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

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

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

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.

Keywords: Spatial Database Systems, Free Form Curves, CIM, Query Processing, Access Methods

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 [Widm91]). 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, for the reason of presentation, but the concept easily can be extended to solid objects in 3-space (see [Heep92]). A more exact definition of the geometric objects is given in section 2.

Desirable retrieval capabilities besides standard SQL-like queries are the search for similar parts ([FrGr82], [SKSH89]) 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 obtained.

2 Definition of free form regions

As mentioned above, we will focus on surfaces in Euclidean 2-space, called regions.

Definition 1: A region is defined by a closed, not self-intersecting curve C which is given by:

$C := \{ P \in \mathbf{R}^2 , P = (x,y) \mid x = f_x(t) , y = f_y(t) , t \in [0,1] \}$

$|C|\ 1$; f_x, f_y: $[0,1] \to \mathbf{R}$ continuous; $f_x(0) = f_x(1)$ and $f_y(0) = f_y(1)$ (closed);

$f_x(t_1) = f_x(t_2)$ and $f_y(t_1) = f_y(t_2)$ iff ($t_1=0$ and $t_2=1$) or ($t_1=t_2$) (not self intersecting).

According to the Jordan curve theorem, the curve C divides the Euclidean plane into three sets: the points of the curve C are the boundary points of the region; the points to the left of C are the interior points int(C) and the points to the right of C are the exterior points ext(C) of the region defined by C.

In the case of a polygonal region, the curve can be described by the vertices of the polygon. For the description of smooth curves, usually much more information is needed. 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. The technique used is the description of the curve by a sequence of control points (a control polygon).

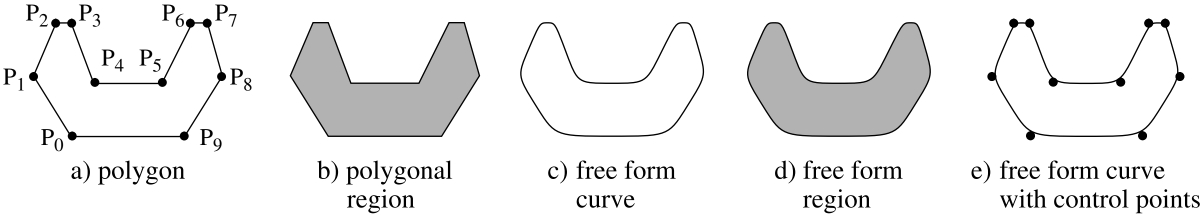


Figure 1: Examples of curves and regions

Definition 2: A free form region is defined by a closed, not self intersecting curve C which is described by a sequence of Bézier or B-spline curves $C_1, .., C_n$ with the following properties:

$f_{ix}(1) = f_{(i+1)x}(0)$ and $f_{iy}(1) = f_{(i+1)y}(0)$ $i = 1(1)n\text{-}1$, and $f_{nx}(1) = f_{1x}(0)$ and $f_{ny}(1) = f_{1y}(0)$.

2.1 Bézier curves

Bézier curves are defined by a sequence of control points that all have influence on all points of the curve. The number of points determines the type of curve.

Definition 3: A Bézier curve of degree m is defined by a sequence of m+1 points P_i (i=0(1)m):

$$r(t) = \sum_{i=0}^{m} B_{i,m}(t) \cdot P_i \text{ with } B_{i,m}(t) = \binom{m}{i} \cdot t^i \cdot (1-t)^i \text{ (Bernstein polynomial).}$$

The Bernstein polynomial is a weighting factor of the points that varies with the value of t (over the curve). We focus on Bézier curves of order 4 which are described by cubic polynomials.

Definition 4: A cubic Bézier curve (order 4) is defined by a sequence of four points P_0, P_1, P_2 and P_3:

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

According to definition 2, a closed curve can be constructed by Bézier patches such that the resulting curve is C^2- continuous in all points of the curve (see figure 2).

2.2 B-spline curves

An unpleasant property of Bézier curves for an interactive curve definition is that each point of the control polygon may influence the whole curve. Contrarily, B-splines allow local control, i.e. the modification of a control point affects only a limited interval of the curve. A polynomial spline interpolation describes a curve as a set of polynomials with low degree which are assembled fulfilling some continuity properties.

Definition 5: Given a vector T = (t_0, ..,t_n) of real numbers with $t_i \le t_{i+1}$, i=0(1)n-1. A function is a polynomial spline function of degree k iff:

1.) on all intervals [t_i , t_{i+1}] (i=0(1)n-1) S is a polynomial of degree k-1;

2.) S is C^{k-2} continuous ((k-2) times differentiable and continuous).

The points t_i are named knots.

Definition 6: A polynomial spline function of degree k which has only vertices with integer values and has weighting functions that are zero on all but k intervals is called B-spline (basic spline). The basis of the B-spline function is given by the following recursion:

$$N_{i,1}(t) = \begin{cases} 1 & t_i \le t \le t_{i+1} \\ 0 & \text{otherwise} \end{cases} \quad , \quad N_{i,m}(t) = \frac{t - t_i}{t_{i+m-1} - t_i} \cdot N_{i,m-1}(t) + \frac{t_{i+m} - t}{t_{i+m} - t_{i+1}} \cdot N_{i+1,m-1}(t) \quad \text{for } m \ge 2.$$

Definition 7: A B-spline curve of order 4 with control points P_0, P_1, P_2 and P_3 is given by

$X(t):= ((1-3t+3t^2-t^3)*P_0 + (4-6t^2+3t^3)*P_1 + (1+3t+3t^2-3t^3)*P_2 + t^3*P_3) / 6.$

A multiple use of a control point P (up to m times) is possible and causes the curve to pass closer to the control point P. If the control point is used m times, the curve passes through the control point. For the special case where only the first and the last control point are used m times the B-spline functions are equal to the Bernstein polynomials ([GoRi74]). Thus B-spline curves can be considered as a generalization of Bézier curves.

