to the "normal". We then describe our approach to anomaly detection.

2.2. Building blocks

At a high level, our approach answers the following question: *How similar is a time series of sensor measurements to a given "reference" time series?.* Suppose we are given two time series, $D^{new}[t]$ and $D^{ref}[t]$, where $D^{new}[t]$ is the time series of new sensor data, and $D^{ref}[t]$

[1]Most of these were designed in other contexts, but constitute possible directions that could have been taken for sensor systems anomaly detection.

is the reference time series. Then, an anomaly detection method can: (1) Construct models corresponding to $D^{new}[t]$ and $D^{ref}[t]$; (2) Compare these two models using a similarity measure; and (3) If the model for $D^{new}[t]$ is not sufficiently similar to the model for $D^{ref}[t]$, conclude that there are anomalies in the time series $D^{new}[t]$. Thus, our method involves solving three main problems: (1) how to construct the models for $D^{new}[t]$ and $D^{ref}[t]$, (2) which similarity measure to use for comparing these models, and (3) how to decide whether the models for two different time series data are sufficiently similar, given our similarity measure.

Piecewise linear model. We use a piecewise linear model to represent $D^{new}[t]$ and $D^{ref}[t]$. Figure 1(b) depicts an example piecewise linear representation of sensor measurements collected by the SensorScope deployment [12]. Each line segment represents a small subset of sensor readings, determined using linear least-squares regression. The advantages of a piecewise linear representation of time series data are: (a) It is *succinct*, since only a few line segments are needed to represent a large amount of time series data; (b) It is *representative* as essential information (e.g., significant patterns) in the data is captured; (c) It is *robust* to changes in model parameters as well as to faults and noise in sensor measurements (as demonstrated in Section 4).

A succinct, representative, and robust piecewise linear model of sensor data time series is desirable for *online* anomaly detection. First, we can compute such a model in near real-time (Section 2.3). Second, it enables us to create a *data driven* reference model that is easy to update – hence, we do not need prior knowledge about the types of anomalies that sensor data might contain. Third, because it is succinct, it enables us to compare two different time series efficiently and transmit models with low overhead. Finally, because it is representative of the sensor data *patterns*, it enables accurate detection of anomalous patterns.

Due to their usefulness in modeling time series data, linearization based approaches have also been used in other contexts. For example, [10] developed an efficient technique to search for occurrences of a *known pattern* within a time series. However, the problem of searching for a known pattern in time series data is different from anomaly detection because often we do not have any prior information about the patterns exhibited by anomalous sensor readings.

Linearization Error. In order to compute a piecewise linear model, we need to define the *linearization error* between a sensor data point j and the line segment l covering it. We define this error as the perpendicular distance between the point j and the line l. Accordingly, we define the linearization error ϵ for a piecewise linear model representing a time series $\{D[t], t = 1, 2, 3..., n\}$, as the maximum linearization error across all the data points in $D[t]$.

How many line segments to use? We also need to determine the number of line segments, k, to use. Intuitively, using a large number of line segments will result in a small linearization error – as explained below, this leads to lower computation cost but larger communication cost. (This tradeoff is explored in detail in Section 4.2.)

We automatically determine the number of line segments in our piecewise linear model based on the maximum allowed linearization error ϵ, which is a (free) parameter in our approach. For a fixed choice of maximum linearization error ϵ, we use a *greedy* approach to determine the number of line segments needed to represent a time series. We start with the first two data points of the time series and fit a line segment, (say) l_1, to them. Then we consider the data points one at a time and recompute l_1 using linear least-squares regression to cover a new data point. We compute the distance of the new data point from the line l_1. If this distance is greater than ϵ, then we start a new line segment, l_2 such that the distance between the new data point and l_2 is at most ϵ. We keep repeating this process until we exhaust all data points. Note that our approach is suited for both *offline* and *online* processing. In an *online* setting, whenever the sensor collects a new reading, we can either recompute the current line segment to cover it or start a new line segment (depending on the linearization error).

We represent the k line segments that constitute a piecewise linear model of a time series using their end points $\{(X[i], Y[i]), i = 1, 2, \ldots, k\}$, where $X[i]$ denotes a sample number (or the time at which a sample was collected). The corresponding $Y[i]$ is one of the end points of a line segment and represents an estimate of the actual sensor reading collected at time $X[i]$. For example, in Figure 1(b), the line segments approximate actual sensor readings (shown using dots) – here we indicate two measurement collection times, $X[j]$ and $X[j+1]$ that correspond to two end points, $Y[j]$ and $Y[j+1]$, that are part of a piecewise linear model.

Similarity measure. Let $\{(\hat{X}[i], \hat{Y}[i]), i = 1, 2, \ldots, \hat{k}\}$ and $\{(\tilde{X}[i], \tilde{Y}[i]), i = 1, 2, \ldots, \tilde{k}\}$ denote the piecewise linear representation of two time series $\hat{D}[t]$ and $\tilde{D}[t]$, respectively. In order to define a similarity measure between any two piecewise linear representations, we need to first align them so that their $X[i]$ values (end points on the x-axis) line up. For example, consider two representations $\{(\hat{X}[i], \hat{Y}[i]), i = 1, 2\}$ and $\{(\tilde{X}[i], \tilde{Y}[i]), i = 1, 2, 3\}$ such that $\tilde{X}[1] = \hat{X}[1]$ and $\tilde{X}[3] = \hat{X}[2]$, and hence, $\tilde{X}[2] < \hat{X}[2]$. In order to align the two representations, we choose the X values as $\{X[1] = \hat{X}[1] = \tilde{X}[1], X[2] = \tilde{X}[2], X[3] = \hat{X}[2] = \tilde{X}[3]\}$. Hence, after alignment, the new representations are $\{(X[i], \hat{Y}[i]), i = 1, 2, 3\}$, and $\{(X[i], Y[i]), i = 1, 2, 3\}$, where $Y[1] = \hat{Y}[1]$, $Y[3] = \hat{Y}[2]$ and the $Y[2]$ value (corresponding to the sample at time $X[2]$) is computed using the equation of the line segment joining $Y[1]$ and $Y[3]$.

We define the *difference* between the (aligned) piecewise linear representations of two time series $\hat{D}[t]$ and $\tilde{D}[t]$ as:

$$S(\hat{D}, \tilde{D}) = \frac{1}{k} \sum_{i=1}^{k} |Y[i] - \tilde{Y}[i]| \tag{1}$$

Here, $S(\hat{D}, \tilde{D})$ represents the average difference between the Y values of the piecewise linear representations of $\hat{D}[t]$ and $\tilde{D}[t]$ over the k line segments. We chose this metric because it is efficient to compute, and it indirectly captures the difference between the two time series.

Threshold computation. We set the threshold γ (for deciding whether $S(\hat{D}, \tilde{D})$ is sufficiently large) to the standard deviation of the *initial* $D^{ref}[t]$. We remove any CONSTANT anomalies (described in Section 3), before computing the standard deviation - intuitively such measurements are not a good indication of variability in sensor data as they typically correspond to faulty data, e.g., due to low voltage supply to the sensor [5]. Intuitively, the standard deviation is a reasonable indication of the variability in the “normal” data. A multiple of standard deviation could also be used, but our more conservative approach already results (Section 3) in a reasonably low false positive rate; more sophisticated (than threshold-based) approaches are part of future efforts.

Putting it all together. Given a time series of new sensor data, $D^{new}[t]$, and a reference time series, $D^{ref}[t]$, our Segmented Sequence Analysis (SSA) based approach to anomaly detection utilizes the following steps (all detailed above):

1. **Linearization**: We apply our linearization technique to obtain the two piecewise linear models $\{(X^{new}[i], Y^{new}[i])\}$ and $\{(X^{ref}[i], Y^{ref}[i])\}$.
2. **Alignment**: We align the two linear representations so that they have the same X values.
3. **Similarity computation**: We compute the similarity, $S(D^{new}, D^{ref})$, between the reference model and the model for new sensor data using Equation (1).

4. **Anomaly detection**: We detect an anomaly using a simple threshold-based approach. Specifically, if $S(D^{new}, D^{ref})$ is greater than a threshold γ, then we conclude that the sensor readings $D^{new}[t]$ contain an anomaly.

We now describe in detail our SSA-based anomaly detection framework.

2.3. *Using* SSA *on a tiered sensor network*

We perform anomaly detection in a tiered sensor network in two stages – (1) a *local* step, executed at each mote, followed by (2) an *aggregation* step, executed at the master nodes. In the local step we exploit temporal correlations (in sensor readings), and in the aggregation step we exploit spatial correlations, as described next.

