Assessing the Completeness of Sensor Data

Jit Biswas[1], Felix Naumann[2], and Qiang Qiu[1]

[1] Institute for Infocomm Research (I^2R), Singapore
{biswas, qiu}@i2r.a-star.edu.sg
[2] Humboldt-Universität zu Berlin, Germany
naumann@informatik.hu-berlin.de

Abstract. In this paper we present a quality model highlighting the completeness of sensor data with respect to its application. The model allows consistent handling of information loss as data propagates through a sensor network. The tradeoffs between various factors that influence completeness are quantified thereby allowing an integrated view of completeness at various levels in a system. The paper is presented in the context of the fast emerging field of *smart spaces*. All concepts in the paper have a foundation in real-life problems arising in this context. Preliminary implementation results are presented to illustrate the value of the completeness based approach versus one that does not use completeness.

1 Introduction and Motivation

Applications in the fast emerging field of smart spaces [1, 2] use data collected and aggregated from sensors of many modalities to monitor entities and environments and assist in decision making. Information rich sensor data, such as image and audio, is used in conjunction with basic context information such as location, identity, and time to carry out classification, inferencing, and other categories of recognition tasks.

A major problem in such systems is information- or data-loss. Data may be lost at many stations from the sensor to the application. Besides the sensors themselves, losses may arise from the transmission medium, e.g., radio attenuation or network congestion. In this paper we present a quantitative model for such loss. The model is based on an intuitive notion of sensor data completeness that measures the amount of data reaching a point of consumption compared to the maximum amount of data possible at that point. In related work [3] we have investigated data loss that arises from delay or congestion within the networking layer, and present an admission control scheme for regulation of such loss.

1.1 A Smart Home Application

We consider as an example, an application that analyzes the pacing behavior of elderly dementia patients in their homes [4]. This application relies on sensor readings that need to be updated and reasoned about rapidly, while at the same time analyzing the knowledge thus generated to answer complex high level queries about the patient. For example, in connection with the home of a dementia patient (Fig. 1), a doctor might ask "Does the person become highly agitated while he is exhibiting abnormal pacing behavior in the living room?" [5]. An answer to this query involves accessing a set of location sensors that determine position and can accurately classify pacing behavior from other forms of movement, and a number of body worn accelerometers, which generate 2D or 3D instantaneous acceleration data. From this data it is possible to infer through some high-level pattern classification algorithms, what was the type of behavior that the person was exhibiting.

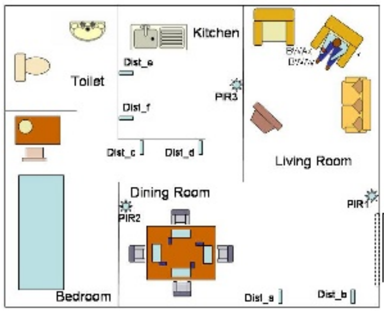


Fig. 1. Home Layout

Each of three rooms of Fig. 1 is equipped with a passive infra-red sensor PIR1, PIR2 and PIR3, passing information to a proxy (Fig. 2), whether a person is in that room.

Because we assume only a single person in the house, the proxy translates this data into a single attribute stating the room in which the person is. In addition, each room is equipped with two ultrasound distance sensors (D_a, D_b, ..., D_f), which measure the distance of some object to the sensor. Usually, the distance of the opposite wall is reported, but when the person is in the room, the sensors report the distance to that person. The data from the two sensors of each room are passed to the proxy, which derives the position of the person in the room. Finally, the person is equipped with an on-body sensor, namely a 2D body worn accelerometer, which generates two readings BWAx and BWAy.

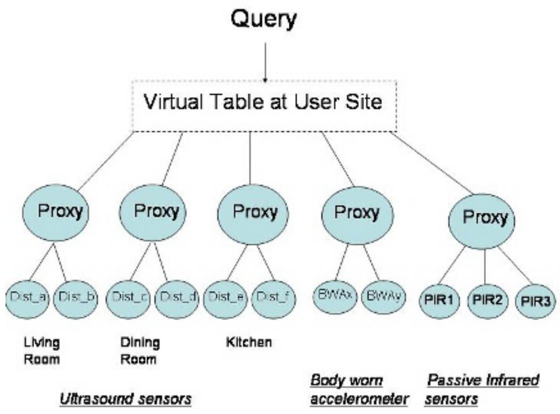
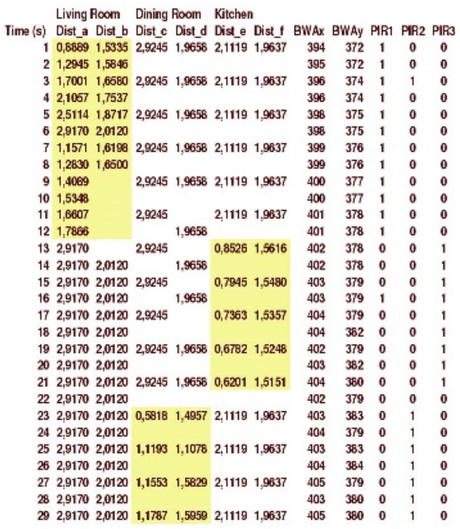


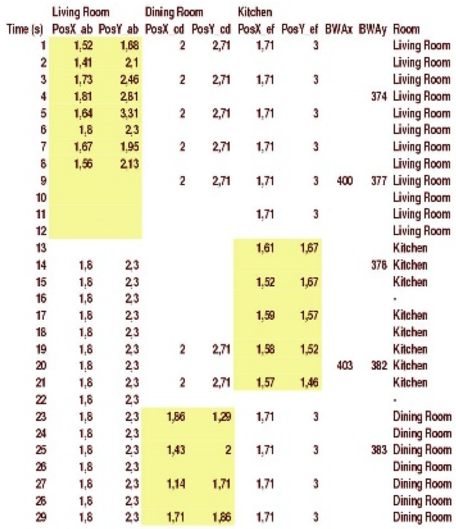
Fig. 2. An Example Sensor Query System Scenario showing the Virtual Table

