Vladimír Mařík Jiří Lažanský
Roland R. Wagner (Eds.)

Database and Expert Systems Applications

4th International Conference, DEXA '93
Prague, Czech Republic, September 6-8, 1993
Proceedings

DEXA 93

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Table of Contents

Invited Talk

Topic 1: Data Models

Topic 2: Distributed Databases

Topic 3: Advanced Database Aspects

Topic 4: Database Optimization and Performance Evaluation

Topic 5: Spatial and Geographical Databases

Topic 6: Expert Systems and Knowledge Engineering

Topic 7: Legal Systems

Topic 8: Other Database and AI Applications

Topic 9: Software Engineering

Topic 10: Hypertext/Hypermedia and User Interfaces

Query Processing of Geometric Objects with Free Form Boundaries in Spatial Databases

Hans-Peter Kriegel, Stephan Heep, Andreas Fahldiek, Norbert Mysliwitz

Institute for Computer Science, University of Munich
Leopoldstr. 11B, D-8000 Munich 40

Abstract: The increasing demand for the use of database systems as an integrating factor in CAD/CAM applications has necessitated the development of database systems with appropriate modelling and retrieval capabilities. One essential problem is the treatment of geometric data which has led to the development of spatial databases. Unfortunately, most proposals only deal with simple geometric objects like multidimensional points and rectangles. On the other hand, there has been a rapid development in the field of representing geometric objects with free form curves or surfaces, initiated by engineering applications such as mechanical engineering, aviation or astronautics. Therefore, we propose a concept for the realization of spatial retrieval operations on geometric objects with free form boundaries, such as B-spline or Bézier curves, which can easily be integrated in a database management system. The key concept is the encapsulation of geometric operations in a so-called query processor. First, this enables the definition of an interface allowing the integration into the data model and the definition of the query language of a database system for complex objects. Second, the approach allows the use of an arbitrary representation of the geometric objects. After a short description of the query processor, we propose some representations for free form objects determined by B-spline or Bézier curves. The goal of efficient query processing in a database environment is achieved using a combination of decomposition techniques and spatial access methods. Finally, we present some experimental results indicating that the performance of decomposition techniques is clearly superior to traditional query processing strategies for geometric objects with free form boundaries.

1 Introduction

One essential goal in engineering applications is the storage of all product definition data in an integrated form. In mechanical engineering, a product is usually defined by a set of parts which are again described by manufacturing (CAD/CAM-) data. Thus, the description of parts and products has a complex structure and contains geometric data. For the purpose of sophisticated retrieval capabilities, a database management system for engineering applications has to offer modelling capabilities for complex structured as well as geometric data.

We will concentrate on the handling of geometric data in the way known from spatial database systems (see [Wi91]). The most relevant geometric objects in engineering applications are solids in Euclidean 3-space and surfaces in Euclidean 2-space; in both cases the objects can be considered as regular point sets. We will restrict our considerations to free form surfaces in Euclidean 2-space, free form regions for short that are described in section 2, but the concept easily can be extended to solid objects in 3-space (see [He92]).

Desirable retrieval capabilities besides standard SQL-like queries are the search for similar parts (see [SK89]) and geometric queries, i.e. queries with geometric selection conditions. Geometric queries and concepts for their realization are elaborated further in section 3. The concept of a query processor is adapted to region data in section 4. The query processor is a very flexible tool increasing query performance by using different representations for regions. In section 5, we focus on possible representations for free form regions. Contrary to polygonal regions, there are no decomposition techniques known for query processing of free form regions. We propose two types of decompositions called heterogeneous and trapezoid decomposition and present some analytical results.

In section 6 results of an experimental comparison of the different representations are presented. After a comparison of the decomposition methods, we consider the construction of the different representations. Finally, we propose query processing strategies and present experimental results comparing the performance of these strategies for different representations with respect to selected types of queries, i.e. the PointQuery and the WindowQuery defined in section 3. The paper concludes with a summary of the results.

