A Raster Approximation for the Processing of Spatial Joins

Geraldo Zimbrão
Systems and Engineering Program
Graduate School of Engineering
Federal University of Rio de Janeiro
PO Box 68511, ZIP code: 21945-970
Rio de Janeiro - Brazil,
zimbrao@cos.ufrj.br

Jano Moreira de Souza
Systems and Engineering Program
Graduate School of Engineering
Federal University of Rio de Janeiro
PO Box 68511, ZIP code: 21945-970
Rio de Janeiro - Brazil,
jano@cos.ufrj.br

Abstract

The processing of spatial joins can be greatly improved by the use of filters that reduce the need for examining the exact geometry of polygons in order to find the intersecting ones. Approximations of candidate pairs of polygons are examined using such filters. As a result, three possible sets of answers are identified: the positive one, composed of intersecting polygon pairs; the negative one, composed of non-intersecting polygon pairs; and the inconclusive one, composed of the remaining pairs of candidates. To identify all the intersecting pairs of polygons with inconclusive answers, it is necessary to have access to the representation of polygons so that an exact geometry test can take place. This article presents a polygon approximation for spatial join processing which we call four-colors raster signature (**4CRS**). The performance of a filter using this approximation was evaluated with real world data sets. The results showed that our approach, when compared to other approaches presented in the related literature, reduced the inconclusive answers by a factor of more than two. As a result, the need for retrieving the representation of polygons and carrying out exact geometry tests is reduced by a factor of more than two, as well.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and / or special permission from the Endowment.

Proceedings of the 24th VLDB Conference
New York, USA, 1998

1 Introduction

The field of spatial databases has recently experienced a very fast development. Many systems represent data having spatial attributes. Such systems, known by the denomination of Geographic Information Systems (GIS's), use spatial databases generally constructed upon relational databases, and need specially developed algorithms to meet their specific requirements. As an example, we should mention the efforts that have been made to create a SQL standard for spatial queries [SQL95, SAI94].

Such systems generally allow us to make spatial queries using some operators similar to those found in relational algebra. Thus, it is extremely important to have an efficient algorithm to perform spatial joins, so that an effective evaluation of the queries can be suitably done. [BRI94] presents the definition of a modular spatial join processor. This model has been frequently cited in many subsequent works. For that reason this work has been developed in such a way to be fully compatible with that processor.

Thus, our purpose is to present a raster approximation for the processing of spatial joins, along with the results obtained through it. The work has been divided in sections, as follows. Section One is an Introduction. Section Two defines the problem itself and shows the need of an efficient accomplishment of the spatial joins. Section Three surveys the related literature and indicates the results on which the present work is based. In Section Four, we present the raster approximation in detail, as well as the results it has provided. Section Five is dedicated to our conclusions and to future developments of this work. Finally, Section Six lists the references we used.

2 Defining the Problem

A significant amount of all the recent research work on databases deals with spatial databases. Although several questions are still open, there is a consensus about some requirements that such databases must meet: for example, a spatial database must offer to spatial data at

least the same facilities that a relational database offers to conventional data. Therefore, a spatial database must give support to *ad-hoc* queries involving the stored spatial attributes. As in relational databases, such queries must be decomposed in smaller, simpler queries, which can be implemented using a small set of spatial operators.

Two of the most frequent operations performed by Geographic Information Systems nowadays are the superposition and intersection of maps. An example of a simple query that falls into this category is presented in [VEE95]: "retrieve all rural areas bellow the sea having soil type equal to sand within three miles of polluted lakes." Several spatial joins involving the thematic planes of soil destination, soil type, pollution and elevation must be made to answer this query. Therefore, we can say that the join operation is of great importance to both spatial and relational databases.

In this work, spatial objects are characterized by possessing at least one attribute describing its spatial extension by means of one or more polygonal lines. It is important to note that the only restriction made relatively to the shape of a spatial object is that it must be closed, although there may be holes in it (for example, lakes) or it may be disconnected (for example, islands).

2.1 Spatial Joins

Drawing another analogy with relational algebra, we can view the spatial join as a subset of the Cartesian product of two sets, A and B, not necessarily distinct, containing, respectively, *m* and *n* elements. This subset is composed of the Cartesian product elements that meet a given spatial predicate. The overlap of spatial objects is of special interest in practical applications. Therefore, this work is concerned solely with the join of intersecting polygons. Nonetheless, as [BRI94] points out, the results obtained considering this predicate can be easily applied to other kinds of predicate.