Local step. During the local phase (executed at individual motes), each mote m performs the following tasks: (1) construct or update a reference time series, $D_m^{ref}[t]$, for its sensor readings, (2) collect new sensor readings $\{D_m^{new}[t], t = 1, 2, \ldots, T\}$ over a period T, (3) construct or update linear models for $D_m^{new}[t]$ and $D_m^{ref}[t]$, and (4) perform anomaly detection using the SSA-based method (refer to Section 2.2).

Reference time series. To construct a reference time series at mote m, $D_m^{ref}[t]$, we use the following approach. For physical phenomena such as ambient temperature or humidity variations that exhibit a diurnal pattern, we initially start with a time series $D[t]$ consisting of measurements collected over a period of 24 hours, (say) on day 1 of the deployment. Let $D^{new}[t]$ be the new sensor readings collected by mote m over time period T corresponding to (say) 9-9:30 a.m. on day 2 of the deployment. For these new readings, we define the data points in $D[t]$ that were collected between 9-9:30 a.m. (on day 1) as $D^{ref}[t]$. We first look for anomalies in the new sensor readings $D^{new}[t]$, and then use the data points in $D^{new}[t]$ to update $D^{ref}[t]$ using weighted averaging. For example, we can use exponential weighted moving averaging to (pointwise) update $D^{ref}[t]$, i.e., $\tilde{D}^{ref}[t] = (1-\alpha) \times D^{ref}[t] + \alpha \times D^{new}[t]$, where $\tilde{D}^{ref}[t]$ denotes the updated reference time series.

Figure 2(a) depicts the time series of humidity readings collected by a sensor from the Jug Bay deployment [13] along with two reference time series for it, constructed using $T = 12$ hours (36 data points with one data point collected every 20 minutes). The reference time series labeled “Reference time series (including anomalous readings)” is computed using both non-anomalous as well as anomalous readings in $D^{new}[t]$ to update the reference time series, while the “Reference time series (excluding anomalous readings)” excludes the anomalous readings in $D^{new}[t]$. The humidity measurements contain two anomalies – sharp changes in the sensor reading (marked by bounding rectangles in Figure 2(a)) which cause the humidity readings to increase sharply and then decay over time. It is important to detect these sharp changes in sensor readings.

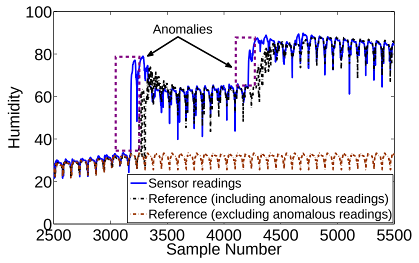
As shown in Figure 2(a), excluding the anomalous readings in $D^{new}[t]$ when updating the reference time series causes $D^{ref}[t]$ to diverge from the sensor data time series. A diverging $D^{ref}[t]$ is likely to result in an increase in false positives (due to lots of samples being flagged as anomalies) and failure to “zoom in” on the samples where sharp changes in sensor readings occur. If we include the anomalous readings in $D^{new}[t]$ for updating of the reference time series, then the reference time series exhibits the same patterns as $D^{new}[t]$ but with a *time lag*. Our evaluation results in Section 4 show that this lag is long enough for SSA to identify the anomalous readings. There is a potential downside in using anomalous readings in updating $D^{ref}[t]$. If an anomaly affects a large number of samples, then SSA will fail to detect many of them. We discuss this in detail in Section 4 and show that for long duration anomalies, SSA can identify anomalous samples that correspond to the start and end of these anomalies, which is also quite useful.

For scenarios where the "normal" pattern of sensor readings might not be known or might not exhibit any periodicity – e.g., sensors deployed for monitoring of birds' nests [11], in the absence of any domain expertise, we assume that the sensor readings collected over a large duration (a 24 hour period in most cases) capture the normal patterns in the sensor data time series, and start with such a time series as our reference. Clearly, the performance of our local anomaly detection step depends on the quality of the reference data. A reference data that does not capture the normal sensor readings or is corrupted by anomalies can lead to false positives and/or false negatives. In Section 4, using real-world sensor readings for periodic (e.g., ambient temperature) as well as aperiodic (e.g., soil moisture variations) phenomena, we show that our approach for selecting and updating $D_m^{ref}[t]$ is robust and works well in practice.

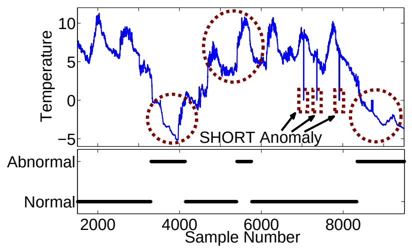
Aggregation step. After performing its local step, each mote m sends its linear model, $\{(X_m^{new}[i], Y_m^{new}[i]), i = 1, .., k\}$, for the new sensor readings, $D_m^{new}[t]$, and the results of its local anomaly detection step to its master node. For each mote m, the master node performs another round of anomaly detection by comparing its linear model against the models from other motes (treating them as reference). Hence, a master node managing n slave motes performs $O(n^2)$ model comparisons. The design of our aggregation step is based on the observations from several real-world deployments that often the readings from sensors deployed at different locations are correlated [12]. The aggregation step exploits these spatial correlations to detect additional anomalies (if any) that might not have been detected during the local step.

The final set of anomalies is the union of the anomalies detected during the local and aggregation steps. In our current framework, the master node does not provide feedback to its slave motes. Hence, the anomalous readings from mote m detected only by the aggregation step are currently not leveraged to improve the accuracy of the local anomaly detection step. Incorporating a feedback mechanism between the aggregation and local steps is part of future efforts.

Online fault detection. To achieve online detection, we run the local and aggregation anomaly detection steps periodically, every T minutes. For example, if $T = 30$ min, we first collect new sensor readings for half an hour and then perform anomaly detection using the framework described above. The anomaly detection interval, T, controls the trade-off between real-time anomaly detection and resource consumption, as discussed in detail in Section 4.2.



(a) Reference time series



(b) Short anomalies (marked by rectangles)

Figure 2: Examples of Anomalies in Sensor Readings

2.4. *Hybrid approach*

As noted in Section 2.2, our piecewise linear representation is very succinct – in practice, a small number of line segments is sufficient to capture the essential information (diurnal patterns, trends lasting for a long duration, etc.) in a time series. However, because it is designed to capture significant trends, a piecewise linear representation will mask faults or anomalies that affect a very small number of sensor samples. The top plot in Figure 2(b) shows a temperature reading time series from the SensorScope datasets [12], and the bottom plot shows whether each sensor reading was identified as "normal" or "anomalous" by SSA. While SSA is able to detect instances of long duration anomalies (marked by circles) it fails to detect the three very short duration anomalies (marked by rectangles in the top plot). To improve the accuracy of SSA on short duration anomalies, next we propose a *hybrid* approach.

Combining SSA with Rule-based methods. We can view data faults in sensor readings as short duration anomalies (refer to Section 6). Thus, it is reasonable to adapt techniques designed for fault detection for identification of short duration anomalies. Specifically, [14, 6] are representative of such techniques and they consider: SHORT anomalies (a sharp change in the measured sensor readings between two successive samples), NOISE anomalies (increase in the variance of sensor readings) and CONSTANT or "Stuck-at" anomalies (the sensor reports a constant value). Thus, we use the Rule-based methods [6] (originally designed for fault detection), for detection of short range anomalies in our hybrid approach by adding the following rules.

SHORT Rule: To detect SHORT anomalies in the time series $\{D[t], t = 1, 2, 3...\}$, we keep track of the change in sensor readings between two successive samples, $|D[t] - D[t-1]|$. If this value is larger than a threshold σ_s, then we flag $D[t]$ as anomalous.

CONSTANT Rule: To detect CONSTANT anomalies we calculate moving variance statistics of time series $\{D[t], t = 1, 2, 3...\}$. Let $V[t] = variance(\{D[j]\}_{j=t-c+1}^{j=t})$ be the variance of c consecutive data readings prior to time t. If $V[t]$ is less than a threshold σ_c, then we flag the set of samples $\{D[j]\}_{j=t-c+1}^{j=t}$ as anomalous.

A rule-based method also exists for detecting NOISE data faults. But, as shown in Section 4, SSA is accurate at detecting NOISE faults anomalies; thus, we do not include the NOISE rule as part of our hybrid method.

To automatically determine the detection thresholds, σ_s and σ_c, we use the histogram-based approach [6]. We plot the histogram of the change in sensor readings between two successive samples (for SHORT rule) or the variance of c samples (for CONSTANT rule) and select one of the modes of the histogram as the threshold.