2.3 Properties of Bézier and B-spline curves

Both types of free form curves have the following properties in common:

- The curves are bounded by the convex hull of their control points.
- Curves can be assembled by using a set of patches.

These properties are depicted in the following figure.

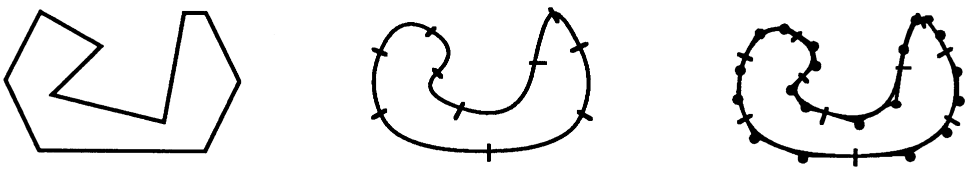


Figure 2: Examples for B-spline and Bézier curves

The following properties of B-splines can be seen as advantages over Bézier curves:

- The modification of control points has only local effects (local control).
- The construction of curves by assembling patches is easy and continuity can be guaranteed.

On the other hand, Bézier curves have the following desirable properties:

- The beginning and the end point of a Bézier curve are given explicitely.
- Bézier curves pass more closely to the control points than B-splines.

From this discussion we can derive the following statements:

1.) B-splines are more suitable for the interactive definition of free form curves.

2.) Bézier curves are better suited for computations on a readily defined curve.

For these reasons, we assume in the following that curves are given as B-splines and that their storage as Bézier curve should be realized. Therefore, we give some essential algorithms for the required computations.

2.4 Basic algorithms on Bézier and B-spline curves

For the realization of the concept proposed above, first of all we need a conversion algorithm from B-spline to Bézier curves. In this section, we consider two further algorithms required for the decomposition of regions.

2.4.1 Conversion algorithm

Solutions to this problem are considered for example in [HoLa89]. Two algorithms are described that solve the conversion from cubic B-spline to cubic Bézier curves as special case: one algorithm by Boehm and one by Cohen, Riesenfeld and Lyche called Oslo-algorithm. Both algorithms are suitable for curves with uniform as well as with non-uniform knot vectors and can be extended to free form surfaces (see [HoLa89]).

Lemma 1: A cubic B-spline segment with uniform knot vector defined by the control points P_0, P_1, P_2 and P_3 is given by

$X(t) := ((1-3t+3t^2-t^3)*P_0 + (4-6t^2+3t^3)*P_1 + (1+3t+3t^2-3t^3)*P_2 + t^3*P_3) / 6.$

This curve is exactly defined by the cubic Bézier curve:

$X(t) = (1-t)^3 * Q_0 + 3*t*(1-t)^2 * Q_1 + 3*t^2*(1-t) * Q_2 + t^3 * Q_3$, with

$Q_0 = (P_0 + 4 * P_1 + P_2) / 6;\ Q_1 = (2 * P_1 + P_2) / 3;$
$Q_2 = (P_1 + 4 * P_2 + P_3) / 6;\ Q_3 = (P_1 + 2 * P_2) / 3.$

It should be mentioned that the conversion of one B-spline segment can be performed using a constant number of arithmetic operations.

2.4.2 Preprocessing of Bézier curves

The preprocessing used for the decomposition is equal to the preprocessing of the Koparkar-Mudur-algorithm that partitions a curve in those points having a horizontal or vertical tangent (see [HoLa92]). In the general case, the computation of those extreme points is very costly because it envolves for example the evaluation of a non-linear system of equations. Here we only present the computations for the simple case of cubic Bézier curves with uniform knot vectors.

2.4.2.1 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 that can be normalized.

Another form of the Bézier function is $X(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$ with $\begin{array}{l} x(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0 \\ y(t) = b_3 t^3 + b_2 t^2 + b_1 t + b_0 \end{array}$

With this notation Lemma 2 follows directly ([FaMy92]).

Lemma 2: A cubic Bézier curve with uniform knot vectors given by a vector equation B(t) may possess extreme points for the following values of t, derived from $B'(t) = 0$:

$$t_{1,2} = \frac{-2 \cdot a_2 \pm \sqrt{(2 \cdot a_2)^2 - (12 \cdot a_3 \cdot a_1)}}{6 \cdot a_3} \qquad t_{3,4} = \frac{-2 \cdot a_2 \pm \sqrt{(2 \cdot a_2)^2 - (12 \cdot a_3 \cdot a_1)}}{6 \cdot a_3}$$

One of these values is an extreme value if and only if $t_i \in [0,1]$ and $x''(t_i) \neq 0$ $(i \in \{1,2\})$ or $y''(t_i) \neq 0$ $(i \in \{3,4\})$.

This shows that a cubic Bézier curve can have at most two extreme points per axis.

2.4.2.2 Partitioning a cubic Bézier curve

The last 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 ([HoLa89][BaGo88][FaRa88]).

