Detection and Exploration of Outlier Regions in Sensor Data Streams

Conny Franke
Department of Computer Science
University of California at Davis
franke@cs.ucdavis.edu

Michael Gertz
Institute of Computer Science
University of Heidelberg, Germany
gertz@informatik.uni-heidelberg.de

Abstract

Sensor networks play an important role in applications concerned with environmental monitoring, disaster management, and policy making. Effective and flexible techniques are needed to explore unusual environmental phenomena in sensor readings that are continuously streamed to applications.

In this paper, we propose a framework that allows to detect outlier sensors and to efficiently construct outlier regions from respective outlier sensors. For this, we utilize the concept of degree-based outliers. Compared to the traditional binary outlier models (outlier versus non-outlier), this concept allows for a more fine-grained, context sensitive analysis of anomalous sensor readings and in particular the construction of heterogeneous outlier regions. The latter suitably reflect the heterogeneity among outlier sensors and sensor readings that determine the spatial extent of outlier regions. Such regions furthermore allow for useful data exploration tasks. We demonstrate the effectiveness and utility of our approach using real world and synthetic sensor data streams.

1 Introduction

Driven by major advancements in sensor technology, sensor networks have become a critical infrastructure component for environmental monitoring and disaster management [5, 12, 17]. Space and airborne sensors, on the one hand, have the advantage of delivering high spatial resolution data of monitored regions but are typically less flexible in terms of continuous coverage (temporal resolution) and rapid deployment for regions of interest. Sensor networks comprised of stationary, ground-based sensors, on the other hand, provide for a continuous monitoring of regions of interest, but at a more coarse-grained spatial resolution. While continental scale sensor networks are still being developed and partially deployed, such as the National Ecological Observatory Network (NEON) [3], other more small scale sensor networks have been in place for many years. For example, the California Irrigation Management Information System (CIMIS) [1] comprised of about 120 weather stations across California has been in place for almost three decades and provides organizations with various important data products, such as evapotranspiration maps for managing water resources.

Besides the continuous delivery of data products, sensor networks play also an important role in the detection of anomalous environmental or climatological phenomena, such as plumes and clouds of hazardous depositions, pollution, wild fires, and flooding, to name only a few. In such settings, individual sensors in the network stream measured variables, such as temperature, (solar) radiation, humidity, wind speed and wind direction to applications. These applications aim at (1) discovering *anomalous sensor readings* and (2) constructing *outlier regions* that delineate normal regions from regions that exhibit environmental phenomena. Such applications rely on data mining techniques, more specifically *outlier detection* approaches, to detect outlier sensors and to construct outlier regions from these sensors. While there are some approaches that support the detection of outlier sensors (e.g., [4, 6, 16]) and boundaries in sensor fields (e.g., [7, 9, 15]), in particular the composition of these approaches suffers from two weaknesses. First, sensors and sensor readings from a data stream are only categorized in normal and outlier sensors. Such a binary decision, of course, provides little to no insights into the degree of anomaly of a sensor reading. Second, regions constructed from outlier sensors are assumed to be homogeneous, that is, all sensors comprising an outlier region are simply considered anomalous.

In this paper, we introduce a novel approach that addresses the above weaknesses of current techniques. By extending an existing approach for detecting outliers in sensor data streams, we utilize the concept of *degree of outlierness* for sensor readings. Using this more fine-grained information about outlier sensors, we construct *heterogeneous outlier regions*. Such regions not only better describe a delineation between normal and outlier sensors but they also

provide more detailed information about the composition of outlier regions in terms of outlier sensors that show different degrees of outlierness, an important aspect in support for the further (continuous) exploration of outlier regions.

In the following Section 2, we further detail the proposed approach and the specific objectives. In Section 3, we discuss related work on outlier detection and region and boundary detection. Our algorithm for detecting degree-based outliers in sensor data streams is presented in Section 4. Section 5 then introduces a technique for the identification and exploration of heterogeneous outlier regions using degree-based sensor outliers. In Section 6 the techniques for degree-based outlier detection and region detection are evaluated. Finally, Section 7 concludes this paper and outlines ongoing work.

2 From Outlier Sensors to Outlier Regions

In the following, we first describe the characteristics of the sensor network assumed for our approach. In Section 2.2, we then introduce the concept of degree-based outliers and motivate their usefulness in the detection and exploration of environmental phenomena. In Section 2.3, outlier regions and their properties are presented. Finally, in Section 2.4 we illustrate a two-tiered approach that utilizes both degree-based outlier detection and outlier region detection to find and explore regions of environmental phenomena in a sensor network.

2.1 Sensor Network

We assume a sensor network S comprised of m stationary sensors $\mathcal{S} = \{s_1, \ldots, s_m\}$. Each sensor $s \in \mathcal{S}$ has a *spatial attribute*, $\langle x_s, y_s \rangle$, which defines its location in 2D space. The sensors are distributed non-uniformly in the network. Each sensor monitors environmental *variables* such as temperature, humidity, or wind speed. In the proposed framework, we assume one-dimensional sensor data, i.e., the *same* variable is monitored by all sensors. However, the proposed method can be extended to multi-dimensional data as well, e.g., by normalizing all attribute values with respect to their standard deviation and then computing a (weighted) sum of the values, as done by Angiulli and Fassetti in [4].

For a sensor s, a measurement of a variable is represented as $r_{s,t}$, with the timestamp t indicating when the variable reading was obtained. We assume a *centralized approach* where all sensors periodically stream their data to a central server for processing and analysis. The network in our setting is *synchronized*, i.e., all m sensors report new measurements near simultaneously such that a set of m new measurements arrive at the server each time period. Depending on the type of sensors, such a period can range from a few seconds to several hours.