Tab. 1 (left) shows the raw data as produced by the sensors. We assume that the smallest granularity time period (system data rate) for this application is 1 second. We will measure completeness as the ability of the overall system to produce data values for each relevant measure in each time period. The following completeness-relevant observations can be made from Tab. 1:

Table 1. Raw data from sensors (left) and data produced by the set of proxies (right)



	Living Room		Dining Room		Kitchen						
Time (s)	Dist_a	Dist_b	Dist_c	Dist_d	Dist_e	Dist_f	BWAx	BWAy	PIR1	PIR2	PIR3
1	0,6689	1,5335	2,9245	1,9656	2,1119	1,9637	394	372	1	0	0
2	1,2945	1,5846					395	372	1	0	0
3	1,7001	1,6580	2,9245	1,9658	2,1119	1,9637	396	374	1	1	0
4	2,1057	1,7537					396	374	1	0	0
5	2,5114	1,8717	2,9245	1,9658	2,1119	1,9637	398	375	1	0	0
6	2,9170	2,0120					398	375	1	0	0
7	1,1571	1,6198	2,9245	1,9658	2,1119	1,9637	399	376	1	0	0
8	1,2830	1,6500					399	376	1	0	0
9	1,4089		2,9245	1,9658	2,1119	1,9637	400	377	1	0	0
10	1,5348						400	377	1	0	0
11	1,6607		2,9245		2,1119	1,9637	401	378	1	0	0
12	1,7866			1,9658			401	378	1	0	0
13	2,9170		2,9245		0,8526	1,5616	402	378	0	0	1
14	2,9170	2,0120		1,9656			402	378	0	0	1
15	2,9170	2,0120	2,9245		0,7945	1,5480	403	379	0	0	1
16	2,9170	2,0120		1,9656			403	379	1	0	1
17	2,9170	2,0120	2,9245		0,7363	1,5357	404	379	0	0	1
18	2,9170	2,0120					404	382	0	0	1
19	2,9170	2,0120	2,9245	1,9658	0,6782	1,5248	402	379	0	0	1
20	2,9170	2,0120					403	382	0	0	1
21	2,9170	2,0120	2,9245	1,9656	0,6201	1,5151	404	380	0	0	1
22	2,9170	2,0120					402	379	0	0	0
23	2,9170	2,0120	0,5818	1,4957	2,1119	1,9637	403	383	0	1	0
24	2,9170	2,0120					404	379	0	1	0
25	2,9170	2,0120	1,1193	1,1076	2,1119	1,9637	403	383	0	1	0
26	2,9170	2,0120					404	384	0	1	0
27	2,9170	2,0120	1,1553	1,5829	2,1119	1,9637	405	379	0	1	0
28	2,9170	2,0120					403	380	0	1	0
29	2,9170	2,0120	1,1787	1,5959	2,1119	1,9637	405	380	0	1	0



	Living Room		Dining Room		Kitchen				
Time (s)	PosX ab	PosY ab	PosX cd	PosY cd	PosX ef	PosY ef	BWAx	BWAy	Room
1	1,52	1,88	2	2,71	1,71	3			Living Room
2	1,41	2,1							Living Room
3	1,73	2,46	2	2,71	1,71	3			Living Room
4	1,81	2,81						374	Living Room
5	1,64	3,31	2	2,71	1,71	3			Living Room
6	1,8	2,3							Living Room
7	1,67	1,95	2	2,71	1,71	3			Living Room
8	1,56	2,13							Living Room
9			2	2,71	1,71	3	400	377	Living Room
10									Living Room
11					1,71	3			Living Room
12									Living Room
13					1,61	1,67			Kitchen
14	1,8	2,3						378	Kitchen
15	1,8	2,3			1,52	1,67			Kitchen
16	1,8	2,3							-
17	1,8	2,3			1,59	1,57			Kitchen
18	1,8	2,3							Kitchen
19	1,8	2,3	2	2,71	1,58	1,52			Kitchen
20	1,8	2,3					403	382	Kitchen
21	1,8	2,3	2	2,71	1,57	1,46			Kitchen
22	1,8	2,3							-
23	1,8	2,3	1,86	1,29	1,71	3			Dining Room
24	1,8	2,3							Dining Room
25	1,8	2,3	1,43	2	1,71	3		383	Dining Room
26	1,8	2,3							Dining Room
27	1,8	2,3	1,14	1,71	1,71	3			Dining Room
28	1,8	2,3							Dining Room
29	1,8	2,3	1,71	1,86	1,71	3			Dining Room

- The ultrasound sensors have long stretches of almost constant data. During those periods they are measuring their distance to the opposite wall.
- Sensor D_b stopped working for seconds 9 - 13.
- At second 22 no passive infrared sensor detects a person. This can happen during transition of the person from one room to the other.
- Sensor D_d has a too fast internal clock. Originally it was synchronized with sensor D_c to deliver distance data simultaneously. This is important to derive exact position data at the proxy. After a while, D_d reports data for second 12, even though the measurement actually occurred in second 11. After corrective actions of the proxy in second 19, the two sensors are in synch again.