A point of a Bézier curve is determined by recursive computation of supporting points:

$$S_{i,m}(t) = (1-t) * S_{i-1,m-1}(t) + t * S_{i,m-1}(t) \quad \text{with} \quad S_{i,0}(t) = P_i \text{ (control points).}$$

Figure 3 shows the geometric visualization of the de Casteljau algorithm. The points are usually specified in a schema called Casteljau schema. The border points of the de Casteljau schema (these are the points P_0, $S_{1,1}(t)$, $S_{2,2}(t)$, X(t) and X(t), $S_{3,2}(t)$, $S_{3,1}(t)$, P_3 respectively) are the control points of the two Bézier curves that originate from the decomposition of the Bézier curve in point X(t).

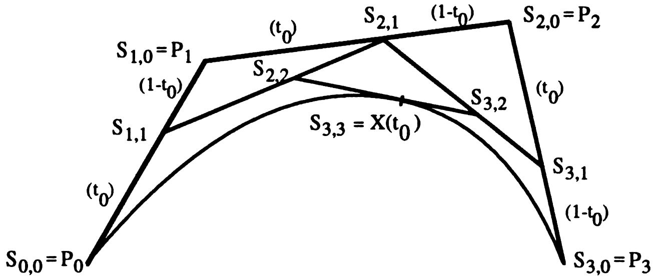


Figure 3: Geometric visualization of the de Casteljau algorithm (see [HoLa89])

3 Known concepts from spatial databases

The demand for the storage of 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 [KHHS91][Widm91]). For achieving a compact definition of sets of geo-objects, we restrict ourselves to the geometric information of objects ignoring other information.

Definition 8: 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 have been considered very often and are summarized here (see for example [Widm91] or [KHHS91]).

Definition 9: The most relevant geometric queries to a given scene S of geo-objects are:

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 $x_1 \le x_2$, $y_1 \le y_2$. 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.

3.2 Methods for efficient query processing

Because a sequential treatment of all geo-objects for the computation of the result of a query is not an efficient solution, the development of access methods for multidimensional point and region data has been a precondition for the implementation of spatial database systems. 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 [KSSS89], [SeKr 88], [Widm91]), strategies for efficient query processing for 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. The most important contribution of this paper is that, for the first time, free form regions are considered for efficient query processing in the context of spatial databases. Due to the approximation, query processing is now a two step process visualised in figure 4:

1.) *Filter step:* The (approximated geo-objects) rectangles are stored in a spatial access method for rectangles. A retrieval operation is represented by an appropriate query on the access method that yields a set of candidates. This set contains all objects fulfilling the original query because the approximation by the MBB encloses the represented region (conservative approximation [OrMa86]).
2.) *Refinement step:* All candidates of step 1 are considered to determine whether they really fulfill the query condition or not.

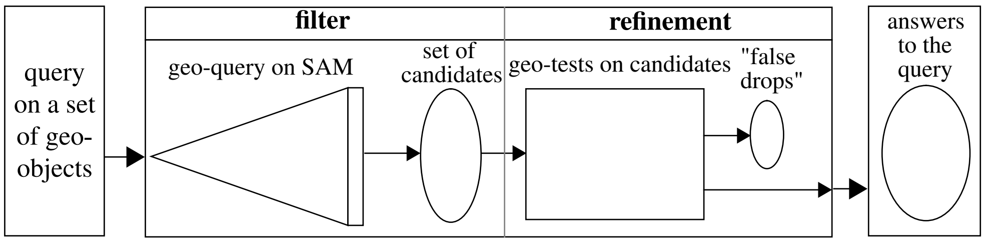


Figure 4: Query processing using a filter

Instead of the MBB, other simple geometric objects could be used for approximation if there are efficient spatial access methods for that type of objects. Examples of such objects are spheres (sphere tree [Oost90]) and convex regions (the cell tree [Günt88]).

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 [OrMa86]). The basis is a recursive (multidimensional) spatial decomposition scheme, called quadtree, that decomposes a finite part of the Euclidean plane into cells. A region is identified with the cells that intersect the region. Because it is possible to enumerate these cells (z-order), they can be coded as one-dimensional points. In [OrMa86], algorithms for the above queries are presented using a B^+-tree as access method for one-dimensional point data.

One problem of this representation is the bad approximation of objects by one cell (compared to the MBB approximation). Therefore, the representation of one region by a set of cells is considered. Such a representation is called a redundant representation. Figure 5 shows the profit of an approximation by more than one cell.

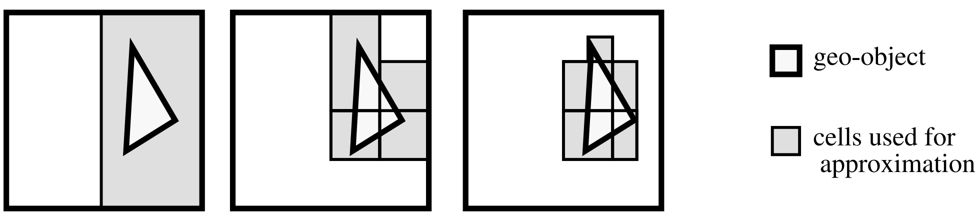


Figure 5: Quadtree approximation of regions with a varying number of cells

Because of the approximation, the z-order representation can only be used as a filter. In the PROBE-project (see [OrMa86]), 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. Figure 6 shows the query processing using a redundant representation in the filter step.

