In-Networks Spatial Query Estimation in Sensor Networks

Silvia Nittel, Guang Jin, Yoh Shiraishi
National Center for Geographic Information and Analysis
Spatial Information Science and Engineering
University of Maine
Orono, ME 04469-5711, USA
nittel, jin@spatial.maine.edu

Abstract

Recently technology developments enable the production and deployment of sensor networks that provide outstanding ability to monitor discrete and continuous phenomena in physical space. From a database perspective, a sensor network can be seen as a virtual distributed database system (DBS) with sensor nodes that are tiny DBS themselves. Thus, a user can interact with the sensor network as a whole, and send queries to it. The sensor nodes run tiny footprint DBS locally, and participate in global query execution. Nowadays, spatial queries over sensor network mostly retrieve discrete information measured at the location of sensor nodes. In environmental applications, however, the estimation of continuous phenomenon such as a toxic cloud or a temperature field is of great interest. Thus, strategies need to be available to evaluate spatial queries about continuous phenomena, and estimate results based on discrete sensor measurements. Kriging is a traditional spatial interpolation method with a guaranteed minimal estimation error, and a smooth, unbiased estimation. However, its execution is computationally expensive. In this paper, we present an approach to Kriging that allows inexpensive in-network execution to evaluate queries over a sensor network. Our approach, called QUAKE, optimizes the selection of sample size and sensor nodes for the requested estimation value of any point location in the sensor network.

1 Introduction

From telescopes to microscopes, we have developed instruments to monitor and observe the world in ways that are not obvious to the human eye. With recent and projected advances in small, low cost microelectronic and mechanical systems (MEMS) with limited on-board processing and wireless communication capabilities, and the development of new sensor materials, we can envision a new generation of technology that enables us to build large collections of untethered, battery powered sensors with various sensing functions that are distributed densely over a geographic region. These sensor networks are able to measure traffic conditions, weather development, seismic activity, or track the movements of toxic fumes at a level of detail that was not possible before. The continued trend towards miniaturization and inexpensiveness of sensors makes it possible that such sensor networks are less than a cubic millimeter in size[12], and sensor networks are made up of thousands or even millions of sensors. At that scale, wireless sensors could permeate the physical world, and help us vastly increase to increase our understanding of our physical environment.

Sensor networks, however, have the following constraints that pose new challenges from a system and application development standpoint:

- **Power consumption:** Sensor nodes are limited with regard to their battery supply, and energy conservation is a major system design principle. Also, communication is a much larger battery drain than local computation on a node.
- **Low-range communication:** The bandwidth and range of wireless communication is limited. Since communication is much higher drain on the energy consumption than on-board processing, optimizing communication within the sensor network is a major system design consideration.

- **Limited computing and storage capabilities:** Sensor nodes have, at least, for the foreseeable future limited on-board computational, and volatile and persistent storage capabilities. Thus, on-board data processing of using available memory and CPU is also limited.
- **Self organization**: Due to the large number of sensor nodes, the failure rates of nodes, and the often unattended deployment, task management and handling in sensor networks is decentralized and self organizing. Thus, some level of local autonomy must be provided for the devices.

Over the last years, much research work has focused on the design of robust, and energy-efficient communication protocols and the development of sensor materials and devices. More and more research and commercial prototype platforms become available to be deployed in real-world applications. First prototypical employments of sensor network can be found in the environmental domain, such as non-invasive habitat monitoring [3, 4]. From the application developer perspective, the main purpose of sensor networks is *collection, aggregation and processing of data* as well as the actuation of the environment. Thus, today simple, easy-to-use programming interfaces for sensor networks become important. A user, often a domain scientist, would like to define the necessary tasks in a user-friendly way, and delegate the optimization and ultimately self-adaptive execution to the run-time environment without having to worry about the details.

Traditionally, database management systems (DBMS) have had the purpose of making managing and querying very large amounts of data simpler and robust. The current generation of DBMS, based on extended relational database technology, is well-understood, and this type of DBMS is used almost pervasively in business as well as in scientific applications. Relational DBMSs provide a simple, query language, SQL, to model the data and formulate queries over the data. The mathematical foundation of SQL on the relational algebra enables automated reformulation and optimization of user queries to speed up the execution of queries significantly.

