Distributed Histograms for Processing Aggregate Data from Moving Objects

Hairuo Xie Egemen Tanin Lars Kulik
NICTA Victoria Laboratory
Department of Computer Science and Software Engineering
University of Melbourne, Victoria 3010, Australia

Abstract

For monitoring moving objects via wireless sensor networks, we introduce two aggregate query types: distinct entries to an area and the number of objects in that area. We present a new technique, Distributed Euler Histograms (DEHs), to store and query aggregated moving object data. Aggregate queries occur in a variety of applications ranging from wildlife monitoring to traffic management. We show that DEHs are significantly more efficient, in terms of communication and data storage costs, than techniques based on moving object identifiers and more accurate than techniques based on simple histograms.

1 Introduction

We develop a novel approach for applications that track a large number of moving objects using wireless sensor networks (WSNs). Since a WSN's throughput per node decreases if the number of sensors increases [10], we envision applications in which sensors cooperatively store and process moving object data for later retrieval of aggregate information. For example, a zoologist may be interested in the global movement patterns of the wildlife in a park using a habitat monitoring system based on movement sensors.

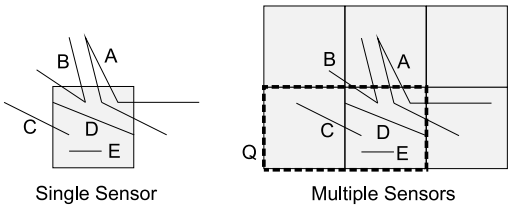

Figure 1. 5 moving object paths intersecting the responsibility regions of 1 and 6 sensors.

We introduce two main aggregate query types for monitoring moving objects: the total number of *distinct entries* and the total number of *distinct objects*. We illustrate those queries in a simple scenario (see Figure 1): a sensor (not visible) is placed in the center of its rectangular responsibility region and 5 distinct objects move in that region along different paths. There are 6 distinct entries to the region of the sensor because the object A enters the region twice.

Assume a deployment of 6 sensors as in Figure 1 to monitor traffic patterns. One approach to monitor the distinct entries and the distinct number of objects for the query rectangle Q is to use object identifiers, such as radio frequency identification (RFID) tags, in tandem with aggregation techniques (e.g., [14]). However, this approach requires a large amount of tracking data to be passed between the sensors. Alternatively, we could simply store object counts on the sensors to reduce communication and storage costs. However, such a histogram-based method cannot accurately answer the two basic queries for more than one sensor, for example, object C in Figure 1 is counted twice in Q. The sum of individual sensor counts can significantly deviate from the correct answer for both query types if the number of sensors is large. Although the sum of the individual counts is useful, it is the *total traffic* observed in Q, the two basic aggregation queries require a different approach.

In this paper, we develop the concept of *Distributed Euler Histograms* (DEHs) to answer both query types (as well as total traffic queries) with a high level of accuracy and without using IDs. We show that DEHs significantly reduce storage and communication costs, which leads to considerable energy savings for WSNs. Our approach is ideal for situations where moving objects cannot be easily identified due to the high costs in RFID tag placement. In addition, privacy and security concerns might prohibit individual identifiers, e.g., in traffic monitoring. As a DEH is a counting-based technique, we only need simple motion detectors and counters that are readily available. We introduce DEHs in Section 3 and show their efficiency in experiments compared to competing approaches in Section 4.

2 Related Work

Aggregate-query processing in WSNs is a major research area [14, 17, 21] that has mostly focused on answer-

ing non-spatial queries such as finding the minimum temperature in a WSN. Spatial data structures for WSNs have only recently been introduced: an R-tree structure in [4], a k-d tree based structure in [12], a multi-resolution grid-based structure in [5], and in [9] a multi-rooted quadtree that incorporates hashing and maps events observed in a network onto the network itself. Gao et al [6] propose a quadtree-based structure to store sensor readings in the network. Although Gao et al use aggregation methods, they do not address moving objects. Meka and Singh [15] present a quadtree variant for managing complex moving phenomena, such as toxic plumes, and create summary information, such as compact individual toxic plume descriptions, but do not apply aggregate data storage and query processing.