Thus, in scenarios where both short and long duration anomalies are expected, we propose a *hybrid* approach for anomaly detection. Specifically, every T minutes, we use the Rule-based methods to detect and mark short duration anomalies, and then use SSA to detect the remaining anomalies[2]. We evaluate our hybrid approach using real-world datasets in Section 4 and show that it is effective at detecting short and long duration anomalies. Our evaluation also shows that Rule-based methods boost the efficacy of SSA only in situations where we are interested in detecting short duration anomalies along with interesting long duration events or anomalies (e.g., changes in sensor readings patterns). Hence, in situations where detecting short duration anomalies is not of interest, the additional complexity of using Rule-based methods is not needed. Note that we do not need to remove short duration anomalies (or data faults) from the data – e.g., by replacing the sensor readings $D[j]$ corrupted by a SHORT anomaly with the average of its adjacent samples $D[j-1]$ and $D[j+1]$ – in order for SSA to be able to detect long duration

[2]It is possible to combine other detection methods with SSA to design variants of our hybrid approach, e.g., [6] proposes other techniques, such as HMM-based methods, for detecting sensor data faults. However, other methods in [6] are much more computationally intensive and require a much longer training phase (than our reference model).

anomalies. Our evaluation results in Section 4 show that the presence of short duration anomalies does not impact the accuracy of SSA when it comes to detecting long duration anomalies.

Complexity and Overhead. Of all the steps in SSA, linearization requires the most computation, with the worst case complexity being $O(n^2)$, where n is the number of measurements accumulated in a time slot of length T. Since we use linear least-squares regression to determine the best-fit line segment, the cost of approximating d (one dimensional) data points with a line segment is $O(d)$. However, our greedy approach performs a least-squares regression fit every time a new sensor sample is recorded. In the worst case, we may need to perform least-squares regression n times (once for each data point) resulting in $O(n^2)$ computational complexity for the linearization step, and hence, for SSA. In practice, SSA is quite efficient, as shown in Section 4 (as n is typically not very large). We note that the Rule-based methods used in our hybrid approach are simple and have $O(n)$ computational complexity; thus, they do not increase the overall complexity of the hybrid approach.

SSA incurs a communication overhead every time a mote conveys its linear model to its master node. Note that a mote needs to convey 4 data points per line segment – two $X[i]$ values (sample times) and the corresponding two $Y[i]$ values. Since a mote's linear model consists of k line segments, the communication overhead is $O(k)$. Note that this overhead is incurred every T minutes since a mote recomputes its linear model once every T minutes.

3. Experimental setup

Sensor datasets. The sensor data time series used in our evaluations come from the SensorScope [12] and the Life Under Your Feet [13] projects. Both projects represent the state-of-the-art in sensor systems and collect measurements in very different environments. Hence, the two datasets allow us to evaluate SSA on representative and diverse sensor system data.

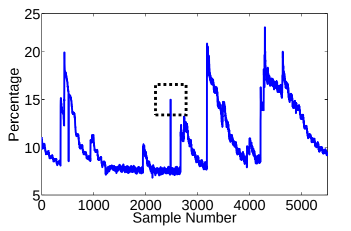
In the SensorScope project, large networks of sensors are deployed to collect environmental data such as temperature, humidity, solar radiation, etc. In this paper, we use temperature readings collected from 23 sensors deployed in the Grand St. Bernard pass between Switzerland and Italy in 2007. Each sensor collected samples every 2 minutes for 43 days. Since the temperature measurements exhibit a diurnal pattern, the sensor data time series are periodic with the period being 720 data points (collected every 24 hours). In what follows, we show results for all 23 sensor data time series. We refer to these time series as SensorScope 1 through SensorScope 23.

Our second sensor data source is from the Life Under Your Feet project [13], which studies soil ecology in a number of locations. We use data sets collected at the Jug Bay Wetland Sanctuary in Anne Arundel County, Maryland between June, 2007 and April, 2008. In this deployment, sensors were placed in the nests of Box Turtles to study the effect of soil temperature and soil moisture on the incubation of turtle eggs. Measurements of soil temperature, soil moisture, box temperature, and box humidity are collected every 20 minutes for more than 5 months. These measurements exhibit very diverse patterns. For example, as depicted in Figure 3(a), the soil moisture data are non-periodic – here the soil moisture readings are close to 8% when it is not raining, but they exhibit a sharp jump followed by a gradual decay when it rains. Hence, for the soil moisture time series, instances of rainfall are the anomalies (or events) of interest that we try to detect using SSA. In contrast, the box humidity data sets are periodic with a period of 72 data points (or 24 hours). The Jug Bay dataset consists of readings from 3 different sensors. In what follows, we show results for soil moisture readings collected (we refer to them as Soil Moisture

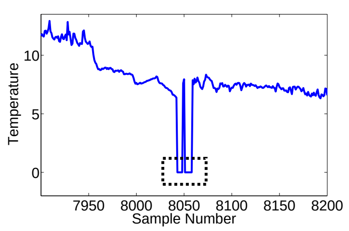
Anomaly	Description	Duration
Change in mean	Anomalous readings differ significantly from average value of normal readings (e.g., as in Figure 1(a))	Long
Change in variance	Anomalous readings exhibit less or more variability than normal readings (e.g., as in Figure 3(c))	Long & Short
Short spike	Equivalent to SHORT fault data type [5] (e.g., as in Figure 3(a))	Short
Constant reading	Sensor reports a constant value over a period of time (e.g., as in Figure 3(b))	Long & Short
Change in shape	Anomalous readings differ in mean and/or variance from normal readings but with shorter duration than *Change in mean* and *Change in variance* (e.g., as in Figure 3(d))	Long & Short

Table 1: Anomaly categories

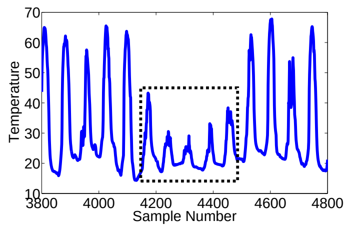
1, Soil Moisture 2, and Soil Moisture 3), as well as the box humidity data time series (we refer to them as Box Humidity 1, Box Humidity 2, and Box Humidity 3).



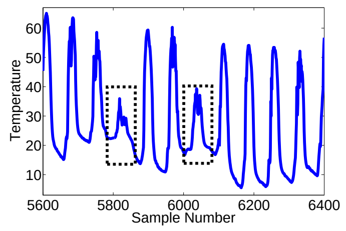
(a) Soil Moisture Readings with Short anomalies



(b) Anomaly: Constant



(c) Anomaly: Change in Variance



(d) Anomaly: Change in Shape

Figure 3: Soil Moisture Readings, Constant, Change in Variance and Change in Shape anomalies

Anomalies in the data sets. To the best of our knowledge, there are no publicly available data sets with anomalies already annotated. Thus, to obtain the ground truth, we visually inspected the sensor data time series from the SensorScope and the Jug Bay deployments, to identify long and short duration anomalies. This is consistent with current practice for other data sets (e.g., Internet traces in [1]) that lack ground truth. To identify long duration anomalies, we used the (subjective) criterion of "what kind of patterns would a human find interesting?". The short duration anomalies that we identified resemble sensor data faults types (SHORT, NOISE, and CONSTANT faults) described in [5, 6]. We categorize these identified anomalies into five groups shown in Table 1. We note that this categorization is done for ease of results presentation (in Section 4) *only* and is *no way used* in our anomaly detection approach.

4. Experimental Results

We now evaluate our SSA-based approach and illustrate its goodness using the following criteria (a comparison with related literature is presented in Section 6).

- *Accuracy*: SSA alone detects most long duration anomalies (plus a significant fraction of the short duration ones), and our hybrid approach detects both, long and short duration anomalies accurately.
- *Sensitivity*: Our results are not sensitive to SSA's parameter, namely to the settings of the linearization period T, and the maximum linearization error ϵ.
- *Cost*: SSA has low computation and memory cost, and hence it can be effectively implemented in sensor systems.
- *Robustness*: SSA is robust to the presence of sensor data faults in the reference time series (i.e., there is no need to "clean" the data before running SSA).

4.1. Accuracy evaluation

We first describe our method's accuracy, using the data sets and the ground truth identification described in Section 3. We use (1) number of false positives (detecting non-exist anomalies), and (2) number of false negatives (not being able to detect an anomaly) as our metrics. Specifically, the results in the tables below are presented as follows - the x/y number indicates that x out of y anomalies were detected correctly (corresponding to $y - x$ false negatives) plus we also indicate the number of corresponding false positives (FP). Note that a long duration anomaly may consist of many consecutive data points. In this paper, we focus on detecting these events rather than on identifying each and every anomalous data point within an event. (Thus, when 50% or more of the data points of a long duration anomaly are identified by SSA as anomalous, we consider this to be a successful detection[3].)

