In-Network Surface Simplification for Sensor Fields

Brian Harrington and Yan Huang
Computer Science and Engineering
University of North Texas
{brh,huangyan}@cs.unt.edu

ABSTRACT

Recent research literature on sensor network databases has focused on finding ways to perform in-network aggregation of sensor readings to reduce the message cost. However, with these techniques information about the state at a particular location is lost. In many applications such as visualization, finite element analysis, and cartography, constructing a field from all sensor readings is very important. However, requiring all sensors to report their readings to a centralized station adversely impacts the life span of the sensor network. In this paper we focus on modeling sensor networks as a field deployed in a physical space and exploiting in-network surface simplification techniques to reduce the message cost. In particular, we propose two schemes for performing in-network surface simplification, namely (1) a hierarchical approach and (2) a triangulation based approach. We focus on a quad tree based method and a decimation method for the two approaches respectively. The quad tree based method employs an incremental refinement process during reconstruction using increasingly finer levels of detail sent by selected sensors. It has a guaranteed error bound. The decimation method starts with a triangulation of all sensors and probabilistically selects sensors not to report to prevent error accumulation. To demonstrate the performance, the two simplification techniques are compared with the naive approach of having all sensors report. Experimental results show that both techniques provide substantial message savings compared to the naive algorithm, usually requiring less than 80% as many messages and less than 50% for some data sets. Furthermore, though the decimation algorithm does not provide a guaranteed error bound, for our experiments less than 4.5% of the interpolated values exceeded the given bound.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Spatial databases and GIS*; H.2.4 [**Database Management**]: Systems—*Distributed databases*

GIS'05, November 4, 2005, Bremen, Germany.

General Terms

Algorithms, Management, Performance

Keywords

Energy Efficient, Sensor Field, In-Network Surface Simplification

1. INTRODUCTION

Postage-stamp to palm size computerized sentries embedded in homes, offices, cars, factories, environments, and even human bodies or clothes monitor human and structural health, measure environment variables, track inventories, detect ground vibrations, and identify toxic chemical spills. When dozens or even thousands of sentries are networked together, a dynamic and powerful organism is formed yielding a smart physical world. Previous work [19, 13] indicated that treating a sensor network as a database embedded in a physical world and providing users with a declarative language is an effective way to interact.

Unlike in the Internet era where the world becomes small and location is not important, location is a first class citizen in sensor networks. Therefore, a spatial database paradigm is a more suitable model for a seamless integration of sensors with physical space. Many queries posted to sensor network databases are spatial queries. Due to the energy deprived nature of the current sensor networks, constantly sensing and continuously streaming back the sensed values by all the sensors to a centralized server depletes the energy of a sensor network very quickly. Recent research literature on prolonging the life span of sensor networks has focused on in-network aggregations (e.g. average temperature of a region) to reduce communication cost due to the fact that communication is much more expensive than computation. Sending/receiving a bit consumes orders of magnitude more energy than executing an instruction [21].

However with aggregation, information about the state at a particular location is lost. Many applications need sensor readings from individual sensors to represent a surface over the whole physical space. For example, when sensors are used for visualization, e.g. chemical fume intensity and temperature, a surface over the physical space needs to be created from sensor readings. In engineering, finite element analysis is used for applications such as structural analysis of bridges and simulating electromagnetic fields. Sensors embedded in bridges and underground may help with the structural analysis and electromagnetic field construction respectively. Polygonal meshes are generated to represent

a field or surface. In cartographic modeling, analysis on the whole field or regions of a field are needed. In fact, analytic field operations operations are grouped into local, focal (neighborhood), and zonal classes based on the spatial scope of the operation. Primitives for each category are defined which allow flexible recombination to express complex spatial analysis. Sensors may be employed to a field which support complex spatial analysis.

Surface simplification is useful and has been extensively researched in many application domains such as remote sensing, computer graphics, computer vision, cartography, and computational geometry. The goal of surface simplification is to use a subset of points to interpolate a surface which is as close to the real one as possible. Requiring all sensors to send their readings to a centralized server to reconstruct a surface is very energy consuming and is often unnecessary. For example, a flat temperature surface does not need the values for all sensors to represent it. In general, a rough surface needs more sensors while a smooth surface only needs a few sensors to be represented. Thus in-network surface simplification is useful in order to reduce message cost and save energy. In this paper, we propose to model dense sensor networks as a sample of a field and investigate in-network surface simplification methods to reduce the number of sensor nodes responsible for reporting their readings for a centralized server to "faithfully" reconstruct the field. We now formally define the problem: Let S be a set of randomly deployed resource and communication constrained sensors in a physical field. The readings from S for a time snapshot can be used to form a surface representation, e.g. for a temperature surface or an electromagnetic field. The goal is to design algorithms to select a subset S' of all the sensors in S to report to the central site so that the central site can reconstruct the surface using S' with the objective of reducing the message cost and bounding the error.

There are two primary challenges: 1) the resource constraint of sensor nodes limits the intensity of computation each node can perform. Thus, approximation rather than optimal solutions are pursued; 2) the communication constraint of sensor nodes limits the total number of hops traveled by all messages. Thus, a localized algorithm which does not require much long haul communication with remote sensors is preferred.

We propose two approaches for in-network surface simplification in this paper, namely a hierarchical approach and a triangulation based approach. Both of them require simple in-network computation and localized communication. In particular, we focus on a quad tree based method and a decimation method for the two approaches respectively. The quad tree based method employs an incremental top-down refinement process during reconstruction using increasingly finer levels of detail sent by selected sensors. For a given desired error bound ϵ, it guarantees the reading received by the central site is within ϵ of the real sensor readings. This method reduces message hops by aggregating homogeneous readings. The decimation method starts with a triangulation of all sensors. A localized Vonoroi cell construction is used to create the initial triangulation. However, the typical iterative node deletion process could not be used in refining the triangulation because it requires global error information. Concurrent error calculation and deletion by all sensors may result in error accumulation. We propose a probabilistic approach to select sensors not to report. To demonstrate the performance, the two proposed simplification techniques are compared with the naive approach of having all sensors report. Experimental results show that both techniques provide substantial message savings compared to the naive algorithm, usually requiring less than 80% as many messages and less than 50% for some data sets. Furthermore, though the decimation algorithm does not provide a guaranteed error bound, for our experiments less than 4.5% of the interpolated values for a surface were more than ϵ from the correct value.