Tab. 1 (right) shows the data produced by the set of proxies. The following completeness relevant observations can be made:

- The system data rate remains (by definition) at 1 second. This rate is globally set and true at all levels, except for the query data rate as specified in the user requirements.
- The distance data was used to derive positional information for each room. Accordingly, as soon as one of the two distance values was missing, no positional value can be derives (lines 9-13).
- Some rounding has taken place.
- Both the accelerometer readings have undergone PCA (Piecewise Constant Approximation) compression. Thus, only a few values remain.
- The passive infra-red (PIR) data was aggregated to form the “Room” attribute values. In cases where the PIR values conflict (line 16) or give no information (line 22) no value is provided.

1.2 Sample Queries with Completeness Awareness

In this section we present three sample queries that illustrate how completeness may be used in the system. The high level queries are as follows: a) “Is there a person in one of the three rooms?”, b) “Is the person in the room pacing abnormally?” and c) “Is the person highly agitated?”

Passive infrared sensors are used to determine the presence of a person in one of the three rooms. Upon detection, ultrasound sensors are used to get accurate pacing trajectories to classify them as normal or abnormal. Once abnormal pacing is identified, body worn accelerometer readings are taken to detect the onset of agitation.

Query 1: “Every 10 seconds generate a PIR sensor reading for each room.”

The above sample query indicates whether a person is present in one of the rooms. The completeness level of the information provided could be increased by upgrading to a greater requirement of completeness to get precise position information once a person is detected to have entered a given room.

Query 2: “Every second generate the position of the person in the Living Room.”

The previous sample query gathers position information through external observation, with a high degree of completeness. Once the pacing is detected to be abnormal, an additional query at a still higher level of completeness is issued to the body worn accelerometer to gather data and collect it for fine grained motion analysis at the analyst’s workstation.

Query 3: “Every 0.1 second generate a set of accelerometer readings.”

In a typical sensor query system, sensors can switch between *sleep*, *idle*, and *active* states to prolong battery life. Due to lack of any well defined conventions, sensors are generally operating at a predefined constant sampling rate while switching into a active state. In addition, to meet the requirements of a certain range of applications, sensors are easily to be operated at a sampling rate that is higher than necessary for some applications. Such over-fed data unnecessarily reduces the query processing performance and wastes system resources in terms of transmission bandwidth and battery power.

1.3 Contributions and Paper Structure

The main contribution of this paper is a model highlighting the *completeness* of sensor data: (i) The model allows consistent handling of information content losses as data propagates through a sensor network. (ii) The model considers factors that influence completeness and allows trade-offs between the sensor data completeness and system resource consumption to be configured based on application requirements. (iii) An implementation of the model in a “smart home” application context demonstrates all the concepts introduced in the paper. Our implementation results illustrate the value of the completeness based approach versus one that does not use completeness. Query running times are greatly reduced and system resources are conserved as over-fed data are cleared from the data operation and transmission paths.

We have presented thus far, a simple example of a “smart home” that is able to monitor the health and well-being of its resident. We have emphasized the importance of information completeness in such an application. We have also put together the concepts discussed in this paper into an example system that illustrates various types of processing taking place, and some sample queries in this environment. In order to keep the presentation and analysis simple, we have only introduced only three modalities of sensors. In our current deployment of sensors in a hospital ward ([5]) we have a much richer diversity of sensing modalities including pressure sensors, microphones and video cameras. Feature extraction allows us to filter massive amounts of source data into streams of relevant information which are then stored in a relational database. The remainder of the paper is structured as follows. In Sec. 2 we present a completeness model for sensor data. The model permits us to express query completeness as well as data completeness in a unified framework. Sec. 3 introduces the notion of *predicting quality* highlighting the fact that such predictions allow fine-tuning and optimization of sensor configuration. Sec. 4 discusses a proof of concept implementation set up in our laboratory, and introduces preliminary results. Sec. 5 discusses related work; conclusions and future work are discussed in Sec. 6.

2 A Completeness Model for Sensor Data

Completeness is but one of several information quality criteria, albeit an important one. It is a measure for *how much* data reaches a point of consumption compared to the *maximum amount* of data possible at that point. The more complete some data is, the higher the quality of conclusions on that data are. Examples for conclusions are aggregations on the data, which again have a completeness value, and medical diagnosis, which are not measured by their completeness but by their clinical success.

In this section we lay the foundation of our completeness model for sensor data by first defining what the maximum amount of data is, i.e., our reference point. Then we formally define completeness and show in the following sections how it is affected by different operations along the sensor query system components and operations performed by them.

2.1 Data Rates in Sensor Systems

Given a component, the rate at which data is produced by the component is called its data rate. Rates are defined as the average number of data items produced within some fixed duration, usually millisecond. There are five types of data rates that are important for us:

System data rate (SysDR). The system data rate is the maximum data rate that prevails for all sources and end users in a system. The system rate should be at least the maximum of all other data rates. It is fixed once at setup and merely serves as a point of reference. SysDR is chosen so that all other rates can be defined as multiples of SysDR.

A system producing data at system data rate is the maximum a system can produce. Typical rates at the user-end of a sensor system are much lower. The typical system data rate of a monitoring system, such as the one described in Sec. 1.1, is 1/msec (or 1000/sec), i.e., no component produces data at a higher rate than that.

Sampling data rate (SampDR). The sampling data rate is the rate at which a sensor obtains samples from its environment, as determined by physical limitations or by configuration settings. The sampling rate may be set during setup or changed dynamically by an application or by the sensor itself.

Typical sampling data rates are 0.1/sec for the IR sensors, 0.1/sec for idle distance sensors, and 1/sec for active distance sensors.

Sensor data rate (SensDR). The sensor data rate is the rate at which a sensor communicates data to the outside world—usually to a base station, or to the next hop in a multi-hop sensor network. Sensor data rate can be lower than the sample data rate to save energy. For instance, a sensor might measure temperature once per second, but is configured to only communicate if there is a change in temperature.