The accuracy results of our hybrid approach on all data sets are given in Tables 2 and 3. Our hybrid method is able to detect long duration and short duration anomalies, with a small number of false positives, and often without any false negatives. Most of the false positives are due to the Rule-based part of the hybrid approach rather than to the SSA part (as explained below).

Tables 2 and 3 also show that long duration anomalies – particularly the *Change in Mean* and *Change in Shape* anomalies – occur quite often in the SensorScope and the Jug Bay datasets (refer to the last row of both tables). For example, over the course of 43 days, a total of 84 instances of *Change in Mean* and 139 instances *Change in Shape* anomalies occurred in the SensorScope datasets; on average, 2 instances of *Change in Mean* and 3 instances of *Change in Shape* anomalies per day. Previously, others have shown that short duration anomalies or data faults (Short spikes, Constant readings, Noise faults) are quite prevalent in real-world datasets [5, 6]; this work is the first to demonstrate that long duration anomalies occur quite often as well.

Under our hybrid approach, anomalies can be detected at three different stages – the Rule-based methods, the local step in SSA, and the aggregator step in SSA. For both the SensorScope and the Jug Bay datasets, we found that the aggregator step in SSA did not add significantly to the accuracy of SSA. This is because the combination of the Rule-based methods and the

[3]Of course, more sophisticated approaches are possible, but as our results indicate, this simple approach already results in good accuracy and allows us to focus on evaluation of SSA, without additional complications.

local step in SSA was able to detect most of the anomalies. We now focus on understanding the contribution to our hybrid approache's accuracy of SSA vs. the Rule-based methods. The first two rows of Table 4 show the results of applying SSA alone (without the use of Rule-based methods) on the Soil Moisture 1 and the SensorScope 1 time series. Based on these results, we make the following observations: (1) SSA is accurate at detecting long duration anomalies such as *Change in Average*, *Change in Variance*, and *Change in Shape*, and (2) SSA can fail to detect short duration anomalies such as *Short spikes*. For example, while it is able to detect more than 70% of the Short spikes in Soil Moisture 1, it detects only about 50% of the Short spikes in SensorScope 1. This makes sense, as SSA is intended more for longer duration anomalies.

The utility of the hybrid approach can be seen, e.g., by comparing the *Short* results for Soil Moisture 1 and SensorScope 1 in Tables 2 and 3 with those in Table 4. The hybrid approach outperforms SSA on short duration anomalies as it uses Rule-based methods, designed specifically for short duration anomalies like *Short spikes* and *Constant readings*. However, our hybrid approach incurred a higher false positive rate than SSA, and a detailed inspection of the samples falsely identified as anomalous revealed that this increase was due to the Rule-based methods. Prior work [6] showed that Rule-based methods can incur a high false positive rate mainly because the histogram method for determining their fault detection threshold (Section 2.4) does not always identify a good threshold. We also verified this by computing the histogram using the entire data set, which significantly reduced the false positive rate. However, such an approach would not be *online* and hence not used here.

We also verified that the Rule-based methods alone do not provide any benefits in detecting long duration anomalies. For instance, this can be seen by comparing the results for Soil Moisture 1 and the SensorScope 1 data in Tables 2 and 3 with those in Table 4, where our hybrid method performs the same as SSA w.r.t. to detecting long duration anomalies like *Change in Average* and *Change in Shape*. In fact, the Rule-based methods can perform quite poorly when used for identifying long duration anomalies. The last row of Table 4 shows the results of applying the Rule-based methods *alone* on the Box Humidity 2 data - compare that to the Box Humidity 2 results in Table 3. As expected, the Rule-based methods detect the short duration anomalies (*Short* spikes and *Constant* readings), but fail to detect most of the long duration anomalies.

4.2. Sensitivity evaluation

The linearization period T and the maximum linearization error ϵ are the two main parameters in our SSA-based approach. Next, we present an analysis of the sensitivity of our results to these parameters' settings.

Impact of T. SSA computes the similarity measure $S(\hat{D}, \tilde{D})$ every T time units. The smaller the value of T, the more real-time is SSA's anomaly detection. However, if T is too small, there may be too few data points for SSA to accurately capture the pattern of a long duration anomaly. Thus, T controls the trade-off between (near) real-time anomaly detection and the accuracy of detecting long duration anomalies.

To characterize the sensitivity of SSA's accuracy to T, we ran SSA using different values of T. For SensorScope datasets, we used T values ranging from 30 minutes (the time to collect 15 data points) to 8 hours (the time to collect 240 data points). For Jug Bay datasets, we varied T from 2 hours (the time to collect 6 data points) to 24 hours (the time to collect 72 data points).

We found that changing T's value did not affect SSA's accuracy w.r.t. *Change in Average* and *Change in Variance* anomalies, but it did affect the accuracy w.r.t. *Change in Shape* anomalies. We show examples of SSA's performance in detecting instances of the *Change in Shape* anomaly

Data Set	Change in Mean	Change in Var	Change in Shape	Short	Constant	False Positives
SensorScope 1	3/4	0/0	6/7	90/90	2/2	7
SensorScope 2	8/8	0/0	6/7	86/86	6/6	6
SensorScope 3	7/7	2/2	9/10	64/64	5/5	12
SensorScope 4	5/5	2/2	12/13	220/222	13/13	27
SensorScope 5	6/6	4/4	9/9	726/819	34/34	0
SensorScope 6	7/7	0/0	8/10	206/206	1/1	7
SensorScope 7	8/8	4/4	12/12	555/567	54/54	0
SensorScope 8	6/6	0/0	9/10	243/243	2/2	4
SensorScope 9	6/6	2/2	11/12	65/65	23/23	6
SensorScope 10	5/5	0/0	10/12	46/46	2/2	3
SensorScope 11	7/7	0/0	8/10	122/122	1/1	6
SensorScope 12	7/7	0/0	11/13	84/84	13/13	7
SensorScope 13	8/8	2/2	13/14	250/250	15/15	5
SensorScope 14	5/5	4/4	5/7	595/633	26/26	19
SensorScope 15	6/6	2/2	8/9	464/475	24/24	12
SensorScope 16	4/4	2/2	7/7	120/120	12/12	25
SensorScope 17	5/5	4/4	6/6	166/166	17/17	13
SensorScope 18	6/7	2/2	16/18	56/56	9/9	12
SensorScope 19	3/3	0/0	4/6	98/98	1/1	15
SensorScope 20	3/3	0/0	3/4	77/78	1/1	9
SensorScope 21	2/2	1/1	3/4	332/337	26/26	5
SensorScope 22	3/3	0/0	3/5	88/88	1/1	11
SensorScope 23	3/3	0/0	4/4	84/84	2/2	17
Total	121/123	31/31	183/209	4837/4999	290/290	238

Table 2: Hybrid method: SensorScope

Data Set	Change in Mean	Change in Var	Change in Shape	Short	Constant	False Positives
Soil Moisture 1	7/8	1/1	8/10	53/55	1/1	0
Soil Moisture 2	8/9	1/1	9/11	74/74	1/1	2
Soil Moisture 3	5/5	0/0	6/7	42/42	0/0	4
Box Humidity 1	2/2	4/4	18/19	15/15	0/0	2
Box Humidity 2	5/5	7/7	27/28	16/16	2/2	2
Box Humidity 3	3/3	2/2	1/1	17/17	2/2	3
Total	30/32	15/15	69/76	217/219	6/6	13

Table 3: Hybrid method: Jug Bay

in SensorScope 2 and Box Humidity 1 time series (for different T values) in Tables 5(a) and 5(b), respectively. Very small T values (corresponding to few data points in a linearization period) result in a significant number of false positives. As T grows, the number of false positives improves and becomes reasonably insensitive to T. The false negative rate is quite insensitive to the value of T, with a small increase for very large values of T. Intuitively, this can be explained as follows. For a small value of T, SSA considers only a few data points at a time and even small differences in these data points (e.g., due to measurement noise) can cause SSA to misclassify these points as an instance of *Change in Shape* anomaly resulting in an increase in the false positive rate.[4] The small increase in false negatives for large values of T is due to very short duration *Change in Shape* anomalies being "averaged out" (with a large T).