We believe that in-network simplification is a relatively under-explored area which deserves more attention. Specialized in-network simplification techniques would be in great demand for specific application domains.

The rest of the paper is organized as follows. We review related work in sensor network databases in section 2. A general framework of in-network surface simplification is first described and then our two in-network surface simplification approaches are presented together with their energy implications in section 3. The comparison of the proposed approaches are discussed as well. An experimental design and performance comparisons of the proposed approaches and the naive approach on real datasets are presented in 4. We conclude and discuss future work in section 5.

2. RELATED WORK

There has not been much research work focusing solely on in-network surface simplification. Nevertheless, our work is closely related to the broad area of sensor network databases. Several pioneering works and research prototypes [2, 4, 3, 13, 19, 21, 20] have demonstrated the feasibility of representing and querying a sensor network as a distributed database system. In an acquisitional sensor network database, a collection of sensors of the same type may be treated as a table in a database. The sensor table has a table schema in the format of $< location, time, reading >$. The rows of the table are distributed among sensors in a physical space. Users can interact with the network using declarative database query languages. The query is inserted into the network by either broadcasting or targeted routing.

We classify related work in the broad area of sensor network databases into three categories: in-network aggregation, correlation based sensor reporting, and sampling. In addition, we look at some of the work that has been done on surface simplification.

Due to the resource and communication constrained nature of current sensors, query optimization schemes to reduce the energy consumption are the focus of much research effort [14, 12, 18]. In particular, in-network aggregation is considered an effective way to reduce the messaging cost for aggregation queries (e.g. sum and average) at the cost of simple in-network computations. The rationale is that communication is much more expensive than computation. In the TAG system [12], an aggregation tree is created when a query is broadcasted to the sensors. The tree is used to aggregate the sensor readings from children to a parent all the way up to the root where the query originates. For distributive (e.g. sum, min, and max) and algebraic (e.g. average), the TAG method significantly reduces the messaging cost by reducing message hops. Recent work [17] pointed out that aggregation without considering the area that a sensor is representing may not be adequate for spatial aggregations such as average. Spatial interpolation methods in-

cluding Spatial Moving Average, Voronoi Diagram and Triangulated Irregular Network were proposed for answering spatial average queries [17]. With aggregation queries, detailed locational information may be lost. Our work focuses on faithfully representing a field using a minimum number of sensors.

The other stream of work under the in-network aggregation category focuses on re-distributing/aggregating sensor readings to designated in-network storage nodes to facilitate future queries [11, 6]. In [11], sensor readings are discretized into ranges and routed to zones which are organized as a zone tree. Future queries based on sensor reading ranges will be routed to the corresponding nodes. In [6], a multi-resolution storage is created based on sensor space. Nodes in higher levels store aggregated sensor readings of lower levels. On-line analytical processing style queries such as roll-up and drill-down may be performed on such a multi-resolution storage structure. In-network storage does not reduce the number of sensors responsible to report in order to reconstruct the surface. Our work does not assume in-network storage but may be extended to include them.

Correlation based sensor reporting techniques utilize spatial and/or temporal correlation. Goel *et. al.* exploited the temporal correlation in the readings of sensor nodes to reduce the communication costs in sensor networks in [7]. Sensor nodes are clustered into groups and a group leader represents the whole group. Grouping and leader rotation were left for future research in [7]. Each sensor node calculates a function based on history readings to predict the readings of the node in the near future and sent it to its group leader. In case of high temporal correlation, a time series is condensed into a single function, thus reducing communication cost. We are more interested in the spatial correlation in this paper. Our approach is orthogonal to Goel's approach.

Sampling from all sensors is one way to avoid requiring all sensors to report. Bash *et. al.* proposed an energy efficient uniform sampling scheme for sensor networks [1]. A random sampler from the central station probes the sensor network to select the set of samples. A sensor uses its Voronoi cell to decide a probability to accept or reject the probe. Uniform sampling is useful for application domains such as querying the average sensor battery life. For other application domains such as finite element analysis uniform sampling may not be suitable.

Numerous surface simplification techniques have been developed in many disciplines including remote sensing, computer graphics, computer vision, cartography, finite element analysis, and computational geometry. An excellent survey on polygonal surface simplification algorithms by Heckbert *et. al.* is available in [9]. Surface simplification approaches may be classified into two categories: hierarchical and flat polygonal simplification approaches. Hierarchical approaches divide the space into hierarchical subdivisions, e.g. quad tree or k-d tree. Polygonal simplification approaches employ more general subdivisions and triangulations to represent a field. Rough areas, i.e. where the sensor readings in spatial proximity fluctuate, need more vertices than smooth areas.

3. IN-NETWORK SURFACE SIMPLIFICATION

The framework for in-network surface simplification is as follows. Initially, a fixed structure over the sensor field such as a hierarchical tree structure or polygonal elements are constructed. It is assumed that the central site knows the positions of all the sensors, so this structured may either be constructed at the central site and distributed to all the sensors or constructed locally in each sensor. Since the topology of a sensor field can change significantly, e.g. due to energy depletion of sensors or failed linkage, the structure may need to be periodically updated. The cost of this step may be amortized by many queries utilizing the structure.

When a surface query which requires constructing a field from sensor readings comes in, the query is inserted into the sensor network using a broadcasting scheme to allow it to reach every sensor. The parameters for a surface query include error bound, simplification method, sample frequency and duration. For example, a surface query may be: "construct surfaces with error bound ϵ using a decimation method every 5 seconds for two hours." Upon receiving such a query, a sensor invokes a localized algorithm to perform in-network surface simplification, possibly using the fixed structure constructed before. The result of such an algorithm is a decision of whether the sensor needs to report its reading to the central site. Reporting sensors use a geographic aware routing algorithm to report their readings to the central site. The central site reconstructs the field using the readings and locations of the sensors which report. Greedy perimeter stateless routing (GPSR) [10] is a typical geographic aware routing algorithm. Assuming each sensor knows its own location, GPSR [10] makes greedy forwarding decisions using only information about a sensor's immediate neighbors in the network topology. It chooses the neighboring sensor which is closest to the destination. When a packet reaches a "void" region where greedy forwarding is interrupted, the algorithm recovers by routing around the perimeter of the region. For typical topologies in two dimensional space with N randomly distributed sensors, geographic routing schemes that route from one sensor location to any other require $O(\sqrt{N})$ messages.