From a DBMS perspective, a sensor network can be viewed as a *large distributed database system*. Here, each sensor node behaves like a tiny DBMS running on the sensor node platform and accepts, processes and answers queries. Furthermore, a sensor node DBMS also participates in the global execution of distributed queries. Consequently, the user can interacting with the sensor network as a whole.

For example, imagine the storm petrel sensor network application on Great Duck Island in Maine (USA) conducted by scientists from UC Berkeley, Intel Research Labs and the College of the Atlantic in Bar Harbor [3]. Great Duck Island is a 90-hectare island consisting of rock and grass off the coast of Mount Desert Island and is home to one of the world's largest breeding colonies of Leach's storm petrels. Researchers installed a collection of monitoring devices (motes) in the petrels' nesting burrows with sensors that monitor light, humidity, pressure, and heat. The data is transmitted via a radio transceiver to nearby motes. The available sensor data is modelled via a relation called *Sensors* whose schema consists of the node identifier, attributes for a node's on-board sensors such as light and temperature, and a time stamp for each data tuple. The tuples are created on demand by nodes in the sensor network based on queries sent to the sensor network. Sensors nodes only start sampling based on the query.

For example, a scientific observer might be interested in **periodic measurements** of pressure values to monitor the inhabitation of birds in nests with the following SQL query:

```
SELECT sensor.id, pressure FROM sensors WHERE pressure>threshold
SAMPLE PERIOD 60sec
```

Every 60 seconds, the query is evaluated at all participating sensor nodes, and those nodes with pressure values larger than the given threshold return a data tuple, and others return a null value. Instead of periodic observations, the scientist might only look for **events** such as ”when and how long does a storm petrel occupy a burrow?”. Events can be derived from collected data outside of the sensor network based on periodic measurements or, they can be processed directly in the sensor network. A third group of queries are **aggregation queries** such as simple aggregates of ”how long are nests occupied on average?” or a query to find the nest with the highest weight:

```
SELECT sensor.id, MAX(pressure) FROM sensors WHERE
pressure>threshold SAMPLE PERIOD 60sec
```

Note, that queries define the request data in a declarative way, and the user does not need to know the the availability or node identifier of sensors, nor does he/she deal with the details of the execution of the query. The DBMS routes the query to the sensor nodes of interest. System-internally, multi-hop georouting or other routing algorithms ([1, 2]) can be applied to