[4]The *Change in Average* and the *Change in Variance* anomalies occur over longer periods of time; thus, to cause a

Method	Data Set	Change in Mean	Change in Var	Change in Shape	Short	Constant	FP
SSA	Soil Moisture 1	7/8	1/1	8/10	40/55	1/1	0
SSA	SensorScope 1	3/4	0/0	6/7	46/90	1/2	1
Rule based	Box Humidity 2	2/5	0/7	5/28	16/16	2/2	2

Table 4: SSA vs. Rule-based methods

T Value	0.5	1	2	4	8
Detected	7/7	7/7	6/7	6/7	6/7
FP	8	4	0	0	0

(a) SensorScope 2

T Value	2	4	6	8	12	24
Detected	19/19	19/19	19/19	19/19	18/19	18/19
FP	17	10	6	4	2	2

(b) Box Humidity 1

Table 5: Sensitivity test for T: SSA with different T values; Change of shape anomaly

In summary, our evaluation shows that our method is not sensitive to the linearization period T, provided it is long enough to collect a reasonable number of data points. The main reason for this is that beyond a certain value of T, our similarity metric $S(D^{new}, D^{ref})$ does not change significantly with T, as illustrated next.

For a fixed T value, we ran SSA on SensorScope 1 and SensorScope 2 separately, and recorded the similarity values (w.r.t. to the reference time series) computed every T time units. For example, for $T = 1$ hour, SSA computes a similarity value for new data points collected every hour using Equation (1). We then computed the mean and the variance of the differences in the similarity values for different values of T for SensorScope 1 (and separately for SensorScope 2). For example, consider a set of SensorScope 1 data points collected over 2 hours. Let α_2 be the similarity value for these data points when $T = 2$ hours, and for $T = 1$ hour, let α_{11} and α_{12} be the similarity values for data points collected during the first and the second hour, respectively. The mean and variance of the differences in the similarity values for $T = 1$ hour and $T = 2$ hours are computed using the values $|\alpha_2 - \alpha_{11}|$ and $|\alpha_2 - \alpha_{12}|$. These values capture the difference in the similarity values associated with a set of data points for different T values.

Table 6(a) shows the results, where $|S_q[t] - S_r[t]|$ is the difference in similarity values corresponding to $T = q$ hrs and $T = r$ hrs and the (x, y) entry is the corresponding mean and variance of that difference. The similarity values for $T \geq 2$ are close, but the similarity values for $T = 1$ hr are different from those for $T = 2$ hrs. Recall that SSA compares similarity values against a threshold to detect anomalies. Hence (for SensorScope 1 and SensorScope 2), SSA's performance with $T = 1$ hr differs from its performance with $T = 2$ hrs; but for $T \geq 2$, SSA's performance is insensitive to the choice of T. We observed a similar behavior for the other SensorScope time series. For the Jug Bay dataset, we observed similar behavior for $T \geq 12$ hrs. The range of T values over which SSA's performance is insensitive is different for the SensorScope and Jug Bay datasets primarily because of the differences in sampling intervals (2 minutes for SensorScope and 20 minutes for Jug Bay). So, it makes sense that it takes much longer to collect a sufficient number of samples in the Jug Bay data sets and hence requires a larger T to achieve robust SSA performance.

Impact of ϵ. As discussed in Section 2.4, for n data points collected during an interval of length T, the worst case running time of our linearization algorithm is $O(n^2)$. Such worst case scenarios arise when a small number of line segments are sufficient to model all n data points. That is, in the extreme case where a single line segment is sufficient to cover all the n data points, our greedy approach will be forced to solve larger and larger instances of the least-square regression problem in each iteration – the first iteration will have 2 samples, the second 3 samples, and the (n-1)st will have n samples, resulting in $O(n^2)$ complexity. At the other extreme,

false positive corresponding to these anomalies, (random) noise would have to affect a much greater number of samples, which is unlikely to happen.

Data sets	SensorScope 1	SensorScope 2
$\|S_1[t] - S_2[t]\|$	$(1.35, 1.40)$	$(1.28, 2.05)$
$\|S_2[t] - S_3[t]\|$	$(0.22, 0.08)$	$(0.25, 0.12)$
$\|S_3[t] - S_4[t]\|$	$(0.25, 0.11)$	$(0.29, 0.17)$
$\|S_4[t] - S_6[t]\|$	$(0.29, 0.14)$	$(0.30, 0.20)$

(a) Similarity metric as a function of T

ϵ value	SensorScope1		SensorScope2	
	# Lines	Running Time	# Lines	Running Time
0.01	92.55	0.08	86.51	0.10
0.05	47.49	0.08	50.50	0.11
0.10	25.45	0.10	29.39	0.12
0.50	3.38	0.34	3.80	0.29
1.00	1.85	0.46	1.98	0.40

(b)Impact of ϵ

Table 6: Robustness to parameters.

is the case where each pair of consecutive samples defines a new line segment, leading to $O(n)$ line segments and $O(n)$ computation. The number of line segments, k, used to model a sensor data time series depends on the desired linearization error ϵ. Intuitively, a small value of ϵ will force us to use more line segments. which would lead to a lower computational cost.However, the communication overhead of our approach is $O(k)$. Thus, ϵ controls the trade-off between computational cost and communication overhead (or size of the model).

We found that in practice (e.g., in SensorScope and Jug Bay datasets) a small value of ϵ results in each line segment covering a small number of points. Intuitively, this happens as typically sensor readings exhibit non-linear patterns (e.g., diurnal or sharp increases in value when an event occurs), and approximating non-linear patterns using line segments results in only a few points being covered by a single line. Table 6(b) shows the average number of line segments used to model 120 data points collected when $T = 4$ hrs, for SensorScope 1 and SensorScope 2, for different ϵ values. As the ϵ value is reduced from 1 to 0.01, the average number of line segments increases from 1.85 (1.98) to 92.55 (86.51) for SensorScope 1 (SensorScope 2). (Note that we can use at most 119 line segments to model 120 data points). Table 6(b) shows our linearization approache's running time (on a PC with a 2.8 GHz processor with 2GB of RAM) on the *entire* time series; as expected, it is smaller for smaller ϵ values.

Table 6(b) results support our intuition that choosing a small value for ϵ results in faster execution time. However, the overhead of the communication between a mote and its master is $O(k)$ (Section 2.4) - a small value of ϵ reduces the computation time at the expense of a larger communication overhead. In scenarios where the aggregator step does not boost the accuracy of SSA (as is the case with SensorScope and the Jug Bay datasets; Section 4.1), we can either do away with the aggregator step or invoke it less frequently than after every T minutes. This can help us reduce the computational cost of the local step (by selecting a small ϵ) while not incurring a significant communication overhead.

The choice of ϵ can also determine how well a sensor time series is approximated by our piecewise linear model. Intuitively, we should choose an ϵ value that is very small compared to the threshold γ against which the similarity measure $S(D^{new}, D^{ref})$ is compared to detect anomalies (see Section 2). With $\epsilon << \gamma$, it is unlikely that linearization errors will significantly impact the similarity measure, and hence, not impact the accuracy of SSA[5]. In this paper, we conservatively set $\epsilon = 0.1$. We also investigated how ϵ affects SSA's accuracy by trying different values between 0.1 and 1 for it and found that the accuracy of SSA was the same for the different values of ϵ. As shown in Table 6(b) $\epsilon = 0.1$ achieves a good computational cost vs. communication

[5]As discussed in Section 2, we set γ to be equal to the standard deviation of the initial reference time series; γ for the SensorScope and Jug Bay Box Humidity datasets was within the interval $[4, 7]$ and $[6, 9]$, respectively.

overhead trade-off – choosing a smaller value did not reduce the running time significantly but led to a large increase in the number of line segments k.

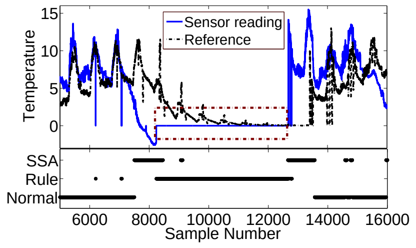
4.3. CPU and Memory Usage

In Section 2.4, we discussed the computation complexity and overhead of SSA. We also measured the running time and memory usage of SSA on a low-end netbook with Atom 1.6 GHz processor and 1.5 GB of RAM. The processing power and available memory of this netbook are comparable to that of the emerging master class devices used in today's tiered sensor network deployments [11]. We first run a monitoring program in the background, and then run SSA over all 23 time series in the SensorScope data sets. We record the running time for each times series is recorded. The monitoring program also records the memory usage by SSA every second.