We describe a hierarchical approach and a triangulation based surface simplification approach to demonstrate the *in-network surface simplification*. In particular, we focus on a quad tree based method for the hierarchical approach and a decimation based method for the triangulation approach.

3.1 Hierarchical Tree Simplification

One way to simplify the surface of the field is to impose a hierarchy and allow sensors that are higher in the hierarchy to represent lower levels when possible. In this section, we first discuss several methods for imposing the hierarchy followed by explaining a general method of exploiting a hierarchy to simplify the surface and then reconstruct it at the central site. Then we analyze the energy consumption benefits of using this technique.

3.1.1 Tree Hierarchies

There are numerous ways to impose a hierarchy on the sensors in a field. In general one can classify techniques into two categories based on whether or not a technique depends on the positions of the sensors. We discuss aggregation trees as an example that depends on the sensor positions and quad trees as an example that does not.

In current sensor database systems such as TAG [12], a

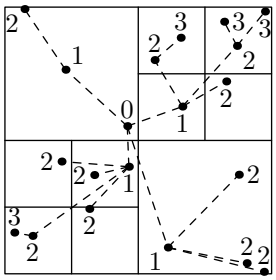


Figure 1: Quad tree of sensor field.

query is broadcast to all sensors from a central site. When the query is broadcast, a tree rooted at the base station is created. The query sent has an associated counter that is incremented with each retransmission. This counter is 0 at the root and specifies the level of a node from the root of the tree. Each sensor chooses one of its neighbors which is one level higher in the tree to be its parent. This tree is also called an aggregation tree because when the aggregation result is routed back to the central site using this tree, each node aggregates its reading with the readings from its children. This tree may be a candidate for a hierarchical decomposition of the field. However, aggregation trees are not unique. In order for the central site to reconstruct the field, it must know what tree was formed. This can be done by having every sensor report its parent at the beginning of a query. If the query has a long duration, e.g. every 5 seconds for 2 hours, this might be acceptable. However, in many cases this may be too costly.

To overcome these problems, it is useful to have a tree that does not change with each query. In a quad tree each node represents a rectangular region in a two dimensional space [15]. At each level the space is divided into four quadrants as shown in figure 1. As it forms a data[1] independent hierarchy, it can be used to divide the region into fixed cells for each level. Having fixed cells means that the centralized site can determine which sensors a reporting sensor is representing based on the location of the sensors and the level of the sensor that is reporting. That is a sensor will represent other sensors in the same space, and the level of the reporting sensor gives the size of the space that is represented.

Quad trees are useful in that the same tree can be used for many queries. A sensor can easily determine which cell it belongs in if it knows its location, the boundaries of the space, and the desired number of levels. It is generally assumed that a sensor knows where it is. The boundaries of the space and the number of levels can be calculated in the central site and broadcast to the sensors. However, this only allows a sensor to know which cell it is in at the lowest level. For the simplification algorithm some sensors must represent higher levels. The central site needs to take the responsibility of choosing the interior sensors and constructing the tree using locations of all sensors. The advantage of having the central site choose is that it can pick the sensors to minimize the average number of hops for a sensor to get a message to its parent. Then the central site tells all of the sensors who their parent is. Figure 1 shows an example with the level each sensor is at in the hierarchy. Level 0 is the root which represents the whole field.

[1]The data referred to is the sensor positions.

3.1.2 Simplification Using the Hierarchy

This section details an in-network algorithm for surface simplification when a hierarchy of the sensors is available. Each node calculates a few aggregates, e.g. sum, max, min, avg, and count. These values are used to estimate the homogeneity of a sub-region and find extreme values to be sent to the central site individually. A leaf node simply assigns values to the aggregates based on its own reading. An internal node calculates the aggregates based on its children's aggregate values and its own reading. Then the internal node identifies the child that has the largest deviation from the average based on its min and max values. If the identified child's value deviates from the average by more than ϵ, then its value will be forwarded individually to the central site. The aggregates are re-calculated for the remaining nodes. This process goes on multiple times until no such child exists any more. Then the final aggregates are sent to the current node's parent. Algorithm 1 summarizes the algorithm.

Algorithm 1 Hierarchical Aggregation (HA)

1: **if** (A node s is a leaf node) **then**
2: Let $sum(s) = max(s) = min(s) = avg(s) = s.value$ and $cnt(s) = 1$;
3: **else**
4: When the values from all s's children C arrive, calculate the aggregation of $C \cup s$ as: $sum(s)$, $max(s)$, $min(s)$, and $cnt(s)$;
5: Calculate the average of s as : $avg(s) = \frac{\sum_{c \in C} sum(c) + s.value}{\sum_{c \in C} cnt(c) + 1}$;
6: **end if**
7: **while** (There exists a child $c \in C$ whose $max(c)$ or $min(c)$ deviates from $avg(s)$ the most and is above a threshold ϵ) **do**
8: The value $avg(c)$ is sent individually to the central site;
9: $C = C - c$; Re-calculate $sum(s)$, $max(s)$, $min(s)$, $cnt(s)$, and $avg(s)$ excluding c;
10: **end while**
11: Send the aggregates of s to parent;

To reconstruct the surface, the central site first sorts the messages based on the hierarchy from higher to lower levels. Then for each message, find the point that sent the value. The average that was sent will be applied to that point and all of its descendents. This algorithm reconstructs the surface gradually by working its way down the hierarchy. The order is important as it allows more precise readings in lower levels of the hierarchy to overwrite the estimate values from the higher levels. Figure 2 shows an example of how a surface is reconstructed using this technique. The actual surface is shaded to indicate the sensor readings for the sensors. We number the south-west, north-west, south-east, and north-east quadrants as 00,01,10, and 11 respectively. This numbering scheme is recursive. The reading from level 0 is used first as the value for the whole field. This reading is an average of its reading with the level 1 readings from quadrants 01,10, and 11 since the level 1 reading from quadrant 00 deviates too much from the average and is excluded. Then any values received from level 1 sensors are applied