An initial approach for the processing of spatial joins consists in the application of the nested loops algorithm that came from relational databases. This algorithm consists in confronting each element in set **A** with every element in set **B** to verify whether the condition is satisfied or not. This simple algorithm illustrates the most expensive operations performed during the processing of spatial joins: the transfer of large objects from disk to memory and the polygon intersection test. It is certainly possible to minimize both the number of times that an object is to be read and the number of intersection tests to be performed. This is the aim of our work.

Another approach for the processing of spatial joins is to use indexes previously built on each data set and simultaneously traverse these indexes, searching for polygon intersections. This approach corresponds to a sort-merge approach in relational databases. Typically, an index is composed of two parts: the index structure, that only stores the data keys, and the data structure, that stores the data itself. Thus, a spatial index should store the objects in a spatial structure according to a geometric key [BRI94]. Due to its simplicity, MBR is the most popular geometric key. When we use MBRs, the complexity of a spatial object is reduced to four parameters which retain the most important characteristics of the object: position and extension. Nonetheless, cartographic objects of the real world are very poorly approximated by MBRs.

The current approximation techniques can be divided into three classes [BRI93b]: *conservative*, *progressive* and *generalizing*. An approximation is said conservative if and only if the boundary of the original object is entirely contained in the approximation. It is termed a progressive approximation when all the points pertaining to the approximation are contained in the object. Finally, a generalizing approximation aims at simplifying the objects' boundary, for example reducing the number of vertices. In general, there is no topological relation between the generalizing approximation and the original object, that is to say, neither is the object entirely contained in the approximation, nor is the approximation entirely contained in the object. For that reason, generalizing approximations can not be used for deciding polygon intersection. Examples of conservative approximations are MBRs, and the minimum bounding n-corner (n-C, for example, **5C**), which is the smallest n-corner polygon enclosing an object. Examples of progressive approximations are the maximum enclosed rectangle (**ER**) and maximum enclosed rectilinear line segments (**EL**). More examples of approximations can be found at [BRI93b].

Conservative approximations can identify mainly negative and inconclusive answers. Progressive approximations can identify mainly positive and inconclusive answers. Generalizing approximations can not be efficiently used for deciding about polygon intersection because it leads to many inconclusive answers. So, in order to test polygons' intersections while traversing a spatial index, one should use a progressive and a conservative approximation.

3 Related Works

In a frequently mentioned reference, [BRI94] presents the Multi-step Spatial Query Processor (MSQP), designed to perform spatial joins. In such processor, the evaluation of a spatial join is divided in three steps. Each step is implemented by a module that can be replaced by other equivalent module. Our work is fit to this structure so as to be able to replace (with advantage) one of the steps of MSQP. Next we describe MSQP with more details:

Step 1 - MBR join: First, the Minimum Bounding Rectangle (MBR) is used as an approximation instead of the exact representation of the polygons to compute an approximate spatial join. This step returns what is called a set of candidates, since it contains all the pairs of polygons that belong to the answer plus other pairs that have only MBR intersection;

Step 2 - Filter Application: In this step, approximations that are more accurate than MBR are used for refining the answer, so that some false candidates are eliminated. Moreover, positive answers can also be identified by means of such approximations, without needing to access the exact geometric representation of the spatial objects. *Our work deals primarily with this step*;

Step 3 -- Exact Geometry Intersection: Eventually, all the remaining pairs of candidates are examined at this stage. This step requires access to the exact geometric representation of the spatial objects, and it is frequently the most time-consuming step: it requires CPU time to compute the exact intersection test, and I/O time to read the spatial objects from disk. Nevertheless, the amount of time spent in this step can be reduced using better approximations in the previous stage.

Two great results came from [BRI94]: first, the conclusion that the exact intersection test is the most time-consuming step in the processing of spatial joins; and second, the modular structure proposed for the processor. In other words, it shows us research directions to improve the processing of spatial joins: the MBR join using R*-Tree or other index structure, and the filter step. Improvements done in the 3rd step tend to be of little impact in the total time spent in the processing of spatial joins, since its effects can be canceled by improvements done in earlier steps.

In Figure 1, we summarize other related works that describe algorithms that can be used to replace any of the steps of the MSQP. Next, we shortly comment some of these works and we indicate which step each one of them can replace. This is not a comprehensive list of recently published papers related to this area, but it includes the most important results reached, as far as we know.

