Comparison of Approximations of Complex Objects Used for Approximation-based Query Processing in Spatial Database Systems

Thomas Brinkhoff, Hans-Peter Kriegel and Ralf Schneider

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

Abstract

The management of geometric objects is a prime example of an application where efficiency is the bottleneck; this bottleneck cannot be eliminated without using suitable access structures. The most popular approach for handling complex spatial objects in spatial access methods is to use their minimum bounding boxes as a geometric key. Obviously, the rough approximation by bounding boxes provides a fast but inaccurate filter for the set of answers to a query. In order to speed up the query processing by a better approximation quality, we investigate six different types of approximations. Depending on the complexity of the objects and the type of queries, the approximations 5-corner, ellipse and rotated bounding box clearly outperform the bounding box. An important ingredient of our approach is to organize these approximations in efficient spatial access methods originally designed for bounding boxes.

1 Introduction

Geographic Information Systems (GIS) are characterized by massive volumes of data, both spatial and non spatial. In a geographic database, the number of objects goes easily into the millions [3]. Therefore, the data are stored on a secondary storage medium. To achieve efficient and persistent storage of these objects, a GIS is based on a *spatial database system*. A geometric object is characterized by a geometric component that determines shape and position of the object in space. In most geographic/cartographic applications spatial objects are two-dimensional built on points, lines and polygons as the basic primitives. As shown in [12], non-point objects can be well represented by *simple polygons with holes.*

The management of geometric objects, for instance in cartography, is a prime example of an application where efficiency is the bottleneck; this bottleneck cannot be eliminated without using suitable access structures. In a spatial database system, the objects are organized and accessed by *spatial access methods (SAMs)*. Commonly, these objects are modelled by simple polygons with holes which are extremely irregular and vary in the number of points as well as in the number of holes. Hence, SAMs are not able to organize such complex polygons directly. *Approximations* maintain the most important features of the objects (position and extension) and therefore, they are used as geometric keys in a spatial access method.

The smallest aligned rectangle enclosing an object, the *minimum bounding box*, is the most popular approximation. Spatial access methods map objects on a secondary storage medium with fixed size blocks that can be addressed directly. Therefore, adjacent objects are combined in *regions* that are associated to blocks. On the one hand, typical block sizes are between 1 and 8 kbytes. On the other hand, data files with object sizes of 3 to 10 kbytes in the average are not uncommon [6]. Therefore, only a few exact object descriptions can be stored in one block. Obviously, clustering a large set of spatially adjacent objects physically on one block can only be achieved on the level of approximations because approximations are short in their description and can reference to the exact object representations [28].

There are several other reasons for using approximations. Often the computation of an estimated value, quickly determined on the basis of approximations, is sufficient for preparing and processing geometric queries. Furthermore, many spatial queries can be answered, partially or completely, using approximations. The *approximation-based query processing* is performed in two steps [18]: First, the *filter step* identifies a superset of the response set by using approximations as a geometric key. Second, the *refinement step*, inspects the exact representation of each object of the superset. In this step, complex and CPU-time intensive algorithms are used for deciding whether the objects fulfil the query condition. Obviously, the performance of approximation-based query processing depends on which type of approximation is chosen for the objects. A suitable object approximation is crucial for both, reducing the size of the candidate set and identifying answers on the basis of approximations.

The commonly used minimum bounding boxes are rather rough and inaccurate object approximations. However, they profit from a very efficient organization using spatial access methods designed for bounding boxes. Several other approaches have been suggested to maintain non-rectangular approximations by adequate spatial access methods, e.g. circles in the sphere tree [17], or convex polygons in the cell tree [8], polyhedra-tree [11] or P-tree [21]. From our point of view, these structures are rather complicated such that the processing of operations and queries as well as insertions is very CPU-time intensive (see also [17]).

Our approach is first to use approximations which are suitable for query processing on geometric objects and second to manage these approximations in spatial access methods originally designed for bounding boxes. Thus, on the one hand we use robust and efficient spatial access methods and on the other hand we improve the approximation-based query processing essentially. Using this approach, two important questions arise:

- Which type of approximations is suitable for geometric objects?
- How efficient is the management of non-rectangular approximations in a spatial access method originally designed for bounding boxes?

In the rest of this paper, we examine which type of approximations is suitable for an approximation-based query processing in spatial database systems. First, we introduce some relevant classes of approximations. Then, in section 3 empirical results are presented that investigate the suitability of the different types of approximations for geographic applications. Section 4 presents our approach and an empirical performance evaluation of approximation-based query processing. In particular, we discuss and investigate in detail the interaction of non-rectangular approximations organized in a SAM originally designed for bounding boxes. The paper concludes with a summary pointing out the main contributions and test results and giving an outlook to future activities.

2 Approximation-based query processing

In this section, we introduce a query processing mechanism for managing large sets of complex polygonal objects.

2.1 Two-step query processing

From the literature no standard set of spatial queries fulfilling all requirements of spatial applications is known [23]. Thus, it is necessary to provide a small set of basic spatial queries which are efficiently supported by the database facilities. Application specific queries, e.g. presented in [17], typically using more complex query conditions, can be decomposed into sequences of such basic spatial queries. We propose the following set of basic spatial queries:

- *Point query:* Given point p, find all objects containing p.
- *Window query:* Given an aligned window w, find all objects intersecting w.
- *Region query:* Given a simple polygon with holes (SPH) p, find all objects intersecting p.
- *Enclosure query:* Given a SPH p, find all objects which are contained by p.
- *Containment query:* Given a SPH p, find all objects containing p.
- *Nearest neighbour query*: Given a point or SPH p, find the nearest object(s) to p.
- *Spatial join*: Given 2 sets S and S' of SPHs. Find all pairs (O, O') of intersecting objects where $O \in S$ and $O' \in S'$.