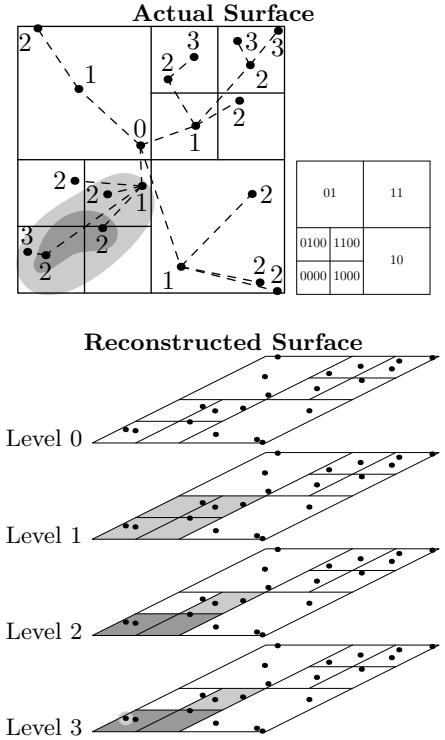


Figure 2: Top down reconstruction of surface as values from each level of the hierarchy are applied.

to the quadrant they represent. The only value from level 1 received is from quadrant 00. This value is an average of its reading and the reading for quadrant 1100. Three level 2 readings are received for quadrants 0000, 0100, and 1000. Finally there is one level 3 reading that is applied.

Using this method, the surface we reconstructed has an error bound ϵ where error is defined as deviation of the actual sensor reading from the reconstructed value at the central site. We have the following lemma.

LEMMA 1. *The deviation of the reconstructed value for any sensor s using algorithm 1 is within ϵ from the actual reading of s.*

Proof Sketch: By using the aggregates $max(c)$ and $min(c)$ of a child c, the steps from 7 to 10 of the algorithm HA make sure that if any of a sub-region's sensor readings deviates from the average by more than ϵ, then the child's value is reported individually and excluded from the average. The iteration results in a final group whose values do not deviate from their average by more than ϵ. The reconstruction process makes sure that finer readings overwrite the average of other readings from higher levels. So the lemma holds.

3.1.3 Analysis On Energy Consumption

This algorithm simplifies a surface by using averages when values are within some threshold. The threshold would need to be given in the query as the desired accuracy which varies depending on the application. The worst case cost is when all of the sensors have values that are over the threshold, leading all of them to individually report their value. In this case, each interior sensor will forward each child's value using an efficient geography aware routing scheme[10]. These sensors will still need to send their value to their parent. Therefore, if the average number of hops from a sensor to the central site is L and the average number of hops from a sensor to its parent is P, the worst case cost is $L \times N + P \times N$. Note P is likely to be 1 for a query tree formed during query insertion. But for a quad tree structure, P may be greater than 1. The best case is when the surface is flat, or at least flat enough that none of the values are over the threshold from the computed averages. In this case each sensor sends one message to its parent. The cost is then $P \times N$. In general this method is useful if the inequality shown in equation 1 holds.

$$P \times N + F \times L \times N < L \times N \quad (1)$$

The left hand side is the approximate cost of this algorithm. The term F is the fraction of sensors which need to report individually. Manipulating equation 1 algebraically we find that $F < 1 - P/L$. We expect P to be close to one[2] for aggregation trees. Others have shown that when using geographic routing schemes that routing from one sensor location to any other requires $O(\sqrt{N})$ messages [10]. This can be used to approximate L and as a worst case for P when using quad trees. However, the central site can choose the parents to try and minimize P so it is likely that P will be significantly less than L even for quad trees. Thus, we think it is quite likely that equation 1 will hold in practical settings.

In addition, we also have the fixed one-time cost of constructing the quad tree at the central site and disseminating the information to all of the sensors. This cost is once per sensor life time or may be several times due to sensor failures and energy balancing concerns. This cost is $L \times N$ since each sensor receives one message from the central site.

3.2 Triangulation Simplification

Triangulation based simplification approaches may be further divided into two sub-categories. Refinement methods work top-down. They start with a minimal approximation, then select and add points in multiple passes to build the final triangulation. Refinement methods may not be a good candidate for in-network surface simplification for sensor fields since they require long haul communication to inform remote sensors about the global information in order to select the right sensors to be added. Decimation methods work bottom-up. They begin with a triangulation of all the input points, select and delete points iteratively to gradually simplify the approximation.

3.2.1 Constructing the Initial Surface

The first step in a decimation surface simplification algorithm is to construct the initial surface. This surface is used to calculate the resultant errors if vertices are merged/deleted. The surface itself is updated iteratively during the surface simplification process until a stopping criteria is met (e.g. an error threshold). In a sensor network, this step means constructing the initial surface using all of the sensors. Among all types of meshes, triangles are most frequently used to approximate surfaces [9]. Because energy is the primary concern in a sensor network, it is desirable to have a triangulation that can be computed once when the sensor net-

[2]As P is the number of hops from a sensor to its parent, one is the smallest possible value.

work is setup and used for all subsequent field based queries. One popular triangulation method is Delaunay triangulation which can be derived from the Voronoi diagram of a set of points [5]. Figure 3 shows our example sensor field with the Voronoi diagram indicated as dashed lines and the Delaunay triangulation indicated as solid lines.

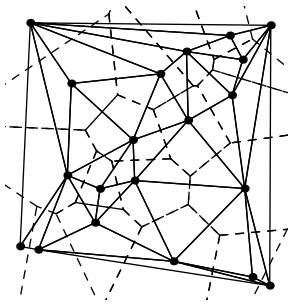


Figure 3: Sensor field with Voronoi diagram and Delaunay triangulation.

The Voronoi diagram is created by finding the Voronoi cells for each point, which divides the space into non-overlapping partitions. The Voronoi cell for a sensor s, is the convex polygon that contains all of the points that are closer to s than any other sensor [5]. The Delaunay triangulation is found by connecting points that share an edge in the Voronoi diagram.

For a pair of sensors, the bisector line of the edge which connects the two sensors divides the space into two half planes. The brute force way to compute a Voronoi diagram for a set of sensors S is: for each sensor s_i, compute the half-planes $hp(s_i, s_j)$ for sensor pairs s_i and s_j where $j \neq i$ and $s_j \in S$. The common intersection of all these half-planes that contains the sensor s_i is the Voronoi cell for s_i. This algorithm requires that each sensor knows the positions of all the other sensors either by locally storing the information or by communicating with all the sensors. A plane sweep algorithm sweeps a horizontal line from top to bottom over the plane and constructs the Voronoi digram progressively [5]. In [8] they look at using the sweep line algorithm in sensor networks but found it is too expensive because of rather involved floating point arithmetic to compute the parabolas.

