Proc. 5th Int. Symp. on Spatial Data Handling, Charleston, S.C., 1992, pp. 200-209

Indexing the Spatiotemporal Monitoring of a Polygonal Object

Ralf Schneider and Hans-Peter Kriegel

Institut für Informatik, Universität München,
Leopoldstr. 11B, D-8000 München 40, Germany

e-mail: ralf@dbs.informatik.uni-muenchen.de, kriegel@dbs.informatik.uni-muenchen.de

Abstract

In Geographic Information Systems spatial objects may change their spatial location and/or shapes over time. Query processing of such spatiotemporal objects consists of spatial query conditions and temporal query conditions. Our study focuses on representing the spatiotemporal monitoring of a polygonal object and aims at developing an efficient data structure supporting queries and operations that refer to both spatial and temporal monitoring. In this paper, we present a first performance comparison which indicates that spatial indexing offers a possibility of efficient query processing of a spatiotemporal object.

1. Introduction

In many applications of Geographic Information Systems (GIS) spatial objects are generally perceived to be time varying. Thus, complex spatiotemporal objects have to be managed. An important class of such spatiotemporal objects are two dimensional polygonal objects that may change their spatial location and/or shapes over time. To mention an example, in an interdisciplinary project we designed a bioindication database (Kriegel and Schneider, 1990) in which lichens were used as a biological indicator for environmental pollution. Growth and change of the lichen's shape are a quantitative and qualitative measure of the environmental stress to the lichen. To investigate this stress the spatiotemporal monitoring of the lichen has to be represented. Before solving the general problem of efficiently handling of spatiotemporal objects, in this paper we will restrict the problem to the temporal sequence of a single object as it occurs for lichens or historical maps, e.g.

In Geographic Information Systems there are two types of representations for spatial data: vector and raster representations. In this paper, we restrict our considerations to vector representation because there are two main disadvantages of raster representation (Oosterom, 1990): (1) Raster data depends on a specific projection. This leads to massive problems when combining raster maps from different sources. A scaleless database cannot be implemented using a raster representation. (2) Objects in raster maps generally are not handled individually. Thus, a support by access methods is more difficult. Additionally, raster data need considerably more disk space.

Versioning mechanisms, e.g. mentioned in (Katz, 1985), are hardly practicable for polygonal objects, because representing changes from one version to another is often as costly as representing the complete version. Our experiences in designing the bioindication database confirmed this.

Xu et al. proposed in (Xu, Han and Lu, 1990) an improved index structure for spatiotemporal databases. In their approach the spatial objects are approximated by minimum bounding boxes. As a consequence, their index structure does not exactly evaluate a query, but only yields a set of candidates, that may fulfill the query condition. Therefore, these candidates have to be examined in a second step. In this step, complex algorithms known from the field of computational geometry are applied to the original spatial objects detecting those objects finally fulfilling the query condition. In (Kriegel, Horn and Schiwietz, 1991) it is shown that query processing of complex spatial objects is dominated by the complex and time consuming computational geometry algorithms. Therefore, the primary goal for efficient query processing is to reduce the computational geometry time. An index on minimum bounding boxes (MBBs) of spatial objects yields marginal improvement because the approximation using MBBs is often so bad that the number of candidates is not considerably reduced (Kriegel et al., 1991). Due to this observation also the interesting versioning approaches proposed in (Lanka and Mays, 1991) and (Driscoll et al., 1989) do not efficiently support the spatiotemporal indexing of the exact representation of polygons.

An important characteristic of a spatiotemporal object is the very high overlap of the polygons that represent the instances of the spatiotemporal object over time. Therefore, using the cell tree (Günther, 1989) or the edge quad tree (Samet, 1990) for spatial indexing of the exact representation of the spatiotemporal object is not recommendable, because the efficiency of these index structures degenerates when the overlap of the polygons increases. In this paper, we will demonstrate that processing time of spatiotemporal queries can be essentially reduced by investing time in preprocessing the object representation using decomposition. Decomposition of complex objects into simple components removes complex computational geometry algorithms from query processing replacing them by simple and fast algorithms for the components.

