Query Processing of Spatial Objects: Complexity versus Redundancy

Michael Schiwietz and Hans-Peter Kriegel

Institute for Computer Science, University of Munich
Leopoldstr. 11B, D-8000 Munich 40, Germany
e-mail: {michael,kriegel}@dbs.informatik.uni-muenchen.de

Abstract: The management of complex spatial objects in applications, such as geography and cartography, imposes stringent new requirements on spatial database systems, in particular on efficient query processing. As shown before, the performance of spatial query processing can be improved by decomposing complex spatial objects into simple components. Up to now, only decomposition techniques generating a linear number of very simple components, e.g. triangles or trapezoids, have been considered. In this paper, we will investigate the natural trade-off between the complexity of the components and the redundancy, i.e. the number of components, with respect to its effect on efficient query processing. In particular, we present two new decomposition methods generating a better balance between the complexity and the number of components than previously known techniques. We compare these new decomposition methods to the traditional undecomposed representation as well as to the well-known decomposition into convex polygons with respect to their performance in spatial query processing. This comparison points out that for a wide range of query selectivity the new decomposition techniques clearly outperform both the undecomposed representation and the convex decomposition method. More important than the absolute gain in performance by a factor of up to an order of magnitude is the robust performance of our new decomposition techniques over the whole range of query selectivity.

1 Introduction

The trend of digitizing the wide field of graphic and geographic real world objects results in a strongly growing amount of spatial data with no end in sight. The handling of spatial data is an essential feature in a wide range of applications, such as geographic information systems, image databases, multimedia databases, engineering (CAD/CAM/CAE), as well as geographic and medical applications. Furthermore, the same set of data such as maps often has to be provided for different application areas and different access patterns. Due to data centralization efforts supporting different applications by one central database system, these data have to be managed by a flexible and extensible query processing system which can be extended regarding application specific requirements. Therefore, the demand for a support by a global integrated spatial database system is considerably increasing.

One important characteristic of most spatial applications is the occurrence of complex spatial objects and their algorithmic treatment including query processing. In a relational database system, spatial objects would artificially spread over serveral relations. As a consequence, they have to be rebuilt on an operational level. Thus, spatial query processing is inefficient and unflexible. Therefore, the management of complex spatial objects imposes stringent new requirements on integrated spatial database systems and particularly, on an efficient processing of spatial queries.

An important class of objects occurring in spatial applications are two-dimensional spatial objects. A spatial object is embedded within a global spatially oriented data space. Points, lines, and rectangles are known as simple spatial objects, because their complete geometry and structural description is given by a small and fixed number of parameters. In contrast, complex spatial objects with an application specific complexity, such as contour lines, limits of lots, and contours of CAD objects are often shaped in the form of simple polygons. Complexity properties of such polygonal objects, such as the shape, the number of vertices, the area or the smoothness of the contour are difficult to predict.

Spatial objects occurring in real applications are typically not homogeneous with respect to the number of vertices, the smoothness of the shape, or the object-area. As outlined in [Fra 91] these parameters underly an extreme variance even for objects of one and the same application. Because simple and complex spatial objects are heterogeneously stored within the same set of data, an object-oriented organisation and query processing is essential.

Spatial queries combine the requirements of a spatial locality search and an exact evaluation of complex geometric properties. Therefore, the handling of complex spatial objects concerns their management on secondary storage as well as the evaluation of main memory algorithms from the field of computational geometry. The traditional approach uses bounding box approximations representing their spatial location clustered by spatial access methods (SAMs; see e.g. [NHS 84], [See 89]) and applies complex computational geometry algorithms to their exact representation. This appoach reveals strong disadvantages caused by the coarse approximation and the expensive computational geometry algorithms. These drawbacks are avoided by object decomposition techniques introduced in [KHS 91]. However, object decomposition techniques use a set of simple components representing a complex spatial object. The number of components, called *redundancy* in the remainder of this paper, results in a storage and query processing overhead. Due to the high amount of redundancy of the traditional object decomposition techniques, e.g. triangulations, the storage overhead is unacceptable and the efficiency of spatial queries decreases with an increasing size of the query region. Therefore, the development of new object decomposition techniques is essential for providing a locality based object treatment and for strongly limiting the amount of redundancy.

In this paper, we will in detail examine the correlation between spatial clustering given by SAMs and computational geometry techniques (e.g. [PS 88]). We will introduce new object-oriented decomposition methods that combine both a locality based representation and a small amount of redundancy. An evaluation of spatial query processing shows the superiority of our new decomposition methods compared to known approaches for typical queries of spatial applications.

The paper is organized as follows: the next section contains an overview of known techniques combining spatial clustering and computational geometry using object decomposition techniques for the improvement of geometric algorithms. Section 3 discusses complexity and redundancy aspects. Our new locality based spatial object decomposition techniques regarding object and application oriented requirements are introduced and evaluated in the sections 4 and 5. In section 6, we present a performance comparison between these new representation schemes and the traditional methods. Section 7 concludes the paper.

2 Spatial query processing based on object decomposition

A typical property of spatial queries is their strictly restricted spatial location within the global data space. Only that location and some limited neighboring area can contribute to the query result. Depending on its restricted spatial locality, a query is called *selective* or *non-selective*. Selective queries limit the area of concern and, therefore, the number of objects relevant to the query, whereas non-selective queries require the examination of large portions of the data space.

Obviously, spatial queries are order-preserving. Objects lying close together in the data space are often accessed together. Therefore, a physical clustering of objects with respect to their spatial location is essential for providing an efficient locality based and selective access to the objects. A good spatial clustering is maintained by SAMs commonly known from the literature. In the absence of such spatial clustering, no spatial locality can be exploited by the query processing algorithm. Each stored object has to be explicitly evaluated against the query condition often requiring very time consuming algorithms. This causes a poor query performance which is further decreasing with an increasing number of objects and object complexity. Therefore, one essential ingredient of a spatial database system with respect to an efficient locality based spatial query processing is a good spatial clustering of the objects with respect to their natural spatial location.

A direct handling of complex spatial objects requires a high amount of algorithmic complexity in managing spatial accesses. For this reason, no SAM is available for complex spatial objects and for the class of simple polygons particularly.

For providing efficient handling and query processing of spatial objects traditionally the following approach is applied. Every complex spatial object is placed within a rectilinear rectangle of minimum area forming a simple *container*. A simple spatial object is called container iff any point inside the contour of the object is also contained in the contour of the container. This yields a so-called conservative approximation. For any closed spatial object, a unique minimum rectangular container can be generated. The container object substitutes the original object with respect to its spatial clustering, representing its approximate location and extension within data space.

As simple containers just provide a coarse approximation of arbitrary spatial objects, query processing is performed in a two step approach. The first step, called filter step, reduces the entire set of spatial objects to a subset of candidates applying a process of spatial locality search. This process exploits the SAM facilities in managing simple containers. It is based on the following property:

If the container does not fulfill the query condition, then this is also true for the corresponding spatial object.