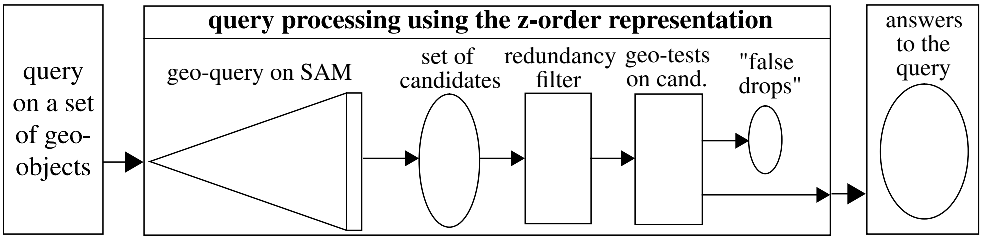


Figure 6: Query processing using z-order representation

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 [KHHS91] is therefore the reduction of the processing time in the refinement step.

A decomposition representation as proposed in [KHHS91] is the combination of:

1.) a decomposition technique partitioning a region into a set of component objects of simpler shape (triangulation for example)

2.) 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. Figure 7 shows the query processing strategy for a decomposition representation.

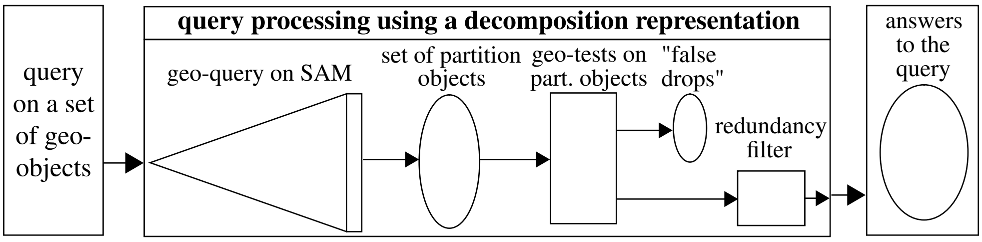


Figure 7: Query processing using a decomposition representation

A decomposition representation seems to be a good alternative especially for free form regions because the computations in the refinement step can be very expensive. This is due to the fact that for most geometric selection conditions, all patches defining the bounding curve have to be considered.

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 and some extensible database management systems allow the integration of new data types in the form of an abstract data type ([CDFG86], [BBGS86], [WSSH87]). 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 interface 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. If we compare it with a data type 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 [Güti89]. Some basic operations are given in figure 8. As we can see, the predicates presented can be used as selection condition to define the queries on sets of geo-objects listed in section 3.1.

predicates	REGION x REGION REGION x rectangle REGION x point	→ BOOLEAN → BOOLEAN → BOOLEAN	intersects, is_contained intersects point_in_object
functions	REGION REGION	→ REAL → rectangle	area MBB
operations	REGION x REGION	→ REGION*	intersection, union, difference

Figure 8: Elementary predicates, functions, and operations of a data type region

4.2 The implementation of the query processor for region data

In contrast to an abstract data type, we do not use a straightforward implementation of the interface but allow some degrees of freedom to achieve efficiency and data independence (see [KHHS91]). 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 [Katz85][Heep92]). In such an environment, 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.

4.3 Types of representations and their use in a query processor

As we have seen in section 3.2, different types of representations can be used for query processing. If we take a closer look at these representations, we can identify two distinguishing features of representations for sets of regions:

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 such approximate representations can only be used in an optional filter step, query processing requires at least one exact representation for the refinement step.

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.

The examples given above are also examples for redundant (z-order) and non-redundant (MBB) representations. As we have seen above the application of a redundant representation requires the use of a component called redundancy filter (see figures 4, 6, and 7).

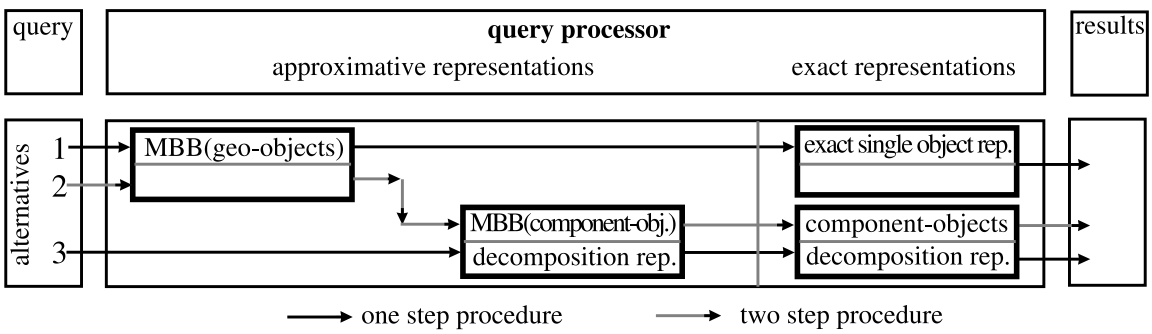


Figure 9: Configuration of a query processor and possible query plans

While figures 4, 6, and 7 visualize the algorithmic side of query processing, figure 9 shows a query processor in a more schematic view only presenting the representations used. The configuration of a query processor is characterized by the type of representations and their interaction. Figure 9 depicts a query processor using all (but one) types of representations described so far and also shows possible query plans.

The two approximations and the two exact representations allow at least the three alternative query plans shown in figure 9; one of these query plans uses a two step procedure and the other two consist of a one step procedure. 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 of all, we consider a non-redundant and exact representation that can be used in an MBB representation. Then we concentrate on representations based on decompositions.

5.1 A non-redundant and exact representation

The exact and non-redundant representation used in the query processor is a boundary representation that is capable of describing exactly one geo-object (single object representation, SOR for short). The starting point is the definition of regions in section 2 that define the boundary of a free form region as a sequence of 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 described in subsection 2.4.