The *approximation-based query processing* (see Fig. 1) is performed in two steps [18]: The first step, the so-called *filter step*, examines the approximations of the objects and provides a fast but inaccurate filter for the response set. Using approximations, the filter step identifies a superset of the response set. Since approximations provide no exact object representations, the filter step does not exactly evaluate the query. The filter step yields a set of candidates which may fulfil the query. More exactly, the set of candidates contains all answers to the query and additionally it may contain some objects not belonging to the response set *(false hits)*. Based on the filter step, for some objects we can already decide that they belong to the response set (see later on the example of Fig. 2). In the second step, called *refinement*, the exact representations of these candidates have to be inspected. In this step complex and CPU-time intensive algorithms known from the field of computational geometry are used for deciding which of the candidates fulfil the query condition. Obviously, the performance of approximation-based query processing depends on the quality of the approximation chosen for the objects. A suitable object approximation is crucial for both, reducing the size of the candidate set and identifying answers on the basis of approximations.

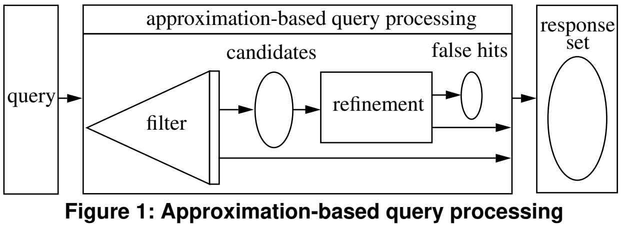

Figure 1: Approximation-based query processing

Fig. 2 depicts an example for the two-step query processing. Assume, the simple polygons are approximated by minimum bounding boxes. In the specified region query we search for all objects intersecting the shadowed query polygon. In the filter step all boxes are determined that intersect the query region. The objects a, b, c, and e belong to the candidate set. Furthermore, at this point we can already decide that b belongs to the response set. In the refinement step, we have to check whether the exact representation of the objects a, c, and e really intersect the query region. In this step, the object a is additionally identified as a correct answer of the query.

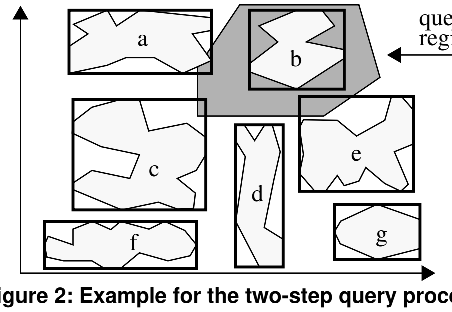

Figure 2: Example for the two-step query processing

From this schema of approximation-based query processing the following criteria can be derived:

- In the filter step, large sets of approximations have to be searched for and have to be tested against the query condition. Therefore, the approximations should be simple in order to yield fast search and test algorithms *(simplicity criterion).*
- The performance of the refinement step depends on the number of refined objects as well as on their complexity. In [13] it is shown that query processing of complex spatial objects is dominated by the complex and time consuming computational geometry algorithms. Therefore, the primary goals for efficient query processing are first to determine for as many answers as possible their membership to the response set and second to reduce the false hits. For that the accuracy of the filter step has to be improved by increasing the quality of the approximations with respect to the original objects *(quality criterion).*
- The time spent for constructing the approximation is a further criterion to evaluate the suitability of a special type of approximation used in query processing. Since the construction of an approximation is only necessary when the object is inserted or updated in the database, higher overhead for the construction may be justified.

2.2 Quality of approximations

In the literature, several alternatives are proposed to measure the quality of approximations. For example, in [1] several metrics are presented and investigated to compute the distance for polygonal objects. However, in our application we are interested in an improvement of the accuracy of the filter step. The accuracy of the filter step is maximized by minimizing the deviation of the approximation from the original object. This deviation is measured by the false area of the approximation which may be positive or negative with respect to the original object. Therefore, we propose as a measure of quality the following parameter called approximation quality G_{Appr}.

Definition 1: ***Approximation quality*** [22]

$$\frac{A(O) + A(O \setminus Appr[O]) + A(Appr[O] \setminus O)}{A(O)} \cdot 100\%$$

A(O) denotes the area of a spatial object *O*, *Appr(O)* describes an approximation of *O* and the symbol \ corresponds to the geometric difference. In the enumerator, the object area and the false areas of the approximation with respect to the inside and outside of the object are summed up. The value of the approximation quality is standardized by division through the area of the object and presented in per cent. An approximation quality of 100% occurs if and only if the approximation is congruent with the original object. An approximation quality of c≤100% describes a (c-100)% false area of the approximation with respect to the area of the original object.