Operator data rate (OpDR). The operator data rate is the rate at which a logical operator in the sensor network produces data. Usually the operation is an aggregation, but could also be a selection, projection, transformation, or even a dispatch-type operation. Operator data rate can be lower or higher than the data rates of its input. For instance, an aggregation operator might aggregate groups of ten data items received from a sensor, and thus have an operator data rate of on tenth of the sensor data rate. On the other hand, an operator might produce data at a higher rate than its input if it fills input gaps with data. In the temperature example above, an operator might receive data only if the temperature changes but produces a constant stream of data, using the last available input data. Also note that input is not necessarily sensor data, but can in turn be data from another operator.

Query data rate (QDR). The query data rate is the rate at which the end user or application would like to see data delivered in a query response. It is dictated by the application requirements. For instance, an clinician might be satisfied with a averaged temperature data for a patient that is based on a temperature reading every minute: The query data rate is $1/60000$ per millisecond. On the other hand, an application tracking fast objects, such as automobiles, can only produce reliable results with a data rate of $1/100$, i.e., 10 readings per second.

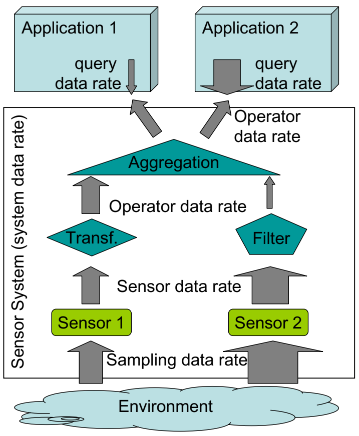


Fig. 3. Different data rates in a sensor system (wide arrows indicate high data rate)

We use the query data rate to measure the quality of the system. If the operator data rate of the final operator is at least as high as the query data rate, the setup and configuration of the sensor system is acceptable. If it is lower, the overall quality is diminished.

For brevity from here on, we omit the term "data" from the different data rates. Fig. 3 shows the points in a sensor system where the individual data rates are defined. The system itself has an overall system rate. Two sensors measure the environment at different sampling rates and produce data at different sensor rates. A transformation operator leaves the data rate unchanged; a filter operator reduces the data rate and thus has a lower operator rate. An aggregation again produces data at some operator rate, which is consumed by two applications, each with different requirements, expressed as query rate. Application 1 has a lower query rate and is thus adequately supplied by the sensor system. Application 2 has higher query rate than the incoming operator rate. Thus, its overall quality is diminished.

The mentioned data rates are determined by different means: Data rates of sensors (SampDR and SensDR) are usually determined by the manufacturer and can be configured using embedded software. The operator data rate is dynamically set during runtime. It depends on the algorithms involved and their internal parameters. Finally, and most important is a means to determine the query data rate QDR. This involves two subproblems: First determining the "right" QDR for the application at hand, and second specifying the QDR in a query. We leave the first problem to the expert and provide a model for the second.

2.2 Completeness

In this section we describe how we translate actual data rates of a running system into a single completeness value measured over a certain period, which can be compared with the requirements of the query. To this end we construct one virtual relation VR for each data transfer level of the system. The first column of VR always represents the time-dimension. Time increments are determined by the system data rate, i.e., if SysDR is 1 /sec, there is a row for each second. The other columns represent different sensors or operators producing data at that level of the system.

We distinguish two kinds of completeness for columns (see definitions later):

- System completeness: Completeness with respect to the system data rate.
- Query completeness: Completeness with respect to the query data rate.

Tab. 1 show the virtual tables of the smart home scenario as described in Sec. 1.1. If we regard each data item in a relation as a cell, some of the cells could be unoccupied or occupied by null value, i.e., SensDR and OpDR are less that SyDR. The proportion of data bearing cells to null valued cells in a set of cells indicates the completeness of the group of cells. Regard the $Dist_b$ column of Tab. 1. The sensor stopped producing output values for five second, thus, the system completeness of this column for the overall period of 29 seconds is $24/29$ or 0.83. Regarding the entire table, we calculate its completeness as $\frac{29+24+15+14+15+15+29+29+29+29+29}{29*11} = \frac{257}{319} \approx 0.81$. Due to aggregation, some columns of Tab. 1 naturally contains less values. To account for this effect, completeness for aggregated columns is calculated differently as we explain in the following paragraphs.

Definition 1 (System completeness). *Let n be the number of sensors represented in the virtual relation VR, let d be the duration of measurement for a given application or query, and let v be the number of non-null values in VR. Then* system completeness *of VR is $SysComp(VR) := \frac{v}{n \cdot d}$.*

In the previous and the next definition we assume all sensors represented in VR to be relevant for a query. Next, query completeness reflects the fact that the usually very high system rate is higher than the requirement of the query. Thus for query completeness, we take as a basis the query rate QDR and not the system rate SysDR.

Definition 2 (Query Completeness). *Corresponding to Definition 1,* query completeness *is defined $QComp(VR) := \frac{SysComp(VR) \cdot SysDR}{QDR}$.*

Consider for instance a system data rate of 1 per millisecond and a query data rate of 1 per min or 1/60000 per millisecond. Consider further a series of sensors and operators producing a final system completeness of 1/1000 (1 per second) at the querying site. Then query completeness is $\frac{1}{1000}/\frac{1}{60000} = 60$, i.e., the query requirements are well met. In fact, completeness is 60 times higher than necessary, indicating potential to decrease sensor data rates. A QDR of 2 per second on the other hand evaluates to a query completeness of $\frac{1}{1000}/\frac{1}{500} = 1/2$, i.e., the query requirements are not met indicating a need to increase sensor data rates or operator data rates.