Query processing of spatiotemporal objects consists of spatial query conditions and temporal query conditions, e.g.: 'Which versions of an object overlap a query region from time t_i to t_j ?'. Our study focuses on representing the spatiotemporal monitoring of a polygonal object and aims at developing an efficient data structure supporting queries and operations that refer to both spatial and temporal monitoring.

2. Queries on a Spatiotemporal Object

First, we have to define which types of spatial objects we want to consider. Our spatial objects are *simple polygons with holes* where simple polygonal holes may be cut out from the simple enclosure polygon (see figure1). A polygon is called simple if there is no pair of nonconsecutive edges sharing a point. From our experience, the class of simple polygons with holes is adequate for GIS applications, which is also underlined in (Burrough, 1986).

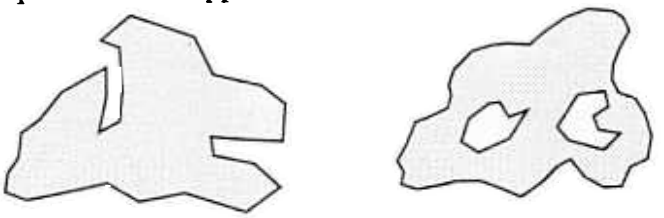

Figure 1 simple polygon simple polygon with holes

In the following, a spatiotemporal object is defined as a sequence of such polygons with incremental time stamps. The instance of a spatiotemporal object at a special point in time is also called a version. A spatiotemporal query is an arbitrary combination of spatial query conditions and temporal query conditions. From the literature, no standard set of spatial as well as temporal queries fulfilling all requirements of GIS applications can be derived (Scholl and Voisard, 1989). Thus it is necessary to provide a set of basic queries that are efficiently supported by a data structure. Application specific queries and operations, e.g., in (Oosterom, 1990), using more complex query conditions, can be decomposed into a sequence of such basic queries. In the following we list a set of basic spatial and temporal queries.

Spatial queries (the query region may be a polygon with holes)

- a *point query* reports those instances of the spatiotemporal object that contain the query point.
- an *intersect query* reports those instances that intersect the query region.
- an *enclosure query* reports those instances that contain the query region.
- a *containment query* reports those instances that are contained by the query region.
- an *intersection query* computes the intersection of the query region with all instances of the spatiotemporal object.

Temporal queries

- report the instance of the spatiotemporal object *at* time stamp t_i.
- report all instances of the spatiotemporal object *from* time stamp t_i *to* time stamp t_j.
- report the k^{th} instance of the spatiotemporal object *after* time stamp t_i.

  report the k^{th} instance of the spatiotemporal object *before* time stamp t_i

For representing a sequence of polygons with incremental time stamps the traditional method is to create an index over the time stamp and to store each polygon separately. Using this method, a spatiotemporal query is answered in a two step manner:

Step 1: The temporal query condition is evaluated using the time index. A set of candidates fulfilling the temporal query condition is selected.

Step 2: Using complex and time consuming computational geometry algorithms the set of candidates is tested whether the spatial query condition is fulfilled.

In the traditional method only the temporal query condition is indexed whereas the time consuming computational geometry algorithms are not supported by an index. In (Kriegel, Horn and Schiwietz, 1991) and (Kriegel et al., 1991) it is illustrated that a spatial index may be constructed for an exact representation of polygons if the complex polygons are decomposed into a set of simple components. In the following, we want to demonstrate how spatiotemporal queries are efficiently performed using a spatial index for decomposed polygons.

3. The Representation of a Spatiotemporal Object Using the TR*-tree

The basic idea of our approach can be summarized as follows: We decompose each polygon of the sequence into a set of disjoint simple components for which state of the art computational geometry concepts are used in order to guarantee efficiency. The time stamp of the corresponding polygon is attached to each decomposition component. Due to this preprocessing step, queries and operations can be simplified. For example, a complex 'Point in Object'-test is replaced by a 'Point in Simple Component'-test and 'Intersection of Objects' is replaced by 'Intersections of Simple Components'. A general consequence of this decomposition is that complex and time-consuming algorithms are replaced by a set of simple and fast algorithms. As a further advantage, the decomposition facilitates the exploitation of the spatial locality of queries and operations (see figure 2). The effect of spatial locality of queries and operations is enforced for spatiotemporal objects, depending on the number of polygons within the sequence.