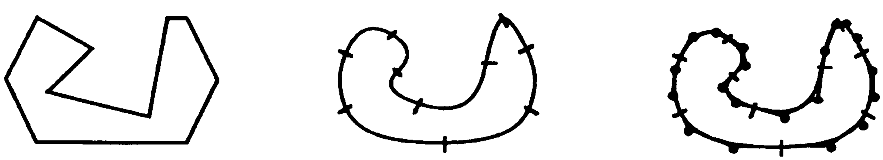
2 Definition of free form regions

As mentioned above, we will focus on surfaces in Euclidean 2-space, called regions. A region is defined by a closed, not self-intersecting curve C which may be given by a sequence of curve segments with common start and end points. According to the Jordan curve theorem, such a curve C divides the Euclidean plane into three sets: the interior, the exterior and the boundary points.

In the case of a polygonal region, the curve can be described by the vertices of the polygon. Methods for the description of smooth curves with only few information are subsumed under the notion of free form curves. Some of the most widely used methods are B-spline and Bézier curves that are defined by a sequence of control points.

We define a *free form region* as a region that is bounded by a closed, not self intersecting curve C which is described by a sequence of Bézier or B-spline curves. In this context the following properties of free form curves are essential:

- B-splines are more suitable for the interactive definition of free form curves due to the local control, i.e. the modification of a control point affects only a limited interval of the curve, and due to the simplicity of assembling patches.
- Bézier curves are better suited for computations on a readily defined curve This is due to the facts that start and end points are given explicitly and that Bézier curves pass more closely to the control points than B-splines.



control polygon for B-spline curve | B-spline curve with segments identified | curve with Bézier control points

Fig. 1. Examples for B-spline and Bézier curves

For these reasons, we assume in the following that curves are given as B-splines and that they are stored as Bézier curves. Therefore, we focus on Bézier curves. A *Bézier curve of degree m* is defined by a sequence of m+1 points P_i (i=0(1)m):

$r(t) = \Sigma_{i=1(1)m} B_{i,m}(t) * P_i$, where $B_{i,m}(t)$ are Bernstein polynomials.

Thus the Bernstein polynomial is a weighting factor of the points varying with the value of t (over the curve). A *cubic Bézier curve* (order 4) is defined by a sequence of four points P_0, P_1, P_2 and P_3:

$$X(t) = (1-t)^3 * P_0 + 3*t*(1-t)^2 * P_1 + 3*t^2*(1-t)^2 * P_2 + 3*t^3 * P_3 .$$

For the realization of the concept proposed above, first of all we need a conversion algorithm from B-spline to Bézier curves. One solution to this problem is the so-called Oslo-algorithm (see [HL89]).

The preprocessing used for the decompositions is equal to the preprocessing of the Koparkar-Mudur-algorithm that partitions a curve in those points having a horizontal or vertical tangent (see [HL89]). In the general case, the computation of those extreme points is very costly because it involves for example the evaluation of a non-linear system of equations. Here, we only consider the computations for the simple case of cubic Bézier curves with uniform knot vectors.

A. Computation of extreme points of cubic Bézier curves

The equation of a cubic Bézier curve with a uniform knot vector is given above in vector form; if we take a look at the component functions, we see that the component functions are polynomials of degree 3. From this fact, it can be derived (see [FM92]) that a cubic Bézier curve can have at most two extreme points per axis.

B. Partitioning a cubic Bézier curve

Another problem considered here is the partitioning of a Bézier curve into two Bézier curves according to a point given by a value of t. This partitioning is an integral part of all decomposition algorithms. The basis of the algorithm is the de Casteljau algorithm to determine a point of a Bézier curve X(t) by a given value of t (see [HL89]). The border points of the de Casteljau schema are the control points of the two Bézier curves that originate from the decomposition of the Bézier curve in point X(t).

3 Known concepts from spatial databases

The demand for storing objects with geometric information (geo-objects for short) in database systems has led to the development of spatial databases. Spatial database systems are concerned with appropriate modelling and retrieval capabilities especially for sets of geo-objects which are grouped for semantic reasons (all parts of a product for example). If such a set of geo-objects is often queried with geometric selection conditions, it should be stored in a cluster (see [KH91][Wi91]). In the following we focus on the geometric information of objects ignoring other information. A set of (free form) regions embedded in the Euclidean plane is called a (two-dimensional) *scene*.

3.1 Retrieval capabilities

The abstract definition of sets of geo-objects also allows an easy definition of the required retrieval capabilities. The most relevant queries using geometric selection conditions to a given scene S of geo-objects are only summarized here (see [Wi91] or [KH91]):

PointQuery: Given a point $P \in \mathbf{R}^2$. Select all geo-objects in S containing P as interior or border point.