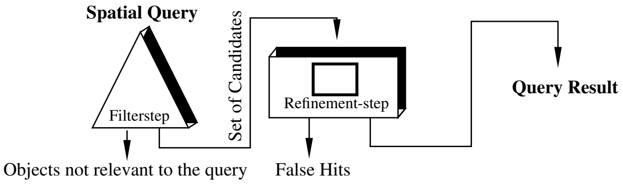


Fig. 1. The two-step approach of a spatial query processing

Therefore, a spatial query on the file of container objects yields a subset of objects definitely including the set of answers to the query. However, the filter step does not exactly evaluate the query, but only yields a set of candidates which potentially may fulfill the query condition. Therefore, those candidates have to be further examined in a second step, the refinement step. The refinement step applies complex spatial algorithms known from the field of computational geometry to the candidates. It detects those objects, actually belonging to the set of answers to the query. Due to these tasks, the filter step is based on a spatial indexing scheme for simple spatial objects, e.g. rectilinear rectangles, representing object containers. The filter step is I/O-intensive, whereas the refinement step using main-memory based computational geometry algorithms is CPU-intensive.

For many years, it was a common agreement that the query performance within database systems is determined by the time necessary for secondary storage accesses. This was due to the fact that main memory operations remained cheap and thus could be neglected. Therefore, minimizing the number of disk accesses was the main goal heading for good performance in a query system. However, if the complexity of the objects and, therefore, the time consumed by main memory algorithms is high, accesses to secondary storage are still important but are definitely no longer the major factor determining query performance. Therefore, in the case of spatial database systems, the more complex spatial objects are, the more dominating are the spatial algorithms performed within main memory. A *´Point in Object´*-test, for example, performed on a polygon with 10,000 vertices, occurring in real world geographic applications, consumes the same amount of time as a few 100 secondary storage accesses, with the exact break-even-point depending on the underlying hardware. Therefore, not only the filter step, but also and more important the refinement step determines the overall performance of complex spatial queries.

At first glance, the conservative approach of directly coupling a SAM and computational geometry algorithms seemed to provide a good and general method for a spatial query processing on arbitrary spatial objects. However, more detailed considerations reveal considerable disadvantages:

- examinations of real world geographic object files turned out the bad approximation provided by rectangular containers with some 100% additionally covered area ([Schi 93]). This leads to roughly twice the number of candidates with respect to the cardinality of the answers for point queries.
- the refinement of one single object is very costly, particularly if the object complexity is high. Complex and time-consuming geometric algorithms evaluating the global shape of the objects have to be applied. Generally, no local restriction can be found limiting the evaluation to local aspects of the object.

The first effort to optimize the performance of the refinement step is obviously an optimization of the underlying spatial algorithms using sophisticated techniques from the field of computational geometry. In fact, this isolated tuning of the algorithms for the refinement step will lead to a considerable enhancement in comparison to ad hoc implementations, but is obviously constrained. The main disadvantage of computational geometry algorithms is that complete spatial objects are handled even if only some local aspects of those objects are relevant to a given query.

The next step for improving query processing concerns the approximation quality. The approximation quality of spatial containers can be enhanced by two ways: using a more complex container or using a set of simple objects as a container. More complex containers introduced in [Schi 93] require more complex SAMs ([Gün 89], [Jag 90], [Schi 93]) and do not affect the complexity of the spatial object itself. In contrast, object decompositions combined with set-oriented containers based on the set of components result in a tuning of both the filter step and the refinement step.

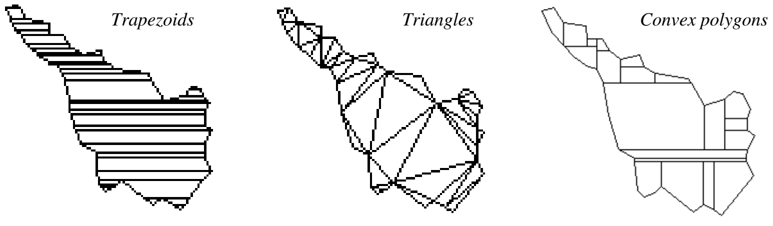


Fig. 2. Decomposition of spatial objects into simple components

In [KHS 91] the spatial and algorithmic overhead of object decomposition techniques have been evaluated and shown to be worthwhile. Following the algorithmic concept of '*divide and conquer*', the decomposition of complex spatial objects into triangles, trapezoids, rectangles, or even convex polygons leads to both a better performance of the filter step and simpler spatial algorithms on simple components. The main drawback is given by the huge amount of components. For triangles and trapezoids (see figure 2), the number of components grows linearly in the number of vertices of the original object. This is due to the fact that the particular components are characterized by topological and not by spatial aspects. Surprisingly, this is also true for the convex decomposition. The reason can be found in the typically high number of 'notches' or reflex angles in the contour of geographic objects violating the condition of convexity caused by digitalization aspects. The large number of components leads to a high amount of redundancy and thus results in a slow performance of low-selectivity spatial queries.

A computational geometry structure providing a spatially restricted search on an object decomposed into a number of trapezoids is given by the trapezoid tree ([PS 88]). Decomposing a spatial object into an $O(n \cdot \log n)$ number of components leads to a totally ordered decomposition scheme. This allows an efficient refinement evaluation of point queries based on binary search over the object shape. However, the high amount of storage, the insufficient handling of extended query regions as well as the applicability to other types of queries prevent the use of this scheme in real world applications.

Similarly, the TR^*-tree ([SK 91], [Schn 92]) tunes the refinement step by introducing a spatial locality on the object shape based on a decomposition into trapezoids. The set of components is organized by the hierarchical index of the R^*-tree ([BKSS 90]). Spatial queries are processed by a tree search directed by the object shape. Because of the locality based definition of the trapezoid decomposition ([KHS 91]), their overlap is small. However, the large amount of components and structural pointers implies that the index needs about the same amount of storage as the original object description. This high amount of additional storage as well as the fact that no spatial storage scheme is provided and complete objects have to be transmitted into main memory is the main drawback of this approach.

However, all these decomposition approaches including the large number of grid based representation schemes (e.g. [Sam 90], [Ore 89], [OM 86]) provide no flexibility for object or application specific requirements to the shape or the complexity of the original object. This is due to the strict condition of the partitioning process given by a predefined constraint to the shape and structure of the components. Therefore, no object-oriented paradigm is applicable controlling the decomposition algorithm.

In the next section, we consider the main aspects of the number and the complexity of the components in spatial object representations.

3 Complexity versus redundancy

The basic idea of any persistent decomposed object representation is to improve query performance by shifting time requirements from query processing to update and restructuring operations. The same principle is applied in any type of access method which provides an additional effort in object organization heading for an improvement of retrieval performance. In typical database applications, retrieval operations occur considerably more frequently in comparison to update operations. Additionally, retrieval operations are usually performed by the user in a dialogue set-up and therefore, have to strive for the best possible performance. Thus, for a reduction of query processing time, it is worth accepting some limited amount of additional time spent for object preprocessing. A typical preprocessing effort is the precomputation of computational geometry aspects.