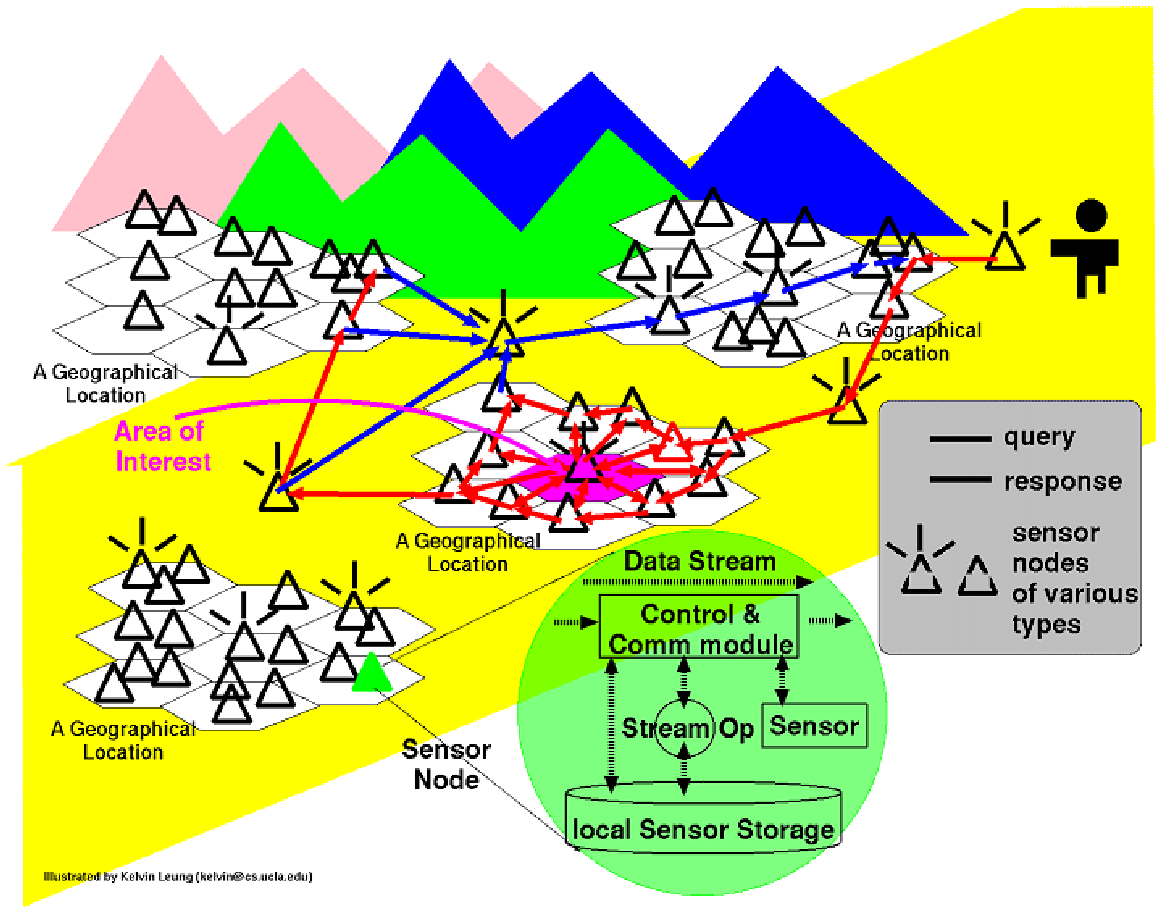

Figure 1: Sensor networks as distributed database system

distribute the query to all relevant nodes, which contribute their measurements as partial query results, and routing algorithm to intelligently and energy-efficiently route partial result back to the usere.

In summary, the sensor network database system translates the declarative user query, and responsible for generating a query execution and optimization plan, and control the query execution as well as its error handling. Although, a sensor network database system provides a similar, easy-to-use interface compared to the traditional database world and its data handling paradigm is promising when dealing with the complexity of data collection and processing in sensor networks, the database technology applied 'under the hood' of the front-end changes significantly from traditional database management systems strategies to deal with the constraints of sensor networks.

2 Database Management Systems for Sensor Networks

Today, first prototype implementations of sensor network database systems are available ([15, 5]). For example, the footprint of TinyDB's run-time is 3 KB which easily fits on a mote's memory which maxes out at 128 KB for programs stored in Flash ROM and 4 KB of RAM. The interface to TinyDB runs on a PC or laptop that communicates with the sensor network.

Overall, the objectives of sensor network database management system are to achieve a

- high-level abstraction of data and tasks,
- simple, declarative user interfaces for aggregated data retrieval tasks,

- and efficient, automated query optimization and robust, energy-efficient, and adaptive execution of queries in the sensor network.

Today's DBMS for sensor network are a light-weight version of DBMS; the query syntax is similar to SQL-style queries that are extended with sampling epochs for long running, continuous queries. Joins over several tables are not available, and all the data is managed in a single relational table. Tuples for this table are created in an append-only style, and tuples are contributed by all nodes in the sensor network. User pose queries to this relation in order to select all tuples or only tuples that qualify for a certain query predicate as specified in the WHERE clause. For example, the query

```
SELECT sensor_node.id, temperature FROM sensors WHERE
temperature>threshold SAMPLE PERIOD 60sec
```