WindowQuery: Given a rectangle $R = ((x_1,x_2),(y_1,y_2))$ with x_1, x_2, y_1, y_2 real numbers and $x_1 \leq x_2$, $y_1 \leq y_2$. Select all objects in S that intersect rectangle R.

RegionQuery: Given a region R. Select all objects in S that intersect region R.

Spatial Join: Given two sets of geo-objects S_1 and S_2. Compute a new set of geo-objects containing all (regular) intersections of pairs of regions (R_1, R_2) with R_1 in S_1 and R_2 in S_2.

This set of queries for region data is easily extendable, but a full list of queries is out of the scope of this paper. Furthermore, the low level queries and operations mentioned above can be used to build complex application specific queries. For an efficient spatial join algorithm for polygonal regions we refer to [BK93a].

3.2 Methods for efficient query processing

For efficient query processing, regions should be clustered according to their spatial location using available spatial access methods (SAMs). A problem concerning region data is the lack of access methods that directly handle any type of region data. Because most access methods for region data are restricted to handle multidimensional rectilinear boxes (see [KS89], [Wi91]), strategies for efficient query processing of region data have to be considered more closely.

3.2.1 Approximation by minimum bounding boxes

This method is most commonly used, because it allows the direct use of spatial access methods for rectangles. A region is approximated by its minimum bounding rectilinear box (MBB for short) and this rectangle is used as spatial key in an access method for rectangles. The MBBs approximate size, location and spatial relationship of regions. This method is also applicable to free form regions. Due to the approximation, query processing is now a two step procedure visualized in figure 2:

Filter step: The rectangles which are conservatively approximating geo-objects (see [OM86]) are stored in a SAM for rectangles. A retrieval operation is represented by an appropriate query on the access method which yields a set of candidates containing all objects fulfilling the original query.

Refinement step: All candidates of the filter step are considered to determine whether they really fulfill the query condition or not.

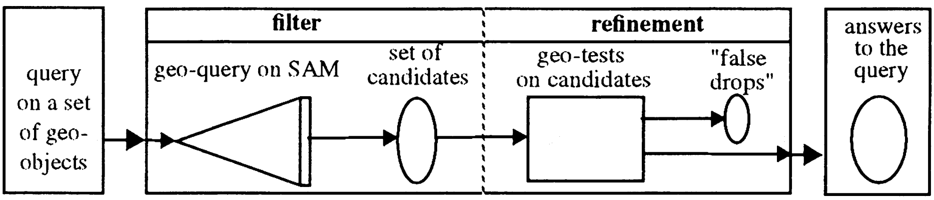


Fig. 2. Query processing using a filter

Instead of the MBB, more efficient approximations may be used if there are spatial access methods for that type of approximations. Examples of such approximations are the convex 5-corner and the rotated MBB (see [BK93b]).

3.2.2 Query processing using a z-order representation

The idea of z-order representation is to represent a region by (a set of) one-dimensional point data that can be stored as a point set in a one-dimensional index like the B^+-tree (see [OM86]). The basis is a recursive (multidimensional) spatial decomposition scheme, called quadtree, decomposing a finite part of the Euclidean plane into cells. A region is identified with the cells that intersect the region. Because the number of cells affects the approximation quality of the objects and therefore the number of candidates, a representation of one region by a set of cells is considered. In the PROBE-project (see [OM86]), the refinement step uses the original representation of the regions. A problem of the use of a redundant representation in the filter step is to eliminate the redundancy in the candidate set.

3.2.3 Query processing using a decomposition representation

The strategies considered so far all use the same refinement step; the original object is tested against the query condition. In the case of free form regions and three-dimensional objects, the amount of work in the refinement step can be enormous. The essential reason for the use of a decomposition representation as proposed in [KH91] is therefore the reduction of the processing time in the refinement step.

A *decomposition representation* as proposed in [KH91] is the combination of:

- a *decomposition technique* partitioning a region into a set of component objects of simpler shape (triangulation for example)
- a *spatial access method* to store the approximated component objects.

The filter step now yields a set of *component objects* (which are triangles in case of a triangulation) that have to be considered in the refinement step. A problem is the amount of redundancy introduced by the decomposition scheme. But a decomposition representation seems to be a good alternative especially for free form regions because the computations in the refinement step can obviously be reduced.