Step 1: In this step, [BRI93a] and [BRI94] use R*-tree as spatial index in order to perform a spatial join. [PAT96] and [MIN95, MIN96] present hash based approaches, while [BRI96] presents parallel algorithms for R*-tree based spatial joins. [BER96] suggests another tree, the X-tree, that can also be used to implement this step. Finally, [HUA97b] proposes a new algorithm to traverse the R*-tree.

Step 2: In this step, [BRI94] uses five-corner polygon approximations, and [BRI93b] discusses other approximations that can be used to filter the candidate set. [VEE95] uses approximations that are constructed by rotating two parallel lines around the object. In this work, we use raster approximations of polygons as filters.

Step 3: In this step, [BRI94] uses plane-sweep and sets of trapezoids to compute the exact intersection test, and [HUA97a] uses an algorithm called Symbolic Intersect Detection to reduce the time spent on plane-sweep algorithm.

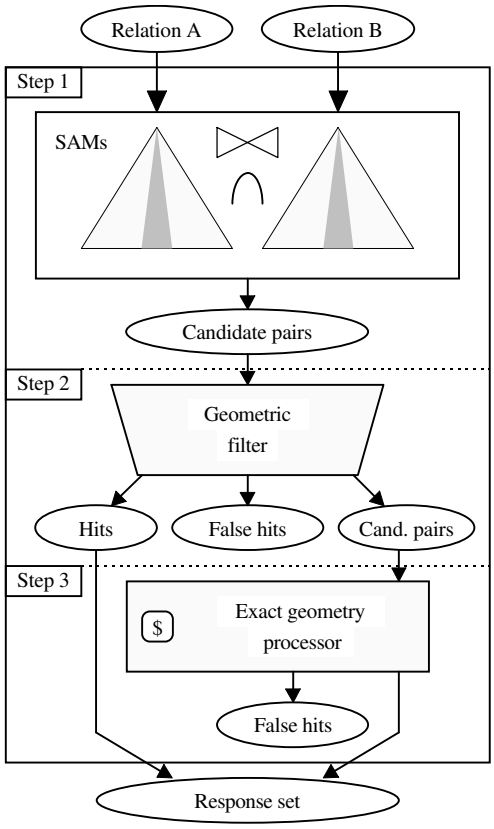


Figure 1: State of art in Spatial Join Processing

Finally, we could mention two other related works: [GÜN93] presents a general model to estimate the cost of spatial joins, and [BRI95] presents a study on complexity of polygons.

4 The Raster Approximation

In this work, we present another kind of filter: the raster approximation of polygons. This approximation combines both progressive and conservative approximations in one single approximation, and it can be used to test if two polygons overlap. Moreover, our approximation can replace [BRI94] approximations with performance gains. Also, it does not interfere with the other steps: the R*-Tree MBR join and the exact geometry intersection test.

The raster approximation of a polygon is a small bit-map of the polygon that uses few colors. This approximation can be viewed as a signature of the polygon. As any other kind of signature, this one should be computed once and stored to be used later. Also, the size of this signature must be kept small enough to be stored in the spatial indexes (for example, R*-Tree) used to provide access to data. In many cases, the bit-maps of two polygons can be used to decide if the real polygons have (or not) intersection area. As we will see in the following sections, there are few cases where the comparison of maps of bits does not lead to a conclusion.

4.1 Raster Signatures of Polygons

Among the results obtained by [BRI94], two are particularly important for our work. The first result is that the step of exact geometric comparison, because of requiring the search of several objects in the disk, is the most time-consuming of all. Thus, efforts should be made to enhance the filters so as to reduce the number of polygons that must be brought to memory. Second, about two thirds of the candidates selected by the step one are pairs of polygons that really intersect each other. Although this proportion may vary from set to set, we have reasons to suppose that it is maintained invariable when we deal with data sets that have no strong correlation with each other, since this happened in random data sets originated from real world data sets. In case of this assumption is confirmed, it would be interesting to improve the filters so as to enhance the detection of intersections during the second step of spatial query processor.

Therefore, we propose an approximation to replace, within the frame of the proposed processor, the **5C** (conservative) and the **ER&EL** (progressive) approximations (both were used in [BRI94]): namely, the four-color raster approximation (**4CRS**). Such approach consists of keeping a raster approximation for each polygon, containing $m \times n$ cells, each one having two bits of information to indicate one of the following possibilities (table 1):

Table 1 - Types of Cell in the 4CRS Approximation

Bit Value	Cell Type	Description
00	*Empty*	The cell does not intersect the polygon
01	*Weak*	The cell contains an intersection of 50% or less with the polygon
10	*Strong*	The cell contains an intersection of more than 50% with the polygon
11	*Full*	The cell is fully occupied by the polygon

As an example, we have the polygons in Figure 2.