Storage and processing of RFID data has attracted significant interest in data management research. Hu et al [11] highlight the key challenge, the high volume of input data created by RFID tags, and apply bitmap-based techniques to store and process RFID information. The system in [8] uses path-dependent aggregates to process and store moving object data in a data warehouse. In contrast to our work, these systems focus on efficient *central* data processing.

Existing moving object indexing techniques (e.g., [18]) mainly focus on centralized spatial data structures, a trend also found for systems that process continuous nearest neighbor and other continuous spatial queries [16, 19, 20]. None of the approaches are designed for a highly distributed environment such as a WSN. Gedik and Liu [7], however, introduce a distributed approach for tracking moving objects and answering continuous queries but focus on efficient retrieval of individual data items rather than aggregate query processing. Spatiotemporal aggregate data storage and processing has been a separate research area (cf. Lopez et al [13] for a comprehensive survey) but most approaches concentrate on server-based data management.

2.1 Euler Histograms

Euler Histograms (EHs) [3] are spatial data structures to efficiently store aggregated spatial data and answer spatial queries that Simple Histograms (SHs) cannot. An example EH is given in Figure 2: 3 rectangular objects are mapped onto a regular space partition, a 4×4 grid, and the corresponding EH is given on the right part of the figure. The SH consists of a set of 16 integers. We increment each cell value for every rectangle intersecting a grid cell leading for the SH to a count of 17 instead of 3. The correct number of 3 rectangles cannot be derived without a separate data set.

An EH consists of three histograms: the counts (of a rectangle such as A) mapping onto the faces, edges, and vertices of each grid cell (as shown on the right part of the Figure 2).

Any query that can be answered using the SH can also be answered with an EH as it includes the *face counts*. The total number of distinct objects T in a given query region is determined by the formula $T = F - E + V$, where F is the sum of face counts overlapped by the (object) rectangles, E is the sum of the edge counts intersecting the rectangles, and V is the sum of vertices enclosed by the rectangles. In Figure 2, we obtain for the query region Q: $T = 3$, which is the exact number of distinct objects intersecting Q.

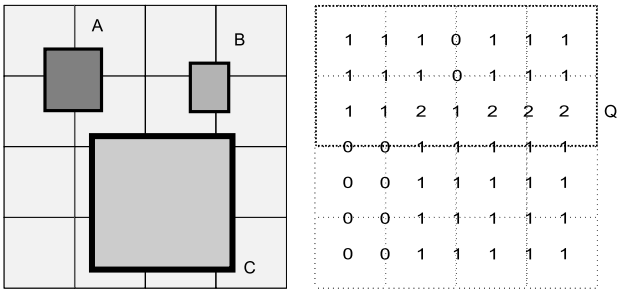

Figure 2. Left: a 4×4 grid with 3 rectangles; right: the resulting EH and a query rectangle.

Currently, EHs are only designed for rectangles on grid-based partitions and centralized architectures. We address these limitations for aggregate-query processing in WSNs.

3 Distributed Euler Histograms

A DEH is a multi-dimensional data structure that maintains histograms at the network nodes. An example DEH is given in Figure 3: sensors partition the space into disjoint convex responsibility regions. The sensors (not shown) are the sites of a Voronoi diagram and their responsibility regions are the Voronoi cells. We assume that each sensor can detect any object in their region, store simple ID information, perform simple counts, and that two sensors in adjacent cells are in communication range.

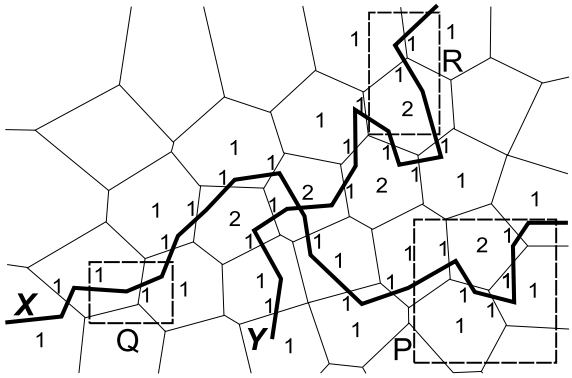

Figure 3. A DEH for two object paths X and Y showing histogram values greater than 0.