4 A query processor for region data

From the viewpoint of a data model of a database management system (DBMS), regions can be seen as an additional data type. It is well-known that there are some extensible database management systems which allow the integration of new data types in the form of an abstract data type. For our presentation it suffices to say that regions are integrated as abstract data types defined by an interface and an implementation.

4.1 The implementation of the query processor for region data

The interface of the new data type 'REGION' has to be comparable to other data types in the data model. In a relational DBMS the interface (used in the query language) consists of a set of predicates, functions and operations. An extensive survey for different kinds of geometric data types is given for example in [Gü89]. Some basic predicates are the test on intersection or the point-in polygon test. These predicates can be used as selection condition to define the queries on sets of geo-objects listed in section 3.1.

In contrast to a straightforward implementation of the interface of an abstract data type, we allow some degrees of freedom to achieve efficiency and data independence (see [KH91]). The concept of the query processor for regions allows us to:

- use different representations of one object in parallel
- use one or more representations to answer one query
- integrate a query optimizer for selecting an optimal strategy (query plan) to answer a given query.

The generation of the representations can be achieved by an extended version of a check-in check-out mechanism (see [Ka85][He92]). The main tasks of a query processor are the control and management of representations, the choice of a query plan for query processing, and the control of the interaction of different representations.

4.2 Types of representations and their use in a query processor

If we take a closer look at the different types of representations for sets of regions presented in section 3.2, we can identify two distinguishing features:

1.) The representation of geo-objects can be approximate or exact.

2.) The representation of geo-objects can be redundant or non-redundant.

Examples for approximate representations are the MBB- and the z-order representation of regions. While an approximate representation can only be used in an optional filter step, query processing requires at least one exact representation for the refinement.

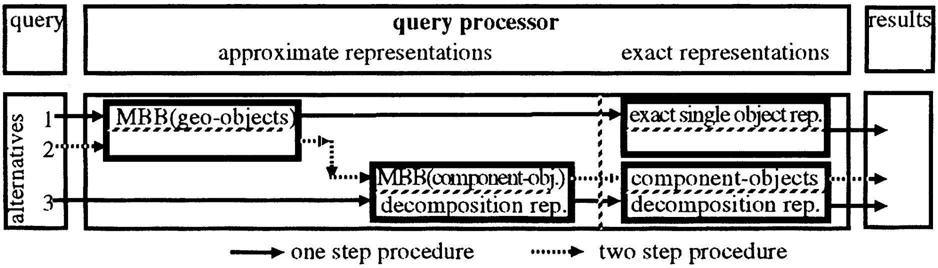


Fig. 3. Configuration of a query processor and possible query plans

Conceptually, it is possible to use more than one filter; in that case the representations used have to be ordered in a sequence of finer filters. The number of filter steps (approximate representations used) is taken to identify the procedure of query processing; the MBB-representation is called a one step procedure because it uses exactly one approximate representation.

While figure 2 visualizes the algorithmic side of query processing, figure 3 shows a query processor in a more schematic view. Figure 3 depicts a query processor using all (but one) types of representations described so far and also shows possible, alternative query plans. An open question so far is the choice of appropriate query plans, especially in the case of decomposition representations.

5 Selected representations for free form regions

In this section, the representations used in the performance comparison are described. First, we consider a non-redundant and exact representation that can be used in an MBB representation. Then we focus on representations based on decompositions.

5.1 A non-redundant and exact representation

As input for the query processor all free form regions are accepted whose boundary is given by a sequence of cubic B-spline or Bézier curves forming a closed curve (external representation). Because of algorithmic and efficiency considerations, we decided to use only cubic Bézier curves for the internal representation. Therefore, the check-in mechanism is responsible for the conversion of this description into a sequence of cubic Bézier curves applying the conversion algorithm mentioned in section 2.

For sets of geo-objects, this representation is combined with a non-redundant filter using a MBB approximation supported by a spatial access method. Thus, the most commonly used query processing strategy, here called the identity representation, is integrated into the query processor.

5.2 Decomposition-based representations

There have been many proposals for decomposition techniques for regions that were surveyed in [KS85] and [KH91] for example. While in the first survey, decomposition techniques were considered with regard to criteria used in computational geometry, the second paper analyzes decomposition techniques with respect to query processing in database systems. Most of the techniques presented there are only suitable for polygons.

