Spatio-temporal Aggregates over Streaming Geospatial Image Data

Jie Zhang

Department of Computer Science
University of California, Davis, CA 95616, U.S.A.
zhangji@cs.ucdavis.edu

Abstract. Geospatial image data obtained by satellites and aircraft are increasingly important to a wide range of applications, such as disaster management, climatology, and environmental monitoring. Because of the size of the data and the speed at which it is generated, computing spatio-temporal aggregates over geospatial image data is extremely demanding. Due to the special characteristics of the data, existing spatio-temporal aggregation model and evaluation approaches are not suitable for computing aggregates over such data.
In this paper, we outline the key challenges of computing spatio-temporal aggregates over streaming geospatial image data, and present three goals of our research work. We also discuss several preliminary results and future research directions.

1 Introduction and Motivation

Driven by major advances in remote sensing technology, geospatial image data from satellites and aircraft have become one of the fastest-growing sources of spatio-temporal data sets. The remotely-sensed imagery collected by NASA alone are expected to exceed dozens of terabytes per day within the next few years. Such data have become increasingly important to a wide range of applications, such as disaster management, climatology, and environmental monitoring.

In a typical data processing setting for streaming geospatial image data, the data are transmitted continuously in the form of raster images. Each image can be regarded as a rectangular grid in which each cell (*point*) consists of a point location and point value. Figure 1 gives an example of a sequence of raster images transmitted from the National Oceanic and Atmospheric Administration's (NOAA) Geostationary Environmental Operational Satellite (GOES) West [4] over a period of about one hour. Since GOES West scans different regions of Earth's surface over time, each image in Figure 1 has a different spatial extent.

In general, such image data continuously arrive at a very high rate and volume. For example, GOES West satellite images are transmitted at 2.1Mbits/second – about 22GBytes/day. As a result, it is very important to have operations that summarize the data. One such operation, known as *spatio-temporal aggregation*, summarizes the data in both spatial and time dimension. For example, a typical spatio-temporal aggregate query is: "Calculate the average soil temperature in

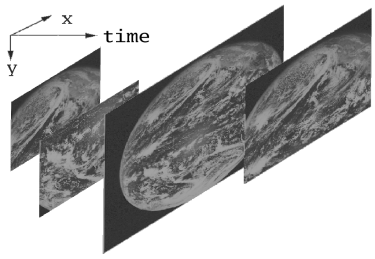

Fig. 1. Sequence of GOES West images. Different regions of Earth's surface are scanned over time, resulting in images with different spatial extents.

Davis, California, from July to September every year for the last ten years." Spatio-temporal aggregates are not only one of the most fundamental operations for various applications, such as detecting changes in the environmental landscape, but also the most demanding in terms of space and time complexity.

The efficient processing of spatio-temporal aggregate queries, which specify a region of interest and some time interval, have been studied in several works [7, 10, 15–18, 20]. However, there are several limitations in these approaches.

First of all, existing **aggregation models** do not provide a meaningful answer to a query over streaming geospatial image data. All existing approaches use traditional aggregation models, in which data that satisfy the given query region is aggregated, and a *single* aggregate value is returned as the answer. This is not always meaningful in the context of streaming geospatial image data, since they falsely imply that high-quality data is always available for the entire query region.

Secondly, existing **evaluation approaches** are not suitable for streaming geospatial image data. Some of these approaches are optimized for static data in such ways that they are not suitable for streaming data. Some approaches have such high construction or maintenance costs that they can not catch up with the arrival rate of streaming geospatial image data. Moreover, some approaches primarily focus on spatial objects, each of which contains its own spatial component. To apply these approaches to geospatial images, they consider each point in an image as an individual spatial object, and store its spatial component by itself. As a result, these approaches do not take advantages of the gridded point set structure inherent to raster image data. In particular, they do not exploit cases where neighboring points have similar or identical point values – but such cases occur frequently for some regions in raster images. Therefore, these approaches tend to have extremely high storage costs, which in turn lead to poor construction and query performance.