We now present a filter-and-refine algorithm for computing Voronoi diagrams in sensor fields. The algorithm has two phases. In the first phase, it computes an initial closed Voronoi cell for each sensor. This cell covers the actual Voronoi cell. In the second phase, it iteratively refines the initial cell until the actual Voronoi cell is found.

It is assumed that a sensor knows about other sensors in radio range and also knows if it is a vertex on the convex hull that bounds all the sensors. An initial Voronoi cell for a sensor is computed using the brute force algorithm just described but only for the sensors within radio range. If the cell is closed then the refinement phase begins. Note that if a sensor is part of the convex hull, then its Voronoi cell will not be closed. For this phase, the cell for a sensor on the convex hull is considered closed when the line segments of the convex hull close the cell. If the cell is not closed, then messages must be sent in the direction of points that would close the cell. Once a reply is received and the cell is closed the refinement phase begins.

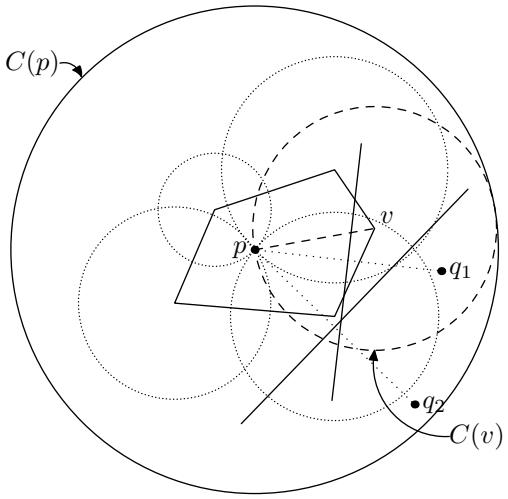


Figure 4: Local Voronoi cell calculation.

Once a closed cell is available for a sensor p, the cell can be gradually refined until the exact cell is determined. The sensors that help to form the Voronoi cell are called contributing sensors. The initial cell may be further refined by a sensor q if the perpendicular bisector line of line pq clips one or more vertices from the initial cell. The work in [17] shows that for each vertex v of the cell, all the sensors that may result in clipping v off must be in a circle $C(v)$ centered at v and with a radius of $|pv|$ where $|pv|$ is the Euclidean distance between p and v. We call $C(p)$ the impact range of sensor p. Figure 4 shows that q_1 may clip v since it lies in $C(v)$ while q_2 will not since it lies outside of $C(v)$. Now define $C(p)$ as the smallest circle that is centered at p and bounds all the $C(v)$s, where v is any vertex that belongs to the initial Voronoi cell. We can conclude that only sensors that are in $C(p)$ may be additional contributing sensors which clip the initial cell.

In [17], the relationship between message propagation delay and distance to the sensor p is used to refine the initial Voronoi diagram. Each sensor sends its position to all other sensors either by flooding or by a routing tree similar to [12]. Each sensor refines its initial Voronoi cell by listening to others position information and finalizes its cell after a while. The rationale is that sensor p will receive position information from the nearby sensors first and nearby sensors are more likely to be the contributing sensors. So if p finalizes its Voronoi cell after a while before some of the remote sensors positional information reaches p, p may still have a reasonable approximation of the actual Voronoi cell. However, although p stops listening and finalizes its cell, the positional information is still flooding in the sensor network. Assuming each sensor sends its position to all other sensors using a routing tree, the total number of position messages sent and forwarded to intermediate nodes is no less than $N \times (N-1)$ since each routing tree has $N-1$ edges. This huge energy consumption may deplete the sensor network quickly.

In this paper, we propose to use an acquisitional approach rather than a flooding approach to exchange positional information. After calculation of the impact range $C(p)$ of sensor p, sensor p initiates a query tree rooted at p similar to the query routing tree in [12]. However, in the message of

constructing the query tree, the position of p and the radius $r(C(p))$ of $C(p)$ are also sent. The tree will stop growing once a sensor's distance is greater than $r(C(p))$. All the sensors that belong to the query tree will send their positional information to p using the query tree constructed. In this way, only a limited number of sensors that may have impact on the Voronoi cell construction will be inquired for there position. This leads to a significant message reduction. Algorithm 2 summarizes the steps for a sensor to construct its Voronoi cell.

Algorithm 2 Voronoi Cell Calculation

1: Construct an initial closed Voronoi cell for p;
2: Calculate $C(p)$;
3: Construct a query tree rooted at p where each sensor's distance to p is no greater than $r(C(p))$;
4: Sensor p refines the Voronoi cell using the positions it received from the query tree.

3.2.2 In-Network Surface Simplification

Decimation algorithms for surface simplification [16] start with the whole set of vertices and gradually reduce the number of vertices that are used. The two most common ways to reduce the number of vertices are deleting and merging. Merging vertices usually results in a new vertex that is positioned to reduce the error. As the sensors represent vertices for our model, this technique cannot be used. Deletion works well for sensor networks. A deleted vertex represents a sensor that does not need to report.

Though it is not used for the triangulation, the value of the sensor readings must be known for all neighbors to calculate the error if this sensor was not to report. The neighbors of a sensor, s_i, is the set of all sensors s_j such that there is an edge from s_i to s_j in the triangulation. This means that all sensors must report their value to their neighbors. As it is possible that some neighbors are not in radio range, it may require several hops to notify all neighbors.

Once the readings from all neighbors have been received, a sensor must calculate some metric to determine the impact that not reporting would have on the surface. In [16] there are two standard metrics which are used depending on the type of vertex. Simple vertices are defined as those that are completely surrounded by triangles and use a distance to plane metric. Conceptually the neighbors of a vertex can be thought of as forming the base of a pyramid, although there may not be a plane that contains all the neighbors so an average plane is constructed, with the vertex itself being the apex. The metric is then finding the height of the pyramid as shown in figure 5 (a). The other type of vertices are boundary vertices. For our purposes boundary vertices are ones that are on the convex hull. For these vertices the distance to boundary metric is used as shown in figure 5 (b). The idea for both metrics is if the distance is small then removing the vertex will not significantly change the surface.