2.3 Classification of approximations

Approximation techniques can be divided into three classes [22]:

- *conservative* approximations
- *progressive* approximations
- *generalizing* approximations

An approximation is called conservative iff any point inside the contour of the original object is also contained in the conservative approximation. Analogously, an object is progressively approximated if the point set of the approximation is a subset of the point set of the object. A generalizing approximation tries to simplify the object contour (e.g. by reducing the number of vertices). Generally, there is no topological relation between the generalizing approximation and the original object, i.e. neither is the object completely covered by the approximation nor is the approximation completely contained in the object. Fig. 3 depicts examples of the different classes of approximations.

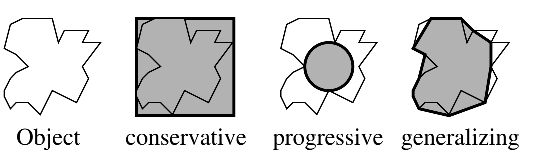

Figure 3: Examples for the three classes of approximations

As already mentioned, in a spatial database approximations should support the two-step query processing incorporating the filter step and the refinement step. Generalizing approximations are useless for query processing because of the missing topological relation, but they are helpful for other applications, e.g. the presentation of maps. By enclosing the object in a conservative approximation, we achieve the following saving: when the low-cost search for the approximation fails, we know that the expensive search for the original object must also fail. By using a progressive approximation, we achieve the analogous effect. Successful searches for progressive approximations yield a definite answer, only failed searches must be followed by the costlier search for the original object. The spatial queries described in section 2.1 are characterized by a high selectivity. Therefore, in the following we discuss the conservative approximations in detail. A progressive approximation may be used in addition to the conservative approximation in order to increase the number of objects which are identified as correct answers already in the filter step.

Conservative approximations can be grouped into *convex* and *concave* conservative approximations. Following the simplicity criterion, we restrict our considerations to the class of convex approximations because computational geometry algorithms for concave polygons are considerably more time intensive than those for convex polygons.

2.4 Approximations

In the following, we will introduce six selected convex conservative approximations which in our point of view are suitable for spatial objects:

Minimum bounding box (MBB): The MBB is the smallest aligned rectangle enclosing an object. It can be represented by four parameters that correspond to the coordinates of the lower left and the upper right vertices of the MBB. The MBB-approximation is unique and translational invariant but not rotational invariant. It can be computed with a simple linear algorithm which determines the minimum and maximum extension of the object in x- and y-direction.

Rotated minimum bounding box (RMBB): If we give up the restriction to align the MBB to the axes and allow rotations, the approximation quality of the MBB can be improved. Obviously, the resulting rotated MBB (RMBB for short) is additionally rotational invariant. It can be represented by the four parameters of the bounding box and one more parameter that correspond to the performed rotation.

Minimum bounding circle (MBC): The circle needs three describing parameters (x-coordinate and y-coordinate of the center of the circle and the radius). The approximation with the minimum bounding circle is unique, translational invariant and rotational invariant. In our tests we used a randomized algorithm with an expected linear complexity [30] which is based on Seidel's optimal linear algorithm [26]. A comparison of further methods can be found in [4].

Minimum bounding ellipse (MBE): Two-dimensional ellipses are determined by 5 parameters. Usually, the ellipse is described by a matrix $\begin{bmatrix} A & B \\ B & C \end{bmatrix}$ and the center $P = (p_1, p_2)$. The center is the intersection of the semiaxis of the ellipse and describes the position of the ellipse in the plane. The MBE-approximation is unique, translational and rotational invariant. A deterministic $O(n^2)$-algorithm for computing the minimum bounding ellipse is presented in [19]. In our tests we used Welzl's randomized algorithm [30] which has an expected linear complexity.

Convex hull (CH): An obvious and popular approximation for simple polygons is the convex hull. The construction of the convex hull of a set of points is one of the best understood problems in computational geometry. We used Graham's simple scan-algorithm [7] with time complexity O (n log n). The construction of the convex hull of a simple polygon is possible in O (n) time [15]. The required storage for the convex hull approximation is determined by the complexity of the object geometry and may vary from object to object.

Minimum bounding n-corner (n-C): To obtain a predefined constant storage requirement, it is possible to compute the minimum bounding n-corner starting from the convex hull of the polygon. An algorithm to construct the n-C-approximation was proposed in [5] for the first time. A detailed description and investigation of this algorithm is presented in [21].

Fig. 4 visualizes the selected approximations using Great Britain as an example. These approximations differ especially in the approximation quality and storage requirement. The convex hull has the best approximation quality. The minimum bounding circle has the lowest storage requirement. A first step of an analytical and qualitative evaluation of approximations can be found in [22].

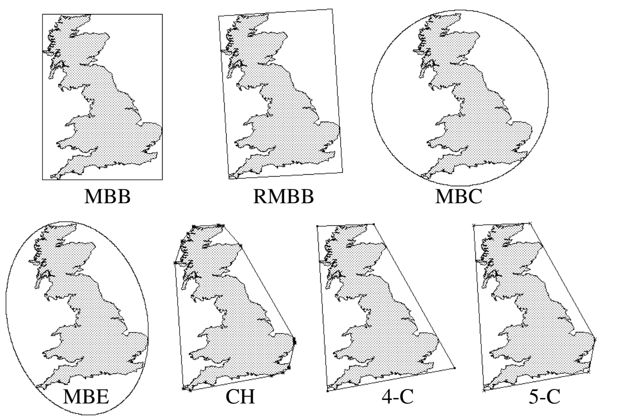

Figure 4: Different presented approximations

3 Empirical comparison of approximations

We have approximated simple polygons with holes of various real maps to get expressive and realistic results on the quality and the storage requirement of the selected approximations. To be as general as possible, we used maps from different sources with different resolutions. The data files contain natural objects such as islands and lakes as well as administrative areas such as counties. Fig. 5 depicts the maps and Table 1 lists their characteristics. N is the number of the polygons in the maps, $m_{\varnothing}$ is the average number of vertices of a polygon, m_{min} and m_{max} the minimum and the maximum number of vertices of a polygon occurring in the map respectively.