In this PhD research work, we aim to *design effective spatio-temporal aggregate computations for streaming geospatial image data*. Here, "effectiveness" represents both spatio-temporal aggregation models and evaluation approaches. The objectives for this goal include:

1. Develop a new spatio-temporal aggregation model that provides more fine-grained (intuitive) results for the spatio-temporal aggregate computation over streaming geospatial image data.
2. Develop a query processing framework to evaluate spatio-temporal aggregate queries using the new aggregation model.
3. Design efficient supporting index structures to evaluate spatio-temporal aggregate queries over streaming geospatial image data, in terms of construction time, space requirements, and query performance.

2 Related Work

Most of the existing approaches for computing spatial and spatio-temporal aggregates [6, 7, 9, 10, 16, 18, 20] focus on spatial objects and not field-based data, such as gridded point data or raster images. The aR-tree [6, 9] is an R-tree in which each MBR (minimum bounding rectangle) of an internal node has a pre-computed aggregate value that summarizes the values for all objects that are contained by the MBR. As a result, the partial aggregate result can be obtained in the intermediate nodes of the tree without accessing all the contained objects.

Papadias et al. [10] presented another structure –the *aRB-tree*– that extends the aR-tree and considers the spatial and temporal information separately. The aRB-tree consists of an R-tree to index the regions of spatial objects, and a B-tree structure associated with each region in the R-tree to store the temporal information. Similar to the aR-tree, partial aggregate results can be obtained from intermediate nodes. The disadvantage of the aR-tree and the aRB-tree is that multiple paths from the root node may be needed to answer an aggregate query. Zhang et al. [21] proposed an indexing scheme –the *BA-tree*– to overcome this limitation. The BA-tree is based on a k-d-B tree. Similar to the aR-tree and aRB-tree, with each region of an internal nodes in a BA-tree a pre-computed aggregate value is associated. In addition to this, the BA-tree maintains some extra data for each region of the internal nodes, which guarantees that only one path is searched during an aggregate computation. Furthermore, based on the BA-tree, Zhang et al. [20] proposed an indexing scheme to maintain aggregate information at multiple temporal granularities for streaming data. Tao et al. [16] pointed out that the aRB-tree has a problem with *distinct counts* and presented an approximate approach to evaluate distinct *count/sum* aggregate queries.

However, none of these approaches is suitable for streaming geospatial image data. The reasons are as follows. First, all the above approaches provide a single value as the final answer to an aggregate query, which is not always meaningful in the context of streaming raster image data, especially when the image data contribute only partially to the query region. Secondly, since these approaches are designed for spatial objects and not for large amounts of points in streaming image data, the location for each point needs to be stored. This results in extremely high space consumption. Finally, the relationship of values among neighboring points is not considered in these approaches, something that one should take advantage of.

3 Preliminary Results

In this section, we describe several preliminary results for this PhD research work. First, we present a new aggregation model – *Segment-based Spatio-temporal Aggregation (SST-Aggregation)* – for answering spatio-temporal aggregate queries over streaming geospatial image data, in particular, raster image data. Secondly, we give a brief description of a query processing architecture for this new model. Thirdly, we present data structures that support such computations. Finally, we present some experimental results.

3.1 Segment-based Spatio-temporal Aggregation (SST-Aggregation) Model

Consider the scenario in Figure 2, which gives an abstract view of a raster image stream (*RIS*). There are four images $I_1, \cdots, I_4$ with different sizes and spatial locations. A typical spatio-temporal aggregate query is, for example, "Calculate the average soil temperature in a given query region R=$[lp, hp]$ (shown as the bold rectangle box in Figure 2) during the time interval $[t_1, t_2]$." Here, we primarily focus on regions with shapes of a multi-dimensional box.