Figure 2: Spatial locality of queries

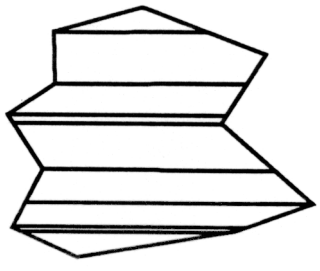

Figure 3: Trapezoid-Decomposition

According to our comparison of decomposition techniques (Kriegel et al., 1991) we decided to decompose a polygon into a set of trapezoids introduced by Asano and Asano

(Asano and Asano, 1983). The components produced by this algorithm are formed as trapezoids containing two horizontal sides. The algorithm uses the plane sweep technique known from the field of computational geometry. The basic idea is to send out for each vertex one or two horizontal rays into the interior of the polygon to the first edge encountered (see figure 3). The inverse operation of the decomposition process generating the sorted list of vertices of the polygon can be performed efficiently using a plane sweep merge algorithm similar to the map overlay algorithm proposed in (Kriegel, Brinkhoff and Schneider, 1991).

The main properties of the decomposition into trapezoids are:

- the trapezoids possess two horizontal edges due to their construction; thus the possibilities of the decomposition process are restricted to the choice of the sweep line.
- the trapezoids are simple components with fixed length
- a polygon with n vertices is decomposed into < n trapezoids.
- the decomposition is easy to implement (Kriegel et al., 1991) and has a run time performance of O(n log n) (Asano and Asano, 1983).

After decomposing the polygons into trapezoids, it is possible to perform the queries and operations on these trapezoids. The success of this kind of query processing depends on the ability to narrow down quickly the set of trapezoids that are affected by queries and other operations. Binary search on these trapezoids is not possible because we cannot define a complete spatial order on the set of trapezoids that are generated by the above decomposition process (Preparata and Shamos, 1988). Therefore, a new approach is necessary that permits efficient spatial search on the trapezoids. This approach must be suitable for all basic spatial queries and operations described in section 2 and it must be completely dynamic, i.e., updates, insertions and deletions of new instances of the spatiotemporal object do not influence the performance of the structure and require no global reorganization. In (Schneider and Kriegel, 1991) we proposed the TR*-tree, a main memory data structure that efficiently fulfill these requirements. The TR*-tree representation of a spatiotemporal object is persistently stored on secondary storage and it is completely loaded into main memory for query processing. The TR*-tree is derived from the R*-tree described in (Beckmann et al., 1990). The R*-tree is an optimized variant of the well known R-tree (Guttman, 1984), a data structure for storing multidimensional intervals.

A TR*-tree stores trapezoids as complete objects without clipping them or transforming them to higher dimensional points. A non-leaf node contains entries of the form (cp, Rectangle) where 'cp' is the address of a child node in the TR*-tree and 'Rectangle' is the minimum bounding rectangle of all rectangles that are entries in that child node. A leaf node contains entries of the form (Oid, Trapezoid) where 'Trapezoid' is a

component of the decomposed polygon with object identifier 'Oid'. Obviously, the structure has to allow overlapping directory rectangles. Thus it cannot guarantee that only one search path is required for retrieval queries.

4. An Empirical Performance Comparison of Spatiotemporal Query Processing

In this section our study focuses on the comparison of two different approaches answering queries on a spatiotemporal object that is represented by a sequence of polygons with an incremental time stamp.

The representation of a spatiotemporal object using the traditional approach:

The time stamps of the instances are stored in a 2-3-tree introduced by Hopcroft (Aho, Hopcraft and Ullman, 1987). A leaf node of the 2-3-tree contains entries of the form E = (pid_i, t_i) where $E.pid_i$ references to the exact representation of the instance of the spatiotemporal object at time stamp t_i.

The representation of a spatiotemporal object using our new approach:

In a preprocessing step each polygon of the sequence is decomposed into a set of trapezoids. Each trapezoid receives the time stamp of the original polygon. The trapezoids of all decomposed polygons are stored together in a single TR*-tree. A leaf node of this TR*-tree contains entries of the form E = (trap, t_i) where 'trap' represents a trapezoid of the polygon at time stamp t_i. A spatiotemporal query is answered as follows: First, a spatial search is performed by traversing the directory of the TR*-tree to find the trapezoids that are affected by the spatial query condition. Thus, a set of candidate trapezoids is selected. In a second step, the spatial query is performed only on those trapezoids fulfilling the temporal query condition. This is done by testing the time stamp of the selected trapezoids against the temporal query condition.