Here, we present two decomposition techniques. The first is a generalization of the trapezoid method for polygonal regions (see [AA83]); the second is called heterogeneous decomposition and is derived from both, the ISOS method ([BD89]) and the heterogeneous decomposition for polygonal regions [KH91].

5.2.1 Decomposition into Trapezoid-like regions

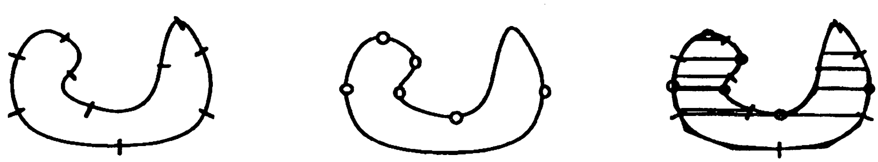
A simple decomposition scheme for polygonal regions is the decomposition into trapezoids proposed in [AA83]. This technique was extended and tuned to be used in a query processor by [He92] and first results are presented in [KH91]. The simple decomposition method is described by the following rule (partition method H in [AA83]):

For each vertex of the polygon draw as many horizontal chords as possible.

Partition method H allows the prediction of the number of components generated in the case of polygonal regions ([AA83]): a polygon with n vertices, w holes, and h horizontal edges is decomposed into at most n+w-h-1 components.

Assuming that free form regions are given in the form of the identity representation described above (i.e. by a sequence of cubic Bézier curves), the decomposition into trapezoid-like components is given by a two step process (see [FM92] and [He92]):

1.) Determine the extreme values of the Bézier curves and decompose each curve into these points, resulting in monotonic curves.
2.) Start partition method H with the set of boundary elements derived in step 1.



Bézier patches — extreme values requiring decomposition — decomposition into trapezoids

Fig. 4. Decomposition into trapezoid-like components

Because all curves are monotonic, partition method H generates *trapezoid-like regions* that are bounded by two horizontal chords and two parts of (monotonic) Bézier curves (see Figure 4 for an example). The monotonic character guarantees that the decomposition method is unique and that geometric predicates are easy to compute. For the decomposition into trapezoid-like components (T-components), we can give some analytical results:

Lemma 1: Assume, a curve is given by b B-spline curves with c control points. Then the number of T-components is bounded by 4*(c-3b).

Because the number of B-spline segments is c-3b, the conversion algorithm generates c-3b Bézier curves. Considering the maximum number of extreme values of a Bézier curve and the number of components generated by partition method H for polygonal regions, the proof is completed.

Lemma 2: The runtime of the decomposition into T-components is O(n*logn) in the worst case where n is the number of control points of the B-spline curve.

With Lemma 1 and the fact that the conversion and the decomposition of the curves can be done in constant time, the runtime of the algorithm is determined by the runtime of the application of partition method H, which is O(n*logn).

5.2.2 A heterogeneous decomposition for regions

The method is similar to the ISOS method reported in [BD89] and to the heterogeneous decomposition for polygonal regions in [KS91]. The idea is to decompose a region into two sets of components (see figure 5 for an example):

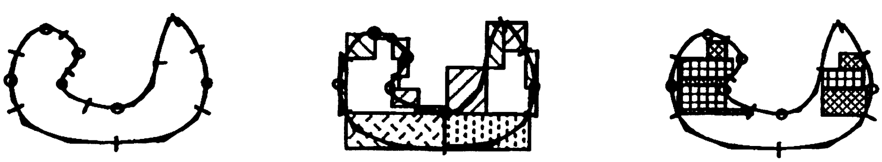
boundary components: A decomposition schema for the bounding curve is used and the components are approximated by rectangles such that the bounding curve is already contained in the rectangles.

interior components: The part of the region not covered so far is decomposed into rectangular components.

A first idea of the decomposition algorithm is given by the following three rules:

1.) Enclose all boundary elements by MBBs.
2.) In the case of pairs of overlapping MBBs, decompose the corresponding boundary elements and replace the old boundary elements by the computed components. Recompute the MBB for the new boundary elements.
3.) The difference between the original geo-object and the MBB of the boundary elements consists of one or more rectilinear regions that are decomposed.