Figure 3 shows two moving object paths, *X* and *Y*. The DEH reflects the counts for *X* and *Y* in each cell. We main-

tain only two histograms instead of three as for EHs. In the DEH we keep a face count for each cell, which is updated when a moving object is detected by a sensor, and edge counts for each intersected edge of any two adjacent cells. Each edge count is assigned to only one sensor (also if an object moves along the edge of two neighboring cells). If an object moves from one cell to another through a vertex, we increase the count for only one edge.

In the example in Figure 3, five cell counts are equal to 2. In three cases, the same object reenters a previously visited cell and in two cases two paths intersect the same cells.

Note that if a query rectangle intersects a cell, its face count is considered relevant to the query. However, a path crossing such a cell may not intersect the actual query rectangle. We assume that the deployment of sensor nodes is dense enough to neglect this inaccuracy. To inject a query from a sensor node, we use the tree-based query broadcast and aggregation techniques proposed in [14].

Figure 3 depicts three rectangles Q, P, and R, that we use to explain our approach to distinct entry as well as distinct object, and total traffic queries. The total traffic query simply requires a summation (at each sensor node) over the cell counts, which leads in case for Q to 2. The number of distinct entries to Q is computed by the formula $T = F-E$, i.e., $2-1 = 1$. For distinct object queries, we also apply the formula $T = F - E$, and obtain for Q: $2 - 1 = 1$. For the query rectangle P, the DEH leads to a total traffic of 6 and to $6 - 5 = 1$ the distinct entries as well as distinct objects. For R the DEH cannot report the number of distinct objects accurately. The DEH computes the total traffic as 4 and correctly $4 - 2 = 2$ distinct entries. However, the DEH also reports 2 distinct objects in contrast to the correct answer of 1 distinct object. Keeping only counts, we cannot maintain the knowledge that the same object has entered R twice.

A vertex count cannot solve this problem. Maintaining ID lists at sensors correctly answers the distinct object query but comes at an additional cost and has the converse problem for distinct entry queries as shown in 3.1. More detailed information, e.g., timestamped paths maintained at each sensor, could provide accurate answers for both query types but is not a viable alternative due to the high cost. Section 4 shows that even simple ID lists have significantly higher data storage and communication costs as a DEH.

3.1 Correctness and Accuracy

In [3] it is shown that a rectangle overlapping a grid-based space decomposition satisfies the formula $F - E + V = 1$, where F is the number of grid cells covered by the rectangle, E is the number of edges it intersects, and V is the number of enclosed vertices. Thus, we only count the rectangle once in a given distinct object query.

We show that DEHs provide correct counts for distinct entry queries if we only keep face and edge counts for moving object paths. We can view a path as a concatenation of line segments, and each segment starts in a cell, crosses an edge, and ends in an adjacent cell. Each line segment can be treated similarly to rectangles. Note that a vertex intersected by an object path is assumed to belong to one of its adjacent edges and line segments cannot span two edges. Hence, the count for V can be treated as 0. Each line segment satisfies the formula $F - E = 0$, except for the end point of the last line segment, which requires an additional face count. Thus, the sum of counts correctly leads to 1 for this path. In the case that an object stays in a cell, a DEH correctly computes one count: the face containing the object path.

We treat the distinct object and distinct entry queries in the same way. However, if a path exits a query region and reenters it, the concatenation-based approach for counting fails for distinct object queries. We observe two end points for a single path without knowing that they belong to the same path. However, as the path entered the query region twice, distinct entry queries are answered accurately. The dual of this problem, for distinct entry queries, occurs for ID-based approach. Assume that two neighboring cells are located at the border of a query rectangle. Then, the two ID lists of the neighboring cells can be merged to find the correct number of distinct objects in the rectangle. However, we do not know if a moving object traveled from one cell to the other via their edge within the rectangle or if any object went outside the query rectangle and returned. Thus, we cannot answer the distinct entry queries accurately.