Opposing strategies are used in the two approaches above. In the traditional approach, the temporal query condition is supported by an index, whereas in our approach the spatial query condition is supported by an index using a new data structure, the TR*-tree. Our approach is guided by the goal to minimize the amount of complex and time consuming computational geometry algorithms in query processing. Complex algorithms are avoided by decomposing the polygons into a set of trapezoids. Thus, only operations on trapezoids have to be performed evaluating the computational geometry part of the spatiotemporal query. Additionally, by using the TR*-tree, an efficient spatial access method, the number of tested trapezoids is minimized.

In the following we want to investigate the performance trade-off of the two methods depending on which query condition the spatial or the temporal dominates the overall query. In (Schneider and Kriegel, 1991) we demonstrated that various types of spatial

queries and operations can be performed efficiently using the TR*-tree representation of polygons instead of implementing several tailor-cut computational geometry algorithms. Therefore, in the traditional approach we use a TR*-tree for each polygon within the sequence evaluating the spatial query condition of the spatiotemporal query. Additionally, this approach allows a standardized comparison of significant performance parameters because the two approaches use the same data structure in evaluating spatial queries and operations. We prefer this way of an empirical performance comparison because CPU-time measurements are generally not comparable and different implementations of algorithms and data structures lead to different CPU-times.

A basic problem of every experimental performance analysis is the selection of an appropriate set of test data. Because of the lack of real data, we generated a sequence of 200 different polygons with each polygon consisting of 75 edges and a small variance in size of the polygons. Each polygon represents an instance of the spatiotemporal object at a different time stamp. To simulate a spatiotemporal object we centralized the location of the polygons in such a way that the centers of all 200 polygons cover the same point. This data is representative for objects occurring in the bioindication application.

In our first experimental performance comparison we analyzed only one example query: *"Report all instances within a time interval that contain a given query point"*. In order to standardize the comparison we investigated successful point queries in randomly distributed time intervals, i.e., at least one instance of the spatiotemporal object contains the query point and is specified by the time interval. Therefore, the query answers vary from 0.5% to 100% of the 200 instances of the spatiotemporal object. The performance of the example query processing is characterized by counting the following three parameters:

- The number of time stamp comparisons are a measure of the expense of evaluating the temporal query condition
- The number of 'Point in Rectangle'-tests (abr. PiR-tests) is a measure of the time that is spent for the spatial search traversing the directory of the TR*-trees.
- The number of 'Point in Trapezoid'-tests (abr. PiT-tests) is a measure of evaluating the computational geometry component of the example query.

In the presentation of the results we do not list the number of the time stamp comparisons evaluating the temporal query condition because this time is negligeable with respect to spatial search (PiR-tests) and computational geometry expense (PiT-tests).

The performance gap between the traditional and our new approach depends on the percentage of the instances of the spatiotemporal object that fulfill the example query. Thus, we depict on the horizontal axis of figure 4 and 5 the percentage of instances of the spatiotemporal object fulfilling our example query. In figure 4 the vertical axis depicts the

number of PiR-tests comparing the efficiency of spatial search of the two approaches. In figure 5 we present the number of PiT-tests, a measure of the computational geometry expense of the two approaches.