It is obvious that the second rule has to be applied recursively. Unfortunately it is not always possible to generate a decomposition without pairs of overlapping MBBs. Solutions depend on metrical facts, so that an analytic prediction of the number of components is not possible.



monotonic Bézier patches — boundary MBB — interior MBB

Fig. 5. Example for the heterogeneous decomposition

One essential difference between the heterogeneous decomposition used for polygonal regions and the decomposition used here concerns the storage and the query processing strategy: the two sets of components (interior and exterior components) indexed by their MBB are stored in two different occurrences of the spatial access method used.

6 Experimental comparison of the representations

First, the decomposition methods are considered with respect to the number of components generated, the time used for generating the decomposition and the construction of the representation. A last criterion is based on the quality of the approximation. Finally, the query performance of the different representations is compared for the different types of queries presented in section 3.

6.1 Experimental setup

For the comparison of the different representations, we generated a set of pseudo random data. We use four sets of free form regions (scenes) varying the complexity of the regions. The complexity is measured by the number of points describing one region. We use four sets of regions that have an average number of 25, 50, 75 and 100 control points per region denoted by U25, U50, U75, and U100, respectively. The control polygons are generated in the same way as we constructed polygonal regions in [KH91].

Each scene is composed of 100 regions. The centers of the regions are distributed uniformly in the unit square given by [0..1] x [0..1]. The size of the regions was chosen such that the sum of the area of all regions is 70% of the size of the unit square. This was done to generate nearly disjoint scenes which we assumed to be realistic for CAD data.

The spatial access method used is the R*-tree which demonstrated a superior performance in the tests for polygonal regions [KS91]. The query processor was implemented in MODULA-2 and the tests were run on SUN 3/60 workstations with a local disk. Performance data were taken from the profiler available under UNIX.

6.2 Comparison of the decomposition methods

We assume that the regions are given as a sequence of Bézier curves (e.g. the internal representation). The check-in mechanism transforming the B-spline curves to cubic Bézier curves is a two step process described earlier. The parameters considered in the experimental performance comparison are:

1.) the number of components generated (C).
2.) the quality of the approximation (A).
3.) the time required for generating the decomposition in seconds (T(D)), the time for generating the representation (T) and the time required for the SAM (T(SAM)).
4.) the number of disk accesses (ACCESSES (SAM)).

The results for the different representations and for the different sets of geo-objects are summarized in table 1. The quality of the approximation is measured by an approximation index defined in the following way:

Sum up the areas of all MBBs of components generated (for the identity representation consider the MBB of the object) and divide this by the sum of the areas of all regions.

The test series reveals the following results:

1.) The heterogeneous decomposition generates twice the number of components of the trapezoid decomposition.
2.) The quality of the approximation of the decomposition methods is much better than that of the approximation of the MBB-representation.
3.) The time required for the construction of the decomposition representations is rather high. With growing complexity of the regions of a scene, this construction time increases for the heterogeneous decomposition faster than for the others.
4.) The time for and the number of disk accesses increase linearly with the number of components.

The results presented here are clearly dependent on the spatial access method used. Tests in [KS91]) have shown, that the R*-tree is relatively slow with respect to insertion, but superior for retrieval. Thus, another choice could give better results for the construction of decomposition representations. But in the opinion of the authors, query performance is more important than insertion time because the construction is considered as a check-in procedure that is performed only once. Therefore, the choice of the R*-tree with slow insertion and high query performance is justified.

Representation	Scene	C	A	T [s]	T(D)	T (SAM)	ACCESSES (SAM)
identity	U25	100	1.40	54.4	0.0	0.36	372
trapezoid		6969	1.06	3699.5	145.6	2267.29	5216
heterogeneous		13154	1.10	4513.5	713.5	1690.78	9711
identity	U50	100	1.47	84.8	0.0	0.72	381
trapezoid		15525	1.04	5835.7	426.8	2288.63	9862
heterogeneous		31254	1.08	11500.2	2508.0	3801.93	15207
identity	U75	100	1.43	108.5	0.0	0.84	360
trapezoid		22706	1.03	8537.9	708.2	3300.53	13572
heterogeneous		43478	1.05	16740.4	4129.3	5287.55	19490
identity	U100	100	1.44	135.3	0.0	1.16	414
trapezoid		31750	1.02	12484.2	1177.4	4754.35	19109
heterogeneous		62489	1.04	24269.2	7137.4	7544.58	26235

Table 1: Performance parameters of the decomposition methods

6.3 Query processing strategies

We want to take a closer look at the query processing strategies described informally by figure 2, because some algorithmic details are not contained in the figure. First, we should mention that for all required evaluations of geometric predicates, a pretest on the basis of the MBB of the geo-object or component is used.