In typical decimation algorithms, after a vertex is removed the hole is re-triangulated and a decision will be made to remove latter vertices using the new triangulation. For sensor networks, that would be too expensive. Since each sensor is performing the error calculation locally and concurrently, it is possible that all sensors will decide to leave, resulting in error accumulation. Figure 6 shows error accumulation

(a) Distance to plane (b) Distance to boundary

Figure 5: Surface distance metrics.

for line simplification. When s_2 to s_4 individually calculate their errors, they are within threshold. But error accumulates quickly when s_2 to s_4 all decide to leave. To counter this we use the local error of each sensor to determine the probability that a sensor should report. The probability that a sensor i needs to report may be determined by: $p_i = \min(\frac{e_i}{\epsilon}, 1)$, where e_i is the error of sensor i and ϵ is the error threshold. After calculating the probability the sensor will make a decision as to whether or not to report by generating a random number r in the interval $[0.0, 1.0)$. If r is less than p_i the sensor will report. This probability has the nice property that if a sensor's error is above the error threshold ϵ, the $p_i = 1$ so it will definitely report. However, if its error is below ϵ, then it may still need to report. The larger the error is the higher the probability. The probability makes it unlikely that all sensors would decide not to report unless their errors were all low (a flat surface). Also it ensures that for a region of sensors all with large errors but below threshold, some of them will still report. The higher average errors are, the more likely it is that sensors from that region will report.

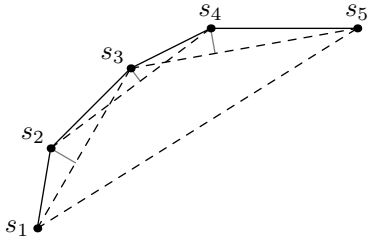


Figure 6: Error Accumulation

3.2.3 Analysis on Energy Consumption

This algorithm is only useful if it can reduce the number of messages sent. In the initial surface construction phase, each sensor needs to construct an initial closed Vonoroi cell and then refine it. In most cases, knowing of other sensors in radio range is a good start for constructing the initial Vonoroi cell. This will require a one hop message carrying the sender's positional information sent from each sensor to its immediate neighbors. The number of messages is N in this step. In the Voronoi cell refinement step, each sensor p calculates its own impact range $C(p)$ and constructs a query routing tree rooted at p. Positional information of nodes in the tree will be routed back using the tree. Let trh be the average height of all the trees, the number of one hop messages in the refinement step can be estimated by $N \times (2 \times trh)$. The cost of constructing an initial Vonoroi diagram is once for all unless there is a change in the sensor network

with sensors coming and leaving. The Voronoi diagram can be used for all future operations which need it, e.g. Voronoi cell based aggregation [17].

In the in-network surface simplification phase, the algorithm requires that every sensor sends its current sensor reading to its neighbors to help its neighbors to estimate their error. The cost of this step is $N \times nh$, where nh is the average number of hops to reach a neighbor. Then the sensors that decide to report must send their value to the collection site. Assuming a network with N sensors and an average length to the collection site of L messages, the number of messages in the reporting step is $L \times N_r$, where N_r is the number of sensors which report.

Without counting the fixed initial surface construction cost, let us compare the cost of the naive algorithm where all sensors send their readings to a centralized site, $L \times N$, and the proposed algorithm. To out-perform the naive algorithm, equation 2 must hold.

$$L \times N > N \times nh + L \times N_r \quad (2)$$

For dense sensor network typically $nh = 1$. Solving for N_r we find that $N_r < (1 - \frac{1}{L}) \times N$. This means that if $\frac{N}{L}$ sensors decide not to report, the proposed algorithm will outperform the naive algorithm. For most sensor networks this is a reasonable assumption.

3.3 Comparison of the Two Approaches

In the hierarchical method, leaf sensors have to perform less computation and potentially send less messages than interior sensors. Furthermore, it is not very robust to sensor failures. If a sensor high in the hierarchy fails, a lot of data about the descendents is potentially lost. However compared with the triangulation method, the hierarchical method has the advantage of having an error bound on the results. The hierarchical method is preferred in heterogeneous systems where sensors with high capacity that are less likely to fail may assume roles high in the hierarchy. Triangulation based techniques for in-network surface simplification have roughly the same amount of computation for each sensor and are more robust to failures. For homogeneous systems where sensors have similar capacity, triangulation based techniques are preferred to avoid data loss due to node failure and energy depletion in part of the network.

4. EXPERIMENTAL DESIGN AND PERFORMANCE EVALUATION

We chose to test the performance of our algorithms using the University of Delaware[3] global surface air temperature and precipitation monthly grids. This data consists of temperature and precipitation readings for 85794 points once a month for 50 years from 1950 through 1999. Using these points as a sensor field, we looked at the effect of sensor density and the error threshold on the number of messages that need to be sent.

To see the difference in the three approaches the naive algorithm of having all sensors report individually was compared to the hierarchical approach, using a quad tree for the hierarchy, and the decimation algorithm. The simulation was done by modeling sensors at the selected points and reporting the values for those points during each iteration. That is the reading for the first iteration would be the value for January 1950 and the next iteration would use the value for February 1950 and so on. The first selected point was chosen to be the central site.

[3]http://www.jisao.washington.edu/data_sets/willmott/

4.1 Density Results

Both algorithms should do better in dense fields, i.e. fields that have a lot of sensors in radio range. To see how density affects the number of messages, a percentage of the points were randomly chosen to be the sensor field. The radio range was selected by finding the minimum range so that each sensor had at least one neighbor in the lowest density. Changing the density alters the number of sensors in the field and as a result can also change the average number of hops to another node or the central site. For the temperature set the ϵ was 5°C and for the precipitation set ϵ was 20 mm.

Figure 7 shows the results. For both temperature and precipitation the naive algorithm grows in terms of the number of messages faster than either the hierarchical or decimation algorithms. For the temperature set our approaches require 10% to 20% fewer messages than the naive algorithm. Both approaches require about 50% as many messages as the naive algorithm for the precipitation data set. Comparing the hierarchical approach with the decimation the hierarchical does slightly better for the temperature data and they are about even for the precipitation data. Furthermore, both algorithms have a similar growth rate which is close to linear.