In Sec. 3 we show how to calculate system completeness without knowing the precise value of v, thus utilizing the completeness measure to gauge the effectiveness of the sensor networks for different applications.

3 Predicting Quality

The main idea of calculating completeness and propagating its scores from the sensors over proxies to the final application is to model the information using a virtual table between each level of nodes. The schema of this table represents the data that is passed between the nodes. This schema includes columns for the "split" values for different sensors. The completeness of this virtual table can easily and formally be defined by counting NULL-values. In real-world scenarios, this virtual table is never materialized, so one cannot count values. Instead, it is necessary to predict the number of null-values based on setup of sensor data rates, properties of the sensors themselves, and operations at the proxy levels. This calculation should be performed bottom up: Given the completeness of the sensor data, completeness values at various higher levels (proxies, application, and query) can be mathematically predicted.

The following paragraphs list formulas to calculate completeness through various operations. Together, these formulas build a completeness model, similar to the known cost-models of conventional DBMS optimizers.

3.1 Completeness of Sensor Output

Sensors output data at their individual sensors data rate, thus "filling" the virtual relation. If $SensDR(s_i)$ is the sensor data rate of sensor s_i, the output data completeness of s_i is, $SysComp(s_i) = \frac{SensDR(s_i)}{SysDR}$. By ignoring for now various factors that influence completeness, the system completeness of VR that is filled with only raw sensor data from n sensors is $SysComp(VR) = \frac{\sum_i SensDR(s_i)}{SysDR \cdot n}$.

As SensDR of a sensor closely depends on its SampDR, which can generally be set to a wide range of different values on the fly, one possible dimension to manipulate data completeness is through dynamically configuring SampDR of a sensor.

3.2 Completeness Through Several Typical Sensor System Operations

Virtual relations are not only filled with raw sensor data, but, at higher levels, by various operations. A comprehensive algebra to manipulate data completeness through every possible sensor system operation is left for further work. In this section we consider a few logical data operators that are most commonly encountered in typical smart home systems of the type described in Sec. 1.1, and their effects on completeness are characterized. It is noted that the formulas listed below for completeness calculation are not necessarily unique, nor the most precise ones, but based on our experience they are simple and effective to provide a good estimation on completeness in real systems. A more general discussion on the completeness influence of data operations is left to future work.

Logic-OR Data Operator: This type of data operation can be abstracted as an operation that integrates N time series inputs,$\{s_1, s_2, \ldots, s_n\}$, into one time series output, s_{or}, and the integration logic is OR, that is, for each time, a non-null value reading from any among the N inputs lead to a non-null value in the output. This type of operation is mainly to describe in-network data aggregation ([6]), which is essential for wireless sensor networks where resources such as bandwidth and energy are limited. In such in-network data aggregation, intermediate nodes may aggregate several events reported from different sources into one event as sensor readings can be correlated, e.g., detection of the same phenomenon. Based on our experience, a good estimation of OpDR for this type of operation is $OpDR_{or}(s_1, s_2, \ldots, s_n) = max(SensDR(s_1), SensDR(s_2), \ldots, SensDR(s_n))$.

Therefore, the system completeness of the aggregated output can be estimated as $SysComp(s_{or}) = max(SysComp(s_1), SysComp(s_2), \ldots, SysComp(s_n))$.

Logic-AND Data Operator: This type of data operation can be abstracted as an operation that integrates N time series inputs, $\{s_1, s_2, \ldots, s_n\}$, into one time series output, s_{and}, and the integration logic is AND, that is, for each time, a null value reading from any among the N inputs lead to a null value in the output. The operation is mainly to describe that columns in VR can possibly be the results by fusing raw sensor data from multiple sources. For example, in an object tracking system, the position data can be the integrated results of fusing the distance readings from two nearby ultrasonic sensors. A typical join operation on sensor data based on only the timestamp can also be described as this type of operation. Based on our experience, a good estimation of OpDR for this type of operation is

$$OpDR_{and}(s_1, s_2, \ldots, s_n) = min(SensDR(s_1), SensDR(s_2), \ldots, SensDR(s_n))$$

Therefore, the system completeness of the fused output can be estimated as,

$$SysComp(s_{and}) = min(SysComp(s_1), SysComp(s_2), \ldots, SysComp(s_n))$$

Compression Operator: In sensor networks, wireless communication is the key factor to consume resources in terms of bandwidth and energy. It is common for sensors to compress time series readings instead of sending them in raw form. One of simplest yet effective sensor data compression method is Piecewise Constant Approximation (PCA) ([7]), which is adopted in the system we built. In PCA, the time data series D to be processed is represented as a sequence of n segments, $PCA(D) = (v_1, e_1), (v_2, e_2), \ldots, (v_n, e_n)$, where e_n is the end point of a segment and v_n is a constant value for time in $[e_{n-1} + 1, e_n]$. Here we assume the value at $e_{n-1} + 1$ is used for v_n. With such approximation, d(i) is estimated as

$$d(i) = \begin{cases} v_1 & \text{if } i \leq e_1 \\ v_m & \text{if } e_{m-1} + 1 \leq i \leq e_m \end{cases}$$

Let $k = e_m - e_{m-1}$, that is, every k samples share a constant, and s_{compr} be the compressed output, OpDR for this type of data compression is, $OpDR_{compr} = \frac{SampDR}{k}$. Therefore, the system completeness of the compressed output can be estimated as, $SysComp(s_{compr}) = \frac{SysComp(D)}{k}$.