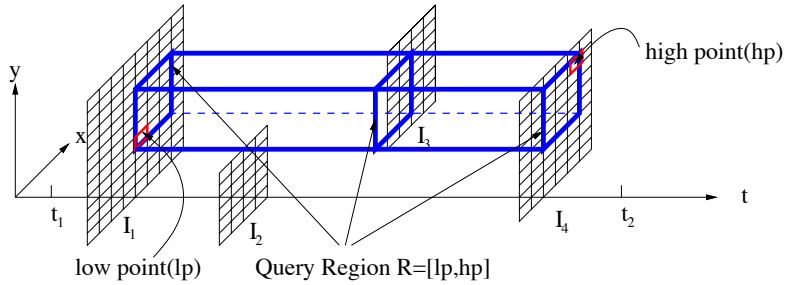

Fig. 2. Example scenario of a spatio-temporal aggregate query over a RIS

Applying existing approaches and index structures to compute spatio-temporal aggregates will provide us with a single value for the query box. An interesting observation for this query is that image I_3 only contributes partially to the query box. That is, at different points in time during $[t_1, t_2]$, not all images contribute all their points (and thus point values) to the query result. Thus, a single aggregate value does not necessarily represent an accurate result since such a result might be skewed, depending on what stream data is available during $[t_1, t_2]$.

Motivated by this observation, we propose a new aggregation model – *Segment-based Spatio-Temporal Aggregation (SST-Aggregation)*. This model overcomes the limitation of the traditional aggregation model in the context of streaming raster image data, and provides more meaningful answers to queries. The definition for SST-Aggregation is given in Definition 2. First, we define a *Segmented Query Region*.

Definition 1. ***[Segmented Query Region]*** *Given a query region $R = [lp, hp]$, and a RIS $\mathcal{I} = \{I_1, I_2, \cdots, I_n\}$. The* ***segmented query region*** *R_{seg} consists of disjoint sub-regions $R_1 = [lp_1, hp_1], R_2 = [lp_2, hp_2], \cdots, R_k = [lp_k, hp_k]$ such that*

1. *$R = R_1 \cup R_2 \cup \cdots \cup R_k$;*
2. *Each sub-region $R_i = [lp_i, hp_i], i = 1, \cdots, k$, has the same timestamp for the low and high point as R;*
3. *For each image $I_j \in \mathcal{I} (j \in [1, n])$ such that $R \wedge I_j \neq 0$, each sub-region $R_i = [lp_i, hp_i], i = 1, \cdots, k$, satisfies exactly one of the following* ***segment properties****:*
   - *$I_j \geq_S R_i$. That is, the image I_j fully covers the spatial extent of R_i. The set of all images that cover R_i is denoted $\mathcal{I}_i$, termed the* ***contribution image set*** *for R_i.*
   - *$I_j \wedge R_i = 0$. That is, the image I_j does not intersect with R_i.*
4. *Given any two sub-regions R_1 and R_2 whose contribution image sets are $\mathcal{I}_1$ and $\mathcal{I}_2$, respectively. If $R_1 \cup R_2$ is a region, then $\mathcal{I}_1 \neq \mathcal{I}_2$.*

Definition 2. ***[SST-Aggregation]*** *A* ***Segment-based Spatio-temporal Aggregation (SST-Aggregation)*** *over a RIS $\mathcal{I}$ is an aggregate operation that computes summarized information (specified by an aggregate function f) for a query region R. The SST-Aggregation constructs the segmented query region R_{seg} for R, and computes an aggregate value for each sub-region in R_{seg} with the aggregation function f. The SST-Aggregation is defined as follows.*

$$SST\text{-}Aggregation\ (\mathcal{I}, f, R) = \{< r_i, f(\mathcal{I}|_{r_i}) > | r_i \in R_{seg}\ for\ all\ i\}$$

The SST-Aggregation model supports those aggregate functions that are defined in the SQL:2003 standard [8]. This model can be easily extended to support continuous queries, including moving window queries.

3.2 Query Processing Architecture

Based on this new SST-Aggregation model, we propose a two-component query processing architecture to evaluate a spatio-temporal aggregate query, as illustrated in Figure 3. When a query is entered into the stream management systems, it is first passed to the *segmentation component*, which segments the query region into a segmented query region, as defined in Definition 1.

Next, the segmented query regions generated by the segmentation are passed to the aggregate-computation component, which computes and constructs the final result to the query.

Both segmentation and aggregate-computation rely on the images (as spatio-temporal objects) from the stream that have entered the system so far. The aggregate-computation component relies on the image point data, while the segmentation component relies on the image metadata, such as the spatial extent and the timestamp of each image.