The identity representation as well as the one step and two step procedure for the trapezoid representation are implemented in the way described earlier. For the heterogeneous representation the two types of components are stored in different occurrences of the spatial access method. This requires a strategy coordinating the use of the two filters. The implementation [FM92] uses the two filters successively.

6.4 The Point and the WindowQuery

The test for the performance of the PointQuery for the different representations is based on a point set of 100 uniformly distributed points in the unit square. For the comparison of the different query processing strategies for window queries, seven sets of rectangles with different sizes were generated. Each set contains rectangles of approximately equal size. The average size of the rectangles in the different sets is 0.01%, 0.05%, 0.1%, 0.5%, 1%, 5% and 10% of the size of the unit square, respectively. They (their centers) are distributed uniformly in the unit square. Each set consists of 100 rectangles.

Figure 6 depicts the mean value for the time required for computing of one resulting geo-object, i.e. one answer to the query, averaged over 100 window or point queries respectively. Because there are no significant differences, only the diagrams for the region files U25 and U100 are shown. The performance comparison implies:

1.) Both two step decomposition representations are superior to the identity representation in all experiments. Particularly for small query sizes, they are up to three times faster in query processing than the identity representation.

2.) The two step procedure of the trapezoid method requires the minimum time for all sizes of query windows and all types of scenes. However, the heterogeneous decomposition is not much worse than the two step trapezoid decomposition.
3.) With growing complexity of the regions in a scene, the superiority of decomposition representations over the identity representation increases.

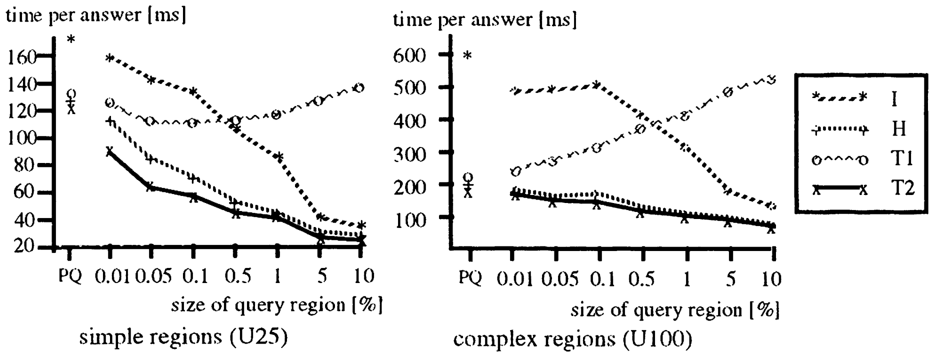


Fig. 6. Performance comparison of the Point and the WindowQuery

An interesting question arises from the fact that the time required for one answer decreases for the identity representation with growing size of the query rectangle. This decrease can be explained by some tuning features contained in the query processing strategy for the identity representation. A first test for the intersection is carried out comparing the MBB of a region with the query rectangle. In the case of large query rectangles, the MBBs are often included in the query rectangle. This suffices to decide that the region intersects the query rectangle such that no comparison with the free form region is required.

7 Conclusions

In this paper, we have proposed the concept of a query processor for the implementation of geometric queries on free form regions in a spatial database system. The query processor allows the application of different representations of regions suitable for processing a query. We survey concepts known from spatial databases and show how to use them in a query processor. We propose two decomposition methods for free form regions called heterogeneous and trapezoid decomposition. Finally, we present experimental results of different query processing strategies using different representations.

The comparison shows that the expense for a redundant representation is more than compensated by the computational savings for the geometric algorithms on free form geo-objects. This is partly due to the efficiency of the query performance of the selected spatial access method (i.e the R*-tree) which allows a fast access to locally bounded components of a scene. The results of the experiments show that the query performance of the two step procedure for the trapezoid representation is up to three times faster than the widely used identity representation (using no decomposition) and is superior to the identity representation in all experiments. For the selected scenes and queries, the two step procedure of the trapezoid decomposition is the best choice, but the query performance of the heterogeneous decomposition is only slightly worse. Another interesting fact is that the two step procedure for the decomposition into trapezoid-like regions is superior to all other techniques used. Thus, a complex query processing strategy with two coordinated filters is a good choice for the types of scenes which were investigated.