For sets of geo-objects, this representation is combined with a non-redundant filter using a MBB approximation supported by a spatial access method. The computation of the MBB is based on the convex hull property of Bézier curves (i.e. all points of a segment are inside of the convex hull of the control points). Because we use that minimum bounding box covering all control points as MBB all interior and boundary points of the region defined by the curves are covered.

This allows us to integrate the most commonly used query processing strategy into the query processor. Because no redundant representation is used, it is called the identity representation.

5.2 Decomposition-based representations

There have been many proposals for decomposition techniques for regions that were surveyed in [KeSa85], [KHHS91] and [Heep92] for example. While in the first survey decomposition techniques were considered with regard to criteria used in computational geometry (optimal number of components, minimal runtime of decomposition algorithm), the other two papers analyse decomposition techniques with respect to query processing time in database systems. The decomposition techniques are classified and compared, but most of the techniques presented are only suitable for polygons.

Nevertheless, we want to use some essential techniques and technical terms also in this paper. We differentiate between spatial and structural decompositions. A spatial decomposition is based on a schema decomposing the underlying space and inducing a decomposition of the geo-objects placed in the space (e.g. the decompositions induced by the quadtree presented in [Same90]). Structural decompositions are defined by conditions that have to be fulfilled by the components of the decomposition (e.g. in a triangulation all components have to be non-overlapping triangles).

All the decomposition techniques presented here are structural decompositions. The first is a generalization of the trapezoid method for polygonal regions ([AsAs83]) and decomposes free form regions into trapezoid-like components. The other decomposition called heterogeneous decomposition is derived from both, the ISOS method ([BDHO89]) and the heterogeneous decomposition for polygonal regions [KHHS91].

5.2.1 Decomposition into Trapezoid-like regions

A simple decomposition scheme for polygonal regions is the decomposition into trapezoids proposed in [AsAs83]. This technique was extended and tuned to be used in a query processor by [KrHe89] and first results are presented in [KHHS91]. The simple decomposition method is described by the following rule (partition method H in [AsAs83]):

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. An estimation is given in [AsAs83]: a polygon with n vertices, w holes, and h horizontal edges is decomposed into at most n+w-h-1 components. A generalization of this method to free form regions is outlined in [FaMy92] and [Heep92].

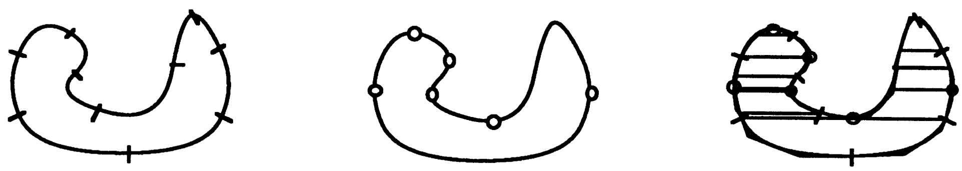
We assume that free form regions are given in the form of the identity representation described above (i.e. by a sequence of cubic Bézier curves).

The decomposition into trapezoid-like components is given by a two step process:

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 new set of boundary elements derived in step 1.

Because all curves are monotonic, partition method H generates trapezoid-like regions that consist of:

1.) two horizontal chords
2.) two parts of (monotonic elements) Bézier curves.



Bézier patches — extreme values requiring decomposition — decomposition into trapezoids

Figure 10: Decomposition into trapezoid-like components

The monotonic character guarantees that the decomposition method is unique and that geometric predicates are easy to compute. Figure 10 shows an example for the decomposition into trapezoid-like components (T-components). For this decomposition technique we can give some analytical results:

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

Proof: Obviously the number of B-spline segments is given by c-3b. The conversion algorithm therefore generates c-3b Bézier curves. Because the number of extreme values of a Bézier curve is limited to four, the number of monotonic Bézier curves is restricted by 4*(c-3b). Considering the number of components generated by decomposition method H for polygonal regions the proof is completed.

Lemma 4: 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 curves.

Proof: With Lemma 3 the number of monotonic Bézier curves is bounded by 4*n; because the conversion and the decomposition of the curves can be done in constant time, the first step of the decomposition can be performed in O(n) time. The runtime of the algorithm is therefore 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 [BDHO89] and to the heterogeneous decomposition for polygonal regions [KHSS91]. The idea is to decompose a region into two sets of components:

- 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 and simple 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. These rectilinear regions are decomposed by the algorithm of [FeSS84].

It is obvious that the second rule has to be applied recursively. Because it is not always possible to generate a decomposition without pairs of overlapping MBBs, some exceptions have to be defined. Two problematic cases are:

1.) The MBBs of two successive curves having a common point may overlap.
   One solution (and the one used here) is to ignore this in the decomposition process and implement an exception handling for all types of queries.
2.) The MBB of two neighbouring but not successive curves may overlap.
   If two curves pass very close to each other, the curves have to be decomposed recursively to generate MBBs without overlap. There are heuristic solutions to reduce the number of components.

Because of the second fact, a heterogeneous decomposition of this form is always dependent on metrical facts and therefore the number of components of the decomposition of a given region is not predictable in an analytic way.

As an example, figure 11 depicts a decomposition of a region using the heterogeneous decomposition.