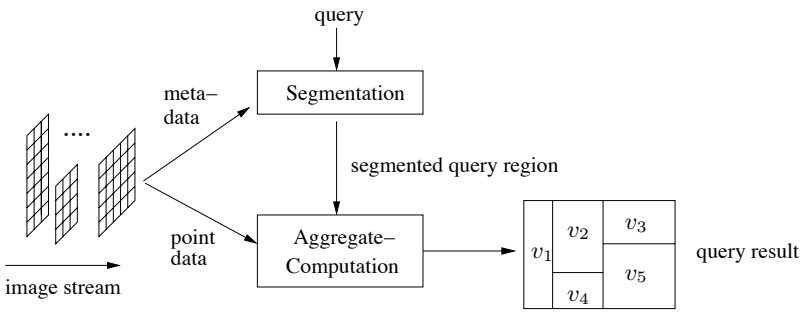

Fig. 3. Conceptual approach for computing SST-Aggregate queries

3.3 Index Structures

To evaluate the effectiveness of our SST-Aggregation model and the proposed query processing architecture, we consider two different index structures for image metadata and point data. Since image metadata is typically much smaller than image point data, we maintain a main-memory data structure –the *Metadata index* – to store the metadata, while disk-based structures are used for image point data.

Metadata Index The metadata index that we proposed in [22] consists of two Red-Black trees [2], one for each spatial dimension. We call these trees the x-tree and the y-tree. Consider an image I created at time t, which has the spatial region represented by two points (x_{low}, y_{low}) and (x_{high}, y_{high}). The x-tree stores the values of x_{low} and x_{high} along with the timestamp t, where each of x_{low} and x_{high} is a key and t is the value. The y-tree has the same structure as the x-tree except that it stores y values. In both trees, we do not store duplicate key values. To achieve this, we maintain a circular buffer inside each node in the trees. This circular buffer stores the list of timestamp values.

Point Data Index Our focus has been on summation-related spatio-temporal aggregate queries (*range-sum* queries), such as *Sum*, *Count* and *Avg*, over user-defined query regions. *Dominance-sum (dom-sum)* is a technique to support the efficient computation of box aggregate queries over objects with non-zero extent in some d-dimensional space [21]. It has been widely used for computations on the data cube [3, 5] and for spatial and spatio-temporal aggregates in the context of spatial objects [20, 21]. In general, for a point set $\mathcal{P}$ in some d-dimensional space, a range-sum query can be computed using the values of 2^d dom-sums. For example, consider the 2-dimensional raster image shown in Figure 4. We want to compute the sum of all values of points located in the region $[(1, 1), (2, 2)]$. Four dom-sum values are required to compute the sum, as illustrated in Figure 4. It is easy to verify that this value is 8, computed from the dom-sum values of four points shown in bold rectangles, that is, $ds((2, 2)) - ds((2, 0)) - ds((0, 2)) + ds((0, 0)) = 8$.

Our design and implementation of point data indexes are based on the dominance-sum technique. We first implemented and improved the BA-Tree [21] proposed by Zhang et al., which is a k-d-B tree [11] with some additional

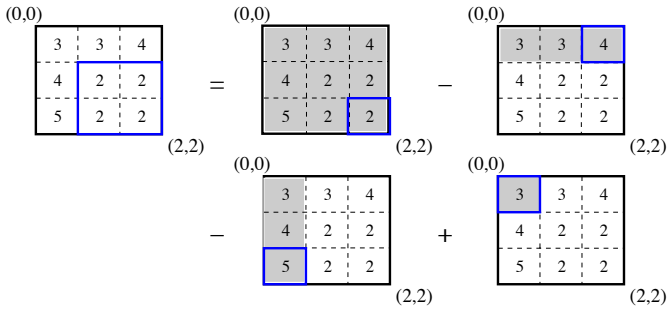

Fig. 4. Computing the range-sum for region $[(1,1),(2,2)]$ by using 4 dom-sum values

aggregate-related information stored at internal nodes. The BA-Tree supports fast dom-sum computations. One improvement we have added to the BA-Tree is as follows. As images stream in, we insert the data for each image spatio-temporal point one by one into the BA-Tree. For a single raster image, all its spatio-temporal points share the same timestamp value. For any two raster images, their spatio-temporal points are likely to share one or both values for the spatial dimensions. In our BA-Tree implementation, we take advantage of this property and introduce an *optimized insert operation* for point data into the BA-Tree. Every time when a point is inserted into the tree, if this point already exists in the tree, we simply add the value of the new point to the value of the existing point. This way, we can significantly reduce the size of the BA-Tree and also improve the performance of querying (refer to [22] for more information).