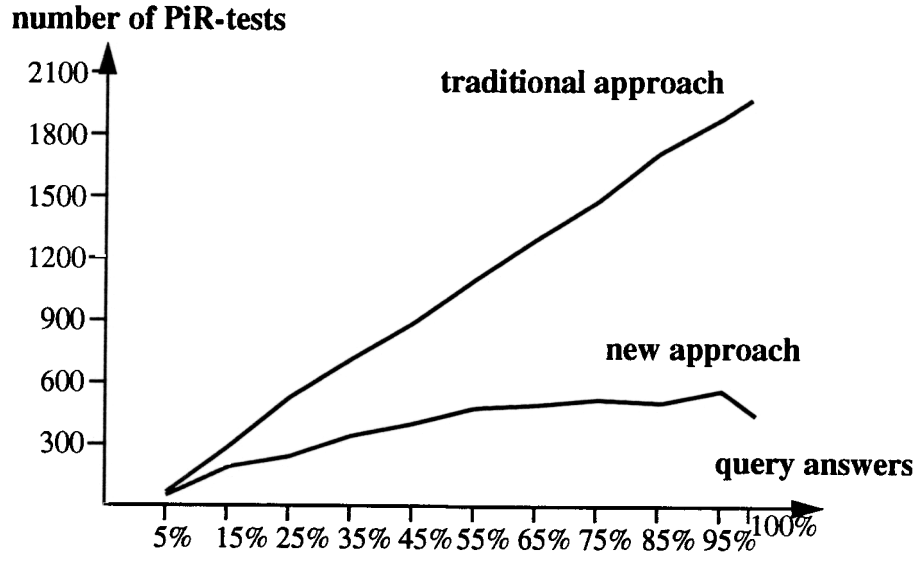

Figure 4: Comparison of the spatial search

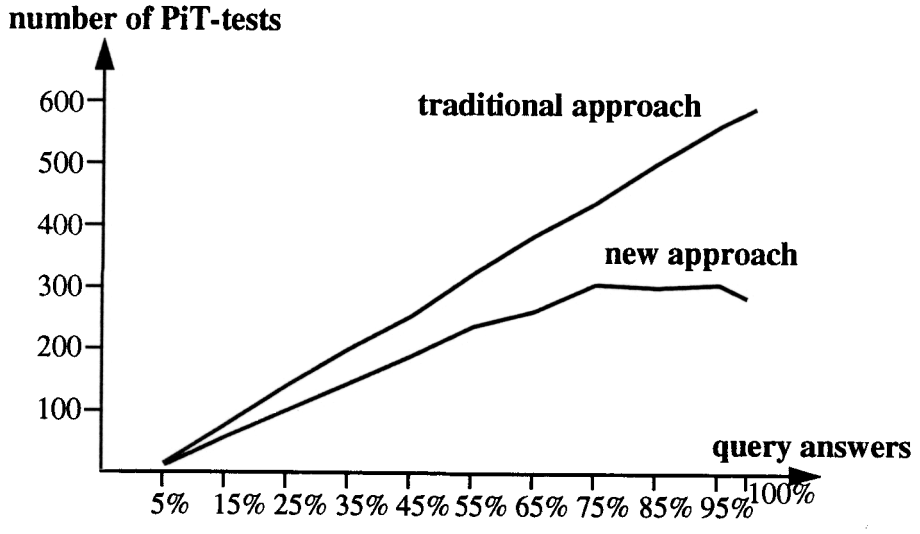

Figure 5: Comparison of the computational geometry expense

Interpretation of the Results:

In the traditional approach the spatiotemporal object is represented by n TR*-trees where n is the number of instances of the spatiotemporal object. Therefore, the spatial search and the computational geometry expense grows linearly because the candidates specified by the time index can only sequentially tested evaluating the spatial query condition. In our query example, m candidates require a point query of m TR*-trees (see figure 4 and 5).

Contrarily, if we use one single TR*-tree representing the spatiotemporal object the spatial search shows a nearly logarithmic growth with respect to the number of found instances fulfilling the example query (see figure 4). This is based on the tree structure of the TR*-tree (Schneider and Kriegel, 1991). Our new approach reduces the computational geometry cost because the TR*-tree is a data structure that clusters overlapping as well as neighboring trapezoids very well. Therefore, only a few unsuccessful PiT-tests are performed. For example, a TR*-tree representing a polygon with 75 edges performs 2.9 PiT-tests on the average to answer a successful point query. However, our approach needs 289 PiT-tests for a point query that specifies all 200 instances of the spatiotemporal object, i.e. on the average 1.4 PiT-tests for one hit.

If less than 5% of the instances of the spatiotemporal object are specified by the temporal query condition, the traditional approach answers the example query always faster. In this case, we have such an high selectivity of the temporal query condition that a spatial indexing is not helpful. At 5% we have the break-even point between temporal and spatial indexing for our example query.