The persistent handling of such precomputations causes a storage overhead in the object representation as compared to the handling of an object as a linear sequence of vertices. Some additional storage is required for maintaining geometric and topologic locality. Typically for the representation of spatial objects, a preprocessing step is used transforming complex spatial objects into a set of simpler components where the number of components determines the degree of redundancy.

Up to now, two main approaches of structural object representation schemes are known for spatial databases:

- the conservative approach inducing no structural redundancy at all (no decomposition)
- a linear number of components as a result of structural decompositions into very simple spatial components.

As shown before, both approaches are not best suitable for a global spatial retrieval system. The conservative approach avoids redundant object representation at all. Therefore, it is lacking of any spatial object structuring. The whole object geometry has to be transmitted into main memory even if resulting in a false hit. Geometric operations are dependent on the global shape and therefore, perform rather slow, particularly for very complex objects.

In contrast, structural object decompositions define a spatial structure on the object shape. However, the object decomposition into components of a simple shape with constant complexity results in a high number of components. As every component independent of the object represents a constant measure of complexity, the cardinality of the set of components grows linearly in the object complexity. Caused by this high redundancy, spatial query processing is burdened by the following problems and deficiencies:

- time consuming filter step in the case of low-selectivity queries
- high amount of structural redundancy and thus high amount of storage
- complex and expensive inversion of the decomposition process, i.e. generation of the original object from the components
- very expensive update- and delete-operations (e.g. updating or deleting one or a few vertices of a spatial object)

The principal intention of any structural decomposition is replacing retrieval complexity by preprocessing and representational redundancy. Considering the drawbacks listed above, the central question is now:

Which degree of redundancy is best suitable for efficient spatial query processing?

Both representations considered up to now reveal an extreme imbalance between the number and the complexity of their components. While the conservative representation (called 'identity' in the remainder of this paper) combines a redundancy free representation with a linear complexity, structural decompositions shift the whole amount of complexity to a linear number of components each incorporating a constant complexity.

According to the results of our experimental analysis in [KHS 91], both approaches, caused by their unbalanced object representations, obviously reveal weaknesses depending on the query selectivity. In the case of high-selectivity queries, the high structural complexity determines the performance of the identity which can frequently be fully answered based on the container-object for low-selective spatial query conditions. The clustering mechanism of SAMs provides a good filtering of redundancy for structural decompositions in the case of high-selectivity queries, whereas low-selectivity queries degenerate due to the high amount of redundancy (see [KHS 91]). From this observation, we conclude the following statement:

A balanced ratio between complexity and redundancy is essential for efficient spatial query processing.

This ratio is determined using an interaction of both steps of query processing. The handling of an adapted degree of redundancy within the filter step is justified by a gain in efficiency within the refinement step. The main criteria are given by an increased approximation quality, i.e. a decreased number of false hits, and particularly by simpler and more efficient geometric operations on local parts of the object.

These considerations lead to the development of newly designed structural decomposition methods that show a good performance for both selective queries because of a good approximation and simple geometric operations, as well as for non-selective queries because of a strongly reduced amount of redundancy. The main issue is the combination of the following opposing criteria:

- geometric and topologic locality in object representation
- low amount of redundancy

In order to fulfill both demands, the goals of structural object decompositions have to be newly defined. Contrary to the traditional object decomposition methods, there are strongly increased degrees of freedom compared to the strict definition of the shape, the complexity, and the topology of the components. This yields a lot of criteria for designing efficient decomposition algorithms which could not be considered by the rigid definition of the component properties in traditional object decompositions.

4 Object-orientied partitioning methods

The central idea of the class of structural decompositions of spatial objects considered below is the complete and disjoint partitioning of a simple polygon into a set of spatial components each of them representing a locally restricted part of the original object with a given complexity. A characterization of this amount of complexity is given by a suitable *constraint of simplicity* (see figure 3). Up to now, for object decompositions this constraint was limited to the shape of a very simple spatial object or was given by the topological convexity property and thus, extremely restrictive. Contrarily, for the identity it is given by the shape of an arbitrary simple polygon and thus, represents no restriction at all.

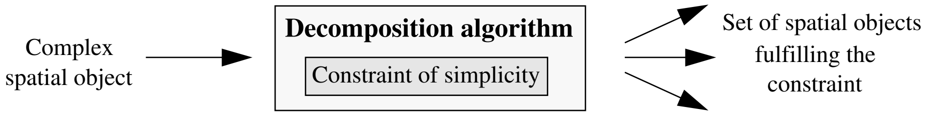


Fig. 3. General scheme of the decomposition algorithm

The correspondence between the number of components, i.e. the redundancy, and their complexity can be expressed by the following formula:

$$Red = \frac{c_{Obj} + a \cdot Red}{c_{comp}} \quad \text{which implies:} \quad Red = \frac{c_{Obj}}{c_{comp} - a}$$

where c_{Obj} and c_{comp} represent the complexity, i.e. the number of vertices, of the original object and of one decomposition component, respectively, and a denotes the structural overhead induced by one step of partitioning. Typical values are $a = 2$ if the partitioning process is restricted to the vertices of the original object, i.e. two original vertices are combined, and $a = 3$ in the case of Steinerpoints.

The constraint of simplicity directly affects both the complexity of the components and the redundancy and therefore, has to be carefully selected. One essential criterion neglected in previous approaches is given by application specific constraints to spatial objects. Such criteria like the existence of holes or the restriction of spatial objects to a maximum number of vertices depending on the application system can be expressed by a corresponding constraint to the partitioning process.

By the instantiation of this constraint, an approximate goal of the decomposition process is established. However, no algorithmic designs are specified. Thus, there remains a lot of flexibility for the decomposition process to be filled by the implementation. In the following, we will first investigate these algorithmic flexibilities before examining the constraint of simplicity with respect to the criteria of redundancy and complexity.

Algorithmic degrees of freedom

The constraint of simplicity defines what type of components have to be generated while the algorithmic instantiation describes *how* to derive them. Thus, the instantiation of the decomposition method is bound to some optimization criteria determining the performance of spatial query processing.

- axis parallel or free oriented partitioning

  Any partitioning with lines parallel to the axes of the data space typically induces Steiner-points. Therefore, as a result, there is a higher amount of structural overhead. Contrarily, free oriented methods with a more flexible selection of partitioning lines may join vertices and thus avoid those drawbacks. However, the container approximation is strongly affected by the orientation of the partitioning lines. Vertical and horizontal lines provide a good bounding box approximation, whereas partitioning lines with angles in $\{k \cdot \pi/4, \ k \in \{1, 3, 5, 7\}\}$ maximize dead space. Therefore, even for orientation free approaches a 'nearly' axis parallel partitioning is advisable.

- geometric and topologic locality

  The central idea of object decompositions is a local and selective processing of complex spatial objects. Therefore, geometric locality is the crucial property of structural decompositions. With regard to applications such as the organization of versioned objects or geographic and topologic maps with neighboring areas expressed by common polylines, the preservation of a topologic locality is essential as well.