The major disadvantages of the BA-Tree are that it stores each point in a raster image separately, and does not take advantage of the gridded point set structure inherent to raster images, in particular of the cases where neighboring points have similar or same point values. This causes significantly high storage cost and consequently bad insertion and query performance. To address these shortcomings, we proposed a novel data structure, called the *compressed raster cube (CRC)*. The CRC integrates the *linear region quadtree* [1, 12, 19] and the dominance-sum computation technique, thus allowing efficient management of streaming image point data in a scalable fashion.

The CRC can be thought of as a cube containing a constantly growing sequence of compressed raster images, each of which is essentially a *linear region quadtree* with some *auxiliary information* with each node, namely dom-sum values for the points contained by the node. These dom-sum values are stored in a novel *dominance-sum compression scheme*, which significantly reduces the number of dom-sum values that need to be stored with nodes in a region quadtree. This compression scheme can be illustrated as follows. In general, given a node k in a quadtree such that k contains $m \times m$ points *that all have the same point value*. Only $2m-1$ dom-sum values need to be stored for k in order to compute the dom-sum value for any point contained by k, as shown in Figure 5. The

dom-sum value for any point (x_1, y_1) contained by k can be computed in the following formula:

$$ds((x_1, y_1)) = ds((x_1, y)) + ds((x, y_1)) - ds((x, y)) + (x_1 - x) \times (y_1 - y) \times v$$

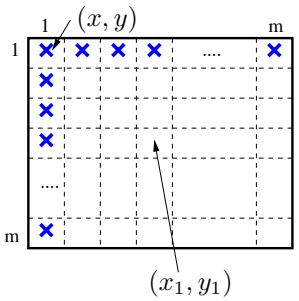

Fig. 5. All required dom-sum values (marked with 'X') for a $m \times m$-size node in which all points have the same point value

To construct such a CRC, we extended the OPTIMAL_BUILD algorithm [12–14] to convert a raster image into a quadtree and at the same time employ our dominance-sum compression scheme. Our experiments on NOAA's GOES West Satellite image data show that the CRC can efficiently evaluate both fix-box aggregate queries and moving window queries.

3.4 Experimental Results

Our experimental data is extracted from NOAA's GOES West satellite data. The GOES satellite carries two types of instruments: the Imager and the Sounder. The Imager scans various regions of the Earth's surface, West-to-East and North-to-South. The data are continually transmitted as frames at a rate of 2.1M bits/sec. In our experiments, we are interested in the image data of the visible band from the Imager. We extract this data from GOES data format into a sequence of lines, each of which has the following format:

```
frameId rowId startColId numOfPoints <point-values>
```

The "frameId" is assigned by the satellite to identify a frame. We use it as a logical timestamp in our experiments. "rowId" represents the absolute id of a row in a scan of an image. "startColId" represents the absolute start column id of a scan. "numOfPoints" gives the number of points in this row. Finally, "<point-values>" represents a list of point values where the list has the size of "numOfPoints".

The first objective of our experiments is to determine how query segmentation affects the overall running time for answering SST-Aggregate queries. We implement our metadata index and the BA-Tree in the Java programming language, and run the programs on a Redhat Enterprise machine with a 2GHz CPU and 512M RAM. The size of each node in the BA-Tree is 4096 bytes.

Figure 6 compares the running time of inserting the image metadata into the metadata index with the running time of inserting the image point data into the BA-Tree. Intuitively, since the amount of the image metadata is much smaller than the amount of point data, the time to insert the metadata should be significantly smaller than the time to insert the point data. Figure 6 verifies this intuition. Figure 7 compares the running time of segmenting a query region with the running time of computing aggregate results for that query region. As one expects, the running time of segmentation is significantly smaller than the running time of computing aggregate results.