We perform two sets of experiments with different linearization period T. In Table 4(a), we show the maximum running time and memory usage of SSA over all the 23 time series. For both $T = 60$ and 120, SSA takes less than 5 seconds to process approximately $30,000$ samples with a small memory footprint. These results show that the computation and memory requirements of SSA are small and well within the resources available on today's master class devices.

T Sample Number	Max Running Time (Sec)	Max Memory Usage (KB)
60	4.876	2048
120	4.796	2052

(a) Running time and Memory Usage of SSA



(b) Data set with faulty readings

Figure 4: (a) Resource usage of SSA and (b) Data set with faulty readings.

4.4. Robustness evaluation

Data faults are quite prevalent in real-world sensor system deployments [5, 6] and can be caused by bugs in hardware or software, improper sensor calibration, or due to motes running out of battery power [5]. Hence, in a real-world scenario, it is quite likely that the reference time series $D^{ref}[t]$ used by SSA may itself contain anomalous readings. Note that, as described in Section 2.3, when updating $D^{ref}[t]$ using new sensor data, we do not discard the anomalous readings. With an anomaly-free reference time series initially, as a result of updating it with these anomalous data points, the reference time series may eventually exhibit the same pattern as the anomalous readings. For example, we observed an instance of this problem in a time series from the SensorScope datasets, as described next.

Figure 4(b) (top plot) shows a SensorScope time series with a long duration *Constant reading* anomaly that lasted for 6 days. The bottom plot in Figure 4(b) shows the readings identified as anomalous by SSA alone and the Rule-based methods. SSA is able to correctly identify the samples corresponding to the start and the finish of the *Constant reading* anomaly but misses most of the "in-between" anomalous samples. This is due to our design choice to update $D^{ref}[t]$

using anomalous samples as well. We can see in Figure 4(b) (top plot) that after (approximately) 400 successive samples corrupted by the *Constant reading* anomaly, the reference time series values are quite close to the anomalous readings, and as a result, SSA does not stops flagging the subsequent readings as anomalous. In Section 2, we justified our use of anomalous readings to update $D^{ref}[t]$ by demonstrating that it helps us "zoom in" on samples where the sharp changes in sensor readings happen (see Figure 2(a)). However, as this example shows, updating $D^{ref}[t]$ using anomalous readings can cause SSA to miss a large number of samples affected by a long duration anomaly, and only identify the beginning and end of a long duration anomalous event. This again motivates the use of our hybrid approach - i.e., for the time series in Figure 4(b) (bottom plot), we identify the samples missed by SSA using the CONSTANT rule.

The SensorScope time series in Figure 4(b) (top plot) also contains two instances of *Short spikes* (that occur before the long duration *Constant* anomaly). Even though SSA alone fails to detect them, their presence does not impair SSA's ability to detect the long duration anomaly that occurs later. Hence, we do not need to "clean" the time series data before running SSA. We can see in Figure 4(b) (top plot) that the SHORT faults do not affect the reference time series significantly, i.e., SSA is robust to faults.

5. Related Work

Anomaly detection is an area of active research. In this section we briefly survey work most related to ours, organized into four categories. Moreover, we select representative techniques from these categories and quantitatively compare them against our hybrid approach, with results of this comparison presented in Section 6.

Anomaly detection in sensor systems. This is not a well-studied area. The only efforts we are aware of that focus on anomaly detection in sensor systems data are [7, 8, 9]. Rajasegarar et al. [7] present a clustering based method based on K nearest neighbor algorithm in [7], and a SVM based approach in [8]. Briefly, these works view measurements collected by a sensor system as coming from the same (unknown) distribution and "pre-define" anomalies as outliers, with main focus of these (*offline*) techniques being on minimizing the communication overhead (in transmitting data needed for anomaly detection) and the corresponding energy consumption. In contrast, we focus on an *online* approach, without "pre-defining" what is an anomaly. However, we do perform a quantitative comparison between our hybrid approach and the clustering based method in [7]. The details of this comparison can be found in Section 6.

Tartakovsky et al. use a change point detection based approach, CUSUM (Cumulative Sum Control Chart), for quickly detecting distribution changes (e.g., mean, variance, covariances) in sensor measurements [9]. We do not provide a quantitative comparison between CUSUM and our hybrid approach as we lack sufficient domain knowledge - i.e., the CUSUM based approach assumes knowledge of the (time varying) probability distribution from which sensor measurements are sampled, and such information is not provided for the SensorScope and the Jug Bay datasets (and would be quite difficult for us to compute accurately).

In general, change point detection can identify changes in the parameters of a stochastic process. Often, such changes are anomalous, and hence, change point detection techniques have been used for change/outlier detection in time series data [15, 16]. However, such existing works (a) assume that sensor measurements are sampled from a known or easily inferred distribution (e.g., Gaussian), an assumption often not true for real world datasets, (b) target a specific subset of anomalies, and (c) do not use real-world sensor datasets for evaluation. In addition, existing change point detection based methods either require a long training phase, or are computationally intensive, or (in some cases) both – hence, they do not meet our efficiency criteria. Moreover, not all changes in a time series might be anomalous, i.e., there could be a change and no anomaly (i.e., a false positive) and vice versa (i.e., a false negative). For example, [15] flags turning points of a time series from increasing to decreasing – as it is often "normal" for a sensor data time series to have periodic behavior, this is (likely) not an anomaly. On the other hand, such an approach would only flag a few turning points, instead of an entire anomalous event, as depicted in Figure 1(a). We believe that change point detection could be a promising approach to anomaly detection, but that it would require significant work to devise an accurate and efficient online anomaly detection method based on such techniques.

Fault detection in sensor systems. Short duration anomalies can be viewed as instances of data faults, errors in measurements, or outliers (see [5]). Sharma et al. focus on SHORT spikes, NOISE faults, and CONSTANT readings data faults and show that these are quite prevalent in real-world sensor datasets [6]. They also evaluate the performance of several methods – Rule-based methods, a least-squares estimation based method, Hidden Markov model (HMM) based method, and a time series analysis (ARIMA model) based method–that are targeted towards detecting transient sensor data faults. However, these methods perform poorly at detecting long duration anomalies. E.g., in Section 4.1 we showed that Rule-based methods are not sufficient for detecting long duration anomalies. Other than that, we also compare the ARIMA model based approach, which works best in case of data faults affecting more than a few samples, against our hybrid approach (with details given in Section 6).

Other approaches to sensor data faults detection include [17, 18, 19, 20]. However, these are not suited for detecting (unknown) long duration anomalies due to one or more of the following assumptions they make: (1) the anomalous data pattern is known a priori [17, 18, 19], (2) the distribution of normal sensor readings is known [19], and (3) focusing on short duration trends is enough to capture anomalies [20].

Network anomaly detection. The work in this area includes detecting anomalies such as DDoS attacks, flash crowds, worm propagating, bulk data transfer in enterprise or ISP networks [1, 2, 3, 4]. Techniques such as Principal Component Analysis (PCA) [1] and wavelet analysis [2, 3, 4], used for detecting network traffic anomalies, are not well suited for *online* detection in sensor systems primarily because they are computationally intensive and difficult to perform in an online manner. Furthermore, we show that even in an offline manner, these methods perform poorly in detecting long duration anomaly on sensor system data (as detailed in Section 6). Our quantitative study (given in Section 6) indicates that a straightforward application of such techniques, as designed for network anomaly detection, is not very effective on sensor data.

Piecewise linear time series models. Piecewise linear models have been used to model time series data in the other contexts (i.e., not for anomaly detection). For example, Keogh developed an efficient technique to search for occurrences of a *known pattern* within a time series using a piecewise linear representation [10]. Specifically, for a time series with n points, Keogh's algorithm starts with $k = \lfloor \frac{n}{3} \rfloor$ line segments and defines a "goodness of fit" metric for k line segments as $B_k = std(e_1, ..., e_k)$, where e_i is the average linearization error for line segment i. It then iteratively merges two consecutive line segments until B_k cannot be reduced any further; this process is continued until a single line approximates the entire time series. This process ends with a family of piecewise linear models for a single time series – one for each value of k, $1 \leq k \leq \lfloor \frac{n}{3} \rfloor$. Each linear model has a B_k value associated with it, and the one with the smallest

B_k is selected as the final representation.

In contrast, our greedy linearization algorithm differs as follows: (1) we start with a single line segment and continue adding more segments, (2) we use a different "goodness of fit" criterion (maximum linearization error; see Section 2) and (3) we compute a single representation instead of a family of piecewise linear models. The main reason behind these choices (rather than, e.g., using Keogh's algorithm) is computational cost as our goal is an efficient *online* approach. Briefly, computing a family of models to find the "best" one is wasteful for our purposes, if it takes significantly greater computation. Hence, we opt for our greedy approach.