Aggregation Query Operator: Aggregation queries, such as `COUNT`, `MIN`, `MAX` and `SUM`, occur frequently in such systems and their completeness handling requires a slight twist. By definition, an aggregation operator has multiple values, $\{s_1, s_2, \ldots, s_n\}$, as input and a single value, s_{aggr}, as output. Completeness associated with such operators should reflect the completeness of the input, i.e., relatively how much data went into the calculation of the aggregate, and not the single output value. Therefore, the system completeness of the single aggregated output value is defined as,

$$SysComp(s_{aggr}) = avg(SysComp(s_1), SysComp(s_2), \ldots, SysComp(s_n))$$

4 Prototype and Experimental Results

In this section, a preliminary prototype implementation of the concepts discussed in this paper is presented. The prototype demonstrates the value of the proposed completeness model through an experimental implementation of the patient behavior monitoring system described in Sec. 1.1.

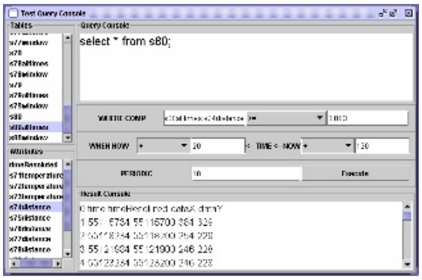
4.1 A Sensor Query Engine with Completeness Awareness

A sensor query engine with completeness awareness is built to perform a preliminary evaluation of the proposed data completeness model. This sensor query engine is implemented by introducing a middleware layer on top of the relational SQL engine, MySQL. The main functions of this layer include,

- Self-discovery of the available sensors in the system,
- Maintenance of a relational table in MySQL for each sensor with real-time update,
- Generation of appropriate views of the full Virtual Relation, by full-outer-join operation on all the relevant tables within a particular time window,
- Adjustment of the sampling rate of each sensor on the fly, based on data completeness.

In the prototype, original SQL queries are supported with two new clauses, `COMP` and `TIME`, as shown in Fig. 4. The `COMP` clause enables the possibility to explicitly indicate the completeness requirement of any available attribute in the VR. The completeness requirement indicated may be used for both configuring the system and accessing the quality of query results. In this initial version of our prototype, completeness is closely coupled with sampling rate adjustment of related sensors for future queries and PCA [7] compression ratio selection for past queries. The `TIME` clause is used to support simple continuous queries by specifying the beginning and end of the monitoring duration as well as the frequency at which the query is to be executed.

4.2 Experimental Setup and Measurements



```
select * from s80;
```

```
0 time timeReceived dataX dataY
1 551 5734 55115700 384 326
2 [illegible] 264 228
3 55121934 55121900 246 226
4 [illegible] 246 228
```

Fig. 4. Sensor Query Engine with Completeness Awareness

A replica of the home layout shown in Fig. 1 is set up in the lab. Each of the three rooms—kitchen, dining room and living room—is equipped with two ultrasonic sensors and one passive infra (PIR) sensor. A person with body worn accelerometer is assumed to be wandering inside the house. All sensors are installed on the CrossBow MicaZ mote platform. The data gathering process is shown in Fig. 2. Due to the small number of sensors used, and also the small spatial separation among sensors, a very simple network topology is employed, in which each sensor directly communicates through a wireless link with a common proxy attached to a PC. Thus, all proxies and the user node of Fig. 2 are actually co-located at the same node. To emphasize the effect of the completeness model, no additional sensor management scheme is employed; in other words, all sensors are kept active during the monitoring period.

As discussed above, in such patient behavior monitoring systems, a wide range of factors are available for us, e.g., sensor sampling rate, data operation or sensor scheduling, to explore the flexibility of meeting query completeness requirements but simultaneously reducing system resource consumption and increasing query performance. In our experiments, we wished to see how the completeness model could help in improving query performance and system resource planning for future queries through the selection of appropriate sensor sampling rate. Without completeness awareness, to satisfy the requirements of all three queries described in Sec. 1.2, the predefined constant sampling rates for PIRs, ultrasonic sensors and BWA needed to be set to at least 0.1, 1, and 10 samples per second continuously. The system with such a set of minimal constant sampling rate settings is used as a comparison against a system with completeness awareness, where the sampling rate of each sensor is dynamically selected based on query completeness requirement, while answering Query 2.

The VR for Query 2, shown in Tab. 1, is generated as a view of combining the output of all five proxies:

```
CREATE VIEW HOUSEHOLD
      SELECT *  FROM P1, P2, P3, P4, P5
      FULL OUTERJOIN ON Time
```

The SysDR is defined here as 1 per millisecond. As indicated, the required QDR in Query 2 for position data, which are represented as X and Y coordinates ($PosX_{ab}$, $PosY_{ab}$), is 1 per second. To achieve full query completeness, the following condition should be satisfied:

$$QComp(PosX_{ab}) \geq 1.$$

Based on the definition of query completeness,

$$QComp(PosX_{ab}) = \frac{SysComp(PosX_{ab}) \cdot SysDR}{QDR},$$

The SysComp requirement to be specified in the query is,

$$SysComp(PosX_{ab}) \geq \frac{QDR}{SysDR}$$

With the new completeness clause `COMP`, Query 2 can be posed as:

```
SELECT PosX_ab, PosY_ab  FROM HOUSEHOLD
WHERE  Room = "Living Room"
AND    COMP(PosX_ab) >= 0.001
```

As shown here, the required QDR for the application, which may be provided by experts, is implicitly specified in the query through the indication of system completeness requirement. For attributes without any completeness requirements a default system completeness value, 0.0001 in this experiment, is assumed.