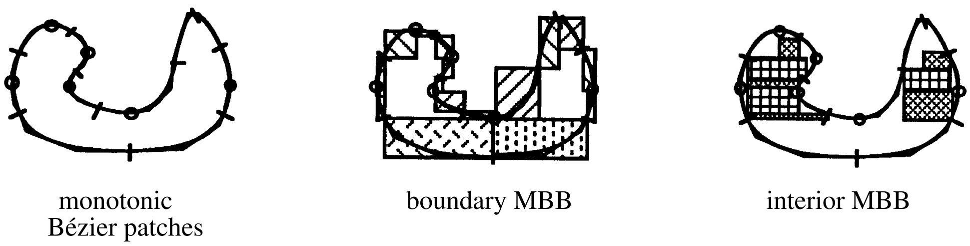


Figure 11: Example for the heterogeneous decomposition

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

The different representations are considered under the following aspects:

1.) Comparison of the decomposition methods
   The decomposition methods are first considered with respect to the number of components generated and the time used for generating the decomposition. A comparison with the identity representation is based on the quality of the MBB approximation compared to the quality of the approximation of the MBBs of the components.

2.) Construction of the representation

In this comparison, the time required to construct the different representations, i.e. the decomposition and the construction of the spatial access method, is considered.

3.) Performance of query processing

The 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 (we use the notation U25, U50, U75, and U100). The control polygons are generated in the same way as we constructed (simple) polygonal regions in [KHHS91].

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. We chose this overlap to get (nearly) disjoint scenes, which we assumed to be realistic for real CAD data.

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 ([GKMK82]).

The spatial access method used is the R*-tree [BKSS90] which demonstrated a superior performance in the tests for polygonal regions [KrHS91].

6.2 Comparison of the decomposition methods

We assume that the regions are given as a sequence of Bézier curves which was assumed to be the internal representation. The check-in mechanism transforming the B-spline curve to cubic Bézier curves is a two step process described earlier. Tests have shown that the transformation of the B-spline segments into Bézier curves and the decomposition at extreme values is not significant. Thus, the time required is not considered any further. The number of Bézier curves generated is affected partially by the form of the control polygon which determines the number of extreme values.

The parameters considered in the experimental performance comparison are:

1.) the number of components generated (C)
2.) the time required for generating the decomposition in seconds (T(s))
3.) the quality of the approximation (A).

The results for the different representations, for the different sets of geo-objects and for the parameters 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 and divide this by the sum of the areas of all regions.

Because the approximation index is also defined for the MBB-representation, table 1 also contains the approximation index for the identity representation.

This test reveals the following results:

1.) The heterogeneous decomposition generates twice the number of components of the trapezoid decomposition which confirms the analytic results.
2.) The time required for generating the decomposition is considerably higher for the heterogeneous decomposition (it could be reduced by applying the algorithm of [LiTL89] instead of the algorithm of [FeSS84] for the decomposition of rectilinear regions).

3.) The quality of the approximation of the decomposition methods is much better than that of the approximation of the MBB-representation. The decomposition into trapezoid-like regions is better than the heterogeneous decomposition, although the number of components is larger.

Scene	U25			U50			U75			U100		
Representation	C	T[s]	A	C	T[s]	A	C	T[s]	A	C	T[s]	A
identity	100	0.0	1.40	100	0.0	1.47	100	0.0	1.43	100	0.0	1.44
trapezoid	6969	145.6	1.06	15525	426.8	1.04	22706	708.2	1.03	31750	1177.4	1.02
heterogeneous	13154	713.5	1.10	31254	2508.0	1.08	43478	4129.3	1.05	62489	7137.4	1.04

Table 1: Performance parameters of the decomposition methods

6.3 The construction of the representations

The main parameter considered here is the time required for the construction of the representation which consists of the decomposition and the generation of a spatial access method, i.e the R*-tree. As table 2 depicts, the time for the construction of the representations is mainly determined by the insertion of the MBB with the corresponding information into the R*-tree. Another parameter is the number of disk accesses performed by the spatial access method.

Representation	Scene	C	T	T (SAM)	ACCESSES (SAM)	Scene	C	T	T (SAM)	ACCESSES (SAM)
identity	U25	100	54.4	0.36	372	U75	100	108.5	0.84	360
trapezoid		6969	3699.5	2267.29	5216		22706	8537.9	3300.53	13572
heterogeneous		13154	4513.5	1690.78	9711		43478	16740.4	5287.55	19490
identity	U50	100	84.8	0.72	381	U100	100	135.3	1.16	414
trapezoid		15525	5835.7	2288.63	9862		31750	12484.2	4754.35	19109
heterogeneous		31254	11500.2	3801.93	15207		62489	24269.2	7544.58	26235

Table 2: Performance parameters of the construction of the representations

Two statements can be derived from the test results:

1.) 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 other decompositions.

2.) 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. As tests in other publications (e.g. [KrHS91]) have shown, 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.

6.4 Query processing strategies

We want to take a closer look at the query processing strategies described informally by figure 9, because there are some algorithmic details that 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. In the case of a window query, for example, each object intersection test is preceded by testing a query rectangle against a MBB. If at least two points of the MBB are contained in the query window or two sides of the MBB are intersected by one side of the query window, the query window intersects the geo-object.

The identity representation as well as the one step and two step procedure for the trapezoid representation are implemented in the way described earlier. However, some remarks on the heterogeneous representation are necessary.

As mentioned in subsection 5.2.2, 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 [FaMy92] uses the two filters successively.

6.5 The PointQuery

For ease of presentation, we first consider the PointQuery. A two step procedure is used to answer point queries. First a PointQuery is performed on the set of MBBs of the geo-objects resulting in a set of candidates. For these candidates, first a point query is performed on the set of interior components. If a component of the candidate fulfills the point query, the object belongs to the set of answers. Otherwise a test using the set of boundary components is necessary. If a point query performed on the boundary candidates is fulfilled, the candidate belongs to the set of answers.