4 Experiments

4.1 Experimental Setup

We designed an experimentation environment using the J-Sim simulator [1]. We generate a simulated deployment of sensors over a rectangular area. The distribution of sensors is uniformly at random. For a given *number of objects* that travel in that area, we generate their travel patterns using the Network-based Generator of Moving Objects [2]. This generator creates moving object paths for a given transportation network, which is for our experiments the Melbourne city transportation network.

For each moving object, we define the concept of a trip: given two randomly generated points in Melbourne, the generator creates the shortest path between them. For each object we create a set of trips, with one shared connection-point, to simulate realistic city conditions where some objects make multiple trips. Thus, the *number of trips* and the *percentage of multi-trip objects* are a part of our simulation parameters. For a given *query rectangle size* expressed as a percentage of the deployment area size, we generate query rectangles over the deployment area uniformly at random. These are injected in the network at a randomly selected point in the network. We run 100 queries for each experiment and report an average.

We have implemented three approaches: DEH, an ID-based System (IDS), for example based on RFID tags, and SH. All approaches use the same routing and aggregation methods but differ in terms (a) their aggregation operations at the node level and (b) the type of the information communicated between sensors. IDS has to compare different ID lists reported from numerous sensors to answer an aggregate query. We compare the answers of all systems with respect to communication costs, storage costs, and achieved accuracy. We excluded computation costs as in WSNs the energy costs for computation are significantly lower than for communication. It is important to point out that DEHs require only simple aggregation operations such as summations, while IDSs require the comparison of large data sets.

We present the results using the two basic query types, distinct entry and distinct object queries. The results for the total traffic query is irrelevant for our discussion as all three approaches can find the result for this query accurately and efficiently by simply reporting a single count per sensor.

4.2 Results

4.2.1 Number of Objects

First, we evaluate the impact of the number of objects on the performance of IDS, DEH, and SH. Our hypothesis is that communication and storage costs for IDS increases as the number of objects increases. In contrast, the costs for DEH and SH are not affected and are lower than for IDS. For DEH and SH, the accuracy in distinct object queries will not be as high as for IDS. For IDS and SH, the accuracy in distinct entry queries will be less than DEH.

	No. of Objects	No. of Trips	Multi-Trip Objects in %	QRS	No. of Sensors
1	100-1000	2	10%	10%	1000
2	1000	1-10	10%	10%	1000
3	1000	2	10-100%	10%	1000
4	1000	2	10%	10-100%	1000

Table 1. Experiment settings

As shown in the first row of Table 1, for the first experiment the number of moving objects varies from 100 to 1000. 10% of the objects make two trips, while 90% are single-trip objects. The query rectangle size (QRS) is 10% of the total deployment area with 1000 sensors. Due to the constraints of our processing power, we had to limit both the number of objects and the number of sensors to 1000.

In Figure 4, we see a significant increase in the number of bytes transmitted for IDS (1816 bytes to 17963 bytes) when the number of objects increase (standard deviations are shown in all charts but may not be visible if they are small). However, the communication costs for SH and DEH are unchanged (508 bytes and 1016 bytes, respectively) and are much smaller than for IDS (9615 bytes in average). For this reason our communication cost charts use a logarithmic scale. A similar situation occurs for the memory costs (Figure 5). The number of bytes stored in IDS increases from 6576 to 46680 with an average of 26412, while the memory cost for SH and DEH remains at 4000 bytes and 15548 bytes, respectively. Note that IDS requires less storage than DEH if the number of objects is less than 300 since every sensor inside the query region has to keep several counts for DEH but may not observe any objects for small numbers of objects. DEH is more expensive than SH as it has to maintain a larger number of counts per sensor.