Finally, we want to mention that for the trapezoid decomposition algorithms can be derived for other geometric queries. In [He92] algorithms for the computation of Boolean operations such as the intersection of polygonal regions are presented. These algorithms can also be generalized to free form regions. In our future work, we will generalize our decompositions from 2D- to 3D-objects.

References

[AA83] Asano, T.; Asano, T.: Minimum Partition of Polygonal Regions into Trapezoids; Proc. 24th IEEE Annual Symposium on Foundations of Computer Science, 1983, 233-241.

[BD89] Beacon, G.R.; Dodsworth, J.R.; Howe, S.E.; Oliver, R.G.; Saia, A.: Boundary Evaluation Using Inner and Outer Sets: The ISOS Method; IEEE Computer Graphics and Applications, March 1989, 39-51.

[BK93a] Brinkhoff, T.; Kriegel, H.-P.; Seeger, B.: Efficient Processing of Spatial Joins Using R-trees; Proc. ACM SIGMOD Int. Conf. on Management of Data, Washington D.C., Mai 1993.

[BK93b] Brinkhoff, T.; Kriegel, H.-P.; Schneider, R.: Comparison of Approximations of Complex Objects Used for Approximation-based Query Processing in Spatial Database Systems; Proc. 9th Int. Conf. on Data Engineering, Vienna, Austria, 1993, 40-49.

[FM92] Fahldieck, A.; Mysliwitz, N.: 'Models for Processing Geometric Objects described by Free Form Surfaces in Database Systems'; Master Thesis, University of Bremen, January 1992 (in German).

[Gü89] Güting, R.H.: Gral: An Extensible Relational Database System for Geometric Applications; Proc. 15th Int. VLDB Conference, Amsterdam, 1989, 33-44.

[He92] Heep, S.: 'Queries on Geometry Data in CAD Database Systems'; Ph.D.Thesis, University of Bremen (in German).

[HL89] Hoschek, J.; Lasser, D.: 'Basics of Geometric Data Processing'; Teubner 1989 (in German).

[Ka85] Katz, R.: Information Management for Engineering Design; Springer, Berlin Heidelberg, 1985.

[KS85] Keil, J.M.; Sack, J.R.: Minimum Decompositions of Polygonal Objects; in: Toussaint, G.T. (ed.): Computational Geometry; Elsevier Science Publishers B.V. (North Holland), 1985, 197-216.

[KH91] Kriegel, H.-P.; Heep, P.; Heep, S.; Schiwietz, M.; Schneider, R.: An Access Method Based Query Processor for Spatial Database Systems; Proc. Int. Workshop on DBMS for Geographical Applications, Capri, May 16-17, 1991; in: Geoegraphic Database Management Systems, Springer, 1992, 272-292.

[KS89] Kriegel, H.P.; Schiwietz, M.; Schneider, R.; Seeger, B.: Performance Comparison of Point and Spatial Access Methods; Proc. Symposium on the Design and Implementation of Large Spatial Databases, Santa Barbara, USA, July 1989, 89-114.

[KS91] Kriegel, H.-P.; Horn, H.; Schiwietz, M.: The Performance of Object Decomposition Techniques for Spatial Query Processing; Proc. 2nd SSD'91, Zürich, August 1991; in: Günther, O.; Schek, H.-J. (eds): Advances in Spatial Databases; published as LNCS 525, Springer, 257-276.

[OM86] Orenstein, J.A.; Manola, F.A.: Spatial Data Modeling and Query Processing in PROBE; Technical Report CCA-86-05, Computer Corporation of America, Four Cambridge Center, Cambridge, MA, October 1986.

[Sa90] Samet, H.: The Design and Analysis of Spatial Data Structures; Addison-Wesley, Reading, MA, 1990.

[SK89] Schneider R., Kriegel H.P., Seeger B., Heep S.: Geometry-based Similarity Retrieval of Rotational Parts; Proc. Int. Conf. on Data and Knowledge Systems for Manufacturing and Engineering, Gaithersburg, Maryland, 1989, 150-160.

[Wi91] Widmayer, P.: Data Structures for Spatial Databases; in: Vossen, G.; Witt, K.-U. (eds.): 'Future Trends in Database Systems', Oldenbourg, 1991 (in German).