Note that the synchronous reporting of sensors is not a strict requirement for our techniques to work. We merely use this assumption for ease of explanation. Alternatively, the most recent measurement of each sensor that is available at the central server can be used when analyzing sensor readings, regardless of the time the measurement was obtained. This would also help to handle faulty sensors that do not communicate, as instead of waiting for new measurements from these sensors, the last measurement they sent before failing can be used.

Finally, based on the spatial attribute of sensors a *spatial neighborhood* $N_f(s_i) \subseteq \mathcal{S}$ can be defined for each sensor $s_i \in \mathcal{S}$. A suitable neighborhood function f allows for different metrics, such as distance based neighbors (given a maximum distance r) or k-nearest neighbors.

2.2 Degree-Based Outliers

An outlier is commonly described as "a data point that is significantly different from the rest of the data points" [6]. The difference between data points is computed using one of various metrics based on, e.g., probability density functions or distance functions. Typically, a user defined threshold is employed to specify when the difference between a data point p and other data points is significant enough to call p an outlier. We call this approach *threshold-based* outlier detection, because a binary decision is made about whether or not a data point is an outlier. If a data point is above[1] a certain threshold, then this point is considered an outlier, otherwise it is considered normal.

Some algorithms use *degree-based outlier detection*, e.g., [18, 19], where an *outlier degree*, $OD \in [0, 1]$, also called *degree of outlierness*, is determined for each data point, as described by Hodge et al. in [11]. By using such a measure to describe a data point, it is taken into account that some data points are more clearly outliers than others. Instead of having just two categories for data points, outliers and non-outliers, a degree-based approach offers a smoother and thus more fine-grained distinction between points in a data set or data stream.

An outlier degree is a useful tool to describe measurements in a sensor network in a more meaningful way and to further explore unusual phenomena. In this process, each sensor s and measurement $r_{s,t}$, respectively, is assigned a value $OD \in [0, 1]$, which changes with each new measurement the sensor obtains. The OD provides immediate feedback about the intensity of a phenomenon at a certain location. The change of a sensor's OD over time indicates how the phenomenon evolves. By comparing the OD's of

[1] Thresholds mark the boundary between outliers and non-outliers, and depending on the definition, a sensor value has to be above or below this threshold in order to be an outlier. For ease of discussion, we write "above the threshold" to refer to both cases.

several sensors within a spatial neighborhood, detailed information about the distribution and spread of an event can be obtained.

To the best of our knowledge, there is currently no degree-based outlier detection algorithm for data streams. In Section 4 our DSTORM algorithm is introduced, which is a degree-based outlier detection algorithm that extends the STORM algorithm proposed by Angiulli and Fassetti in [4].

2.3 Spatial Outlier Regions

The detection of degree-based outliers provides useful information about events in a given sensor network S. However, regions offer a more succinct spatial description of environmental phenomena than individually marked sensors. Therefore, sensors having an OD greater than a threshold φ are composed into one or more *outlier regions*. Formally, an outlier region is defined as a spatial area in the observed space where a subset $S_i \subseteq \mathcal{S}$ of sensors in the sensor network is located such that $\forall s \in S_i : OD_s \geq \varphi$.

An algorithm to detect outlier regions from outlier sensors should satisfy the following properties:

- Regions are described as polygons.
- Multiple, distinct regions can be detected at the same time.
- The placement of a region boundary depends on the OD values of sensors near the region boundary.

Polygonal regions are desirable, because polygons are a common way to describe spatial objects of any kind, and there is a plethora of tools and algorithms to store, manipulate and further process polygons. In contrast to most existing approaches, as will be discussed in Section 3, we aim at detecting an arbitrary number of distinct regions at the same time. In addition, the region boundary should be placed somewhere in the unobserved space between outlier and non-outlier sensors, and the precise placement should depend on the OD values of sensors near the region boundary. If a sensor has a low OD value, the region boundary is placed fairly close to this sensor, whereas a sensor having a high OD repels the boundary such that it is placed further away from this sensor. This novel kind of boundary placement is meaningful, as we do not expect environmental phenomena to stop exactly at the sensor where it is last observed. In contrast, existing algorithms only find clusters of outlier sensors or they find those sensors that comprise the region boundary, but they do not contemplate where exactly in the unobserved space between outliers and non-outliers the region boundary is located.

Finally, using the OD values of sensors in an outlier region, it is possible to further refine the interior of a polygonal description, leading to so-called *heterogeneous outlier regions*. That is, within an outlier region, outlier sensors may exhibit different OD values and thus provide for a more fine-grained description of an environmental phenomenon and in particular its evolution over time. This is a meaningful concept with a high degree of utility, especially if environmental phenomena need to be explored in terms of change in intensity within subregions of the outlier region over time. Just knowing about the (evolving) boundary of an outlier region is clearly of much less help.

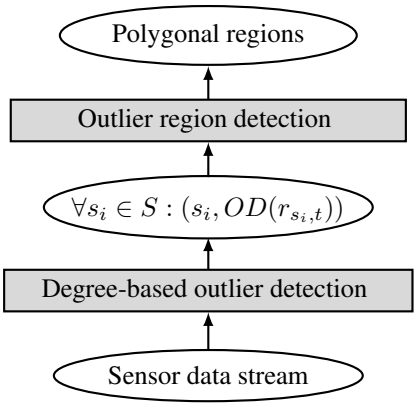


Figure 1. Two-tiered region detection

2.4 Two-Tiered Region Detection

Our overall goal is to develop an algorithm to detect and explore spatial outlier regions using a stream of sensor data as input. In order to achieve this, we use a two-tiered approach, as illustrated in Figure 1. First, a degree-based outlier detection algorithm is applied to the sensor data stream in order to determine an OD value for each individual sensor each time new measurements are obtained. The output of this step are tuples $(s_i, OD(r_{s_i,t}))$, one for each sensor $s_i \in S$ at time t. These tuples are used as input for the second step, where outlier regions are detected by taking the OD values of sensors into account. The output of the outlier region detection algorithm is a set of polygons representing event regions. With each region, the OD values of interior sensors furthermore provide for the description of the heterogeneity of outlier sensors within an outlier region.