is sent to all nodes in the sensor network, and all nodes start sampling temperature values every 60s. However, only those tuples with a temperature measurement above the specified threshold are routed back to the user. Similarly, an *aggregate condition* can be defined on an attribute in the SELECT clause; for example, a user can request ”SELECT MAX(temperature) FROM....”. In this case, only the maximum temperature reading within the entire sensor network per sampling interval is reported back to the user. However, all sensor nodes need to produce tuples, and collaborate in the network on which tuples to forward, and which tuples to drop since they are superseded by a higher measurement. A simple algorithm is to use the hierarchical routing tree, and appoint the parents as evaluators of their child nodes' tuples. Thus, the leaf nodes send their measurements to their parent node, which forwards the tuple with the highest temperature value to its own parent node [15].

Overall, the SELECT part of a query allows to specify the *attributes* of interest (e.g. temperature, wind speed, etc.), and aggregates such as *MIN*, *MAX*, or *AVG* for such attributes. The WHERE clause defines *qualifying conditions* for attributes of the table. If the predicate is spatial, we deal with a *spatial query*. For example, in the query ”SELECT AVG(temperature) FROM sensors WHERE region=”3rdfloor” the query evaluation is limited to a subregion of the sensor network, i.e. the nodes on the third floor.

Furthermore, we distinguish between *snapshot queries* and *continuous queries*. A snapshot query retrieves a single measurement from all nodes in the sensor network, while a continuous query produces periodic query results for the specified epochs.

2.1 In-Network Data Aggregation

Overall, SQL-style queries for sensor networks are rather simple today. Query processing involves the translation of declarative query into query execution plans, and in-network query processing. The goal of in-network processing is to reduce the amount of data that is sent through the network, and eliminate disqualified tuples early on, or process partial results at intermediate nodes. Due to energy, bandwidth, and sleep cycle constraints, queries over sensor nodes are executed in two phases: a) in a *distribution phase* the query is routed and distributed to all sensor of interest in the network, and b) in a *collection phase* partial query results are routed back to the query originating node. More detail on the coordination between parent and child nodes during the collection phase can be found in [15].

Today, several techniques are investigated to reduce the amount of data that is routed through the network. One strategy is to reduce and compress that data directly at a sensor node (e.g. using Kalman Filters [**?**]), or the extraction and encoding of a base signal from a locally stored time series of measurements [**?**]. Another strategy is to store historic data in the network in more powerful sensor nodes.

One of the major challenges of automated query translation and query execution is the creation of 'aggregation trees' for complex aggregate queries [**?**]. A sketch of an execution plan for a MAX aggregate was presented above. However, coming up with execution plans for more complex aggregates such as a computing the median, counting distinctive values or value estimation for non-sampled areas is less straight-forward.

3 Spatial Estimation Queries in Sensor Networks

Most data queries over sensor networks are spatial in their nature due to the embedding of sensor nodes in the physical world. In current work, spatial queries mostly coincide with retrieving the values at the sensor nodes, and assuming that a value is representative 'enough' for the space it is located in, e.g. *”Select temperature FROM sensors WHERE room.id=”334”*.