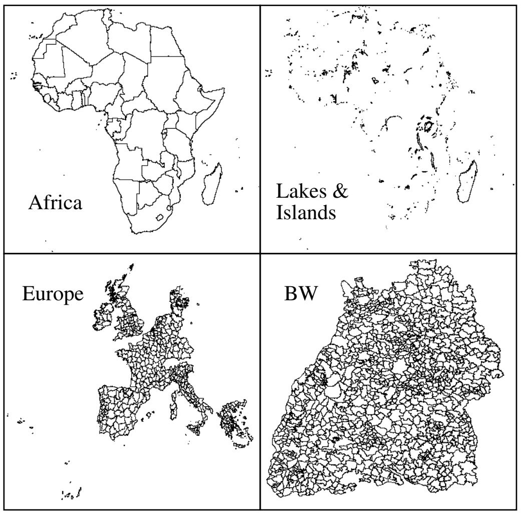

Figure 5: The analysed maps

map	N	$m_{\varnothing}$	m_{min}	m_{max}	source
Europe	809	84	4	869	[29]
BW	1315	572	6	3104	[14]
Lakes & Islands	1253	120	5	9106	[6]
Africa	104	2769	28	16324	[6]

Table 1: Characteristics of the analysed maps

The computed approximation qualities G_{Appr} for the different approximations and maps are presented in Table 2. $G_{Appr\ \varnothing}$ is the average approximation quality of the different maps. The required storage for the approximations is listed in bytes in the row denoted by S_{Appr}. 4 bytes are assumed per parameter.

G_{Appr} (%)	CH	5-C	4-C	RMBB	MBE	MBB	MBC
Europe	125	133	144	163	170	193	213
BW	129	136	149	167	174	193	205
Lakes	119	128	139	157	160	197	230
Africa	123	133	144	161	168	189	211
$G_{Appr\ \varnothing}$(%)	124	133	144	162	168	193	215
S_{Appr}		40	32	20	20	16	12

Table 2: Approximation quality in per cent

The results show that all approximations have an approximation quality which is nearly independent from the tested maps. Obviously, the more parameters are available for the representation of an approximation, the better is the approximation quality. The approximation quality of the 5-corner is nearly the same as that of the convex hull. The storage requirement of convex hulls varies extremely and is on the average much higher than the storage requirement of the other approximations (Europe 104 bytes, BW 184 bytes, Lakes & Islands 136 bytes and Africa 248 bytes). The rotated MBB improves the approximation quality by 16% compared to the MBB, although only one additional parameter is used; the RMBB approximates 4% better than the minimum bounding ellipse which has the same storage requirement. The 5-corner needs 6 additional parameters compared to the MBB paying off in a of 31% gain over the average approximation quality of the MBB.

The main advantages of an improvement of the approximation quality with respect to query processing are:

- Performing a point query, the probability to obtain a false hit in the filter step is proportional to the false area of the approximation normalized by the area of the original object. In other words, the worse the approximation quality is, the more often the exact object representation is unnecessarily loaded into main memory and tested with costly computational geometry algorithms. Therefore, the saving of accesses to the exact object representation is characterized by the difference of the approximation qualities.

  Table 1 points out that spatial objects occupy several blocks (block sizes of 1 to 8 kbyte are common). Therefore, object and approximation are not stored together in one block. This implies that each access to the exact object representation needs at least one, but on the average several block accesses. The relative difference of the number of accesses when using different approximation techniques is calculated by the ratio of the approximation qualities.
- The time for refinement dominates the time for the filter step [13]. Every improvement of the approximation quality results in an essential gain for the refinement step and thus in query processing time. For point queries, the gain in query processing time is proportional to the gain in approximation quality. Also for queries such as the region query, the enclosure query and the containment query, the access frequency to the exact representation depends of the approximation quality. Efficiency increases also for nearest neighbour queries and the spatial join. In particular, for the spatial join a high gain is expected because approximated objects are tested in pairs. Therefore, the improvement in approximation quality pays off quadratically in the total time for query processing.

4 Approximations stored in SAMs

The last section has shown the potential of improving total query processing time when approximations with a high approximation quality are used in query processing. In a spatial database system, such approximations are efficiently organized by spatial access methods. Some data structures were suggested that are designed for special types of approximations. Examples are the sphere tree [17] for circles as well as the cell tree [8], the polyhedra-tree [11] and the P-tree [21] for convex polygons. From our point of view, these structures are rather complex such that the processing of operations and queries as well as updates is very CPU-time intensive (see also [17]). Such access methods have to organize circles or convex polygons in their directory. This is more difficult than organizing simple aligned bounding boxes.

Our approach is to manage the different types of approximations in spatial access methods originally designed for bounding boxes. In several performance analyses and comparisons these spatial access methods have proven their robustness and efficiency. In this section, we demonstrate that also the other approximations are efficiently managed by such access methods.