Note that this approach comprises two orthogonal steps, and therefore each of the two components can be replaced independently. For example, one can use another degree-based outlier detection algorithm, or even a different kind of anomaly detection, e.g., burst detection, as long as there is an outlier degree $\in [0, 1]$ associated with each data point.

Moreover, the separation of outlier detection and region detection facilitates the exploration of the observed environ-

mental phenomena. Based on the OD values computed for each sensor at a certain point in time, several region boundaries can be generated, each using a different threshold φ, which defines the minimum OD a sensor must have in order to be included in the outlier region. This results in the generation of isolines describing the spread and value distribution (i.e., heterogeneity) within an outlier region. Section 5 gives more details on the exploration of the interior of outlier regions.

3 Related Work

Outlier Detection Wu et al. [18] as well as Zhang et al. [19] propose outlier detection algorithms that assign an outlier degree to each value. However, both algorithms are not designed for a streaming data context but only operate on static data sets.

Several algorithms have been proposed to detect outliers in data streams, e.g., [4, 6, 10, 16]. None of them detects degree-based outliers. We chose to use the STORM algorithm proposed by Angiulli and Fassetti [4] as basis to develop a degree-based outlier detection algorithm called DSTORM, which we will detail in Section 4. STORM (STream OutlieR Miner) detects distance-based outliers in streams of data using a sliding window approach. The algorithm takes a sensor's past values as well as past and current values of other sensors in the network into account to determine whether or not a current sensor reading is an outlier. In contrast to other approaches [6, 10, 16], the data structures and methods used in STORM provide for a flexible framework for choosing which sensors should be considered when computing whether a sensor is an outlier, i.e, sensors in the spatial neighborhood vs. all sensors in the network. This is one of the main reasons why we chose STORM over other approaches.

Region and Boundary Detection Several algorithms for boundary and edge detection in sensor fields have been proposed, e.g., in [7, 9, 15]. Chintalapudi et al. [7] propose three approaches for localized edge detection of large-scale phenomena in a sensor field. Their algorithms detect a set of edge sensors, i.e., sensors that are inside the phenomenon and close to sensors that are not inside the phenomenon. This output is therefore a subset of the outlier sensors detected by our DSTORM algorithm that will be presented in Section 4. The output of their algorithm is not sufficient to describe multiple outlier regions in the sensor network, as the boundary sensors are not partitioned into distinct groups and no polygonal boundaries can be easily generated from the set of boundary sensors. Similar arguments apply to the work by Ding et al. [9].

Nowak et al. [15] propose a boundary estimation algorithm. Their objective is more constrained than ours, as they assume that the "sensor network consists of two fields of relatively homogeneous measurements". Thus, the approach cannot detect multiple (heterogeneous) regions in the sensor network at the same time.

Kolingerová et al. [14] propose a boundary detection algorithm where the detected boundaries should be as close as possible to what a human would recognize as the outer shape of a given point set, including holes in the region. Delaunay triangulation [8] is used for this task. This approach is different from ours in two ways: First, if we were to apply the algorithm in [14] to the set of outlier sensors detected by DSTORM, there might be too many holes in the detected regions. This is because the algorithm in [14] generates a hole whenever there is a lack of sensors in an area. In contrast, in our approach, we only generate holes in an outlier region if a non-outlier sensor is located in that region, and we want the region to be continuous otherwise. The second difference is that we do not want the region boundaries to be right next to the outmost outlier sensors, but somewhere in the unobserved space between outliers and non-outliers.

All of the existing approaches mentioned above are not able to take different OD values for sensors into account. For example, the discovery of boundary sensors and region boundaries in the case of [14] would have to be done from scratch if one is interested in a region based on a different threshold for outliers.

4 Degree-Based Outlier Detection

In this section, we present our degree-based outlier detection approach for sensor data streams, called DSTORM. Our technique extends the approach for detecting distance-based outliers realized in the STORM algorithm [4], which is discussed in the following Section 4.1. After the presentation of our DSTORM algorithm in Section 4.2, we provide a runtime analysis of DSTORM in Section 4.3.

4.1 Distance-Based Outlier Detection

Distance-based outlier detection is based on comparing a data point $r_{s,t}$, e.g., a sensor measurement, with a given number of other data points in the stream. If enough data points, i.e., more than a pre-defined threshold k, have values that are similar to $r_{s,t}$, based on some distance measure for variable values, then $r_{s,t}$ is not an outlier. A sensor measurement $r_{s,t}$ can be compared to previous measurements of the same sensor s and to those of other sensors, either in the spatial neighborhood $N_f(s)$ or in the entire sensor network. We call this the *reference* of the outlier detection algorithm. In the STORM algorithm, a sliding window of recent measurements of all sensors is maintained. Formally, an outlier sensor is defined as follows:

Definition 1 (Outlier Sensor) *Let WM be a sliding window containing the w most recent measurements of each sensor in the network. Thus, WM contains sensor measurements r_{s_i,t_j}, $1 \leq i \leq m$, and $t - w \leq t_j \leq t$, i.e., from different sensors and different points in time. In addition, let $k \in \mathbb{N}$, and $r \in \mathbb{R}^+$. Then, s_l is an outlier sensor at time t if less than k measurements in WM lie within a distance r from $r_{s_l,t}$.*

Measurements in WM having a distance less than or equal to r from $r_{s_l,t}$ are called *neighbors* of $r_{s_l,t}$ and must not be confused with spatial neighbors. As also stated in [4], a measurement in WM is not considered a neighbor of itself. In [13], Knorr et al. discuss how to choose meaningful values for r and k. To determine the distance between two measurements the Euclidean distance of the two values is computed. WM contains exactly $w \cdot m$ measurements at every point in time. Recall that we assume the sensor network to be synchronous. Thus, at each point in time t, exactly m new measurements are added to WM.