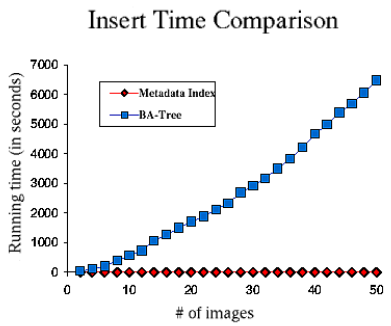

Fig. 6. Time comparison for maintaining two different data structures: Metadata index (bottom line), BA-Tree (upper curve)

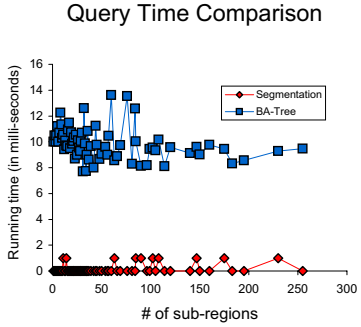

Fig. 7. Time comparison to answer an aggregate query: query segmentation (bottom) and computing aggregate for a single region (top)

The second objective of our experiments is to evaluate the effectiveness of our CRC structure, in terms of space consumption, construction time, and query performance. Our algorithms are implemented using GNU C++. The programs run on a Linux Redhat Enterprise 3 machine with an Intel Pentium 4 $3G$ CPU, $1024K$ cache and $1G$ RAM. For the B^+-tree implementation realizing the linear region quadtree, we choose a node size of 4096 bytes and buffer size of $80M$ bytes.

In this set of experiments, we quantize point values of incoming images by ignoring the $0, 2, 4, 6$ least significant bits, respectively, and collect the results for each of the quantized data. In Figures 8-10, we label these quantized point data sets with CRC-0 ($\hat{=}$ original point data), CRC-2, CRC-4, and CRC-6. For the different quantized data, Figure 8 shows the changes in the number of nodes

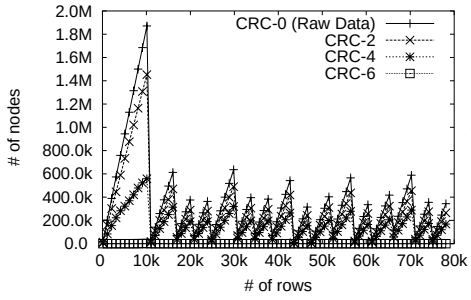

Fig. 8. B^+-tree space consumption for different quantized data

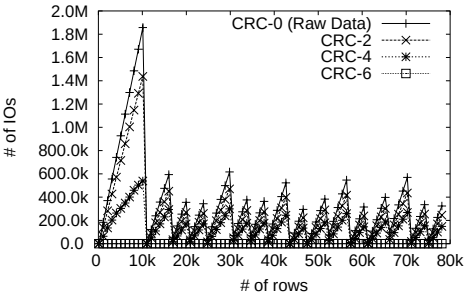

Fig. 9. Number of IOs for different quantized data as points are inserted into the CRC.

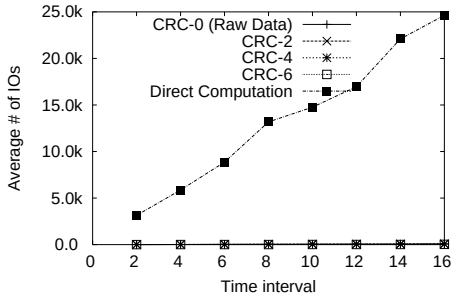

Fig. 10. Comparison of query performance, in IOs, for different quantized data and a direct computation over raw image data. Since the number for a direct computation is much greater than that of any CRCs, all CRCs appear in the same line.

in the B^+-tree for each image as image point data is inserted into the CRC. As expected, the more least significant bits are ignored for the point values, the more the size of the CRC decreases. Figure 9 shows the number of IOs to construct the CRC for the different quantized data. Figure 10 shows the query performance of the CRCs and a direct computation from the raw image data. This direct computation is done as follows. The raw image point data are stored point by point in a file. This direct computation extracts the points that satisfy the query and aggregates their values. As one can see, the CRC significantly outperforms this direct computation on the raw data.