- regularity of the components

  A basic requirement to the representation of spatial objects in real applications is a guarantee for stability and the exclusion of degenerations. This demand is important with respect to spatial, redundancy, and structural aspects. It affects the partitioning process by strict conditions concerning the extension, the number and the complexity of the components. These criteria strongly depend on the object topology and partially describe opposing requirements. Therefore, for designing the partitioning process an integrated consideration of those requirements is necessary. Between opposing parameters, a weighting with a sufficient variability has to be determined for a global consistency and applicability.

  An example for a spatial criterion of regularity can be found in the Delauney-triangulation of simple polygons. It is the '*Lawson-criterion*' ([Law 72]) incorporating a regularity condition on the angles of the triangles. Contrarily, the trapezoid decomposition ([AA 83]) is not bound to any regularity in the shape of the trapezoids and thus, may create degenerated 'line-shaped' components. The decomposition methods investigated in [KHS 91] do not fulfill all regularity criteria outlined above as shown in table 1.

- small number of components (minimum decompositions)

  Beside those degeneration aspects there is the demand for a manageable number of components regarding certain complexity constraints. A minimum number of components generally imposes very expensive algorithms typically with NP-complete or NP-hard time complexity ([Kei 83], [KS 85]). Particularly, for a dynamic decomposition method meeting some regularity conditions, it is inevitable to relax the strict minimality criterion. Instead an algorithmically more simple, suboptimal method with more flexibilities has to be designed.

	shape	redundancy	complexity
trapezoid decomposition	-	+	+
convex decomposition	-	+	-
identity	-	+	-

Table 1: Regularity of spatial object representations

Having surveyed the general aspects concerning the partitioning process of structural decompositions, we next investigate an appropriate degree for the number of components and their structural complexity. As there is no general answer for an arbitrary type of spatial objects and applications, we develop a set of criteria serving as a basis to an adequate redundancy instantiation for a given application. However, topologic aspects remain unconsidered in this context.

The amount of redundancy

Based on the variety and heterogeneity of geometric applications as well as on the structural variance of spatial objects, applicational and object-oriented aspects have to be considered defining the measure of redundancy in object representation. A global and flat rated instantiation is not possible. It rather depends on a set of factors influenced by the conditions of the underlying application. In the following, the important factors determining redundancy are explained.

- *object-oriented aspects*
  - *structural complexity*
    The complexity of a spatial object strongly influences the number of components. Obviously, an object with say 10 vertices, needs no partitioning at all, whereas a complex object consisting of 50,000 vertices should be partitioned into several hundreds of components.
  - *geometric shape and geometric complexity*
    For a simply shaped object, e.g. a convex polygon, less and simpler components are generated than for a strongly meandering object (see figure 5). This is due to the fact that the spatial locality of the components depends on the geometric complexity of the object
  - *object area relative to the data space*
    The strong variance of the area of spatial objects exerts a bad influence on the clustering and therefore, on the selectivity properties of SAMs. Thus, one goal is to homogenize the area and extension of spatial objects. Large objects are partitioned more distinctly than small objects.
- query oriented aspects
  Depending on the selectivity of application oriented spatial queries, a stronger preference for the number of components or for their complexity may be adequate. In the case of high-selectivity queries, e.g. point queries, a higher amount of redundancy is useful and vice versa (see the results in section 6 and [KHS 91]).
- hardware aspects
  The given hardware environment determines the ratio of the time for one secondary storage access to the time for one typical CPU-operation. Because redundancy is I/O-intensive, whereas complexity is CPU-intensive, the best possible balance between redundancy and complexity has to be tuned for a given hardware environment.

- application specific aspects
  Requirements to object complexity may be imposed by the application system. For example, a polygonal object has to fit in a single page on secondary storage and therefore, only a limited number of vertices are allowed.
- general aspects
  - *data page as clustering unit*
    The term of a data page represents the atomic unit of transfer between main memory and secondary storage. This observation leads to a component complexity approximately meeting the space of a data page. In this case, the redundancy corresponds to the number of data pages. As the structural overhead of the partitioning is low, the number of actual data pages is most likely not increased with respect to the linear sequence of pages, necessary for storing the complex object. However, a spatially organized and accessible set of pages takes the place of the sequentially organized traditional representation.
  - *dynamic balancing between redundancy and complexity*
    A 'fair' balance between redundancy and complexity is provided by the *root criterion*:

    > A structural decomposition method fulfills the *root criterion*, if the complexity of the components is in the range of $[c \cdot \lceil \sqrt{n} \rceil, 2 \cdot c \cdot \lceil \sqrt{n} \rceil + 1]$, where c is a real constant and n denotes the complexity of the original object. A typical measure of the constant c is given by $c \approx 1$.
    >
    > As an example, for a polygon with 100 (50,000) vertices the complexity of the 5-10 (112-223) components is in the range [10, 21] ([224, 449]).

Fig. 4. Examples of object-oriented decompositions

These aspects describe different starting-points for an adaptive selection of the number of components. We have shown that the number of components depends on various application specific aspects. Our goal within this paper is not an evaluation of all those aspects and their mutual affects but to present a number of different decomposition methods under the criterion of a constrained redundancy.

The next section will introduce several structural decomposition methods defining a simultaneous restriction of complexity and redundancy. Two basic methods will be introduced: composition methods are based on an object decomposition into very simple spatial components suitably merging those components to more complex components. However, fragmentation methods directly construct the components by a sophisticated definition of partitioning lines.

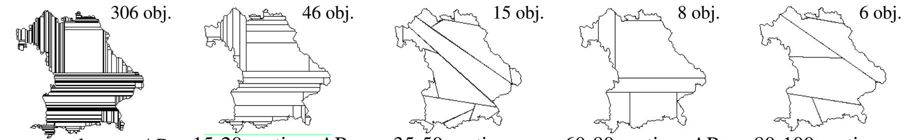
5 Object-oriented decomposition methods

Composition methods

Within this section, we generalize the decomposition methods into convex polygons, trapezoids, and heterogeneous components as described in [KHS 91] by merging a number of components to more complex component objects resulting in a decreased amount of redundancy.

Convex composition method

Convex decompositions are based on the treatment of notches, i.e. vertices violating the convexity property. Recursively introducing partitioning lines, in every step of the 'naive' partitioning process, components with a smaller number of notches are generated (see [CD 85]). This process finishes up with convex components containing no notches.



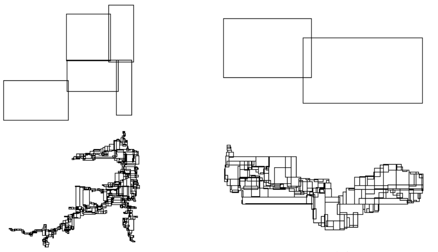
convex polygons AP 15-20 vertices AP 35-50 vertices 60-80 vertices AP 80-100 vertices

Fig. 5. Examples of the convex composition method

The generalized approach imposes a new criterion on the break off of the partitioning process. Components are partitioned only if their complexity exceeds a predefined constant even if they contain notches. On the logical level, this constraint partitioning process corresponds to a merge of neighboring convex components. Figure 5 depicts the result of the partitioning process with a constraint number of vertices for the axis-parallel (AP) and the orientation free convex composition. The axis-parallel method is one of the competitors in the performance comparison given in section 6.