4.2 Error Threshold Results

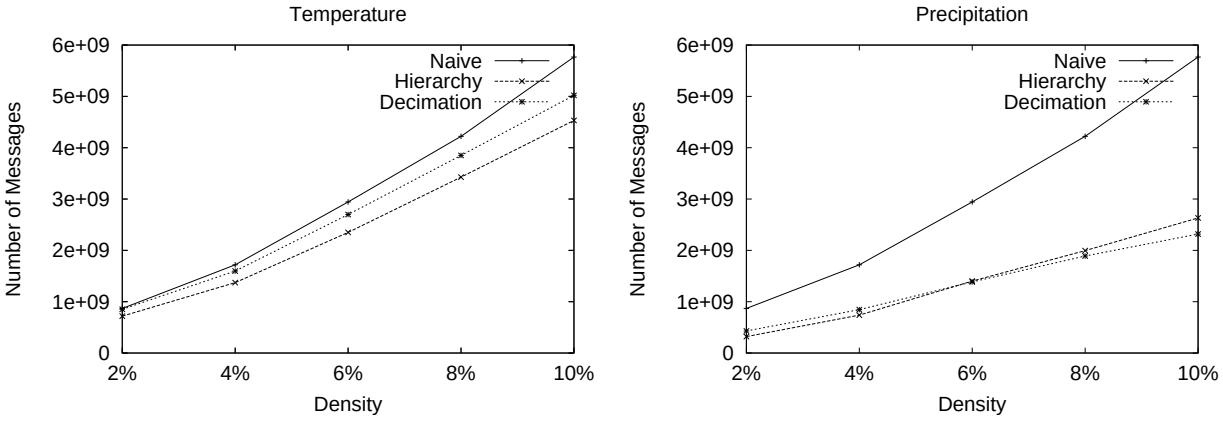
Another import aspect of the algorithms is how much the desired level of detail changes the required number of messages. If less detail is required then fewer messages should need to be sent to construct the surface. the level of detail for our algorithms is controlled by an error threshold. Increasing the threshold reduces the amount of detail.

Figure 8 shows how the level of detail changes the number of messages. The density was fixed at 10% for the temperature data set and at 2% for the precipitation data set. As expected, when the error threshold is increased the number of messages required decreases. For both data sets the hierarchical algorithm performed much better than the decimation algorithm. This is probably due to the sensitivity to density for the decimation approach. For the decimation algorithm a sensor must send to all neighbors in the triangulation. In sparse networks these neighbors are likely to be more than one hop away. For the hierarchical algorithm a sensor sends one message to its parent and thus is not as sensitive to the density.

4.3 Error Bounds For Decimation

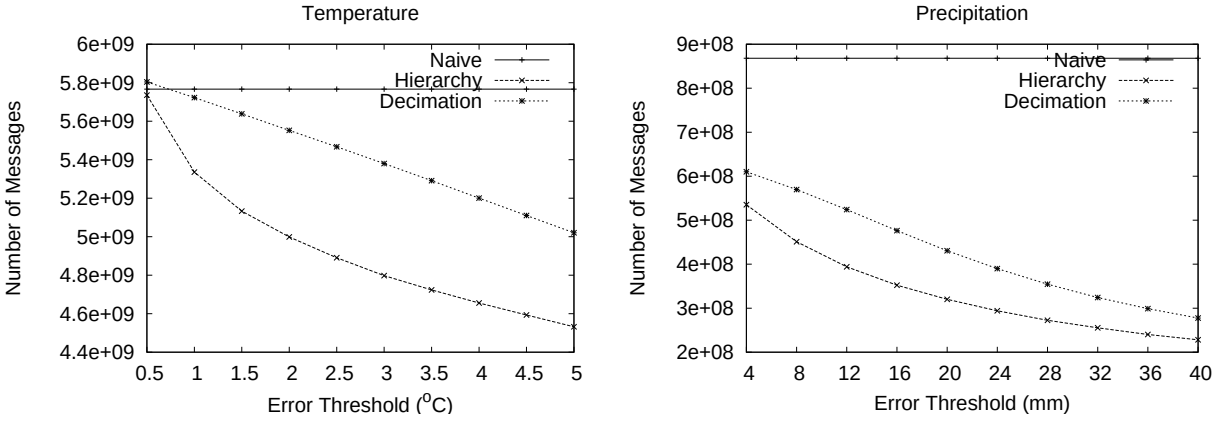
With the hierarchical algorithm all of the sensors are guaranteed to be within the error threshold from the actual reading. The decimation algorithm makes no such guarantee. For sensors that report the exact value is known, but for other sensors the value must be interpolated from reported values. To verify that the decimation algorithm generates a close approximation of the surface, we looked at how many of the interpolated points were outside the error threshold from the exact value.

Figures 9 and 10 show the results. All surfaces had less than 4.5% of the sensors with interpolated values over the threshold. Though a hard bound cannot be guaranteed,



(a) Temperature (b) Precipitation

Figure 7: Number of messages with respect to density.



(a) Temperature (b) Precipitation

Figure 8: Number of messages with respect to error threshold.

empirically it seems that the probabilistic mechanism works fairly well at preventing error accumulation.

5. CONCLUSION AND FUTURE WORK

For many applications including visualization and finite element analysis constructing a field from all sensor readings is essential. In this paper, we introduced two techniques for performing in-network surface simplification including a hierarchical approach and a triangulation approach. Experimental results indicate significant savings of up to 50% compared to the naive algorithm of having all sensors report. Furthermore, the hierarchical approach provides a bounded error for all sensors in the reconstructed field. Though the triangulation approach does not have a bounded error empirically only a small number of sensors, less than 4.5% for our tests, will have an interpolated value that exceeds the bound.

Future work is needed to find ways to make the techniques more robust to changing networks. Ideally the initial construction costs can be reduced and designed to handle sensor and communications failures. In addition, some work is needed to try and combine spatial and temporal correlation to try and further reduce messaging costs.

6. REFERENCES

[1] B. A. Bash, J. W. Byers, and J. Considine. Approximately uniform random sampling in sensor networks. In *DMSN '04: Proc. of the 1st international workshop on Data management for sensor networks*, 2004.

[2] P. Bonnet, J. E. Gehrke, and P. Seshadri. Querying the physical world. *IEEE Personal Communications*, 2000.