Maintaining all recent sensor measurements in a sliding window provides a lot of flexibility in terms of the kind of reference that can be chosen for the outlier detection. We can use a single sensor s_i's previous measurements as the only reference for determining if s_i is an outlier, or we can use values of sensors in $N_f(s_i)$, or we can use the values of all sensors in the network as reference. Different references can be obtained by disregarding certain measurements recorded in WM when determining the number of neighbors of a sensor measurement $r_{s_i,t}$.

Note that the reference for the outlier detection algorithm influences which kinds of outliers can be detected. For example, hazardous events like wild fires and storms can be detected using measurements of the entire sensor network as a reference, as measurements within the storm region will be significantly different from all other measurements obtained in the network. By contrast, a local phenomenon like an anomalously cold valley is more likely to be detected using only sensors in the spatial neighborhood as reference. This is because if the sensor measurements obtained in the valley would be compared to all other measurements obtained in the network, then other naturally cool areas like mountain tops could obscure the phenomenon. In the following, we assume the entire sensor network as a reference, but the DSTORM algorithm can be used with all other kinds of references as well.

4.2 DSTORM Algorithm

If outlier sensors are detected according to Definition 1, as done in the STORM algorithm, only threshold-based outliers can be identified. For the reasons mentioned in Section 2.2, we add a degree-based outlier detection technique to this approach. Accordingly, the new algorithm is called

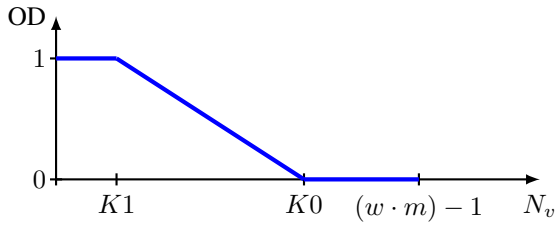


Figure 2. Relationship between degree of outlierness and outlierness bounds

DSTORM (Degree-based STream OutlieR Miner).

Intuitively, the fewer neighbors a sensor measurement has, the higher its degree of outlierness (its OD value) should be. Based on this idea, DSTORM computes the outlier degree of a sensor measurement $r_{s,t}$, denoted $OD(r_{s,t})$, based on the *percentage* of neighbors $r_{s,t}$ has in the window WM. Each measurement in WM can have at most $(w \cdot m) - 1$ neighbors. DSTORM uses two real-valued parameters $k0$ and $k1$, $k0, k1 \in [0, 1]$, called *outlierness bounds*, that replace the parameter k used in the STORM algorithm. $k0$ specifies the minimum percentage of neighbors a measurement in WM needs in order to be considered a non-outlier, i.e., $OD(r_{s,t}) = 0$. To improve readability, $K0$ denotes the actual number of neighbors needed, i.e., $K0 = k0 \cdot ((w \cdot m) - 1)$. Likewise, $k1$ specifies the maximum percentage of measurements in WM that can be neighbors of $r_{s,t}$ such that $r_{s,t}$ has a maximum outlier degree, i.e., $OD(r_{s,t}) = 1$. Also, $K1 = k1 \cdot ((w \cdot m) - 1)$. The radius parameter r for computing the distance between measurements is used like in the STORM algorithm. Let $N_v(r_{s,t})$ be the number of neighbors of measurement $r_{s,t}$ in WM. The computation of $OD(r_{s,t})$ based on $K0$, $K1$, and r is done as follows:

$$N_v(r_{s,t}) \geq K0 \quad \Rightarrow \quad OD(r_{s,t}) = 0 \tag{1}$$

$$K0 > N_v(r_{s,t}) > K1 \quad \Rightarrow \quad OD(r_{s,t}) \in (0, 1) \tag{2}$$

$$N_v(r_{s,t}) \leq K1 \quad \Rightarrow \quad OD(r_{s,t}) = 1 \tag{3}$$

If $r_{s,t}$ has more than $K1$ but less than $K0$ neighbors (Equation 2), its outlier degree is somewhere between 0 and 1 and is computed based on a weighting of the number of neighbors and the outlierness bounds as follows:

$$OD(r_{s,t}) = 1 - \frac{N_v(r_{s,t}) - K1}{K0 - K1} \tag{4}$$

The relationship between $OD(r_{s,t})$ and the outlierness bounds $K0$ and $K1$ is illustrated in Figure 2. One can see that as $N_v(r_{s,t})$ increases $OD(r_{s,t})$ decreases. $OD(r_{s,t})$ is constant for values of $N_v(r_{s,t})$ below $K1$ and above $K0$.

The specification of a measurement's OD value is very flexible, as the parameters $k0$ and $k1$ can be chosen arbi-

trarily from the interval $[0, 1]$ according to the guidelines in [13]. It is also independent of the number of measurements contained in WM, and thus the window size w can be changed without interfering with the effect of the outlierness bounds $k0$ and $k1$. For the same reason, identical outlierness bounds can be used for different references, e.g., the entire sensor network or sensors in the spatial neighbor of a sensor, as $k0$ and $k1$ only specify the percentage of neighbors that are required for a certain outlier degree.

4.3 Runtime Analysis

The sliding window data structure WM can be maintained in a list or a tree structure. We found that a tree is no more efficient than a simple list, because only a small number of elements in WM are outliers. Therefore, a range query with radius r in the tree returns almost all elements in WM, degenerating the query's runtime close to $O(m \cdot w)$. Thus, WM is kept in a list.