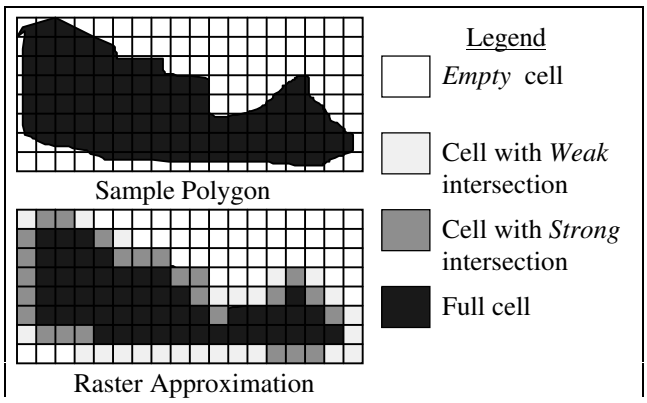


Figure 2 - Cells in a Raster Approximation 4CRS

In order to compare two candidate polygons, we must superimpose their raster approximations in the area where their MBRs intersect each other, having first performed all the required changes in scale. Analyzing each pair of superimposed cells, we have the following possibilities (table 2):

Table 2 - Outcome of cell matching: only three inconclusive cases (*candidate* cells)

Type of *cell* × *polygon* intersection	*Empty*	*Weak*	*Strong*	*Full*
Empty	Discarded	Discarded	Discarded	Discarded
Weak	Discarded	Candidate	Candidate	Accepted
Strong	Discarded	Candidate	Accepted	Accepted
Full	Discarded	Accepted	Accepted	Accepted

If two superimposed cells have each more than 50% of the polygon's area it is obvious that the polygons intersect each other. It is important to notice that if a pair of superimposed cells is accepted, this is a sufficient condition for the pair of polygons they represent to be accepted, too, as depicted in Figure 3

.

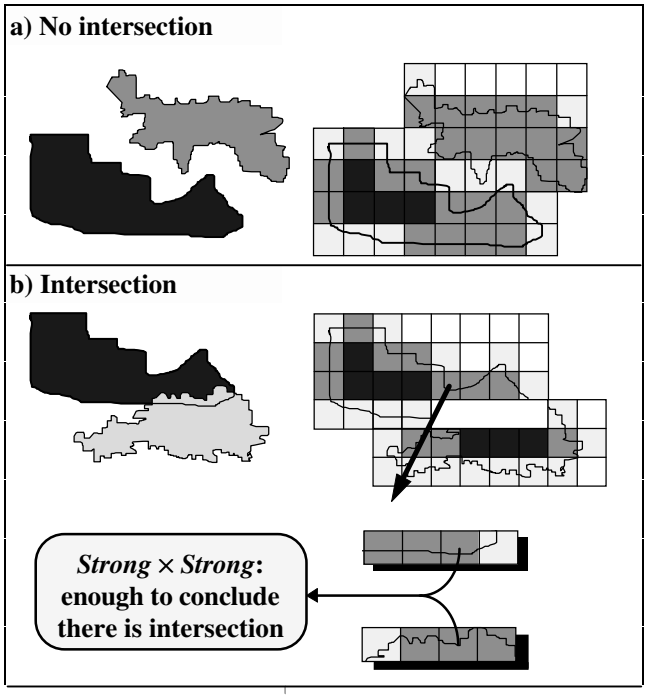


Figure 3 - (a) non-intersecting polygons, (b) intersecting polygons

In order to compare two approximations of distinct polygons, we must first ensure that their cells have the same size and that the intersecting cells have the same corner coordinates. The changes of scale are more readily performed if we require that length of each cell side should be a power of two (2^n), and that the beginning of each cell be a multiple of the same power of two ($a2^n$). In

so doing, we guarantee that, if two cells of the same size intersect each other, then they are perfectly superimposed to one another (Figure 4).