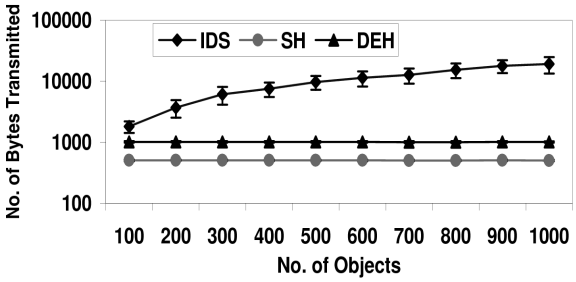

Figure 4. Communication costs

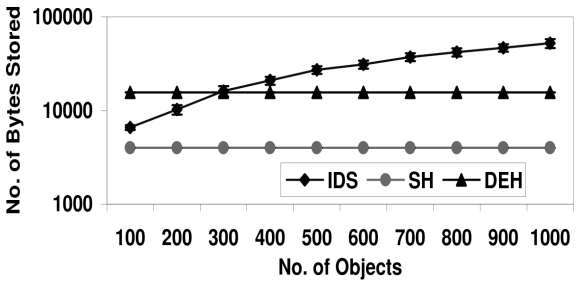

Figure 5. Memory costs

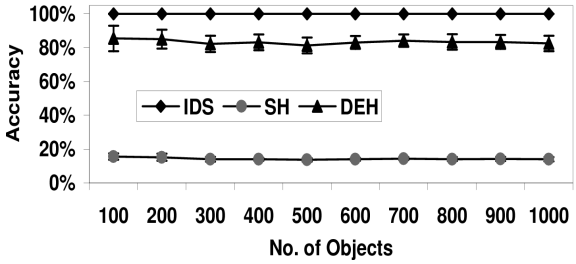

Figure 6. Accuracy for distinct object query

While IDS always achieves 100% accuracy for distinct object queries (Figure 6), DEH achieves an accuracy level of 81% to 85%. In contrast, while DEH always has a 100% accuracy for distinct entry queries (Figure 7), IDS achieves an accuracy level of 81% to 85%. Thus, the two approaches

are dual in terms of their behavior for the two query types. SH fails for both of the query types in terms of accuracy (average at 14% and 17%, respectively).

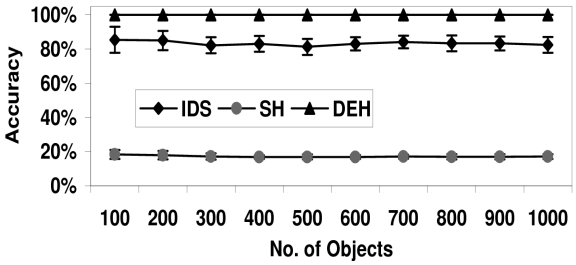

Figure 7. Accuracy for distinct entry query

4.2.2 Number of Trips

We also measured the impact of the number of trips on the performance. Our hypothesis is that the accuracy of DEH and SH in distinct object queries will be less than IDS and will decrease as the number of trips increases. For IDS and SH, the accuracy in distinct entry queries will be less than DEH and will drop as the number of trips increases. The experiment parameters are shown in row 2 of Table 1.

While IDS always achieves 100% accuracy for distinct object queries (Figure 8), the accuracy of DEH has an average at 76% and drops from 85% to 70% as the number of trips increases. This decrease is due to a larger number of disconnected entries to the query region from an increasing number of objects (creating similar examples as for the query rectangle R in Figure 3). Symmetrically, the accuracy of IDS has an average of 76% and drops from 85% to 70% for distinct entry queries (Figure 9), while DEH always has 100% accuracy.

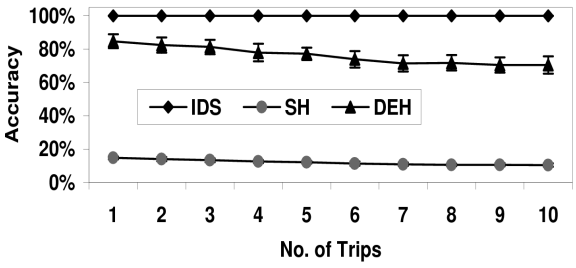

Figure 8. Accuracy for distinct object query

4.2.3 Percentage of Multi-Trip Objects

We studied the impact of multi-trip objects (the settings are shown on the third row in Table 1). The results confirm the results for increasing numbers of trips: while IDS always achieves 100% accuracy for distinct object queries, the accuracy achieved by DEH has an average at 77% and drops