Updating WM every time new measurements arrive, i.e., replacing the m oldest measurements with m new measurements can be done in $O(m)$. Computing the number of neighbors of a measurement $r_{s,t}$ in WM takes $O(w \cdot m)$ time, as it has to be checked for every $r_{s,t}$ which other measurements have values within a distance r. There are m new measurements at every point in time, i.e., at every *cycle* of the DSTORM algorithm. As the number of neighbors has to be computed for each of the m sensors, the process takes overall time of $O(m \cdot w \cdot m) = O(m^2 \cdot w)$. In combination with updating WM every cycle, the runtime per cycle is dominated by the computation of neighbors and thus the overall degree-based outlier detection algorithm has a runtime of $O(m^2 \cdot w)$.

5 Outlier Region Detection

In the following Section 5.1, we present our approach to detect (heterogeneous) outlier regions from outlier sensors. In Section 5.2 we outline a runtime analysis for our technique, and in Section 5.3 we discuss approaches to the exploration of outlier regions.

5.1 Detecting Outlier Regions

A digital elevation model (DEM) is a representation of a topographic surface as a set of x,y,z coordinates where z represents surface elevation. One of the possible representations of a DEM is a *triangulated irregular network* (TIN). Triangulation is the subdivision of the 2D plane into triangles. In the case where the plane is marked by a set of points, as it is in a sensor network, the points are used as the vertices of the triangles. One common triangulation is the Delaunay triangulation [8]. A TIN is created based on a triangulation by elevating the vertices according to the elevation of the topographic surface at that point. If the concept of DEMs and TINs is used in a context different from topological surfaces, this technique is known as *wireframe surface modeling*.

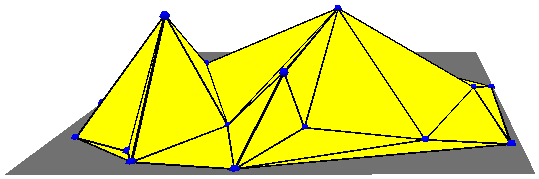
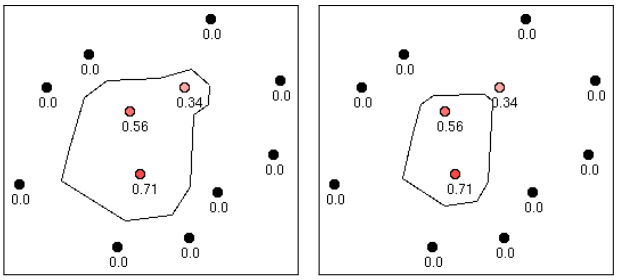


Figure 3. Triangulated wireframe surface representation of an example sensor network

A wireframe surface model based on a triangulation of the sensor network yields an insightful visualization of (possibly heterogeneous) outlier regions in the network. The elevation of a sensor is based on its current outlier degree. Thus, the wireframe surface shows outlier regions as "hills", as illustrated in Figure 3. We refer to this surface model of the sensor network as *TWS* (Triangulated Wireframe Surface). The interior of outlier regions can be visually explored instantly, as differences in the elevation within the TWS represent variations in the degree of outlierness of different sensors. A TWS is an effective method to represent a sensor network with respect to the OD values of the sensors. The triangulation of the entire network is done once and takes $O(m \cdot \log m)$ time. In each cycle the elevation of sensors in the TWS has to be adjusted according to the current OD values of all sensors, which can be done in $O(m)$.

From a TWS one can infer polygonal outlier regions, as described in Section 2.3. For this, the TWS, which is based on the current OD values of all sensors in the network, is intersected with a plane parallel to the x/y plane, at height φ. We call this approach TWISI (Triangulated WIreframe Surface Intersection). The intersection of the TWS with a plane yields a set of line segments, which are projected onto the x/y plane to get one or more polygons describing the boundary of one or more outlier regions. The plane is called the *degree plane*, and its z-coordinate, φ, is called the *height* or *threshold*, of the degree plane. The value of φ determines the minimum OD value a sensor must have in order to be included in an outlier region. Figure 4 illustrates the effect of φ on an outlier region. The figure shows part of a sensor network, where three outlier sensors are located. The sensors are labeled with their OD values 0.34, 0.56, and 0.71. Figure 4(a) shows an outlier region that was determined using $\varphi = 0.25$. In Figure 4(b), $\varphi = 0.4$ and therefore one sensor is excluded from the outlier region albeit having an OD value > 0.

Placing the degree plane at a flexible height using the



(a) Outlier region with $\varphi = 0.25$ (b) Outlier region with $\varphi = 0.4$

Figure 4. Outlier regions based on different values of φ

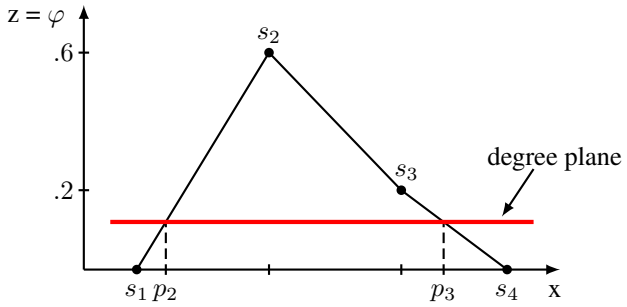


Figure 5. Effect of outlier degree on region boundary placement

parameter φ allows for additional filtering on the detected outlier sensors. The value of φ specifies the minimum OD value required to make an outlier sensor significant enough to be included in an outlier region.

The interpretation of the boundary of an outlier region is as follows: Assume we identified an outlier region using $\varphi = 0.25$. If we were to install a new sensor at some location close to or on the boundary of the detected outlier region, we expect it to measure a value that has an outlier degree of about 0.25. Values measured in the interior of the outlier region are expected to have an OD value above φ, i.e., $OD > 0.25$.