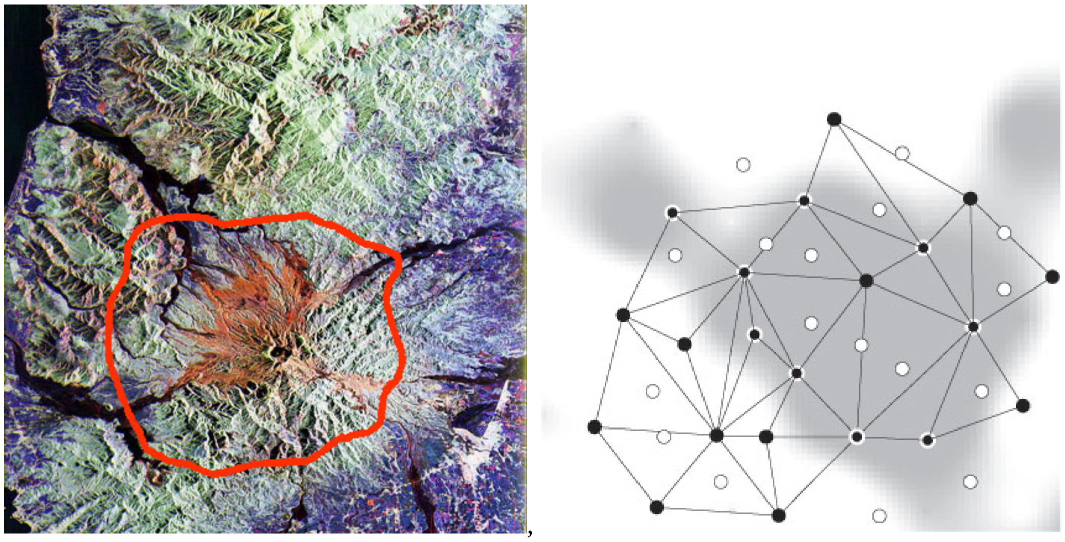

Figure 2: Approximating Spatial Queries for Continuous Phenomena

However, for environmental applications another type of queries is of great importance. Environmental applications, a major application area of geosensor networks [**?**], deal with the observation and monitoring of continuous phenomena (or dynamic spatial fields) such as temperature fields or toxic fumes. Here, the *estimation* of data values in space that are located *between* actual sensor nodes are an important class of sensor network queries. From a database perspective, a interpolation or estimation query can be considered a complex aggregates and is implemented using user-defined functions.

3.1 Querying Continuous Phenomena

Defining a continuous phenomenon formally, we can say that a spatial scalar field represents the variation of some scalar property over a region of space (e.g. wind-speed, or a gas pollutant in the air). Thus, a spatial field is defined as a function from space to a scalar property. Spatial fields that can change through time. Therefore, a *dynamic spatial scalar field* is defined as a function f from a temporal domain T to a spatial field $S \rightarrow V$, $f : T \rightarrow S \rightarrow V$. The temporal domain T may be consist of instants or intervals, and may be linearly or partially ordered.

For applications such as tracking toxic clouds, queries about the underlying continuous phenomenon such as approximating its contour line or areas of similar values are of high interest, and cannot be directly assessed from the discrete sensor readings (see Figure 2), but have to be interpolated.

3.2 Data Interpolation Methods

In the literature, several models and methods have been introduced to approximate values at non-sampled sensor locations. A simple form of estimation is the use of Voronoi diagrams [8, 19]. Via Voronoi diagrams, a region is divided into cells, i.e. a polygon is calculated around each sensor such that all points inside the polygon are spatially closer to the sensor node location than to any other nodes in space. Using Voronoi diagrams to answer a point query, the value of the closest sensor node based on the Voronoi space is selected. A similar approach can be built using Delaunay triangulation. These types of approximation are coarse, and information about the estimation error is not available.

Diverse probabilistic estimation methods with smoother and more accurate estimation have been established in the area of spatial data analysis. For example, Kriging [13] is an estimation method that is used to predict the value of a point in space of a continuous phenomenon by weighing the contribution of sensor samples to the estimated value, and it provides well-defined error guarantee. Kriging also contains information to evaluate the quality of the estimation result, i.e. quantify the error. Several variations of Kriging are available; ordinary Kriging [**?**] has the benefit that it estimates the first order and second order effects of the monitored phenomenon both in one process.

3.3 In-Network Estimation of Data Values

Traditional Kriging is a computationally complex consisting of an iteration of matrix computations. Several ‘extreme’ models for a Kriging computation of sensor network data can be considered: The first approach is to issue a ’snapshot query’ to the sensor network, and compute Kriging outside of the sensor network. Similarly, a single sensor node in the network can take over the role of a ‘Kriging node’, and perform $(n+1) \times (n+1)$ matrix computations locally. In this case, all measurements of all other nodes are sent to this node, and the node has to be computationally powerful. In a third model, traditional Kriging is computed in a distributed way in the network, by applying the iteration algorithms to solve the linear equations of Kriging similar to an approach presented in [7]. To solve the weight matrix, however, since V_+ is not sparse, the communication load would be an extreme burden for the sensor network.