Trapezoid composition

The main drawbacks of the decomposition into trapezoids originate in the linear number of components and in their degenerated shape inducing a lacking spatial locality. However, the demand for axis parallelism in one dimension causes a partial order on the trapezoids characterizing a neighborhood relation on the components. Thus, a merging process on the set of trapezoids is simple and leads to still axis parallel but less degenerated components. This constrained merging process, however, hinders the definition of locality based components.



'Lake Volta' 'Gambia'

Fig. 6. First and second directory-level of an R*-tree for heterogeneously decomposed objects

Heterogeneous composition

The heterogeneous decomposition (see [Schi 93], [KHS 91]) is based on a separation of the contour and the interior of a spatial object. The principal building block of a heterogeneous composition is given by a merge of components describing a locality based part of the object. Joining neighboring edges leads to a topologic locality, whereas their merge with suitable interior components preserves a geometric locality.

One possible instantiation of this merge process is given by the data pages of a clustering SAM. A merge of the components clustered, for example, in one data page of an R*-tree, while checking some connectivity and integrity constraints yields a good spatial locality of the components (see figure 6).

Fragmentation methods

An alternative approach of object-oriented spatial decompositions of simple polygons is given by the class of fragmentation methods. Fragmentation methods directly decompose a spatial object with respect to a given simplicity constraint. The paradigm of a binary space partition is applied. As there is no preceding decomposition into simple components, a much more flexible partitioning process is available with respect to topologic, geometric, or structural object properties.

A wide range of different partitioning methods is available, all based on the fragmentation approach. We do not want to present all those variants, but describe one selected fragmentation method, which is efficient for a number of reasons.

Within the following, $p_i = (p_{i_x}, p_{i_y})$, $1 \le i \le m$, denotes the vertices of a simple polygon $P = (p_1, p_2, \ldots, p_m)$, where the edges of the polygon are given by pairs of consecutive vertices (p_1, p_2), (p_2, p_3), …, (p_{m-1}, p_m), (p_m, p_1). Let $\langle p_i, p_j \rangle$ denote the line connecting the points p_i and p_j. Then $dist_{ind}(i, j) := min\{(i - j + n) \text{ MOD } n, (j - i + n) \text{ MOD } n\}$ denotes the *index-distance* of p_i and p_j. The term of an index-distance describes the number of edges between two given vertices of P.

In the partitioning process, we connect vertices which have a minimum distance from each other under some constraints. Using the Euclidean distance, the following minimization function has to be solved for a polygon $P = (p_1, p_2, \ldots, p_m)$ and a predefined constant d_{min}:

$$\min_{1 \le i \le m} \Big(\min_{i < j \le m,\, dist_{ind}(i,j) \ge d_{min},\, \langle p_i, p_j \rangle \subset P} \Big(\sqrt{(p_{i_x} - p_{j_x})^2 + (p_{i_y} - p_{j_y})^2} \Big) \Big).$$

Obviously, the primary effect of this procedure is a minimization of the unnatural contour and therefore, of the contour of the components. However, there are a number of advantageous consequences to the spatial and topological properties of the components.

- spatial and topologic locality of the components
- no Steinerpoints, i.e. no new vertices induced by the components
- small contact of the components defined by single edges and not by polylines
- object-oriented and not component-oriented approach

The realization of this decomposition method necessitates some additional examinations as described in the following:

- using the Euclidean metric for determining the minimum distances, we can make no statement with respect to the orientation of the partitioning lines. We expect uniformly distributed orientations in the range of $[-\pi/2, \pi/2)$. For approximation and regularity reasons, however, axis parallel lines, i.e. lines close to the orientation values of $\{-\pi/2, 0, \pi/2\}$, are desirable. Therefore, we apply the Manhattan metric instead, penalizing a non-axis-parallel orientation: $dist_{sum}(p_i, p_j) = |p_{i_x} - p_{j_x}| + |p_{i_y} - p_{j_y}|$.
- as introduced above, we use the root-criterion determining the complexity of the components. Therefore, the complexity as well as the number of the components is $O(\sqrt{n})$. To avoid a degeneration in the case of objects with a small number of vertices, we define a lower bound of 10 for the number of vertices of one component, therefore: $Min_{Compl}(n) = max\{\lceil \sqrt{n} \rceil \cdot c, 10\}$. The value of c is defined as $c = 2.0$.
- a major problem of the partitioning process is the preservation of regular components. Thus, partitioning lines must be completely included within the object and may not have an intersection with the object border. Principally, any computation of minimum distances has to check for some visibility constraints resulting in a high computational effort. To re-

duce this effort, we choose a ‘suboptimal’ way to proceed. For this purpose, we define two strict visibility conditions (see figure 7).

Two Points P_1 and P_2 of a simple polygon P are called *locally visible*, if:

$$\exists\, \varepsilon_1, \varepsilon_2 > 0 \text{ with } \left(\{P_1 + \delta \cdot (P_2 - P_1),\ 0 \le \delta \le \varepsilon_1\} \subset P \right) \wedge \left(\{P_2 + \delta \cdot (P_1 - P_2),\ 0 \le \delta \le \varepsilon_2\} \subset P \right)$$

Particularly, two interior points of P are locally visible by definition.

P_1 and P_2 are called *globally visible* or *visible*, if: $\{P_1 + \delta \cdot (P_2 - P_1),\ 0 \le \delta \le 1\} \subset P$.

While checking for the global visibility constraint takes linear time in the number of vertices, the local visibility is evaluated in constant time. Therefore, within the minimization phase for every vertice we only check the local visibility. Instead, the process of searching pairs of vertices of minimum distance checks the global visibility to maintain correctness. This approach, not guaranteeing the global minimum distance of two points joined by a partitioning line, reduces the time complexity by a linear factor.

Local visibility constraint Global visibility constraint

Fig. 7. Global and local visibility condition

- releasing the strict structural constraint

  Based on the arbitrarily but fixed defined constant of the minimum component complexity, a reduction of the natural locality may be the consequence. In order to prevent this, at least partially, we define a range of fussiness $\vartheta(n)$ and a fixed upper bound Δ. Now, we introduce a relaxation of the strict minimum complexity constraint:

  If, for two points P_i and P_j, the following condition holds:

$$dist_{sum}(P_i, P_j) \le (1 - \Delta) \cdot \left(\min_{1 \le k \le n,\ dist_{ind}(i, k) \ge Min_{Kompl}(n)} (dist_{sum}(P_i, P_k)) \right) \text{ with },$$

$$Min_{Kompl}(n) > dist_{ind}(i, k) \ge Min_{Kompl}(n) - \vartheta(n)$$

  then a partition line is introduced between P_i and P_j neglecting the low component complexity. A careful selection of the values of $\vartheta(n)$ and Δ is necessary. If $\vartheta(n)$ is too large or Δ is too small, the above defined criteria become annulled. In our instantiation, we define $\vartheta(n) = 0.05 \cdot n$ and $\Delta = 0.25$.