As a logic-AND type operator is used to derive the position information by fusing distance data from two ultrasonic sensors $Dist_b$ and $Dist_b$, based on the algebra discussed in Sec. 3, we have,

$$SysComp(PosX_{ab}) = min(SysComp(Dist_a), SysComp(Dist_b)),$$

Based on the discussion in Sec. 3 we have

$$SysComp(Dist_a) = \frac{SensDR(Dist_a)}{SysDR}, \quad SysComp(Dist_b) = \frac{SensDR(Dist_b)}{SysDR}$$

and

$$SensDR(Dist_a) = SampDR(Dist_a),\ SensDR(Dist_b) = SampDR(Dist_b)$$

As specified in the query, to satisfy the query completeness we should have

$$SysComp(PosX_{ab}) \geq 0.001$$

With the set of equations above, we found such completeness requirement can be satisfied with

$$SampDR(Dist_a) \geq 1 \text{ per second}, \quad SampDR(Dist_b) \geq 1 \text{ per second}$$

Results were collected by posing Query 2 over our sensor query system with and without completeness awareness respectively. The query period, which is the duration of measurement for a query each time, is varied in experiments.

From the experimental results, we observe that by satisfying just the necessary query completeness, through our completeness planning, we can achieve much better query response performance as shown on the left of Fig. 5. This is the case, even though the overall system completeness is low as shown on the right. In addition, there is greater system resource saving, in terms of bandwidth and energy, as shown on the left of Fig. 6 and better query completeness satisfaction as shown on the right.

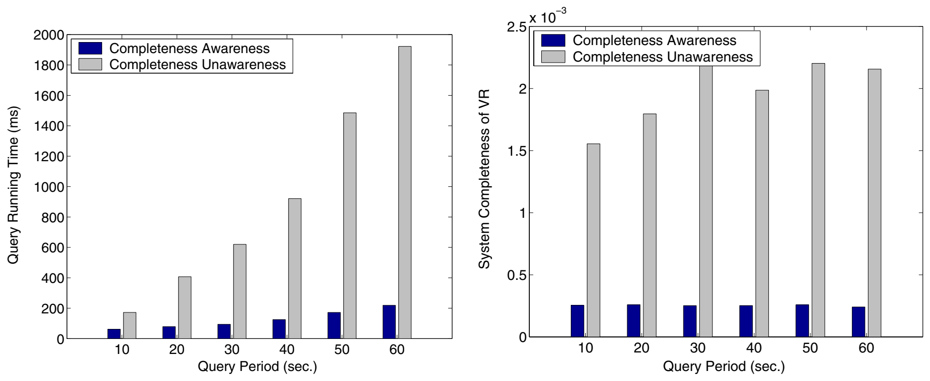


Fig. 5. Running Time (left) and System Compl. of VR (right) of Query 2

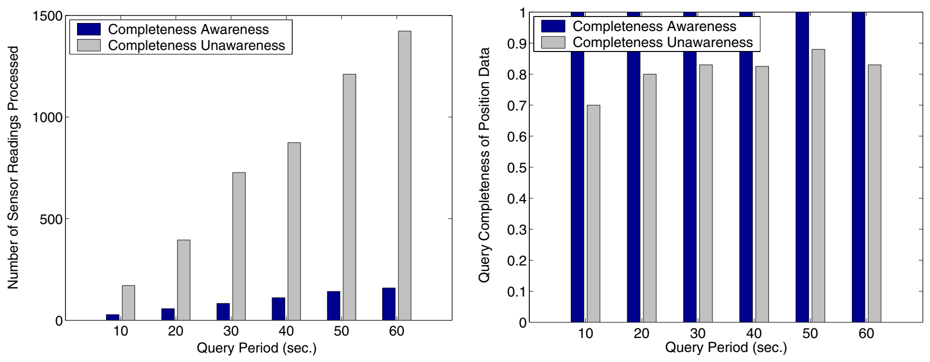


Fig. 6. Amount of Data Processed (left) and Query Completeness of Position Data (right) for Query 2

5 Related Work

There are two major areas of related work, namely information quality and Data Stream Management Systems (DSMS). A systematic and formal approach to the measurement of information quality, and the combination of such measurements for information integration are presented in [8]. These approaches are based on notions of coverage, density of information, ranking of information sources, query-specific attribute weightings, and a number of ways of selecting between multiple sources of data. IQ bounds are discussed in [9] and related to the notion of query completeness in this paper. Determining the "size" of a data source, i.e., its *coverage*, its not a new problem. Most notably, Motro and Rakov define a "completeness" criterion, which matches our coverage criterion [10]. Motro suggests to add "completeness assertions" to the query result, adding more meaning to the result [11]. Completeness assertions are statements, such as "the data contains *all* recordings on the CBS label". These assertions are aggregated along query plans in a similar fashion to our coverage along mapping paths. Thus, the author can give qualitative statements about the completeness of results, but not quantitative statements as we do. Finally, Florescu et al. quantitatively describe the content of distributed autonomous document sources using probabilistic measures [12]. Their model calculates two values: "Coverage" of data sources, determining the probability that a matching document is found in the source, and "overlap" between two data sources, determining the probability that an arbitrary document is found in both sources. These probabilities are calculated with the help of word-count statistics.

The early work in the area of continuous queries led to the identification of stream processing as as new problem [13, 14, 15, 16]. It is found in [15, 16] that a key challenge for the design of a DSMS is to provide approximation and adaptivity in executing continuous queries, because data rates and query load may exceed available resource. Our proposed completeness model can provide a quantitative approach to handle the interaction between resource management and query approximation in such DSMS by simply treating those approximation techniques [16, 17], such as synopsis compression, sampling and load shedding, as various knobs in our model.

The most well known and distributed sensor based system is the Berkeley MOTE. In [18] the authors describe the query language for the MOTE, the database TinyDB and its support for continuous queries. In [19] is presented ideas on tree-based processing of in-network aggregation and the subtleties of its cost-benefit analysis, and *wave scheduling*. In [20] the authors discuss the main issues that apply to sensor based query processing. High level approaches have been outlined in [21] and [22]. Finally, [23] discusses continuously adaptive queries over streams in the face of changing query workloads and data rates.