Traditional Kriging is designed to calculate the value of single points in space with minimal standard error based on all sensor measurements. However, it is obvious that nodes that are in large spatial distance to a point have less influence on the estimation value than sensor measurements in close proximity.

4 In-Network Spatial Query Estimation using Adapted Kriging

In this paper, we introduce a computational approach for high quality data estimation that is processed incrementally in the sensor network. In detail, we introduce the QUAKE approach (Query Answering Using Adapted Kriging Estimation), which encompasses a novel algorithm for ‘in-the-network’ Kriging computation that minimizes the acquisitional cost of query processing for spatial queries. The approach taken in QUAKE is based on calculating the minimal subset of sensor nodes, which are necessary to compute an estimation result that guarantees the user-defined acceptable error tolerance for a query. QUAKE significantly decreases the number of messages sent within the sensor network, and assures that the message size between nodes stays constant throughout estimation process.

4.1 A “sensor” can represent the entire network

Based on a set of sensor nodes, Kriging returns the estimated value and error for a point of interest. We make the following assumption: if Kriging can return the same estimation value and error based on a single point, this point actually represents the entire set of sensor nodes. We use this assumption iteratively, until we actually ’grow’ from the single virtual point to a minimal set of sensor nodes that allows us to reach the user-defined acceptable error. Based on a given variogram model γ for a monitored phenomenon, its reversed function γ^{-1} can be calculated, which returns the spatial distance based on a known variance value. Using a set of sensor nodes, Kriging returns the estimation result for the query point as $\hat{Y}(q)$ with the standard error S_{z_0}. As said, we make the initial assumption that a single ‘representing point’ p can represent the entire sensor network, if the following equations are satisfied:

$$\begin{aligned} S_{z_0} &= \sqrt{\gamma(|p-q|) + \lambda} \\ \gamma(0) + \lambda &= \gamma(|p-q|) \end{aligned}$$

In this case, the variance value $\gamma(|p-q|)$ between the point p and the query point q can be calculated. Via the reversed variogram model, γ^{-1}, we can also calculate the distance between them. In this case, the location of this assumed ’virtual sensor’ is presented as a *circle* around the query point q. The idea is shown in Fig. **??**.

4.2 The QUAKE computation tree

Based on the introduced idea of a virtual representing point, we create a ‘QUAKE computation tree’, which is a conceptual data estimation or incremental data aggregation tree used for the Kriging estimation. It selects the sensor nodes which are in the proximity of the query point, and uses them to compute partial estimation results. It decides whether the tree needs to grow larger based on the results of the partial estimation. Every node is awoken by its parent except for the root node which is awoken by the query and is also the closest point to the query point. A tree has 3 node types: one root node, internal nodes, and leaf nodes.

4.2.1 Creating the QUAKE Tree

Once a query is submitted to a sensor network, the QUAKE tree is created in the following steps:

1. The query node awakes the root. The root assigns 1 as its weight to the Kriging matrix, and 0 to other sensor nodes. Based on equations **??** and **??**, it calculates the estimated value and standard error. If the estimated standard error can satisfy the user-defined error tolerance, then the computation terminates at this point. In fact, a Voronoi diagram around the sensor node is returned as a query result.

   If not, the root awakens n next nearest neighbor sensor nodes to the query point as its children.

2. After being awoken by its parent, a leaf returns its sensed value with its location to its parent.

3. After a parent (an internal node or the root), i, receives the messages from its children, it performs the Kriging estimation among itself and its children. Based on the above discussion, we can determine a single representing point p for the family of this parent node and its children$c1$ and $c2$, and a similar estimation result is returned. As mentioned before, the representing 'point' is a point on a circle around the query point. To minimize the variance introduced by further computations, QUAKE chooses the nearest point on the circle as the representing point p for the previously calculated family of nodes.

   For example, in Fig. **??**, the internal node s_i notifies its parent with its location, its representing point p and its local estimation result, $w_iY(s_i) + w_{c1}Y(s_{c1}) + w_{c2}Y(s_{c2})$.