4.1 Spatial access methods

Spatial access methods are designed for a dynamic organization of geometric data. To store and organize the data on secondary storage, approximations are grouped into *regions*. To perform spatial queries efficiently, spatially adjacent approximations are clustered into one region *(local order preservation* [28]*)*. A region corresponds to a physical block on secondary storage. Spatial access methods are implemented as trees or hashing schemes. In the past few years, a lot of spatial access methods were developed. Most of them use minimum bounding boxes as spatial approximation, e.g. the grid file [16], the buddy tree [25], and the R-tree [9]. A survey can be found in [20].

In [24] three techniques are presented for the organization of complex spatial objects in SAMs: *Clipping* partitions the data space into disjoint regions. The objects are associated with each of the regions they intersect and thus one object or a pointer is stored in each of the corresponding blocks. In general, the technique of clipping may degrade query performance substantially, since the number of objects (copies) to be stored increases, which in turn increases the number of regions, thereby increasing the number of copies, a vicious circle [27]. The *transformation technique* views an object as a point in some parameter space. Since transformations do not preserve the spatial neighbourhood of objects in the original space, and since the distribution of parameter points tends to be extremely skewed, the query efficiency tend to be quite low [28]. The third technique is *overlapping regions*. In this technique each object is assigned to exactly one region. Overlapping regions may induce a higher query time because there may exist several regions potentially containing the searched object. However, the R*-tree has demonstrated that it is possible to organize spatial objects such that the overlap of the regions in the directory is extremely small (see Fig. 6).

The *R*-tree* [2] is based on the well-known R-tree [9]. It manages a set of bounding boxes by grouping them recursively into regions which are described by bounding boxes themselves. The R*-tree uses a sophisticated strategy to split regions. The strategy is based on three design paradigms:

- The dead space in a region, i.e. the area not covered by any bounding box, is minimized.
- The overlap of the regions is minimized.
- The perimeter of a region is minimized.

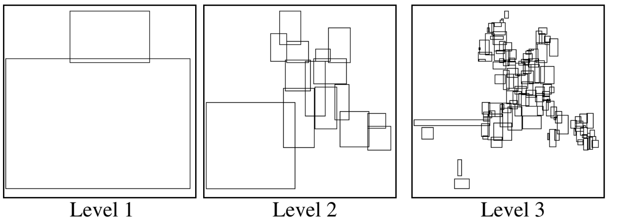

Figure 6: R*-tree partitioning of the map 'Europe'

The R*-tree is a simple, robust, and efficient spatial access method. This has been demonstrated in tests [2] and in a comparison with other access methods [10]. Algorithmically it is simple to organize bounding boxes and to search through the directory of the R*-tree testing bounding boxes. Therefore, it is an interesting approach to use the R*-tree for organizing different types of approximations.

4.2 Organization of approximations in the R*-tree

In section 3, we have pointed out the fruitful effects of approximations to query processing performance. In this section we present the effects to the R*-tree when managing different types of approximations:

- In this case, sets of spatially adjacent approximations are grouped into one region *(data block)* instead of MBBs. A data block is described by a MBB which can be organized by the R*-tree in the normal way.
- More complex approximations require more storage. This increased storage requirement determines the maximum number of entries stored in a data block which influences the performance of the spatial access method.
- In general, the selected approximations have a larger x-extension and y-extension than the MBB. As a consequence, this extension influences the extension of the MBBs describing data blocks of the R*-tree.

Experimental setup

Since spatial access methods are often heuristic data structures, analytical performance evaluations are hardly possible or are restricted to (rarely occurring) uniform distributions. Thus, we investigate the performance of approximation-based query processing in an empirical comparison with queries performed on real cartography data. For our comparison, we select the R*-tree because of its mentioned characteristics simplicity, robustness and efficiency.

The following tests examine R*-trees which store 33,204 objects representing a map of administrative regions (counties) of Europe. For geographic applications, it is realistic to assume that the directory does not exceed height 3. Using block sizes of 4 kbytes, a R*-tree of height 3 can manage more than 300 millions of RMBB-approximations. To model the behaviour of large directory trees, we choose fairly small directory blocks, with a maximum fan out of 25 per directory block. Under this assumption the 33,204 objects generate an R*-tree of directory height 3 using MBB-approximations. The R*-trees use an LRU-buffer with a capacity of 35 directory blocks. From our point of view the buffer size should be larger than the maximum fan out. In our tests the factor between maximum fan out and buffer is 1.4. Larger buffer sizes yield similar results.

Query profile

In the following, we want to compare the performance of the R*-tree storing different types of approximations. For comparing the performance of different R*-trees, we need a parameter to indicate the differences of the structures of the R*-trees which are significant for the performance. R*-trees differ in their sizes as well as in the shape, the area and the overlap of their directory regions. These parameters are presented in an integrated way when uniformly distributed point queries are performed on the R*-trees. Therefore, we have developed a query profile which corresponds as far as possible to real applications: Typical sequences of point queries contain query points hitting an object as well as of points not hitting an object. The objects used in our tests cover only part of the data space. Therefore, uniformly distributed query points over the complete data space do not correspond to real applications, because the average time to answer the queries would be dominated by the query points not hitting an object: To avoid this effect, we define to the covered data space D which is covered by the objects a *query space $Q_\varepsilon(D)$* (short: Q) where:

$$Q_\varepsilon(D) = Q = \{P \mid \exists P' \in D: (\|P, P'\| \le \varepsilon)\}$$

For this query space we can control the ratio of points not hitting an object. In our tests we performed 10,000 point queries uniformly distributed over the query space Q. The parameter p_{succ} describes the probability that a point of Q lies in D. Fig. 7 shows an example for $p_{succ} = 0.41$.