6. Quantitative comparison

We now present results from a quantitative comparison of our hybrid approach and anomaly detection techniques based on ARIMA models [6], PCA [1], wavelets decomposition [3], and K-means clustering [7]. As noted above, these techniques have proven to be effective at detecting sensor data faults and network anomalies. However, our evaluation shows that an "out-of-a-box" application of these techniques for detecting sensor system anomalies performs poorly.

Comparison with ARIMA based method. ARIMA (Autoregressive Integrated Moving Average) models are a standard tool for modeling and forecasting time series data with periodicity [21], and [6] leverages temporal correlations in sensor measurements to construct an ARIMA model of sensor data. This model is used to predict the value of future readings, with new sensor readings compared against their predicted value - if the difference between these values is above a threshold (the 95% confidence interval for the predictions) then the new data is marked as anomalous. We compare our hybrid approach against the ARIMA L-step method from [6], where the ARIMA model is used to predict the next L sensor readings.

We first trained the ARIMA model to estimate its parameters using sensor readings (from SensorScope 1 and SensorScope 2) collected over 3 days as training data (a separate training phase was required for SensorScope 1 and SensorScope 2, [6] also uses training data from 3 days; these are more favorable conditions for ARIMA as SSA uses a shorter period for its reference model). The ARIMA L-step method with $L = 24$ hrs flagged 12,135 (of the 30,356 data points in SensorScope 1) as anomalies. Our inspection revealed a total of 107 anomalies that affect more than 7,500 data points. While the ARIMA L-step method identified most anomalous samples, it also falsely identified a large number of samples as anomalous. The extremely high number of false positives resulting from the ARIMA L-step method reduces its utility.

We failed to train ARIMA on SensorScope 2 due to the training data containing a *Constant readings* anomaly that affects almost two-thirds of the samples. This failure highlights the lack of robustness in ARIMA based methods to long duration anomalies in the training data. SSA's counterpart to the ARIMA training phase is the selection of an initial reference time series, and SSA tolerates anomalies in its initial reference time series (see Section 4.4).

Comparison with network anomaly detection techniques. The techniques such as Principal Component Analysis (PCA) [1], and wavelet analysis [3, 2, 4] used for detecting network traffic anomalies are not well suited for *online* detection in sensor systems primarily because they are computationally intensive and difficult to perform in an online manner. However, one can still ask: *How accurately would those technique perform on sensor systems data?*, i.e., in an offline manner. As a reasonably representative technique, we use the PCA based approach in [1], which collects periodic measurements of the traffic on all network links and splits it into "normal" and “residual” traffic using PCA. The residual traffic is then mined for anomalies using the Q-statistic test in which L2-norm $||y||^2$ of the residual traffic vector y for a link is compared against a threshold. If it is greater than the threshold, then the traffic vector y is declared anomalous [1].

We ran the PCA based method on the data from SensorScope time series 6, 8, 9, 10, 11, and 12. We chose these as their start and end times are the same. The input to PCA was a data matrix with 6 columns (one for each sensor) and 29,518 rows (the total number of samples collected). Note that the PCA based method is applied in an *offline* fashion using the entire time series data from 6 different sensors whereas in our SSA-based hybrid approach, at best, the aggregator step would get access to linear models from 6 sensors during the past $T = 4$ hours only.

The results for the PCA based method are summarized in Table 7. It fails to detect most long duration anomalies (5 out of 38 *Change in Mean* anomalies and 4 out of 67 *Change in Shape* anomalies). It does better at detecting *Short spikes* but is still not as accurate as our hybrid approach. Thus, even under a best case scenario (offline with access to the entire time series), the PCA based method does not perform as well as our hybrid approach. Recall that it identifies anomalies by looking at the L2-norm of the data vector in the residual space. As pointed out in [1], the PCA based method works best when the *intensity* of an anomaly is large compared to the variability of the normal data. This is not the case with most of the long duration anomalies present in the sensor data analyzed in Table 7. For instance, Figure 5(a) shows a *Change in Mean* anomaly in SensorScope 12 time series that the PCA based method fails to detect. It also shows a *Short* anomaly (spike) that the PCA based method is able to detect.

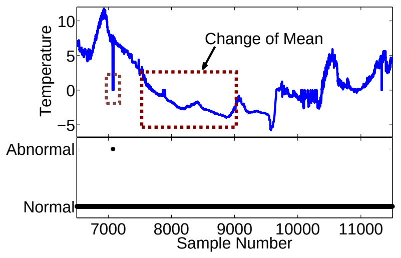
To further illustrate the impact of anomalies’ intensities on the accuracy of a PCA-based method, we injected anomalies (Short, Noise, and Constant) into the time series SensorScope 9, and attempted to detect these injected faults using PCA. The data samples corrupted by Short and Noise anomalies had a higher variance as compared to the normal data whereas, by definition, the variance of the samples corrupted by the Constant anomaly was lower than the normal data. We found that the PCA based method was able to detect most of the samples corrupted by SHORT and NOISE anomalies, but missed the Constant anomaly. We do not present these here due to lack of space; these results can be found in our technical report [22].

Apart from the intensity of an anomaly, the PCA results might also be impacted by several other factors such as sensitivity of PCA to its parameters and lack of data preprocessing. For instance, [23] shows that the performance of PCA is sensitive to the number of principal components included in the normal subspace, and the threshold used for anomaly detection. We note that, in our experiments, we did vary the number of principal components in the normal subspace, and Table 7 depicts the best results obtained (i.e., those with having only 1 principal component in the normal space). To our knowledge, the Q-statistic based technique that we use here is the only known method for automatically deciding the detection threshold. It is also well-known that anomalies can contaminate the normal subspace and hence, avoid detection by PCA [23]. One way to ameliorate this situation can be to preprocess the data to identify and remove large anomalies before applying PCA. However, in our context, defining a *large* anomaly would itself have required us to introduce another heuristic (with its own shortcomings). We do not pursue this (or other PCA related improvements) further, as the main goal of our PCA-based evaluation here was to illustrate that, as in the case of network anomaly detection, it is not straightforward to apply PCA to detect anomalies in sensor data. (This partly contributed to our choice of a different approach.) Of course, we do not claim that the PCA based method cannot be made more effective at detecting sensor data anomalies, but as noted, this is not the goal of our work here.

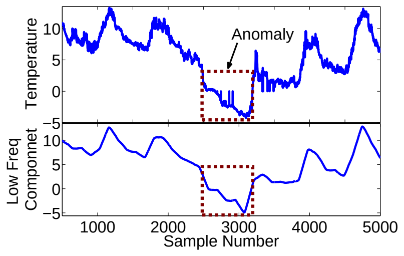
To the same end, we also explore wavelet based methods for detecting sensor data anomalies. We select the method presented in [3] as a representative technique. This method first separates

Data Set	Change in Mean	Change in Var	Change in Shape	Short	Constant
SensorScope 6	1/7	0/0	2/10	45/206	0/1
SensorScope 8	1/6	0/0	1/10	59/243	0/2
SensorScope 9	1/6	1/2	0/12	47/65	4/23
SensorScope 10	0/5	0/0	0/12	31/46	0/2
SensorScope 11	1/7	0/0	0/10	33/122	0/1
SensorScope 12	1/7	0/0	1/13	27/84	6/13
Total	5/38	1/2	4/67	242/766	22/42

Table 7: PCA based method: SensorScope



(a)SensorScope 12: PCA result



(b)SensorScope 11: Wavelet result

Figure 5: Detection of long duration anomalies: PCA and wavelet based methods.

a time series of network data (e.g., aggregate byte counts on a link) into low, medium, and high frequency components using wavelet decomposition. To detect anomalies, first a (time varying) *deviation score* is computed by combining the local variance of the medium and high frequency components. Local variance is defined as the variance of the data falling within a moving time window. The deviation score is then compared against a threshold to detect anomalies.