In our discussion in Section 2.3, it was pointed out that a higher OD value of a sensor should cause the region boundary to be repelled. This property matches the interpretation of the region boundary described above. Figure 5 illustrates that this is accomplished with the TWISI approach. The figure shows the degree plane and the profile of a TWS as a projection on the x/z-plane. In the figure, s_1 and s_4 are non-outliers and therefore a section of the outlier region boundary lies somewhere between s_1 and s_2, and between s_3 and s_4 respectively. Among the outliers s_2 and s_3, s_2 has the higher degree of outlierness (again indicating that a heterogeneous region is described). p_2 and p_3 are points on the x/y-plane where the outlier region boundary intersects. It can be seen in the figure that the relative distance of the region boundary is higher for s_2 than it is for s_3, due to the steeper slope of the TWS between s_1 and s_2. The region boundary got repelled more strongly by s_2's higher OD value, and thus the point p_2 on the boundary is closer to the neighboring non-outlier s_1.

5.2 Runtime Analysis

The outlier region detection is done each cycle after the current OD values of all sensors have been determined, and thus the set of outlier sensors at the current point in time is known. In order to compute the region boundaries based on a set of outlier sensors, first the line segments at the intersection of the TWS with the given degree plane (based on the threshold φ) are computed. In the second step, these line segments are combined into one or more polygons, depending on the number of intersections of the TWS with the degree plane. During the first step, only those triangles in the network triangulation where at least one vertex is an outlier have to be checked for intersection. There are at most $m_\circ \cdot tr_{max}$ such triangles, where $m_\circ$ is the number of outlier sensors and tr_{max} is the maximum number of triangles a sensor in S can be a vertex of. The intersection of the degree plane with an individual triangle yields a line segment that is part of the region boundary. Thus, after the first step, there are at most $m_\circ \cdot tr_{max}$ line segments. Computing them takes $O(m_\circ \cdot tr_{max})$.

Combining the line segments into polygons takes at most $O((m_\circ \cdot tr_{max})^2)$, using a naive approach. If there is only one outlier region, the runtime can be improved to $O(m_\circ \cdot tr_{max} \cdot \log(m_\circ \cdot tr_{max}))$.

5.3 Exploring Outlier Regions

The boundary of an outlier region based on threshold φ is an isoline of the TWS at height φ. The interior of an outlier region can be explored by plotting multiple isolines using different thresholds. These thresholds can be set appropriately to better reflect the heterogeneity among OD values of the sensors in that region. Such a visualization provides valuable insights into the value distribution in the interior of an outlier region, as we assume the interior to be heterogeneous in terms of outlier degrees of sensors within the region. For example, hotspots within an outlier region, i.e., sub-regions containing sensors with particularly high OD values, can be detected by looking at the isolines. As part of our approach, we developed a GUI that visualizes the sensor network and plots outlier regions based on a threshold φ as well as several isolines. Figure 6 shows the GUI, where an example sensor network containing outlier regions

is shown. The threshold φ is set to 0.2 and corresponding isolines are created at increments of 0.1, i.e., at heights of 0.3, 0.4, etc. When looking at the visualization, one can immediately recognize that there are two separate outlier regions. Within the larger of the two regions, there are two hotspots, one towards the north of the region and one towards the south. From looking at the isolines of the larger of the two outlier regions, one can infer that if φ would be set to 0.3 or higher, instead of 0.2, the larger region would be split into two separate regions, yielding three outlier regions in total in the shown sensor network.

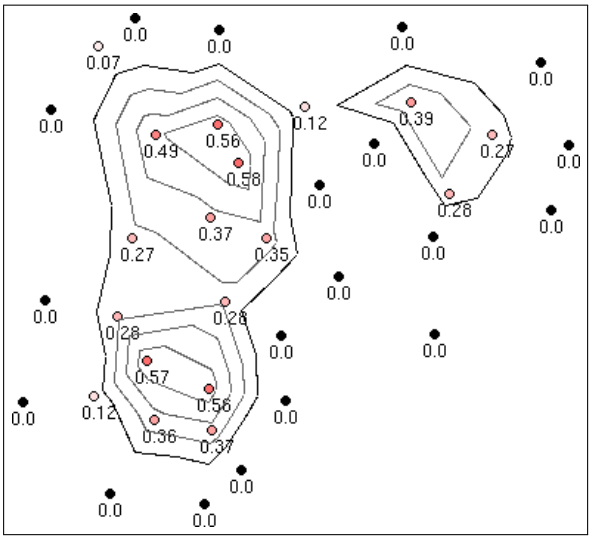


Figure 6. Exploring the heterogeneity of outlier regions using isolines

Generating outlier regions, and thus also isolines, using different values of φ during one cycle of the data stream analysis is efficient. Due to the two-tiered approach, it is not necessary to analyze sensor measurements repeatedly whenever φ is changed or isolines are added or modified. This is because the region detection technique works solely on the OD values determined for each sensor in the current cycle. In contrast, when using other existing approaches for boundary detection, as described in Section 3, all computations would have to be done from scratch when the parameters that determine which sensors are in the interior of a region are changed.

6 Experimental Evaluation

In this section, we present several experimental results using both synthetic and real data streams. We first evaluate the runtime of the DSTORM algorithm, and then demonstrate the flexibility of the TWISI technique to represent heterogeneous outlier regions.

6.1 Evaluation of DSTORM

A prototype of the DSTORM algorithm was implemented in Python, using no custom data structures or other optimizations. In order to evaluate the runtime of the algorithm, synthetic data sets were generated, simulating various sensor network sizes and containing 20,000 measurements for each sensor. The actual data values have no influence on the runtime of DSTORM, so we can safely generalize the obtained results to real data sets. Moreover, the runtime of DSTORM is independent of the number of outliers detected. We therefore omit experiments where the number of outliers in the synthetic data set varies. We tested the algorithm on networks having between 1 and 100 nodes, and in addition varied the window size w from 100 to 1200.