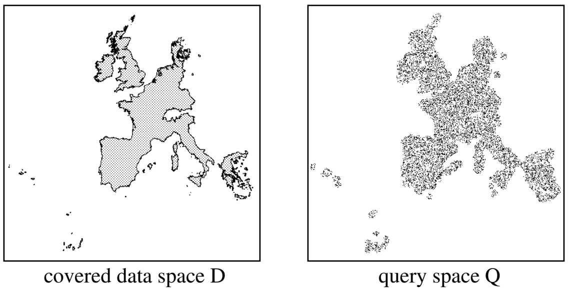

Figure 7: Covered data space and query space

The MBB-approximation is the most popular approximation used in spatial access methods. In the following, we normalized the results of the comparisons in order to show the differences to the traditional approach using the MBB-approximation. The absolute values can be derived from Table 3 where according to p_{succ} the number of directory block accesses (Acc_{dir}) and data block accesses (Acc_{data}) per point query are depicted when MBB-approximations are used. The height of the directory of the tree is 3. Since we want to investigate the time spent in the spatial access method, accesses to the exact object representation during the refinement step are not considered.

	$p_{succ} = 1.0$	$p_{succ} = 0.72$	$p_{succ} = 0.59$	$p_{succ} = 0.41$
Acc_{dir}	1.12	1.04	0.92	0.67
Acc_{data}	1.76	1.39	1.16	0.81

Table 3: Averaged block accesses for a MBB query

Higher storage requirement

More complex approximations have higher storage requirements than the MBB approximation (see Table 2). Therefore less approximations can be stored in one data block. Thus with increasing storage requirement, the number of data and directory blocks is increasing. The influence of higher storage requirement to the performance of the R*-tree is depicted in Table 4. The number of block accesses are normalized on the basis of the MBB-approximation (100%). S denotes the storage requirement for the approximation measured in bytes. S consists of the storage requirement for the representation of the approximation (4 bytes per parameter) and for the pointer to the exact representation (4 bytes).

S (in bytes) Approximation	$p_{succ} = 1.0$		$p_{succ} = 0.59$	
	Acc_{dir} (%)	Acc_{data} (%)	Acc_{dir} (%)	Acc_{data} (%)
16 MBC	92	102	94	103
20 MBB	100	100	100	100
24 MBE/RMBB	127	108	125	108
36 4-C	151	106	141	102
44 5-C	163	108	149	104
108 CH	261	113	231	106

Table 4: Block accesses for different storage requirements

The number of data block accesses varies only slightly. However, Acc_{dir} clearly depends on S; the number of directory block accesses depends roughly logarithmically on the storage requirement. For S = 108 byte (the average storage requirement of convex hulls in the Europe map), the height of the directory was varying from 3 to 4. Therefore, the block accesses in that row are essentially higher and not comparable to the other rows.

Higher extension of approximation

The approximations of a set of objects are grouped into one region. Except for the convex hull, all of the selected approximations have higher x- and y-extension than the MBB-approximation. This yields a higher *area extension* A_{Appr} which is the product of the x-extension and the y-extension. We tested the influence of higher area extensions on the performance of the R*-tree. To get a feeling which area extensions are realistic, we computed the approximation extensions of the real data presented in section 3. Table 5 depicts the average area extension normalized to the MBB-approximation which is set to 100%.

	5-C	4-C	RMBB	MBE	MBC
A_{Appr} (%)	121	144	151	122	142

Table 5: Area extensions of the approximations

Table 5 shows that on the average the area extension of the 5-corner and of the minimum bounding ellipse is about 22% higher than the area extension of the MBB. The area extensions of the 4-corner, of the RMBB and of the circle are 42% to 51% higher than the area extension of the MBB.

The area extension determines the size of the region which describes the data block. Such a region is defined by the MBB of the approximations contained in the associated block. The higher the area of these MBBs, the worse is the performance of the R*-tree, because more queries will access to an extended region. Table 6 depicts the influences of area extensions to the performance. Acc_{dir} is the number of directory block accesses and Acc_{data} the number of data block accesses. Both are normalized to the accesses when MBBs are used which again are set to 100%

A (%)	p_{succ} = 1.0		p_{succ} = 0.59	
	Acc_{dir} (%)	Acc_{data} (%)	Acc_{dir} (%)	Acc_{data} (%)
120	107	109	106	110
140	109	120	106	121
160	114	129	113	130

Table 6: Block accesses for different area extensions

Independent from the query files, we can observe an increase of block accesses which is proportional to the increase of the area extensions. The increase of the directory block accesses is 20% of the extension increase and that of data block accesses 50% of the increase in area extension.

Storing different types of approximations in the R*-tree

In the above subsections we examined higher storage requirements and higher extensions of the approximations to investigate their influence to the performance of the R*-tree. In the following we analyse the performance of R*-trees storing the selected approximations.