4. Similar to 2, after the children of the root send their local estimation result $P1$ and $P2$ to the root (Figure 2b), the root calculates the new estimated standard error and the estimated value. If the new error cannot satisfy the query-defined error tolerance, the root notifies its children to awake more sensors. If the user tolerance is satisfied, the root sends a prune notification to its children.

5. When a leaf receives the notification from its parent to awake more sensors, it wakes up n next nearest sensor nodes to the representing point as its children. Also, this leaf will be a new internal node of the tree.

6. The cycle begins again from 2.

As we discussed, the Quake computation tree can grow exponentially if the number of children per parent is larger than 1. It is possible that several unnecessary sensor nodes may be invoked which are no longer needed to satisfy the error tolerance. Pruning happens during establishing the QUAKE tree. Our tests demonstrated that trees with two children provide the best performance with regard to estimation, number of message and energy conservation.

4.3 Performance Results

The simulation was implemented using two types of data sets. The first data set is the 18 measurements of zooplankton density. The data set has been collected off the coast of Southern California in the CalCOFI survey. We tested QUAKE's performance using one child to four children while building the QUAKE tree.

To demonstrate the difference between the incremental QUAKE approach and a conventional, full-blown, centralized Kriging we used the case of an expensive, but smooth estimation case. We performed the tests using both methods by applying the same tolerance settings in both cases. In Fig. 3, the lighter color indicates higher estimated density of zooplankton, and darker lighter color indicates lower density areas when using QUAKE to estimate the area. As we expected, larger tolerance settings return coarser estimation results. Our results revealed that although for smaller tolerance levels in QUAKE results are are slightly coarser than Kriging, QUAKE still catches the main trends of the phenomenon, and is close to the performance of centralized Kriging at higher tolerance levels.

5 Conclusions and Outlook

Programming and managing data in sensor networks is a complex problem. Database system technology provides beneficial features to interface with sensor networks: data modelling and data queries are defined with high-level, application-oriented languages, while the database management system is responsible for under the hood optimized, adaptive, and robust query

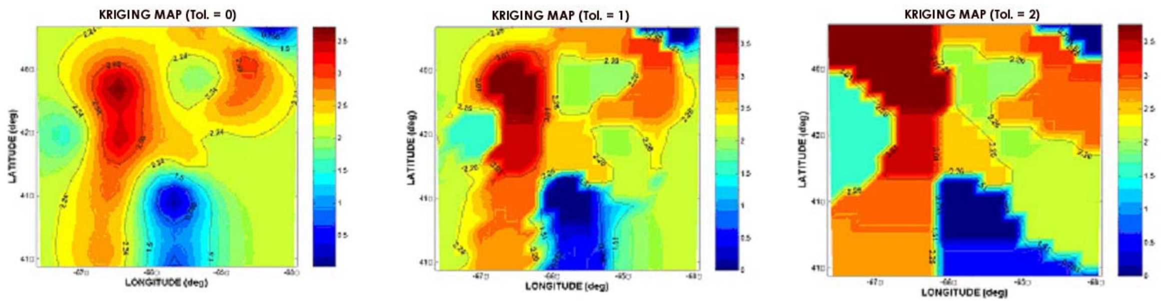

Figure 3: Estimation Results

execution. Naturally, optimization programmed into DBMS are not less complex than other sensor network applications; however, being part of the information system, they are reusable for users' applications while keeping application programming simpler. In the database research field, long-established and understood technology is available, however, sensor network-based environments introduce new research problems that require rethinking and adapting database technology. Due to the application domain of sensor networks, also learning from other domains such as geographic information systems and especially temporal-spatial database systems are ultimately necessary.

Acknowledgements

This work was partially supported by the National Science Foundation under NSF grant number EPS-9983432.

References

[1] Braginsky, D. and Estrin, D., *Rumor Routing Algorithm For Sensor Networks*, First Int'l Workshop on Wireless Sensor Networks and Applications, Atlanta, Georgia, 2002.