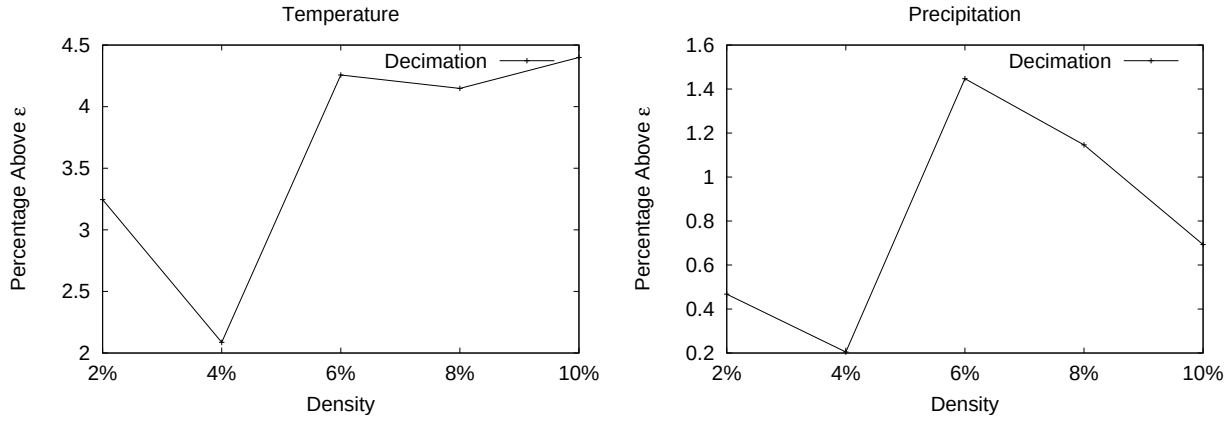
[3] P. Bonnet, J. E. Gehrke, and P. Seshadri. Towards Sensor Database Systems. In *Proc. of Second International Conference on Mobile Data Management*, 2001.

[4] P. Bonnet and P. Seshadri. Device database systems. In *Proc. of the 16th Intl. Conf. on Data Engineering*, 2000.

[5] M. de Berg, M. van Kreveld, M. Overmans, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. 2nd rev. ed. Berlin: Springer-Verlag, 2000.

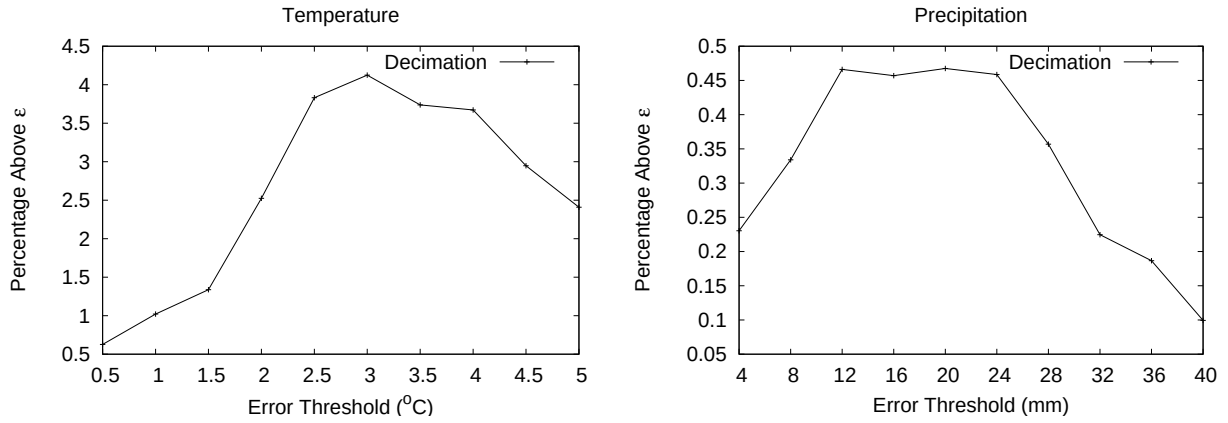
[6] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann. An evaluation of multi-resolution storage for sensor networks. In *SenSys '03: Proc. of the 1st Intl. Conf. on Embedded networked sensor systems*, 2003.

[7] S. Goel, A. Passarella, and T. Imielinski. Using buddies to live longer in a boring world, 2004. Rutgers Department of



(a) Temperature (b) Precipitation

Figure 9: Percentage of points outside threshold with respect to density.



(a) Temperature (b) Precipitation

Figure 10: Percentage of points outside threshold with respect to error threshold.

Computer Science Technical Report DCS-TR-558.

[8] B. Greenstein, E. Kohler, D. Culler, and D. Estrin. Distributed techniques for area computation in sensor networks [wireless networks]. In *29th Annual IEEE Intl. Conf. on Local Computer Networks*, pages 533–541, 2004.

[9] P. S. Heckbert and M. Garland. Survey of polygonal surface simplification algorithms. Technical report, 1997.

[10] B. Karp and H. T. Kung. Gpsr: greedy perimeter stateless routing for wireless networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, 2000.

[11] X. Li, Y. J. Kim, R. Govindan, and W. Hong. Multi-dimensional range queries in sensor networks. In *SenSys '03: Proc. of the 1st international conference on Embedded networked sensor systems*, 2003.

[12] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks, 2002. OSDI.

[13] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Design of an acquisitional query processor for sensor networks. In *SIGMOD*, 2003.

[14] S. R. Madden, R. Szewczyk, M. J. Franklin, and D. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks, 2002. Workshop on Mobile Computing and Systems Applications.

[15] H. Samet. *The design and analysis of spatial data structures*. Addison-Wesley Longman Publishing Co., 1990.

[16] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. In *Proc. of the 19th Annual Conf. on Computer graphics and interactive techniques*, pages 65–70, New York, NY, USA, 1992. ACM Press.

[17] M. Sharifzadeh and C. Shahabi. Supporting spatial aggregation in sensor network databases. In *GIS '04: Proceedings of the 12th annual ACM international workshop on Geographic information systems*, 2004.

[18] N. Trigoni, Y. Yao, A. Demers, J. Gehrke, and R. Rajaraman. WaveScheduling: Energy-Efficient Data Dissemination for Sensor Networks. *Internet Draft*, 2004.

[19] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. In *SIGMOD*, 2002.

[20] Y. Yao and J. E. Gehrke. The cougar approach to in-network query processing in sensor networks. *Sigmod Record*, 31(3), 2002.

[21] Y. Yao and J. E. Gehrke. Query Processing in Sensor Networks. In *Proc. of the First Biennial Conference on Innovative Data Systems Research (CIDR)*, 2003.