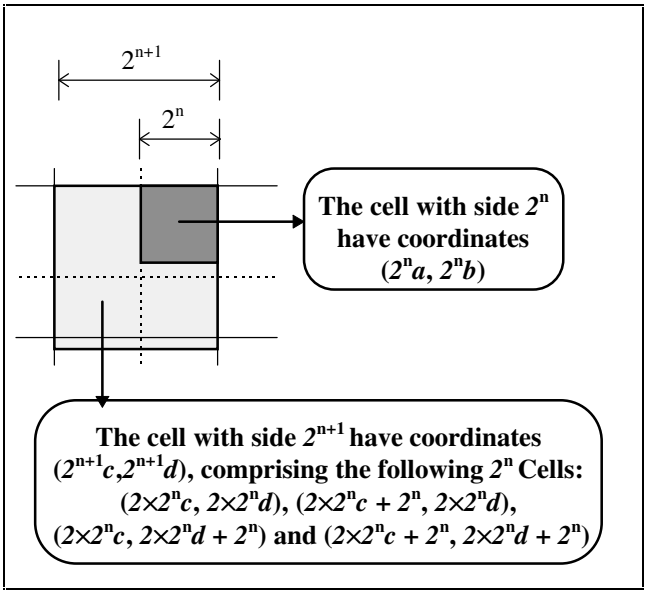


Figure 4 - Line up of Cell corners

Note that it is not possible to subdivide a cell, that is to say, to reduce the length of its side, performing a change of scale since, in the cases of Weak or Strong, this would lead to the erroneous assumption that the polygon's area is uniformly distributed throughout the cell. Hence, whenever a change of scale is necessary (Figure 5), it is accomplished through the grouping of 2^m cells, having in mind that the coordinates of the beginning of each cell are proportional to the length of its side.

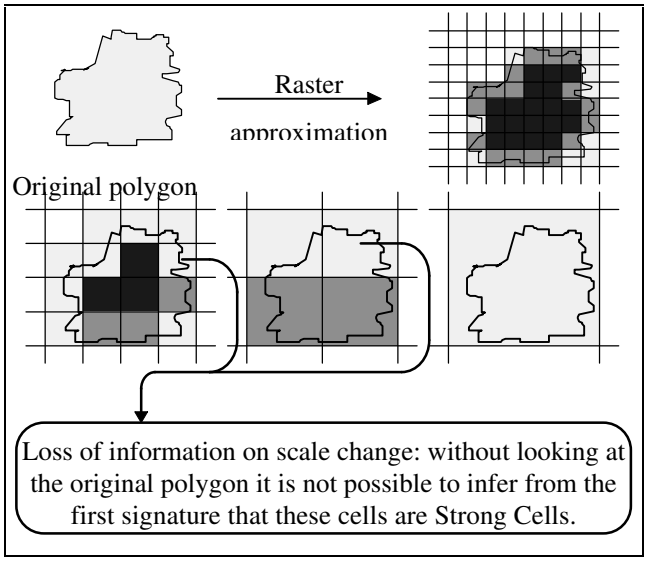


Figure 5 - Change of scale

To perform this scaling, we must use a pessimistic approach. It is of great importance to perform the conversion directly from the original signature, so as to reduce the loss of information. The algorithm for scaling a box with a 2^n side cell, is the following:

1. *if all the cells are empty, the result is empty;*
2. *if all them are full, the result is full;*
3. *otherwise we must evaluate a sum, attributing to each cell a numerical value as follows:*
   - ***a)*** *if the cell value is Empty or Weak, we count it as 0; if it is Strong, we count it as 0.5; if it is Full, its numerical value is 1.*
   - ***b)*** *If the average is less than 0.5, the result is Weak; otherwise, it is Strong.*

The raster approximation combines in only one approximation the conservative and progressive ones, being thus classified as integrated. When we use a progressive approximation, we can only be sure of the intersections, since the approximation area is totally contained within the polygon. On the other hand, a conservative approximation contains the whole polygon plus some additional area that does not belong to it: in fact, it is an improvement of MBR. Obviously, the progressive approximation is wholly contained in the conservative one. Nevertheless, this dual approach tells us nothing about the area that is contained in the conservative approximation but not in the progressive one. We call this an indecision area (Figure 6), since if the approximations for two polygons intersect each other only in this area we cannot affirm anything and must postpone the decision to the step of exact geometrical comparison (3rd step of the processor). It is important to notice that our definition of the indecision area is directly related with the idea of Approximation Quality, defined in [BRI93b].