4 Conclusions and Future Work

In this paper, we have outlined the key challenges of computing spatio-temporal aggregate queries over streaming geospatial image data. Several preliminary results have been presented. Our focus has been on summation-related aggrega-

tions – range-sum aggregations. Range-min/max aggregations are another important fundamental operations that support complex data analysis. We are looking into some evaluation approaches that are able to efficiently support both range-sum and range-min/max aggregate queries.

Since we are dealing with streaming data, the management of large volumes of history data is a very important and challenging task. We are investigating *data vacuuming schemes* to compress or eliminate history data in the stream, and then integrating such schemes into our SST-Aggregation model and query processing framework.

Acknowledgment This work is supported in part by the NSF under award IIS-0326517.

References

1. D. Abel. A B+-tree structure for large quadtrees. *Computer Vision, Graphics, and Image Processing*, 10(2):167–170, 1984.
2. R. Bayer. Symmetric binary B-trees: Data structure and maintenance algorithms. In *Acta Informatica*, 290–306, 1972.
3. S. Geffner, D. Agrawal, and A. E. Abbadi. The dynamic data cube. In *EDBT'00*, 237–253, 2000.
4. *GOES I-M DataBook*. Space Systems-Loral, 1996.
5. C.-T. Ho, R. Agrawal, N. Megiddo, and R. Srikant. Range queries in OLAP data cubes. *SIGMOD Rec.*, 26(2):73–88, 1997.
6. M. Jurgens and H. Lenz. The RA*-tree: An improved R-tree with materialized data for supporting range queries on OLAP-data. In *DEXA'98*, 1998.
7. I. F. V. Lopez and B. Moon. Spatiotemporal aggregate computation: A survey. *IEEE TKDE*, 17(2):271–286, 2005.
8. J. Melton. SQL:2003. *ISO (International Organization for Standardization) and ANSI (American National Standards Institute)*, 2003.
9. D. Papadias, P. Kalnis, J. Zhang, and Y. Tao. Efficient OLAP operations in spatial data warehouses. In *SSTD'01*, 443–459. Springer-Verlag, 2001.
10. D. Papadias, Y. Tao, P. Kalnis, and J. Zhang. Indexing spatio-temporal data warehouses. In *ICDE'02*, 166. IEEE Computer Society, 2002.
11. J. T. Robinson. The k-d-b tree: A search structure for large multi-dimensional dynamic indexes. In *ACM SIGMOD*, 10–18, 1981.
12. H. Samet. *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*. Addison-Wesley, 1990.
13. C. A. Shaffer. *Application of alternative quadtree representations (VLSI, data structures)*. PhD thesis, University of Maryland, Computer Science TR-1672, 1986.
14. C. A. Shaffer, H. Samet. Optimal quadtree construction algorithms. *Comput. Vision Graph. Image Process.*, 37(3):402–419, 1987.
15. J. Sun, D. Papadias, Y. Tao, and B. Liu *Querying about the Past, the Present, and the Future in Spatio-Temporal Databases*. In *ICDE'04*, 202. 2004
16. Y. Tao, G. Kollios, J. Considine, F. Li, and D. Papadias. Spatio-temporal aggregation using sketches. In *ICDE'04*, 2004.
17. Y. Tao and D. Papadias. Historical spatio-temporal aggregation. *ACM Trans. Inf. Syst.*, 23(1):61–102, 2005.

18. Y. Tao, D. Papadias, and J. Zhang. Aggregate processing of planar points. In *EDBT'02*, 682–700, 2002.
19. T. Tzouramanis, M. Vassilakopoulos, and Y. Manolopoulos. Overlapping linear quadtrees: a spatio-temporal access method. In *ACM GIS'98*, 1–7, 1998.
20. D. Zhang, D. Gunopulos, V. J. Tsotras, and B. Seeger. Temporal and spatio-temporal aggregations over data streams using multiple time granularities. *Inf. Syst.*, 28(1-2):61–84, 2002.
21. D. Zhang, V. J. Tsotras, and D. Gunopulos. Efficient aggregation over objects with extent. In *PODS'02*, 121–132. ACM Press, 2002.
22. J. Zhang, M. Gertz, and D. Aksoy. Spatio-temporal aggregates over raster image data. In *ACM GIS'04*, 39–46. ACM Press, 2004.