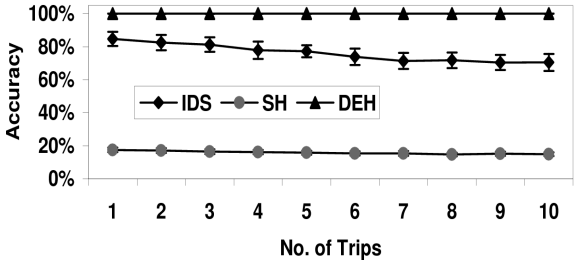

Figure 9. Accuracy for distinct entry query

from 82% to 73% as the percentage of multi-trip objects increases. Similarly for distinct entry queries, the accuracy achieved by IDS has an average at 77% and drops from 82% to 73% as the percentage of multi-trip objects increases, while DEH always achieves an 100% accuracy.

4.2.4 Query Rectangle Size

We also tested the impact of query rectangle size (QRS) on the performance of the systems. Our hypothesis is that the communication costs for all systems increases as the QRS increases. For DEH, the accuracy in distinct object queries is less than for IDS but will increase as the QRS increases. Conversely, for IDS the accuracy in distinct entry queries will not be as high as DEH but will increase as the QRS increases. The settings are given in the fourth row of Table 1.

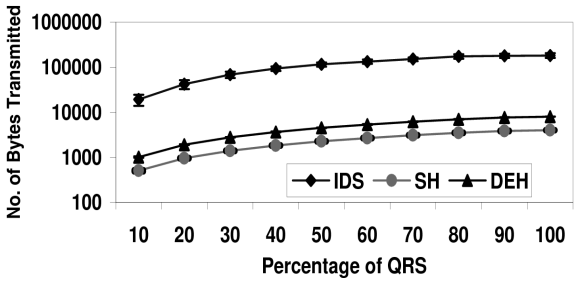

Figure 10. Communication costs

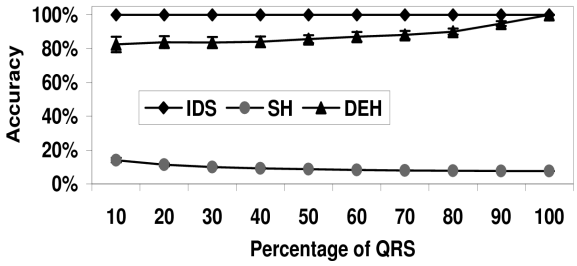

Figure 11. Accuracy for distinct object query

In Figure 10, we see an increase of communication costs in all systems as the QRS increases, because larger QRSs

cover more sensor nodes, which require more data to be transmitted. The cost for IDS increases from 19217 bytes to 182805 bytes with an average of 116218 bytes, the cost for DEH from 1012 bytes to 7992 bytes with an average of 4818 bytes, and the cost for SH from 506 to 3996 bytes with an average of 2409 bytes. While IDS always achieves 100% accuracy in distinct object queries (Figure 11), the accuracy achieved by DEH has an average at 88% and rises from 82% to 100% as the QRS increases. At the 100% level, there are no reentries to the query rectangle as all objects can only travel within the WSN. Correspondingly, DEH always achieves 100% accuracy for distinct entry queries (Figure 12), while IDS's accuracy has an average at 88% and rises from 82% to 100% as the QRS increases.

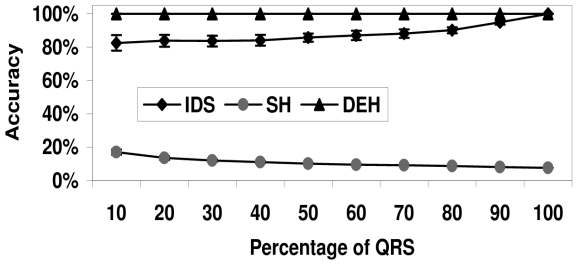

Figure 12. Accuracy for distinct entry query

5 Conclusions and Future Work

We introduced aggregate-query processing of moving object data in WSNs and the concept of DEHs as an efficient method to answer aggregate queries. We presented two basic query types, distinct entries to a region and the distinct number of objects in that region. We have shown that DEHs can decrease the communication costs by an order of magnitude under a large variety of realistic experimental settings compared to approaches based on maintaining moving object identifiers. We have also observed storage cost savings of more three times of what an RFID-based system achieves. Our method shows a high level of accuracy for the two basic query types, unlike simple histograms where answers can deviate significantly from the correct ones.