This first performance comparison shows that spatial indexing offers a possibility of efficient query processing of a spatiotemporal object. In our future work, we will investigate in a more extensive comparison the general performance trade-off of temporal and spatial indexing with respect to different types of spatiotemporal queries and a variation of object parameters such as location, shape and number of vertices, also incorporating real data. Furthermore, we will design a query processor that supports multiple representations of objects, optimizes between temporal indexing and spatial indexing and operates on sets of spatiotemporal objects.

References

Asano Ta. and Asano Te. 1983: *'Minimum Partition of Polygonal Regions into Trapezoids'*, Proc. 24th IEEE Annual Symposium on Foundations of Computer Science, 1983, pp. 233-241.

Aho A., Hopcraft J. and Ullman J. 1987: *'Data Structures and Algorithms'*, Addison-Wesley, 1987.

Beckmann N., Kriegel H.-P., Schneider R. and Seeger B. 1990: *'The R*-tree: An Efficient and Robust Access Method for Points and Rectangles'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Atlantic City, N.J., 1990, pp. 322-331.

Burrough P. A. 1986: *'Principles of Geographical Information Systems for Land Resources Assessment'*, Oxford University Press, 1986.

Driscoll J., Sarnak N, Sleator D. and Tarjan R. 1989: *'Making Data Structures Persistent'*, Journal of Computer and System Sciences, Vol. 38, 1989, pp. 86-124.

Günther O.: *'The Design of the Cell Tree: An Object-Oriented Index Structure for Geometric Databases'*, Proc. IEEE 5th Int. Conf. on Data Engineering, 1989, pp. 508-605.

Guttman A. 1984: *'R-trees: A Dynamic Index Structure for Spatial Searching'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Boston, MA., 1984, pp. 47-57.

Katz R. 1985: *'Information Management for Engineering Design'*, Springer, 1985.

Kriegel H.-P., Brinkhoff T. and Schneider R. 1991: *'An Efficient Map Overlay Algorithm based on Spatial Access Methods and Computational Geometry'*, Proc. Int. Workshop on Database Management Systems for Geographical Applications, Capri, 1991.

Kriegel H.-P., Horn H. and Schiwietz M. 1991: *'The Performance of Object Decomposition Techniques for Spatial Query Processing'*, Proc. 2nd Symp. on the Design of Large Spatial Databases, 1991, in: Lecture Notes in Computer Science, Vol. 525, Springer, 1991, pp. 257-276.

Kriegel H.-P., Heep P., Heep S., Schiwietz M. and Schneider R. 1991: *'An Access Method Based Query Processor for Spatial Database Systems'*, Proc. Int. Workshop on Database Management Systems for Geographical Applications, Capri, 1991.

Kriegel H.-P. and Schneider R. 1990: *'Design of a Bioindication Database System'* (in German), Proc. 5th Symp. Informatik für den Umweltschutz, Wien, 1990.

Lanka S. and Mays E. 1991: *'Fully Persistent B^+-tree'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, 1991.

Oosterom P. J. M. 1990: *'Reactive Data Structures for Geographic Information Systems'*, Ph.D.-thesis, Dept. of Computer Science at Leiden University, 1990.

Preparata F. P. and Shamos M. I. 1988: *'Computational Geometry'*, Springer, 1988.

Samet H. 1990: *'The Design and Analysis of Spatial Data Structures'*, Addison Wesley

Schneider R. and Kriegel H.-P.: *'The TR*-tree: A New Representation of Polygonal Objects Supporting Spatial Queries and Operations'*, Proc. 7th Workshop on Computational Geometry, Bern, 1991, in: Lecture Notes in Computer Science, Vol. 553, Springer, 1991, pp. 249-264.

Scholl M. and Voisard A. 1989: *'Thematic Map Modelling'*, Proc. 1st Symp. on the Design and Implementation of Large Spatial Databases, Santa Barbara, 1989, pp. 167-190.

Xu X., Han J., Lu W. 1990: *'RT-Tree: An Improved R-Tree Index Structure for Spatiotemporal Databases'*, Proc. 4th Int. Symp. on Spatial Data Handling, Zürich, 1990.