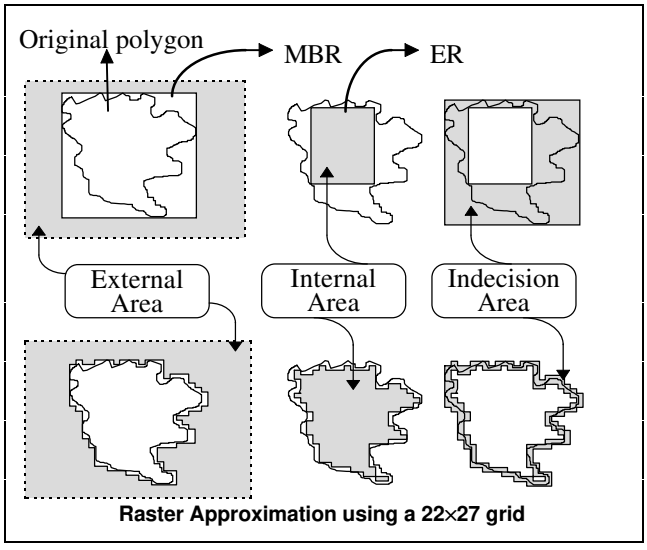


Figure 6 - Indecision area relatively to the object's area

The raster approximation, besides combining two approximations in one, retains information about the

fraction of polygonal area contained in the indecision area. Thus, it is possible to reduce the extent of the approximation indecision area, as the following table shows (Figure 7 and Table 3). The indecision level is reduced not only because we have a smaller indecision area but also because, in some cases, it is possible to come to a conclusion, even though we are in the indecision area.

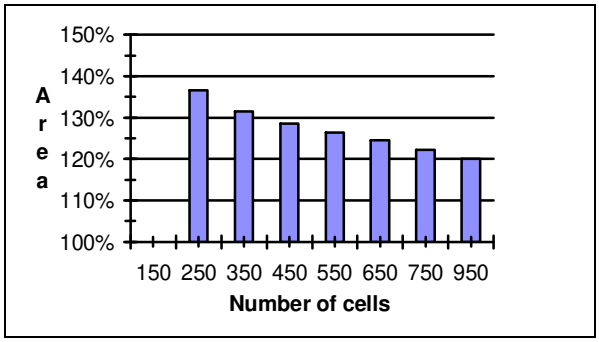


Figure 7 - Average results for 4CRS approximation (Approximation area and polygon area ratio)

Table 3 - Comparison of approximations - average of data sets

	Approximation Area / Polygon Area ratio(%)
4CRS(750 cells)	122%
Convex Hull	125%
5-Corner	133%
MBR	193%

As we can see, the Figure 7 presents a distribution that seems to be asymptotic. Thereby, to increase the number of cells in an approximation will not result in one corresponding improvement in the quality of the approximation. Here, a trade-off between approximation quality and key size must be done. In order to keep the key size small, we decided to use approximations with at most 750 cells. As we will see in section 4.3, this limit on resolution leads to an approximation size in bytes comparable to other approximations. Moreover, the analysis of our data sets, coming from the real world, has shown that the cells present the following approximate distribution (considering a resolution of 750 cells): for each cell having strong area intersection there is a cell with weak intersection, four empty cells, and four full cells. Though it is certainly encouraging, we will see that this distribution is scale dependent. This means that, when we compare polygons whose cells have different sizes, the distribution does not hold any more, since it is necessary to perform a scale change. Even when the polygons we compare are very different in size, in the worst case one of them will be reduced to a cell with weak area intersection, but the other polygon approximation remains unchanged. This means that about 80% of its cells will be full or empty (considering a resolution of 750 cells), leading to a good result when the comparison is made.

4.2 Number of Cells

We have verified experimentally that, as the number of cells in the approximation increases, the filter quality improves. Intuitively, we can explain this result using the argument that, the larger the number of cells, the closer is the approximation to the original polygon. In the limit, we are no longer comparing approximations, but the polygons themselves. This would lead to a rightness of 100%. However, this is a theoretical result. In practice, an approximation with 10,000 cells would be greater than the polygon itself, what would be useless.

Previously, we said that the distribution *area* × *number of points* was apparently asymptotic. We will not prove this statement here, because that is not the purpose of this article. We just provide a schedule to the proof, leaving the demonstration to the reader:

1. When we divide the cell side by two, the number of *full* and the *empty* cells are multiplied by four, as well as the number of some cells (*full* or *empty*) obtained in the following case;
2. Nonetheless, when we divide a cell that has *Strong* or *Weak* intersection with the polygon, we have the following situation: at most, ***r*** cells will be homogeneously divided, that is, they will give rise to four full cells or four empty cells, where ***r*** is the number of vertices in the polygon. The remaining cells, as shown in figure 8, originate at least one *full* cell or one *empty* cell, and at most three cells having *weak* or *strong* intersection;

Hence, for each increment ***d*** in scale, we have (estimated values):

a) $3^d(W + S) + 4r$ - remainder: cells with ***Weak*** or ***Strong*** intersection (at most).

b) 4^d (F + E) + remainder: ***Full*** or ***Empty*** cells (at least).