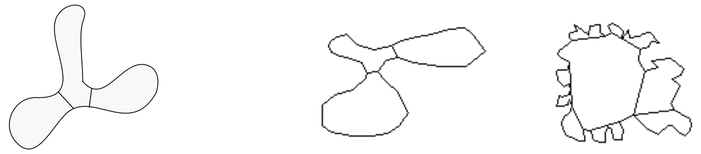
Properties of the fragmentation method

The effect of the algorithm strongly depends on the shape of the object justifying the name of an object-oriented method. We distinguish two basic treatments (see figure 8):

- the join of two dents
- the cut of bulges

If an object includes distinct dents, the minimum distance of a pair of points occurs between those dents under the predefined constraints. This treatment results in a spatial and topologic localized and more regular shape of the components, both underlying a further recursive partitioning. If no such dents exist, i.e. if there is an obvious kernel (see figure 8), parts of the object ‘sticking out’ in the form of bulges will be cut. Successively, the object is reduced to a topologically simple kernel and a set of components along the object border. As every step of the partitioning treats one bulge, two unsimilar components are generated. One represents the bulge yet fulfilling the complexity property, whereas the other describes the rest of the object which is further partitioned.

This decomposition process for spatial objects is particularly determined by the object shape and topology. The two basic treatments guarantee for a high locality with respect to spatial and topological aspects. In particular, the difficult problem of defining an adequate kernel is solved in a direct and pragmatic way. The existence of a distinct kernel preserves spatial locality and supports answering spatial queries by only considering the kernel.


desirable partition

treatment of dents and bulges of the objects

Fig. 8. Instantiation of the object-oriented decomposition method

In the following section, we will evaluate this method of decomposing a complex spatial object with respect to the filter and the refinement step of spatial query processing comparing it to presently known methods.

6 Secondary memory organization and spatial query processing

As outlined in section 2, spatial query processing on complex spatial objects is performed in a two step approach: based on an approximation oriented object representation, the filter step reduces the set of objects to a set of candidates most likely fulfilling the query condition. The refinement step exactly evaluates the query condition. Within this section, we will examine the decomposition methods introduced above, i.e. the convex composition (comp) and the fragmentation method (frag), with respect to this two step approach of spatial query processing. In the performance comparison, we use the identity representation (ide) as well as the convex decomposition (conv) introduced in [KHS 91] as a measuring stick.

This comparison utilizes three different datafiles, two of them corresponding to real geographic objects that do not overlap (county regions of 'Europe' (see figure 9)) and 'Baden-Württemberg' (BW), and the third representing a set of synthetic objects with some 85 vertices and two holes on the average ('sph_85'). 'Europe' and 'BW' mainly differ with respect to the number of objects as well as the average object complexity influencing the amount of decomposition redundancy. The average object complexity of the strongly homogeneous 'sph_85'-file meets the object complexity of 'Europe', while the number of vertices in 'BW' is about six times higher.

The following tables depict the major parameters of the particular representations. They include the number of components and their average and total complexity, the storage needed for the representation and spatial organisation of the decomposition components compared to a linear organization of the objects, each within one or more 2 KByte-secondary memory pages, and the bounding box approximation characterizing the efficiency of the filter step expressed by the number of false hits (see also [BKS 93]).

The convex decomposition turns out to generate a extremely high number of components (about $1/2$ the total number of vertices) and a low and constant component complexity (4.5-5 vertices on the average). The overall complexity factor of about 2.5 is a measure for the strong structural overhead caused by a multiple storage of identical vertices within different components and by Steinerpoints. However, this overhead is small for our new decomposition methods. The convex composition method shows a constant component complexity. Due to the merge of convex components, their complexity is much higher and thus, their number is much lower compared to the convex decomposition. The component complexity of the object-oriented decomposition by definition strongly depends on the complexity of the original objects.

	number of components			relative storage space		
	Europe	BW	sph_85	Europe	BW	sph_85
ide	471	1,298	1,000	0.71	1.00	1.00
conv	22,296	367,815	51,183	2.56	4.41	3.74
comp	2,091	29,810	3,093	0.96	1.68	1.36
frag	1,410	12,602	3,573	0.92	1.52	1.31

Table 2: Decomposition parameters (1)

	medium comp. complexity			overall complexity			bounding box approx.		
	Europe	BW	sph_85	Europe	BW	sph_85	Europe	BW	sph_85
ide	94.9	572.5	83.3	1.00	1.00	1.00	2.15	2.00	1.72
conv	4.94	5.01	4.57	2.46	2.48	2.81	1.31	1.05	1.16
comp	23.7	27.8	29.0	1.11	1.12	1.08	1.92	1.30	1.74
frag	33.1	60.8	24.8	1.04	1.03	1.06	1.93	1.54	1.84

Table 3: Decomposition parameters (2)

Now, our goal is to perform spatial queries based on these representation schemes measuring the performance of the particular methods. For that purpose the objects and the components are organized in an R^*-tree, an efficient SAM for rectangles using the concept of overlapping regions ([BKSS 90]). We define a ‘one-path’ page buffer supporting query processing.

The main criteria determining query efficiency are (see also [KHS 91]):

- disc accesses and main memory page search operations, i.e. ‘*point in rectangle*’ operations and rectangle intersections, performed by the SAM (filter step) and
- exact spatial operations performed on the original objects or their decomposition components (refinement step)

In the case of a decomposed object representation an additional filter of redundancy has to be incorporated within the refinement step to avoid redundant spatial operations. Our instantiation is a temporary main memory binary search tree structure, implemented as an AVL tree.

The tables below present the performance results of point queries and window queries depending on the size of the query window. The percentage values describe the extension of the query window in each dimension of the data space, i.e. a 10%-window query corresponds to a window covering 1% of the data space. As the performance of the access method is independent of the object distribution, the query windows are evenly distributed in the space covered by the data objects. Thus, only successful queries are performed for a good simulation of application specific requirements. The performance values are divided into filter and refinement performance, parametrized as follows:

$$filter = number\ of\ accesses \cdot t_{access} + number\ of\ comparisons \cdot t_{comp}$$

$$ref = number\ of\ duplicates \cdot t_{tree\_search} + spatial\ operations \cdot t_{spatial\_operation}$$

Fig. 9. Partitionings of the 'Europe' file

The time parameters are averaged over the corresponding values measured in numerous experiments on HP 720 RISC workstations under GP MODULA-2. We instantiate the values $t_{access} = 0.12ms$, $t_{comp} = 0.002ms$, and $t_{tree\_search} = 0.015ms$ averaged. $t_{spatial\_operation}$ strongly depends on the type of the spatial operation. In the case of a query region which is large compared to the average object area, the refinement operation is performed by a small number of comparisons, whereas point queries require for the evaluation of the complete shape of the spatial objects. Therefore, this time parameter cannot be globally defined. However, for point queries and small window queries we have approximately: $t_{spatial\_operation} = n \cdot 0.01ms$, where n denotes the object complexity.