Data Set	Change in Mean	Change in Var	Change in Shape	Short	Constant	False Positive
SensorScope 6	1/7	0/0	2/10	205/206	1/1	26
SensorScope 8	1/6	0/0	2/10	243/243	1/2	24
SensorScope 9	1/6	2/2	3/12	65/65	13/23	42
SensorScope 10	1/5	0/0	3/12	46/46	0/2	47
SensorScope 11	1/7	0/0	2/10	122/122	1/1	29
SensorScope 12	1/7	0/0	3/13	80/84	9/13	22
Total	6/38	2/2	15/67	761/766	25/42	190

Table 8: Wavelet based method: SensorScope

We ran the wavelet based anomaly detection method described above on the same set of data that we used for evaluating the PCA based method. The results are summarized in Table 8. While the wavelet based method detects more anomalies as compared to PCA, it does not perform as well as our hybrid approach at detecting long duration anomalies. In particular, it fails to detect most of the *Change in Mean* and *Change in Shape* anomalies. In our evaluation, this method also incurred a large number of false positives. One possible reason for the wavelet based method not being very effective on the SensorScope dataset could be that it looks for anomalies in the medium and high frequency components of a time series, whereas the long duration anomalies fall into the low frequency component. The top plot in Figure 5(b) shows a *Change in Mean* anomaly in SensorScope 11 time series that is captured by the low frequency component shown in the bottom plot. Hence, it is difficult for previously proposed wavelet based methods that

Data Set	Change in Mean	Change in Var	Change in Shape	Short	Constant
SensorScope 6	2/7	0/0	4/10	56/206	1/1
SensorScope 8	3/6	0/0	5/10	54/243	1/2
SensorScope 9	3/6	1/2	5/12	20/65	5/23
SensorScope 10	2/5	0/0	5/12	16/46	1/2
SensorScope 11	3/7	0/0	6/10	49/122	1/1
SensorScope 12	3/7	0/0	5/13	36/84	3/13
Total	16/38	1/2	30/67	243/766	12/42

Table 9: Cluster based method: SensorScope

mine for anomalies only in the medium and high frequency components [3, 2, 4] to detect such anomalies. These approaches can possibly be extended using heuristic(s) that mine for anomalies in the low frequency component. We do not pursue this (or other improvements to the wavelets based approach) further as, like in the case of PCA, the main goal of our wavelet based evaluation of sensor data was to demonstrate that a straightforward application of a wavelet-based technique designed for network anomaly detection is not very effective on sensor data. Of course, we do not claim that wavelet based techniques cannot be made to work on sensor data, but as noted, that is not the goal of our work here.

Comparison with clustering based method. We compare our hybrid approach with the clustering based method in [7], which first groups the vector of readings from several sensors into clusters of fixed-width. After this, the average inter-cluster distance between a cluster and its K nearest neighboring clusters is used to determine the abnormal clusters[6]. We ran this method on SensorScope time series 6, 8, 9, 10, 11 and 12. The inputs to the clustering method are vectors of readings with the same time stamp from the 6 time series. We found that the performance of this method depends strongly on the cluster width w, as also noted in [7]. We tried a large number of w values (from 0.03 to 8) and used the best results found to compare with our method.

This method can only identify which data vectors contain anomalies, but cannot identify anomalous readings within a data vector. We determine the number of detections and false negatives as in the PCA-based method; so, it is not possible to report false positives for individual time series. This method did incorrectly flag 18 data vectors as anomalous; the actual number of false positives is between 18 and 108.We give detailed results in Table 9 and note that the cluster based method performs poorly in detecting long term anomalies such as *Change in mean*, *Change in shape*, and *Constant*. Intuitively, this is because this method is designed to detect outliers and does not exploit temporal correlations within the time series. We can also see that the method has a lot of false negatives in detecting *Short spikes*. This makes sense as a spike is determined by the corresponding data point's relative position to the previous and next point, but in the cluster based method all data points are considered as a whole and this temporal relationship is lost.

7. Conclusions

We proposed an *online* anomaly detection approach for sensor systems measurements. Our approach utilized piecewise linear models of time series, which are succinct, representative, and robust, and therefore enabled us to (a) compute such models in near real-time, (b) create models without prior knowledge about anomaly types that sensor data might contain, and (c) compare

[6]An SVM-based method [8] comparison is omitted as it assumes that few measurements are anomalous; thus it is unlikely to detect long duration anomalies such as the *Constant* anomaly in Figure 4.

and communicate different time series efficiently. Our extensive evaluation study, using real sensor systems deployments data, illustrated that our approach is accurate, robust, and efficient. Future work includes study of (1) dynamic setting of parameters (e.g., the linearization period and the size of the initial reference model), (2) feedback mechanisms between the local and aggregation steps of our approach, and (3) other techniques that are useful in our hybrid approach.

Acknowledgments

This work was funded in part by the NSF 0540420, 0120778, and 0917340 grants. We thank the anonymous reviewers and our shepherd, Anirban Mahanti, for helping us improve the paper. We also thank Fei Sha for his valuable feedback.

References

[1] A. Lakhina, M. Crovella, C. Diot, Diagnosing network-wide traffic anomalies, in: ACM SIGCOMM, 2004.
[2] A. Soule, K. Salamatian, N. Taft, Combining filtering and statistical methods for anomaly detection, in: Proceedings of the ACM conference on Internet Measurement (IMC), 2005.
[3] P. Barford, J. Kline, D. Plonka, A. Ron, A signal analysis of network traffic anomalies, in: Proceedings of the Workshop on Internet measurment (IMW), 2002.
[4] C.-T. Huang, S. Thareja, Y.-J. Shin, Wavelet-based Real Time Detection of Network Traffic Anomalies, International Journal of Network Security 6 (2008) 309–320.
[5] K. Ni, N. Ramanathan, M. Chehade, L. Balzano, S. Nair, S. Zahedi, G. Pottie, M. Hansen, M. Srivastava, Sensor Network Data Fault Types, Transactions on Sensor Networks 5(3), 2009.
[6] A. B. Sharma, L. Golubchik, R. Govindan, Sensor Faults: Detection Methods and Prevalence in Real-World Datasets, Trans. on Sensor Networks 6(3), 2010.
[7] S. Rajasegarar, C. Leckie, M. Palaniswami, J. Bezdek, Distributed anomaly detection in wireless sensor networks, in: ICCS, 2006.
[8] S. Rajasegarar, C. Leckie, M. Palaniswami, J. Bezdek, Quarter sphere based distributed anomaly detection in wireless sensor networks, in: IEEE ICC, 2007.
[9] A. Tartakovsky, V. Veeravalli, Asymptotically Optimal Quickest Change Detection in Distributed Sensor Systems, Sequential Analysis 27 (2008) 441–475.
[10] E. Keogh, Fast similarity search in the presence of longitudinal scaling in time series databases, in: ICTAI, 1997.
[11] J. Hicks, J. Paek, S. Coe, R. Govindan, D. Estrin, An Easily Deployable Wireless Imaging System, in: Proceedings of ImageSense Workshop, 2008.
[12] http://sensorscope.epfl.ch/.
[13] http://www.lifeunderyourfeet.org/.
[14] N. Ramanathan, L. Balzano, M. Burt, D. Estrin, E. Kohler, T. Harmon, C. Harvey, J. Jay, S. Rothenberg, M. Srivastava, Rapid Deployment with Confidence: Calibration and Fault Detection in Environmental Sensor Networks, Tech. Rep. 62, CENS (April 2006).
[15] V. Guralnik, J. Srivastava, Event detection from time series data, in: ACM KDD, 1999.
[16] C. Chen, L.-M. Liu, Joint Estimation of Model Parameters and Outlier Effects in Time Series, Journal of the American Statistical Association 88 (1993) 284–297.
[17] N. Khoussainova, M. Balazinska, D. Suciu, Towards Correcting Input Data Errors Probabilistically Using Integrity Constraints, in: Proceedings of the ACM Workshop on Data Engineering and Mobile Access, 2006.
[18] S. R. Jeffery, G. Alonso, M. J. Franklin, W. Hong, J. Widom, Declarative Support for Sensor Data Cleaning, in: Proceedings of the International Conference on Pervasive Computing, 2006.
[19] E. Elnahrawy, B. Nath, Cleaning and Querying Noisy Sensors, in: Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications (WSNA), 2003.
[20] D. Tulone, S. Madden, PAQ: Time series forecasting for approximate query answering in sensor networks, in: Proceedings of the European Conference on Wireless Sensor Networks (EWSN), 2006.
[21] G. E. P. Box, G. M. Jenkins, G. C. Reinsen, Time Series Analysis: Forecasting and Control, 3rd Edition, Prentice Hall, 1994.
[22] Y. Yao, A. Sharma, L. Golubchik, R. Govindan, Online Anomaly Detection for Sensor Systems: a Simple and Efficient Approach, Tech. Rep. 10-914, Computer Science Department, USC (March 2010).
[23] H. Ringberg, A. Soule, J. Rexford, C. Diot, Sensitivity of pca for traffic anomaly detection, in: ACM SIGMETRICS, 2007.