[2] Intanagonwiwat, C., Govindan, R. and Estrin, D., *Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks*, Proc. of ACM Conference Mobile Computing and Networking (MobiCOM), Boston, MA, August, pp. 56-67, 2000.

[3] Mainwaring, A., Polastre, J., Szewczyk, R. and Culler, D., *Wireless sensor networks for habitat monitoring*, First Int'l Workshop on Wireless Sensor Networks and Applications, Atlanta, Georgia, September 28, 2002.

[4] Cerpa, A., Elson, J. , Estrin, D., Girod, L., Hamilton, M. and Zhao, J., *Habitat monitoring: Application driver for wireless communications technology*, Proc. of the 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean, April, 2001.

[5] Bonnet, B., Gehrke, J. and Seshadri, P., *Towards Sensor Database Systems*, Proc. of 2nd Int'l Conference Mobile Data Management (MobiDE), Hong Kong, January, 2000.

[6] Bonnet, B. and Seshadri, P., *Device Database Systems*, Proceedings of the 16th International Conference on Data Engineering, San Diego, CA, 2000.

[7] Delouille, V. and R. Neelamani, R. and Baraniuk, R., *Robust Distributed Estimation in Sensor Networks using the Embedded Polygons Algorithm*, International Conference "Information Processing in Sensor Networks", Berkeley, CA, 2004.

[8] Ganeriwal, S., Han, C.C. and Srivastava, M., *Going Beyond Nodal Aggregates in Sensor Networks*, Technical Reporst, NESL, EE Department, UCLA, August 2004.

[9] Gilbert, A.C., Kotidis, Y., Muthukrishnan, S. and Strauss, M.J., *Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries*, Proc. of the 27th VLDB Conference, Rome, Italy, 2001.

[10] Lam, N., *Spatial Interpolation Methods: A Review*, The American Cartographer, Vol. 10(2), 1983, pp. 129-149.

[11] I. Lazaridis, I. and S. Mehrotra, *Capturing Sensor-Generated Time Series with Quality Guarantees*, Int'l Conference on Data Engineering (ICDE), Bangalore, India, 2003.

[12] Kahn, J.M. and Katz, R.H. and Pister, K.S.J., *Next Century Challenges: Mobile Networking for "Smart Dust"*, International Conference on Mobile Computing and Networking (MOBICOM),1999, pp. 271–278.

[13] Oliver, M. and Webster, R., *Kriging: A Method of Interpolation for Geographic Information Systems*, Int'l J. Geographic Information Systems, Vol. 4(3), 1990, pp. 313-332.

[14] S. Madden, M. Franklin, J. Hellerstein, W. Hong, *The Design of an Acquisitional Query Processor For Sensor Networks*, Int'l Conference on Data Management (SIGMOD), June 10-12, San Diego, CA, 2003.

[15] Madden, S., Franklin, M., Hellerstein, J. and Hong, W., *TAG: Tiny AGgregate queries in ad-hoc sensor networks*, Proc of the USENIX Symposium on Operating Systems Design and Implementation, Boston, MA, USA, 2002.

[16] Stefanidis, A., and Nittel, S., *GeoSensor Networks*, CRC Press, August, 2004.

[17] Yao, Y. and Gehrke, J., *The Cougar Approach to In-Network Query Processing in Sensor Networks*, SIGMOD Record, Vol. 31(3), 2002.

[18] Ratnasamy, S., Karp, B., Yin, L., Yu, F., Estrin, D., Govindan, R. and Shenker, S., *GHT: A geographic hash table for data-centric storage*, Proc of the ACM Workshop of Sensor Networks and Applications, Atlanta, GA, September, 2002.

[19] Sharifzadeh, M. and Shahabi, C., *Supporting Spatial Aggregation in Sensor Network Databases*, Int'l Conference ACM-GIS, Washington, D.C., 2004.

[20] Worboys, M. F., *Computation With Imprecise Geospatial Data*, Computers, Environment and Urban Systems, Vol. 22(2), pp. 85-105, 1998.