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 each scene, PointQueries with all points of the point set are performed. Table 3 depicts the number of results and the number of disk accesses, summarized over all queries. The number of results corresponds to the overlap of the sets of regions.

	U25	U50	U75	U100
number of answers	72	69	72	67
accesses (identity)	246	257	203	219
accesses (trapezoid, 1-step)	366	357	327	320
accesses (trapezoid, 2-step)	348	351	298	317
accesses (heterogeneous)	380	397	350	409

Table 3: Disk accesses and number of results of the PointQuery test

The time required for the computation of the results of the point queries is depicted in figure 12. One bar of the diagram depicts the time required for a set of point queries on one representation; it is subdivided into the time required for the access method (SAM), the time for the geometric algorithms (GEO), and the time required for management tasks in the query processor (REST).

The query times are given for the identity representation (I), the heterogeneous representation (H) and the trapezoid representation using a one step (T1) and a two step procedure (T2). The query processing strategies are described in figure 9.

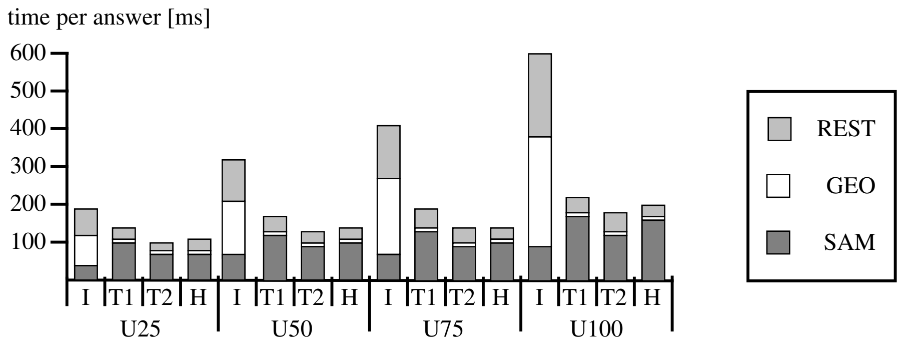


Figure 12: Performance comparison for the PointQuery

The following statements can be derived from the results depicted in figure 12:

1.) The identity representation requires more time than all other query processing strategies using a decomposition method. While the difference between the decomposition methods and the identity representation grows with the complexity of the regions, the difference between the decomposition methods is negligible.

2.) While the query processing time for the identity representation is to a large extend determined by the geometric computations, for decomposition methods, the spatial access methods consume the major portion of time.

Taking a closer look at table 3 and figure 12, it becomes obvious that query processing time is not only determined by the number of disk accesses. The computations of a spatial access method are also essential.

6.6 The WindowQuery

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 13 depicts the mean value of time required for the computation of one resulting geo-object, i.e. one answer to the query, averaged over 100 window queries. Only the diagrams for the region files U25 and U100 are shown, because there are no significant differences in the other test scenes.

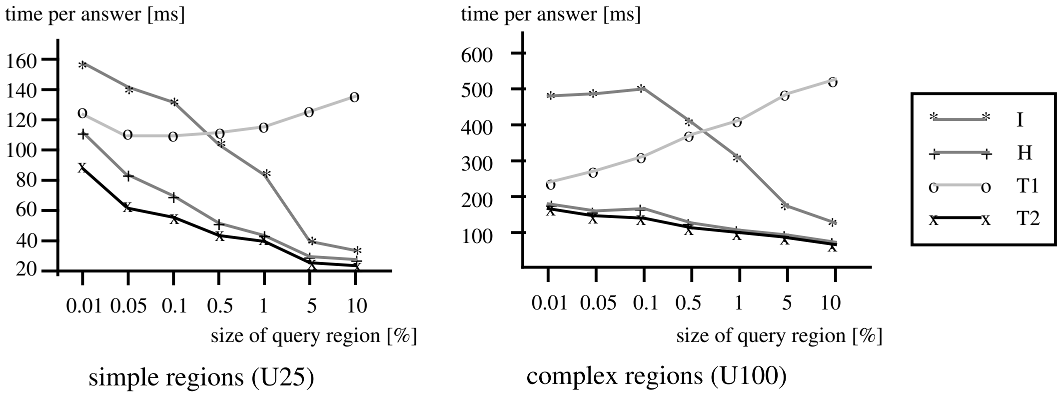


Figure 13: Performance comparison of the WindowQuery

The performance comparison for the WindowQuery implies the following statements:

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. 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.

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. Figure 14 reveals this behaviour.

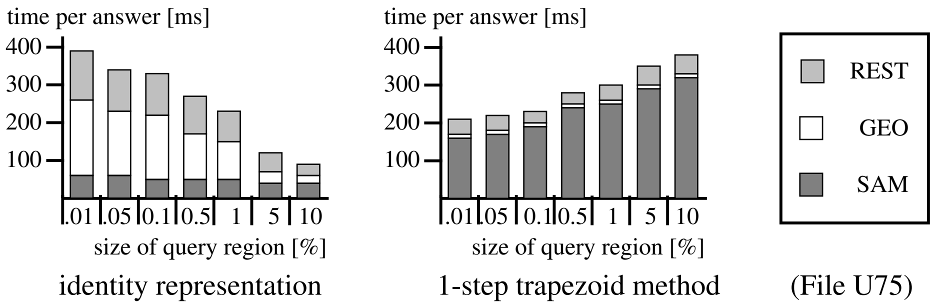


Figure 14: Distribution of the query time for the WindowQuery on different tasks

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 which can be used 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 MBB representation. For point queries and window queries this representation is superior to the MBB representation in all experiments.