Table 7 depicts the performance of the different types of approximations for one point query. Acc is the number of block accesses in per cent normalized to the MBB-approximation. Acc_{Δ} is the necessary number of additional block accesses for a point query not including the savings in the refinement step. For an easier interpretation of the results, we present in Table 7 once more the storage requirement (S), the area extension (A_{Appr}) and the average approximation quality (G_{Appr}) of the selected approximations.

Appr	S (Byte)	A_{Appr} (%)	G_{Appr} (%)	p_{succ} = 1.0		p_{succ} = 0.59	
				Acc (%)	Acc_{Δ}	Acc (%)	Acc_{Δ}
CH	108	100	124	170	2.03	162	1.29
5-C	44	121	133	144	1.29	139	0.80
4-C	36	144	144	143	1.25	138	0.80
RMBB	24	151	162	130	0.85	128	0.59
MBE	24	123	168	118	0.51	115	0.31
MBC	16	142	215	105	0.13	106	0.12

Table 7: Block accesses for different approximations

The following results are derived from Table 7:

- The values show that the circle is not a suitable approximation; it approximates over 20% worse than the MBB (G_{MBB} = 193%) and needs also more block accesses than the MBB.
- The other approximations (except for the convex hull) need 0.5 to 1.2 (for p_{succ} = 1) and 0.3 to 0.8 (for p_{succ} = 0.59) block accesses more than the MBB. However, these additional accesses are generally more than compensated by the accesses saved in the refinement step resulting from the better approximation qualities. In other words, the decreasing number of false hits more than compensates the cost caused by the higher storage requirement and higher area extension of the approximations.
- The approximation quality of the 5-corner (133%) is almost as good as that of the convex hull (124%). But the convex hull needs considerably more storage (the factors are between 2.45 for Europe and 5.7 for Africa.) Thus, the 5-corner approximation outperforms the convex hull approximation, except for very complex objects.
- In the refinement step, we can achieve a substantial performance gain using approximations with a very high approximation quality. The more costly the refinement step is, the more worthwhile is the usage of approximations with a high approximation quality. In queries with an expensive refinement step, the 5-corner seems to be the best candidate.
- Large window queries do not considerably profit from a high approximation quality. To support such queries as well as the other queries, the storage requirement of an

approximation should not be essentially higher than the storage requirement of the MBB. Particularly the MBE- and the RMBB-approximations are the best candidates if approximation quality and storage requirement are equally weighted.

Summarizing, we can say that for the approximations 5-corner, RMBB and ellipse the block accesses for searching the R*-tree are increased. However, the reduced number of block accesses to false hits due to higher approximation quality more than compensates this cost. Thus the cost for the filter step is not increased for these approximations when objects of high complexity occur. In [13] we have shown that the time for the refinement step clearly dominates the total time for two-step query processing. Therefore, the total time for two-step query processing is considerably improved because for a lower number of false hits the expensive refinement algorithms are performed. In particular, we can say that for point queries the gain in performance corresponds to the gain in approximation quality of our new approximations over the minimum bounding box.

5 Conclusions

Approximations of objects which are used for a two-step query processing technique should be simple to provide a fast filter (simplicity criterion) and they should have a high approximation quality (quality criterion) to reduce the number of false hits and to identify as many final answers as possible. There are a several convex conservative approximations which meet these requirements. In this paper, we selected and investigated the minimum bounding box, the convex hull, the minimum bounding 4- and 5-corner, rotated boxes, ellipses and circles. The measured approximation qualities were almost independent of the various tested maps. Let us emphasize that this result is very interesting because we used maps from different sources with different resolutions. A clear order of rank turned out. Naturally, the convex hull has the best approximation quality (124 %), followed by the 5-corner (133 %), the 4-corner (144 %), the rotated box (162 %) and the ellipse (168 %). Compared to the minimum bounding box (193 %), clear gains of approximation quality were obtained. The better the approximation quality, the fewer accesses to the exact object representation are necessary. Thus, many procedure calls of costly computational geometry algorithms are avoided in the refinement step.

Using the R*-tree, we have shown that such approximations can efficiently be organized in a spatial access method originally designed for bounding boxes. The simplicity and robustness of the spatial access method is preserved, because in the directory simple bounding boxes are organized and only in the data blocks more complex approximations are stored. Obviously, in the filter step approximations other than the minimum bounding box need more block accesses when traversing the SAM because of their higher storage requirement and their higher extension in x- and y-direction. However, the reduced number of false hits due to higher approximation quality results in a substantial gain in the refinement step by avoiding time intensive computational geometry algorithms. This gain exceeds the slightly higher access costs by far. It turned out that two-step query processing is essentially more efficient when 5-corner or minimum bounding ellipse or rotated minimum bounding box approximations (stored in R*-trees) are used instead of the popular minimum bounding box approximation. 5-corners have an extremely high approximation quality and deviate only 33% from the exact object representation. The minimum bounding ellipse and the rotated minimum bounding box approximation are the best choices when storage requirement and approximation quality are equally weighted.

As indicated the results of the previous section, it is the reduced number of false hits which yields a considerable improvement in total query time when using our new approximations.

Several interesting aspects and possibilities of improvement are not yet included in this work. For instance, by using compressed representations (e.g. discrete descriptions of approximation coordinates) approximations can be stored in a more compact way. Therefore, the increase in storage requirement is more or less negligible. Furthermore, the selected approximations are one-container approximations. Another possibility is to use multi-container approximations [22].