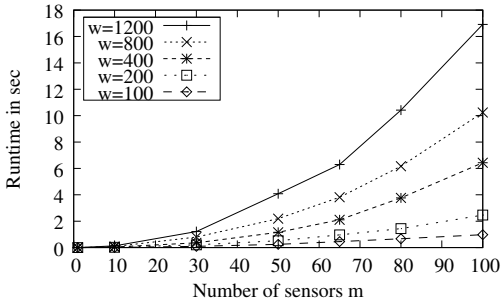
All experiments were run on an 1.86GHz Intel Core Duo machine with 2GB main memory and running Debian Linux. The runtimes presented here are the average of three separate runs for each configuration of parameter values. The results are shown in Figure 7. Runtime is presented in seconds per cycle, i.e., we show how long it takes to compute the OD values for all sensors in the network at one point in time. Figure 7(a) shows the runtime of DSTORM for different values of m. It can be seen that the runtime increases quadratically with respect to m. Figure 7(b) depicts DSTORM's runtime for different window sizes w, and illustrates that the runtime increases linearly with respect to w. Recall that in Section 4.3 we conclude that the runtime of DSTORM per cycle is $O(m^2 \cdot w)$. Therefore the experiments support our runtime analysis.

In accordance with the exact STORM algorithm, the DSTORM algorithm per definition detects the exact set of distance-based outliers in the sliding window, so precision and recall of the algorithm do not have to be measured.

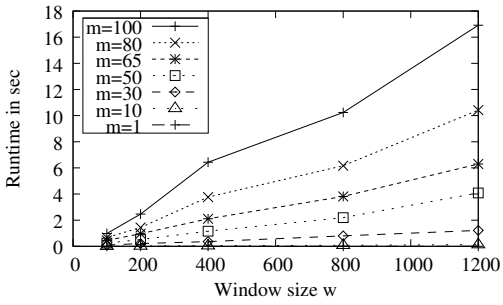
6.2 Evaluation for Region Detection

To evaluate our TWISI approach, we utilize two real data sets. The first one, the Intel lab sensor data [2], provides measurements from 54 sensors deployed in the Intel Berkeley Research lab for a period of two months in 2004, at a granularity of 30 seconds. In our experiments, we use the temperature variable, which is measured in degrees Celsius. As a second real data set we use CIMIS data [1], containing measurements from about 120 weather stations across California, at a granularity of one hour. In our experiments we use the wind speed variable, which is measured in miles per hour. For all experiments, parameter values were chosen based on the suggestions for meaningful values of r and k in [13] as well as knowledge about the period and range of the temperature and wind speed variable.

We analyzed the Intel data set using parameter values $r = 6.0$, $k1 = 0.01$, $k0 = 0.18$, and $w = 240$. Figure 8

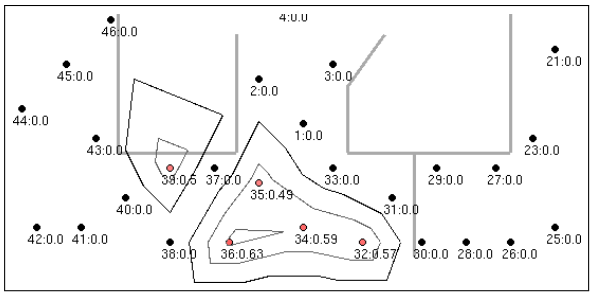


(a) Varying the number of sensors m

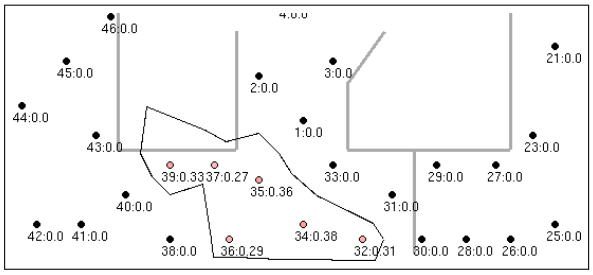


(b) Varying the window size w

Figure 7. Runtime of DSTORM per cycle using different values for m and w



(a) Outlier regions at t = 7:51am



(b) Outlier regions at t = 8:42am

Figure 8. Outlier regions in the Intel lab data

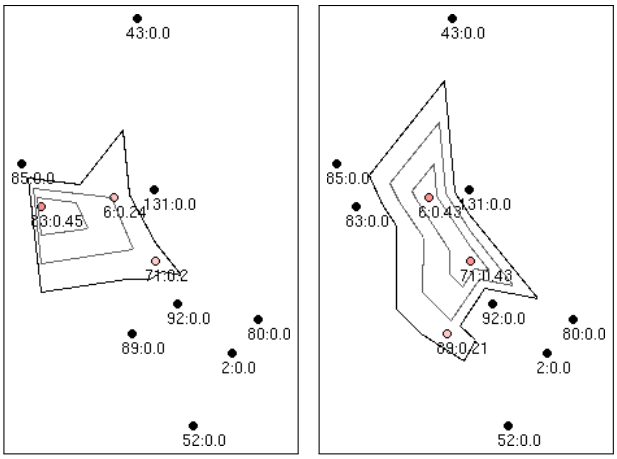
shows the lower part of the sensor network at two different points in time. We do not show the entire sensor network due to space constraints. The outlines of the network layout diagram provided in [2], i.e, the walls of the monitored floor of the research lab, are shown as light gray lines. Each sensor is labeled with its ID as provided in [2], followed by a colon and its outlier degree at the current point in time. Figure 8(a) shows the current state of the network on Feb 28th, 2004 at 7:51am. Two outlier regions are identified, using a threshold value of $\varphi = 0.2$ and inserting isolines at increments of 0.2, i.e., at 0.4, 0.6, etc. The region boundaries demonstrate the flexibility of our technique; we can detect outlier regions of arbitrary shape, i.e., regions do not necessarily have to be convex or have a pre-defined shape. Also, regions can have arbitrary size, and we can detect an arbitrary number of regions at every point in time. In addition, displaying isolines aids in the exploration of the interior of heterogeneous regions, as can be seen in the region on the bottom of Figure 8(a). The upper region spreads beyond the wall, because there is an area in the sensor network where no sensors are located and thus there is insufficient information about the temperature in this area.