We plan to extend our method to allow for communication gaps in a WSN. A further extension are tracking scenarios where the extent of objects is important, such as the monitoring of toxic gas releases. In addition, we plan to address non-rectangular multi-dimensional range queries.

References

[1] www.j-sim.org.

[2] www.fh-oow.de/institute/iapg/personen/brinkhoff/generator.

[3] R. Beigel and E. Tanin. The geometry of browsing. In *LATIN'98*, pages 331–340, Campinas, Brazil, 1998.

[4] M. Demirbas and H. Ferhatosmanoglu. Peer-to-peer spatial queries in sensor networks. In *P2P'03*, pages 32–39, Linkoping, Sweden, 2003.

[5] D. Ganesan, B. Greenstein, D. Estrin, J. Heidemann, and R. Govindan. Multiresolution storage and search in sensor networks. *Trans. Storage*, 1(3):277–315, 2005.

[6] J. Gao, L. Guibas, J. Hershberger, and L. Zhang. Fractionally cascaded information in a sensor network. In *IPSN'04*, pages 311–319, Berkeley, CA, 2004.

[7] B. Gedik and L. Liu. MobiEyes: Distributed processing of continuously moving queries on moving objects in a mobile system. In *EDBT'04*, pages 67–87, Heraklion, Greece, 2004.

[8] H. Gonzalez, J., X. Li, and D. Klabjan. Warehousing and analyzing massive RFID data sets. In *ICDE'06*, page 83, Atlanta, GA, 2006.

[9] B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, and S. Shenker. DIFS: A distributed index for features in sensor networks. In *IEEE ICC'03, Workshop on Sensor Network Protocols and Applications*, pages 163–173, Anchorage, AK, 2003.

[10] P. Gupta and P. Kumar. The capacity of wireless networks. *Trans. Information Theory*, 46(2):388–404, 2000.

[11] Y. Hu, S. Sundara, T. Chorma, and J. Srinivasan. Supporting RFID-based item tracking applications in oracle DBMS using a bitmap datatype. In *VLDB'05*, pages 1140–1151, Trondheim, Norway, 2005.

[12] X. Li, Y. Kim, R. Govindan, and W. Hong. Multidimensional range queries in sensor networks. In *SenSys'03*, pages 63–75, Los Angeles, CA, 2003.

[13] I. Lopez and B. Moon. Spatiotemporal aggregate computation: A survey. *IEEE TKDE*, 17(2):271–286, 2005.

[14] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: A Tiny AGgregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, 2002.

[15] A. Meka and A. Singh. DIST: A distributed spatio-temporal index structure for sensor networks. In *CIKM'05*, pages 139–146, Bremen, Germany, 2005.

[16] M. Mokbel, X. Xiong, and W. Aref. SINA: Scalable incremental processing of continuous queries in spatio-temporal databases. In *SIGMOD'04*, pages 623–634, Paris, France, 2004.

[17] S. Nath, P. Gibbons, S. Seshan, and Z. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *SenSys'04*, pages 250–262, Baltimore, MD, 2004.

[18] S. Saltenis, C. Jensen, S. Leutenegger, and M. Lopez. Indexing the positions of continuously moving objects. In *SIGMOD'00*, pages 331–342, Dallas, TX, 2000.

[19] Y. Tao and D. Papadias. Time-parameterized queries in spatio-temporal databases. In *SIGMOD'02*, pages 334–345, Madison, WI, 2002.

[20] X. Xiong, M. Mokbel, and W. Aref. SEA-CNN: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In *ICDE'05*, pages 643–654, Tokyo, Japan, 2005.

[21] S. Yoon and C. Shahabi. Exploiting spatial correlation towards an energy efficient clustered aggregation technique (CAG). In *IEEE ICC'05*, pages 82–98, Seoul, Korea, 2005.