The query performance of the heterogeneous decomposition is only slightly worse than the two step procedure of the trapezoid representation. For the selected scenes and queries, the two step procedure of the trapezoid decomposition is the best choice.

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 [Heep92] 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.

References

[AsAs83] Asano, T.; Asano, T.: Minimum Partition of Polygonal Regions into Trapezoids; Proc. 24th IEEE Annual Symposium on Foundations of Computer Science, 1983, 233-241.

[BaGo88] Barry, P.J.; Goldman, R.N.: De Casteljau-type subdivision is peculiar to Bézier-Curves; Computer-aided Design Vol. 20, 1988, 114-116.

[BBGS86] Batory, D.S.; Barnett, J.R.; Garza, J.F.; Smith, K.P.; Tsukuda, B.C.; Twichell, B.C.; Wise, T.E.: GENESIS: An Extensible Database Management System, Technical Report, TR-86-07, Department of Computer Science, University of Texas at Austin, March 1986.

[BDHO89] 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.

[BKSS90] Beckmann, N.; Kriegel, H.P.; Schneider, R.; Seeger, B.: The R*-tree: An efficient and robust access method for points and rectangles; Proc. 1990 ACM SIGMOD Int. Conf. on Management of Data, Atlantic City, USA, May 1990, 322-331.

[CDFG86] Carey, M.; DeWitt, D.J.; Frank, D.; Graefe, G.; Muralikrishna, M.; Richardson, J.E.; Shekita, E.J.: The Architecture of the EXODUS Extensible DBMS; Proc. Int. Workshop on Object-Oriented Database Systems, 1986, 52-65.

[FaMy92] 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).

[FaRa88] Farouki, R.T.; Rajan, V.T.: On the numerical conditions of algebraic curves and surfaces, 1. implicit equations: Computer Aided Geoemtric Design, Vol. 5, 1988, 215-252.

[FeSS84] Ferrari, L.; Sankar, P.V.; Slansky, J.: Minimum rectangular partitions of digitized blobs; Computer Vision, Graphics and Image Processing, Vol. 28, 1984, 58-71.

[FrGr82] Freist, C.; Granow, R.: 'Similarity search using cluster analysis'; VDI-magazin,Vol. 124, Part 1: No. 11, Part 2: No 13, VDI Düsseldorf, 1982 (in German).

[GoRi74] Gordon, W.J.; Riesenfeld, R.F.: B-spline curves and surfaces; in: Barnhill, R.E.; Riesenfeld, R.F. (eds): Computer Aided Geometric Design; Academic Press, 1974, 71-87.

[GKMK82] Graham, S.L.; Kessler, P.B.; McKusick, M.K.: gprof: A Call Graph Execution Profiler; Proceedings of the SIGPLAN´82 Symposium on Compiler Construction, SIGPLAN Notices, Vol. 17, Nr. 6, Juni 1982, 120-126.

[Günt88] Günther, O.: Efficient structures for geometric data management; Lecture Notes in Computer Science, Vol. 337, Springer Verlag, Berlin, 1988.

[Güti89] Güting, R.H.: Gral: An extensible relational database system for geometric applications; Proc. 15th Int. VLDB Conference, Amsterdam, 1989, 33-44.

[Heep92] Heep, S.: 'Queries on Geometry Data in CAD Database Systems'; Ph.D.Thesis, University of Bremen (in German).

[HoLa89] Hoschek, J.; Lasser, D.: 'Basics of Geometric Data Processing'; Teubner 1989 (in German).

[Katz85] Katz, R.: Information Management for Engineering Design; Springer, Berlin Heidelberg, 1985.

[KeSa85] 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, 197-212.

[KrHe89] Kriegel, H.-P.; Heep, S.: 'Efficient Boundary representation of polygonal objects on secondary storage'; Abstracts of the 6th Workshop on "Computational Geometry", Siegen, 1990 (Internal report in German).

[KHHS91] 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.

[KrHS91] Kriegel, H.-P.; Horn, H.; Schiwietz, M.: The Performance of Object Decomposition Techniques for Spatial Query Processing; Proc. SSD´91, Zürich, August 1991; in: Günther, O.; Schek, H.-J. (eds): Advances in Spatial Databases; published as LNCS 525, Springer, 257-277.

[KSSS89] 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.

[LiTL89] Liou, W.T.; Tan, J.J.M.; Lee, R.C.T.: Minimum partitioning simple rectilinear polygons in O(n*log log n)-time; ACM, 1989.

[Oost90] Oosterom, P.J.M.: Reactive Data Structures for Geographic Information Systems; Ph.D. Thesis, Department of Computer Science at Leiden University, The Netherlands, 1990.

[OrMa86] 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.

[Same90] Samet, H.: The Design and Analysis of Spatial Data Structures; Addison-Wesley, Reading, MA, 1990.

[SKSH89] 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.

[Widm91] Widmayer, P.: 'Data Structures for Spatial Databases'; in: Vossen, G.; Witt, K.-U. (eds.): Future Trends in Database Systems, Oldenbourg, 1991 (in German).

[WSSH87] Wilms, P.F.; Schwarz, P.M.; Schek, H.-J.; Haas, L.M.: Incorporating Data Types in an extensible Database Architecture; Internal Report, IBM Research Division, Almaden Research Center, San Jose, 1987.