Figure 8(b) depicts the situation about one hour later, at 8:42am. As can be seen, the outlier regions have merged due to sensor 37 becoming an outlier. All sensors within this region are located in the same room in the lab, so it is likely that there is a common cause for these outlier sensors. Thus, the outlier regions provide a good summary of a phenomenon occurring in this room during the given time.

As a second real data set, we analyzed CIMIS data. On January 4th, 2008, a severe storm hit Davis, CA, and the surrounding area. We used our approach to detect this storm (post-mortem) in the recorded CIMIS data and explore its movement and change in intensity, i.e., its heterogeneity. For our analysis, parameters were set to $r = 5.0$, $k1 = 0.01$, $k0 = 0.2$, and $w = 48$, i.e., the sliding window contained measurements of the most recent two days. Figure 9 shows some of our findings. We set $\varphi = 0.15$ and drew isolines at increments of 0.1. The sensor with ID 6 is located in Davis. Figure 9(a) illustrates that the storm approached Davis from the west at 10am. The storm is most intense in Santa Rosa (sensor ID 83) at this point in time. Figure 9(b) depicts the situation one hour later. One can see that the storm moved south-east, and the most intense region of the storm is now spanning from Davis to Modesto (sensor ID 71).

7 Conclusions and Ongoing Work

In this paper we have presented a two-tiered approach to detect and explore outlier regions in sensor data streams, which addresses an important application need in environmental monitoring. We proposed a technique to determine *degree-based outliers* in data streams, which has the advantage over traditional binary outlier models as it provides more fine-grained information about outliers. Using the information about degree-based outliers, we then showed how outlier regions can effectively be constructed. The novelty



(a) Storm region at 10am (b) Storm region at 11am

Figure 9. Storm region detected in CIMIS sensor data

of this technique is not only the approach of placing region boundaries but also the representation of the interior of outlier regions to better reflect the *heterogeneity among sensor readings*. Our experimental evaluations using real and synthetic data sets demonstrated that our approach is both efficient and practically useful.

As part of our ongoing work, we are developing an approach to analyze and predict how outlier regions evolve over time, both in terms of movement and change in spatial extent. In addition, we are working with real data sets from different sources to verify the interpretation of the outlier region boundary we propose. Preliminary results using the Intel lab sensor data set [2] are encouraging. We are also investigating how information about obstacles in a given sensor network, like buildings, rivers, or lakes, can be incorporated into our approach. Such obstacles might damp the effects of certain phenomena and thus we want the region detection technique to incorporate such information into the process of generating an outlier region.

Moreover, we currently develop a distributed version of our DSTORM and TWISI approach, in order to enable in-network processing of the sensor data. A cost model for in-network processing will be devised to evaluate whether a distributed approach is more cost efficient for some scenarios than the centralized approach proposed in this paper.

We plan to extend our algorithm to work with high-dimensional sensor data. This will facilitate outlier region detection in projected space, such that the detected regions indicate which dimensions of the feature space are correlated in certain outlier regions.

Acknowledgments. This work is in part supported by the National Science Foundation under Award No. ATM-0619139.

References

[1] California irrigation management information system (CIMIS). http://wwwcimis.water.ca.gov.

[2] MIT Computer Science and Artificial Intelligence Lab: Intel lab sensor data. http://db.csail.mit.edu/labdata/labdata.html.

[3] National ecological observatory network (NEON). http://www.neoninc.org.

[4] F. Angiulli and F. Fassetti. Detecting distance-based outliers in streams of data. In *CIKM '07: Conference on information and knowledge management*, pages 811–820, New York, NY, USA, 2007. ACM.

[5] J. Artiola, I. L. Pepper, and M. L. Brusseau. *Environmental Monitoring and Characterization*. Academic Press, 2004.

[6] S. Basu and M. Meckesheimer. Automatic outlier detection for time series: an application to sensor data. *Knowledge and Information Systems*, 11(2):137–154, Feb 2007.

[7] K. K. Chintalapudi and R. Govindan. Localized edge detection in sensor fields. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 59–70, May 2003.

[8] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 2008.

[9] M. Ding, D. Chen, K. Xing, and X. Cheng. Localized fault-tolerant event boundary detection in sensor networks. In *INFOCOM 2005: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 902–913, March 2005.

[10] D. J. Hill, B. S. Minsker, and E. Amir. Real-time bayesian anomaly detection for environmental sensor data. In *Proceedings of the 32nd conference of IAHR*, Venice, Italy, 2007.

[11] V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artif. Intell. Rev.*, 22(2):85–126, 2004.

[12] J. R. Jensen. *Remote Sensing of the Environment: An Earth Resource Perspective (2nd Edition)*. Prentice Hall, 2006.

[13] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *VLDB'98*, pages 392–403, New York City, NY, USA, August 1998.

[14] I. Kolingerová and B. Zalik. Reconstructing domain boundaries within a given set of points, using delaunay triangulation. *Comput. Geosci.*, 32(9):1310–1319, 2006.

[15] R. Nowak and U. Mitra. Boundary estimation in sensor networks: Theory and methods. *Information Processing in Sensor Networks*, 2634:80–95, 2003.

[16] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online outlier detection in sensor data using non-parametric models. In *VLDB'06*, pages 187–198, 2006.

[17] G. B. Wiersma. *Environmental Monitoring*. CRC Press, 2004.

[18] W. Wu, X. C. M. Ding, K. Xing, F. Liu, and P. Deng. Localized outlying and boundary data detection in sensor networks. *IEEE Transactions on Knowledge and Data Engineering*, 19(8):1145–1157, 2007.

[19] J. Zhang, M. Lou, T. W. Ling, and H. Wang. Hosminer: a system for detecting outlyting subspaces of high-dimensional data. In *VLDB'04*, pages 1265–1268, 2004.