As pointed out before, in approximation-based query processing the spatial join profits especially from a higher approximation quality. In this operation, objects are tested for intersection in pairs and thus the improvement of the approximation quality quadratically influences the performance improvement. Therefore, a more detailed investigation of spatial joins is necessary.

The use of approximations seems to be useful also for three-dimensional objects. Such objects occur in databases for CAD/CAM. Because of the additional dimension, there is a greater scope to improve the approximation quality. Obviously, we need suitable approximation techniques for three-dimensional objects. Not all of the selected approximation techniques for 2D can easily be generalized to 3D. From ellipsoids and rotated 3D boxes we expect a substantial improvement of approximation quality over the aligned 3D box.

Acknowledgements

We would like to thank Emo Welzl and Michael Schiwietz for making their implementations of approximations available to us. We thankfully acknowledge Alexander Braun.

He integrated these and other approximations in a tool for analysing geometric objects and also performed some of the tests in this paper.

References

[1] Alt H., Blömer J.: *'Resemblance and Symmetries of Geometric Patterns',* Data Structures and Efficient Algorithms, Lecture Notes in Computer Science, Vol. 594, Springer, 1992, pp. 1-24.

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

[3] Crain I.K.: *'Extremely Large Spatial Information Systems - A Quantitative Perspective',* Proc. 4th Int. Symp. on Spatial Data Handling, Zurich, Switzerland, 1990, pp. 632-641.

[4] Dörflinger J., Forst W.: *'Approximation by Circles'* (in German), Manuscript, 1991.

[5] Dori D., Ben-Bassat M.: *'Circumscribing a Convex Polygon by a Polygon of Fewer Sides with Minimal Area Addition',* Computer Vision, Graphics, and Image Processing, Vol. 24, 1983, pp. 131-159.

[6] Gorny A.J., Carter R.: *'World Data Bank II: General Users Guide'*, Central Intelligence Agency, Washington, D.C., 1987.

[7] Graham R. L.: *'An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set',* Information Processing Letters, Vol. 1, 1972, pp. 132-133.

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

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

[10] Hoel E.G., Samet H.: *'A Quantitative Comparison Study of Data Structures for Large Line Segment Databases'*, Proc. ACM SIGMOD Int. Conf. on Management of Data,, San Diego, CA., 1992, pp. 205-214.

[11] Jagadish H. V.: *'Spatial Search with Polyhedra',* Proc. 6th Int. Conf. on Data Engineering, Los Angeles, CA., 1990, pp. 311-319.

[12] Kriegel H.-P., Heep P., Heep S., Schiwietz M., Schneider R.: *'An Access Method Based Query Processor for Spatial Database Systems',* Int. Workshop on Database Management Systems for Geographical Applications, Capri, Italy, 1991, in: Geographic Database Management Systems, Springer, 1992, pp. 273-292.

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

[14] Landesvermessungsamt Baden-Württemberg, Stuttgart, 1991.

[15] Melkman A.A.: *'On-line Construction of the Convex Hull of a Simple Polyline'*, Information Processing Letters, Vol. 25, No. 1, 1987, pp. 11-12.

[16] Nievergelt J., Hinterberger H., Sevcik K. C.: *'The Grid File: An Adaptable, Symmetric Multikey File Structure',* ACM Trans. on Database Systems, Vol. 9, No. 1, 1984, pp. 38-71.

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

[18] Orenstein J.A.: *'Redundancy in Spatial Databases',* Proc. ACM SIGMOD Int. Conf. on Management of Data, Portland, USA, 1989, pp. 294-305.

[19] Post M.J.: *'Minimum Spanning Ellipsoids'*, Proc. 16th Annual Symp. on Theory of Computing, 1984, pp. 108-116.

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

[21] Schiwietz M.: *'Organization and Query Processing of Spatial Objects'* (in German), Ph.D. thesis, Institute for Computer Science, University of Munich, 1993.

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

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

[24] Seeger B., Kriegel H.-P.: *'Techniques for Design and Implementation of Efficient Spatial Access Methods',* Proc. 14th Int. Conf. on Very Large Databases, Los Angeles, CA., 1988, pp. 360-371.

[25] Seeger B., Kriegel H.-P.: *'The Buddy Tree: An Efficient and Robust Access Method for Spatial Databases',* Proc. 16th Int. Conf. on Very Large Data Bases, Brisbane, Australia, 1990, pp. 590-601.

[26] Seidel R.: *'Linear Programming and Convex Hulls Made Easy',* Proc. 6th Annual ACM Symp. on Computational Geometry, Berkeley, CA., 1990, pp. 211-215.

[27] Six H.-W., Widmayer P.: *'Spatial Searching in Geometric Databases',* Proc. 4th Int. Conf. on Data Engineering, Los Angeles, CA., 1988, pp. 496-503.

[28] Six H.-W., Widmayer P.: *'Spatial Access Structures for Geometric Databases',* Data Structures and Efficient Algorithms, Lecture Notes in Computer Science, Vol. 594, Springer, 1992, pp. 214-232.

[29] Statistical Office of the European Communities: *'Regions'*, Luxembourg, 1990.

[30] Welzl E.: *'Smallest Enclosing Disks (Balls and Ellipsods),* Paper B91-09, Freie University of Berlin, 1991.