The following table 4 depicts the average time required for the evaluation of one single query. Due to space limitation, the results for thr 1%-window queries are not presented in the table. The values are given in microseconds. For a clearer evaluation they are divided into the time for the filter step and the refinement step. In the table, we have shadowed the best performing method for each type of query. A summation of both values resulting in the total time needed for the processing of one spatial query is represented in figure 10.

	point queries		0.5 % - window qu.		2 % - window qu.		5 % - window qu.		10 % - window qu.		25 % - window qu.	
	filter	ref.	filter	ref.	filter	ref.	filter	ref.	filter	ref.	filter	ref.
	Europe - file (471 objects / 44,716 vertices)											
ide	0.79	3.05	0.83	2.53	0.97	1.06	1.32	0.64	2.08	0.35	5.6	0.1
conv	0.69	0.05	0.83	0.13	1.33	0.52	2.67	1.84	5.78	5.31	21.2	23.2
comp	0.48	0.29	0.55	0.25	0.75	0.16	1.26	0.21	2.35	0.48	7.6	1.8
frag	0.50	0.47	0.55	0.38	0.73	0.17	1.18	0.15	2.20	0.28	7.1	1.0
	BW - file (1,297 objects / 743,137 vertices)											
ide	6.98	11.2	7.55	14.0	9.67	13.8	14.8	13.0	26.0	8.31	75.4	2.2
conv	1.01	0.05	1.76	0.60	5.91	4.54	21.7	22.5	65.6	75.0	302	370
comp	0.81	0.32	1.14	0.76	2.73	1.85	8.26	2.21	23.7	6.50	101	29.5
frag	0.78	1.32	1.04	1.47	2.37	1.63	7.13	1.78	20.0	4.06	88.1	11.7
	sph_85 - file (1,000 objects / 83,291 vertices)											
ide	2.70	14.7	2.88	15.1	3.46	17.6	4.74	22.3	7.3	17.8	17.7	11.3
conv	1.78	0.55	2.07	0.78	3.09	1.62	5.86	2.84	12.6	9.4	47.3	49.5
comp	2.12	5.17	2.31	4.84	2.91	3.75	4.32	3.63	7.3	2.2	20.2	2.7
frag	2.12	4.69	2.30	4.07	2.88	3.71	4.32	3.15	7.1	2.1	19.6	3.2

Table 4: Average time per query (in ms)

These results reveal strong differences in query performance. As expected, for each of the test files the decomposition based representations perform good for high selectivity queries, whereas for queries with low-selectivity, i.e. for extremely large window queries, their performance gets poorer. In principle, for very large queries a spatial clustering is not supportive anymore. Then, a sequential scan over the objects may turn out to be the best choice. Typical application-oriented queries extract small portions of the objects and therefore, are restricted to small query windows. In this case, any of the object decomposition methods clearly outperforms the identity representation up to an order of magnitude. The identity representation is superior to the decomposition methods only for very large queries when the performance of the decomposition methods degenerates.

The break-even point of the performance curves depends on the characteristics of the underlying objects. The identity representation performs considerably worse than the decomposition methods for a wide range of the window sizes, if the overlap of the objects is strong and the average object area is high (see file ‘sph_85’). However, for small objects and for large window queries, the average computational geometry cost becomes cheap in comparison to the cost for handling the high amount of redundancy. Thus, window queries exceeding a break-even point which is in the range of [0.1, 0.35] of the size of the data space in each dimension, are performed most efficiently by the identity representation. However, in most applications these very large window queries rarely occur.

Caused by the reduced amount of redundancy and still a small complexity of the components, the two new methods outperform the convex decomposition already for small window queries and improve their superiority with increasing query size. Only for point queries, where the whole amount of redundancy is neutralized by the clustering effect of the SAM, due to its bounding box approximation quality and its simple refinement operations, the convex decomposition displays the best performance. However, in a wide range of small and medium sized window queries decomposition methods based on a balance between complexity and redundancy take advantage of much less redundancy, and, nevertheless, a reasonably small object complexity (see figure 10). To be more precise, the following table 5 normalizes the total query time for each method and each type of query with respect to the most efficient method. The table clearly shows that for small and medium sized window queries (window sizes in the range of 0.5% to 10% of the data space in each dimension) our new methods are superior to the traditional methods by a factor which varies between 1.24 and 11.34. Possibly even more important, our new methods are very robust in performance over the whole range of queries. For the BW-file there is no query type where their performance is worse than the best method by a factor of more than 2. Since it has been demonstrated that the performance of the R*-tree is basicaly independent of the object parameters, the above results reflect the performance of the decomposition and not the performance of the access method. .

	point queries	0.5 % - window	1 % - window	2 % - window	5 % - window	10 % - window	25 % - window
	BW - file						
ide	17.15	11.34	8.82	5.87	3.12	1.43	1.00
conv	1.00	1.24	1.68	2.61	4.96	5.84	8.66
comp	1.07	1.00	1.00	1.15	1.18	1.26	1.68
frag	1.98	1.32	1.13	1.00	1.00	1.00	1.29

Table 5: Performance of spatial queries normalized with respect to the most efficient method

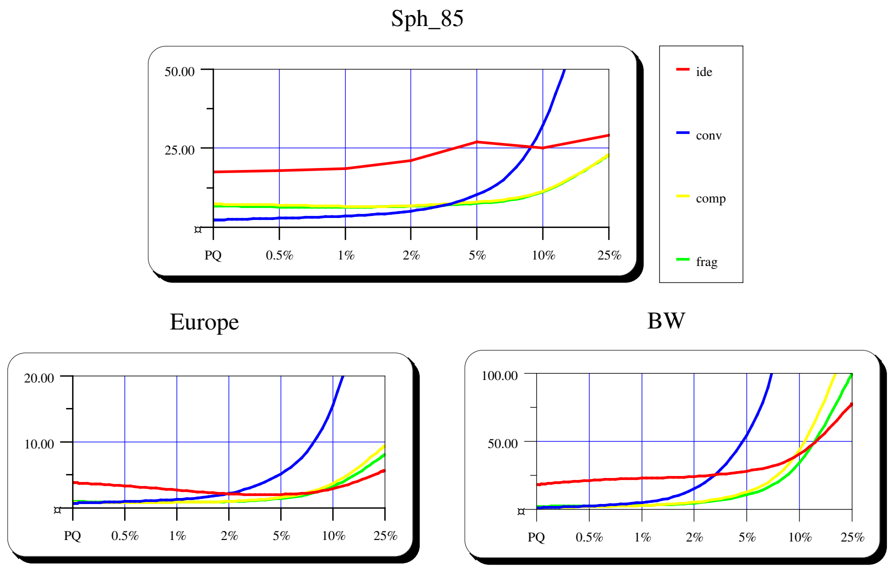


Fig. 10. Total query time (in ms) depending on the window size of spatial queries

Due to the strict axis-parallel partitioning process of the convex composition method, its approximation is superior to the fragmentation method. This property is particularly influencing the performance of small sized window queries. For larger query windows the spatial locality of the fragmentation method inducing a better clustering results in a better performance. However, the performance of the two methods is very similar which is due to the similar complexity and redundancy.