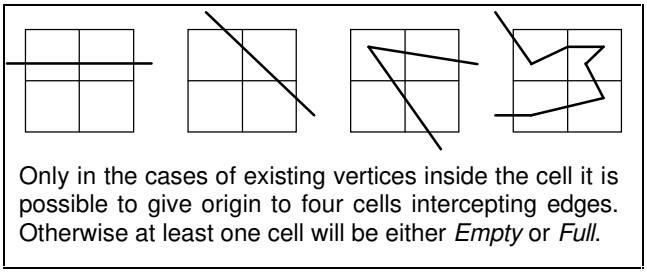


Figure 8 - Different cases of cell splitting

As ***d*** increases, the exponential terms dominate and the ratio (**a**)/(**b**) tends to decrease, being zero when ***d*** tends to infinity. Well, (**a**) is just the number of indecision cells: cells with weak or strong intersection. If the raster approximation is constituted only of full or empty cells, the percentage of rightness is equal to 100%. Nevertheless, we can also observe that the filter

improvement is not linear, reflecting the function that dominates the ratio (**a**)/(**b**), that is, $(3/4)^n$.

We can thus develop some criteria to choose the limit ***p*** to the cells used in the raster approximation. As we have seen, this limit influences the average size of each key and thus involves a cost/benefit relation. The choice is related with the average number of points per polygon, as well as with the minimum distance between two points in each polygon. Yet, the minimum distance between two points is directly related with precision and, thus, with the average number of points per polygon. The exceptions to this rule are those polygons that have large edges, generally defined artificially - an example is given by the case of several large American states and counties, as well as by large constructions, cities and real estates in general.

[BRI95] presents a study of the polygon complexity and [GAE96] presents a study about the fractal dimension of polygons, which can serve as a basis to estimate the number of cells necessary to make a good raster approximation. Although the preliminary results are encouraging, more detailed tests are needed to achieve conclusive results. As we have stated before, in this work the limit for ***p*** is 750 cells, in order to keep approximation size competitive. Of course, any other number close to 750 could be used with similar results.

4.3 Computing the 4CRS Approximation

We implemented a straightforward algorithm, that was developed using routines that were already written by others. We didn't worry about the optimization of the algorithm since it will just be used once to compute the approximations, that will be stored and used several times later. Although we didn't measure the exact time spent on the calculation of the 4CRS approximations we can affirm that it is comparable with the time spent on the calculation of the other approximations: 5-C and ER&EL.

The MBR of the polygon is used to calculate the 2^n-MBR, that is a MBR whose vertexes are in the form ($2^n x_0$, $2^n y_0$) and ($2^n x_k$, $2^n y_k$), where x_0, x_k, y_0, y_k and n are integers. Besides, n is chosen in such a way that $(x_k - x_0)(y_k - y_0) < N+1$, where N is the maximum number of cells of the approximation (for example 750). Used algorithm (brute force): initially maximum n is chosen so that 2^n is smaller than the side of original MBR. Then, in successive iterations, n is decreased of 1 while the above conditions hold.

The points x_0, x_1,... x_k defines a set of parallel lines to the vertical axis that we will call X_0, X_1,... X_k. In a similar way, we define the horizontal lines Y_0, Y_1,... Y_k . For each pair of vertical lines (X_i, X_{i+1}) we compute its intersection with the original polygon. At this time, we are using a simplified version of the algorithm of Sutherland-Hodgman [ROG86]. We call each resulting polygon by P_i. For each polygon P_i, we perform the clipping process for the pairs of horizontal lines (Y_j, Y_{j+1}). The result of each clipping is the intersection of the original polygon with the cell (i, j). Finally, for each cell, we compute the area of the intersection of the original polygon with the cell, and we classify the cell: *Full*, *Strong*, *Weak* or *Empty*. The cells *Full* and *Empty* can be classified by inspection, without need to compute its area. The only optimization that is taken place is to store the intersections of the line X_{i+1} that were computed for the pair of lines (X_i, X_{i+1}) and to use them for the calculation of the pair (X_{i+1}, X_{i+2}). The same is done for the horizontal lines.