6 Conclusions

As a first step we have introduced a simple model for information completeness, which is a criterion for information quality. The model is evaluated in a unique application setting that is based primarily on sensor data sources. Factors affecting completeness are characterized and a simple analytical model illustrates how a trade-off can be made between avoidable and unavoidable factors that affect completeness, thereby giving some means for achieving desired completeness levels without paying too high a price. Although a system should contain components such as reasoning systems, knowledge-bases, databases, and stream management, only the lower portion of such a system has actually been modeled and analyzed in this paper. Higher level components of such systems must be integrated to gauge the effectiveness of the entire scheme of dealing with information completeness.

On the positive side, query completeness is a notion that can be used in many ways and to many ends. This paper illustrates how query completeness can be used to alleviate sensors from unnecessarily high sampling rates. Other uses are conceivable, such as sensor selection and sensor management as well as resource management. In future work we shall be continuing to develop a sound understanding of important criteria for information quality such as completeness. Our aim is to apply these ideas in a manner that permits easy development of context aware applications for smart spaces.

We also intend to develop a formal model for completeness based on the informal model presented herein.

Acknowledgment. This research was supported in part by the German Research Society (DFG grant no. NA 432).

References

1. Wang, X., Dong, J.S., Zhang, D., Chin, C.Y., Hettiarachchi, S.R.: Semantic space: An infrastructure for smart spaces. IEEE Pervasive Computing Magazine (2004) 32–39
2. Bardram, J.E.: Applications of context-aware computing in hospital work - examples and design principles. In: Proceedings of the 2004 ACM Symposium on Applied Computing. (2004)
3. Tolstikov, A., Biswas, J., Chen-Khong, T.: Data loss regulation to ensure information quality in sensor networks. In: Proceedings of the 2005 Intelligent Sensors, Sensor Networks and Information Processing Conference. (2005)
4. Biswas, J., Das, S., Qiu, Q., Chava, V.S., Thang, P.: Quality aware elderly people monitoring using ultrasonic sensors. In: Proceedings of the International Conference On Smart Homes and Health Telematics (ICOST). (2005) 107–115
5. Biswas, J., Yap, P., Foo, V., Qiu, Q., Aung, A.P.W., Thang, P.V., Guopei, Q.: Use of pervasive monitoring technology as compared to direct observational methods using the soapd scale in the measurement of agitation in patients with dementia. In: Research Collaboration between Institute for Infocomm Research (I2R) and Alexandra Hospital, Singapore. (2005)
6. Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed diffusion: A scalable and robust communication paradigm for sensor networks. In: Proceedings of the International Conference on Mobile Computing and Networking (MobiCom). (2000)
7. Lazaridis, I., Mehrotra, S.: Capturing sensor-generated time series with quality guarantees. In: Proceedings of the International Conference on Data Engineering (ICDE). (2003)
8. Naumann, F., Freytag, J.C., Leser, U.: Completeness of integrated information sources. Information Systems **29**(7) (2004) 583–615
9. Leser, U., Naumann, F.: Query planning with information quality bounds. In: Proceedings of the International Conference on Flexible Query Answering Systems (FQAS). Advances in Soft Computing, Warsaw, Poland, Springer Verlag (2000)
10. Motro, A., Rakov, I.: Estimating the quality of databases. In: Proceedings of the International Conference on Flexible Query Answering Systems (FQAS), Roskilde, Denmark, Springer Verlag (1998) 298–307
11. Motro, A.: Completeness information and its application to query processing. In: Proceedings of the International Conference on Very Large Databases (VLDB), Kyoto (1986) 170–178
12. Florescu, D., Koller, D., Levy, A.: Using probabilistic information in data integration. In: Proceedings of the International Conference on Very Large Databases (VLDB), Athens, Greece (1997) 216–225
13. Terry, D., Goldberg, D., Nichols, D., Oki, B.: Continuous queries over append-only databases. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD). (1992)
14. Babu, S., Widom, J.: Continuous queries over data streams. In: SIGMOD Record. (2001) 109–120
15. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proceedings of the Symposium on Principles of Database Systems (PODS). (2002)

16. Motwani, R., Widom, J., Arasu, A., Babcock, B., Babu, S., Datar, M., Manku, G., Olston, C., Rosenstein, J., Varma, R.: Query processing, resource management, and approximation in a data stream management system. In: Proceedings of the Conference on Innovative Data Systems Research (CIDR). (2003) 245–256
17. Abadi, D.J., Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., Zdonik, S.: Aurora: A new model and architecture for data stream management. VLDB Journal **12**(2) (2003) 120–139
18. Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W.: Tinydb: An acquisitional query processing system for sensor networks. ACM Transactions on Database Systems (TODS) **30**(1) (2005) 122–173
19. Demers, A., Gehrke, J., Rajaraman, R., Trigoni, N., Yao, Y.: The Cougar project: A work-in-progress report. In: SIGMOD Record. Volume 32. (2003)
20. Yao, Y., Gehrke, J.: Query processing for sensor networks. In: Proceedings of the Conference on Innovative Data Systems Research (CIDR). (2003)
21. Bonnet, P., Gehrke, J., Seshadri, P.: Towards sensor database systems. Technical Report TR2000-1819, Cornell University (2000)
22. Gedik, B., Liu, L.: Mobieyes: Distributed processing of continuously moving queries on moving objects in a mobile system. In: Proceedings of the International Conference on Extending Database Technology (EDBT). (2004)
23. Madden, S., Shah, M.A., Hellerstein, J.M., Raman, V.: Continuously adaptive continuous queries over streams. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD). (2002)