These trends come out even clearer for region queries where the query is specified by a polygonally shaped object. Region queries are typically performed by the sequence of a window query using the query window formed by the bounding box of the query region followed by refinement operations based on the exact shape of the query region. Particularly for the new decomposition methods inducing a low amount of redundancy, a decomposition of the query region combined with an iterated query processing of simple and local query regions becomes a promising approach.

Generally, the refinement operation of the identity representation strongly suffers from an increase of the query region complexity. Typically, the algorithmic complexity is $O(r \cdot n)$, where r and n denote the complexity of the query region and the complexity of the refined object, respectively. As the decomposed spatial objects are considerably less complex, their refinement complexity is much lower, particularly for complex query regions.

A map overlay of two object files (e.g. [KBS 91]), an essential operation for geographic applications, results in a set of object pairs, one of each file, fulfilling a predefined join condition. Again, a two step approach is applied by first computing a set of candidate pairs based on their bounding box approximation, followed by the evaluation of the overlay condition on the exact object shapes. The identity representation as a first approach reveals two strong drawbacks with respect to its performance: as a result of the bad approximation, a high number of object pairs has to be considered and their evaluation takes the same quadratic effort as outlined above for region queries. Contrarily, the convex decomposition exhibits both an excellent approximation and very simple refinement operations. However, as the redundancy of decomposition methods takes quadratic effort, it has to be carefully controlled. Our new decomposition schemes provide simpler spatial algorithms combined with a still small measure of redundancy. Therefore, first results confirm that their performance improvement for map overlay operations is much higher than for simple spatial queries. Final results are left to a future performance comparison of our new decomposition methods for the map overlay operation.

7 Conclusions

Spatial query processing of complex spatial objects is typically performed by a two step approach. The first step, based on a coarse spatial approximation of the objects embodies a simple spatial filter yielding candidates, whereas in a second step the exact spatial properties of the candidates are evaluated. Up to now, the so-called identity representation is commonly used based on an integrated evaluation of the complete object shape organized by a bounding box approximation of the object. First approaches gaining in efficiency for small sized query regions by decomposing the complex object into simple components are given in [KHS 91]. However, the high number of components involves some inherent disadvantages: a high amount of storage and redundancy, an expensive inversion of the partitioning process, and, most essential, a strong degeneration of the query performance for medium and large sized query regions. Therefore, we have defined a group of decomposition methods, taking into account both a small number of components and a low component complexity. The implementation and evaluation of two selected methods show a clear gain in efficiency in comparison to the identity representation and the convex decomposition method, the winner in [KHS 91]. For small and medium sized window queries (window sizes in the range of 0.5% to 10% of the data space in each dimension) this gain in performance is by a factor in the range of 1.24 to 11.34. Possibly even more important, our new methods are very robust in performance over the whole range of window queries. Furthermore, for spatial queries more complex than window queries, this gain will be even higher. Thus, for all applications where extremely large queries do not occur, in other words for most practical applications, our new decomposition methods turn out to be a good choice.

References

[AA 83] Ta. Asano and Te. Asano, *Minimum partition of polygonal regions into trapezoids,* Proc. 24th IEEE Annual Symp. on Foundations of Computer Science, 1983, 233-241.

[BKS 93] T. Brinkhoff, H.-P. Kriegel, and R. Schneider, *Comparison of approximations of complex objects used for approximation-based query processing in spatial databases*, in Proc. 9th Int. Conf. on Data Engineering, Vienna, Austria, 1993.

[BKSS 90] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, *The R*-tree: An efficient and robust access method for points and rectangles*, Proceedings ACM SIGMOD Int. Conf. on Management of Data, Atlantic City, NJ, 1990, 322-331.

[Bra 92] A. Braun, *A graphic-oriented tool for analyzing spatial objects: 'its design and application to real world data'*, Master thesis, Institute for Computer Science, University of Munich, Germany, 1992, (in German).

[CD 85] B. Chazelle and D.P. Dobkin, *Optimal convex decompositions*, in *computational geometry*, Proceedings Comp. Geometry, Elsevier Science, Netherland, 1985, 63-134.

[Fra 91] A.U. Frank, *Properties of geographic data*, Proceedings 2nd Symp. on Large Spatial Databases, SSD'91, ETH Zurich, 1991, 225-234, in: Lecture Notes in Computer Science, Vol. 525, Springer, 1991.

[Kei 83] J.M. Keil, *Decomposing polygons into simpler components*, Ph.D. thesis, Department of Computer Science, University of Toronto, 1983.

[KS 85] J.M. Keil and J.R. Sack, *Minimum decomposition of polygonal objects*, in *Computational Geometry*, G.T. Toissant (Ed.), Amsterdam, Netherland, 1985, 197-216.

[KBS 91] H.-P. Kriegel, T. Brinkhoff, and R. Schneider, *An efficient map overlay algorithm based on spatial access methods and Computational Geometry*, Int. Workshop on Database Management Systems for Geographical Applications, Capri, Italy, 1991, in: Geographic Database Management Systems, Springer, 1992, 194-211.

[KHS 91] H.-P. Kriegel, H. Horn, and M. Schiwietz, *The performance of object decomposition techniques for spatial query processing*, Proceedings 2nd Symp. on Large Spatial Databases, SSD'91, Zurich, 1991, 257-276.

[Law 72] C.L. Lawson, *Generation of a triangular grid with application to contour plotting*, CIT Jet Propulsion Laboratory, Technical Memorandum 299, Pasadena, CA, 1972.

[NHS 84] J. Nievergelt, H. Hinterberger, and K.C. Sevik, *The Grid File: an adaptable, symmetric multikey file structure*, ACM Transactions on Database Systems, Vol. 9, 1, 1984,38-71.

[Ore 89] J. Orenstein, *Redundancy in spatial databases*, Proceedings 1st International Symposium on Large Spatial Databases, SSD'89, Santa Barbara, CA, 1989.

[OM 86] J. Orenstein and F.A. Manola, *Spatial data modeling and query processing in PROBE*, Technical Report CCA-86-05, Xerox Advanced Inform. Technology Devision, 1986.

[PS 88] F.P. Preparata and M.I. Shamos, *Computational Geometry*, Springer, New York, 1988.

[Sam 90] H. Samet, *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, 1990.

[Schi 93] M. Schiwietz, *Storage and Query Processing of Complex Spatial Objects*, Ph.D. thesis, (in German), Inst. for Computer Science, University of Munich, Germany, 1993.

[Schn 92] R. Schneider, *A Storage and Access Structure of Spatial Database Systems*, Ph.D. thesis, (in German), Institute for Computer Science, University of Munich, Germany, 1992.

[SK 91] R. Schneider and H.-P. Kriegel, *The TR***-tree: a new representation of polygonal objects supporting spatial queries and operations*, Proceedings 7th Workshop on Computational Geometry, Bern, Switzerland, 1991, in: Lecture Notes in Computer Science, Vol. 553, Springer, 1991, 249-264.

[See 89] B. Seeger, *Design and implementation of multidimensional access methods*, Ph.D. thesis, (in German), Depart. of Computer Science, University of Bremen, Germany, 1989.