4.4 Compression

The fact that the major part of the cells is either empty or full (80% for 750 cells) led us to study the applicability of (no loss) compression algorithms to the raster approximation. In fact, using patterns of 3×3 cells, we have got very good compression levels, the compressed approximation having only 40% of its original size. Using patterns of 2x2 cells, the compression rate was not so good. On the other hand, using patterns of 4x4 cells we have gotten a little better compression rate (about 35% of original approximation size), but at the expense of decompression speed and space requirements for decompression structures. Thus, we decided to use the 3x3 cell patterns.

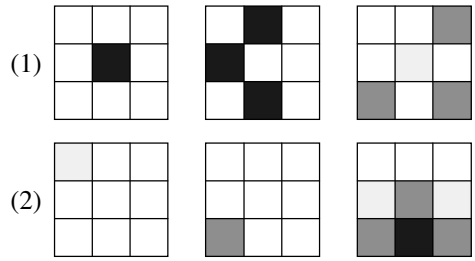
The raster approximation compression allows for a larger information density and makes it possible to use larger number of cells that, as we have seen, improves the filtering process and the decision ability in step 2. Nevertheless, this approach makes evident a problem that we have not faced before: the fact that the approximations will be of different sizes. One must have in mind, though, that this problem is by no means a result of compression. When we say that the approximation should use p cells, we are in fact determining the upper limit of cells for such approximation. Certainly, there will be approximations $m \times n$ so that the number of cells is different of p (and therefore less than p). Moreover, to increase the number of cells in an approximation we must divide each cell size in a half, since we have constrained the cells sizes to be powers of two. As a result, the total number of cells would be multiplied by four. Since we can only affirm that $2m \times 2n$ is less than p if $m \times n$ is less than $p/4$, we have that, if $m \times n$ is larger than $p/4$ then it is not possible to increase the number of cells in the approximation. This result means that the difference between the larger and the smaller approximation can be at most equal to four times. Hence, even if we do not use compression, we still have to deal with approximations of different sizes.

4.4.1 The Compression Algorithm

The compression algorithm used here has been developed through statistical tests involving the data. The tests clearly demonstrated the predominant occurrence of full and empty cells over the cells with weak and strong intersection. Moreover, one of the most important requirements in the determination of the compression algorithm was its simplicity, since compression should

by no means pose any disadvantage for the query as a whole.

Having this in mind, the cells of a particular signature have been split in groups of 3×3 cells, and each group was attributed a fixed code. Since there are 4^9 possibilities for the cells, we need 18 bits to code each of them. Nevertheless, some kinds of cell are improbable of occurring, while some others are very frequent (Figure 9). Just to illustrate this fact, we made a statistical study of the various data sets having 750 cells' signatures. The results have shown that the full group (the group in which all the cells are full) and the empty group (the group in which all the cells are empty) taken together represent about 40% of all cell groups.

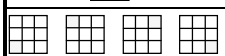
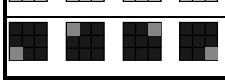


(1) Groups of improbable occurrence, and (2) groups of very frequent occurrence in any data set.
In practice, it is hardly observed the occurrence of a full cell next to an empty cell, as in a "chess table", since the boundaries of spatial objects tend to be more complex and irregular, so that their vertices hardly match the cells coordinates.

Figure 9 - Example of cell groups

Our compression algorithm is an adaptation of Huffman's, in which the set of groups has been partitioned in four classes. The compression tree is constructed once, and it is based on the analysis of a large set of data. Basically, the compression algorithm assigns codes with few bits to cell groups of very frequent occurrence, and vice-versa. Table 4 shows some of these codes and cell groups. It is important to note that code prefix must be unique.

Table 4 - Sample patterns, their codes and frequency (Brazil, 550 cells). Prefix 11 is used for coding other patterns that use more bits.

Pattern	# of occurrences	Code (*binary digits*)
	22%	00
	18%	01
	14%	10000, 10001 10010, 10011
	12%	10100, 10101 10110, 10111

It is then stored as an algorithm parameter, separated from the compact data. Hence, the compression, although less efficient, can be done more readily and the tree need not be stored with the data. Moreover, the unpacker can be optimized manually for a particular tree, determined before the compilation of the unpacker. In our tests the compression tree reached as much as 15 Kbytes and the compression rates were about 40% of the original key size. The compressed keys' sizes are smaller than those in [BRI94] (Table 5). Moreover, our results cell out to an almost total independence between the data set and the tree. However, the dependence between the approximation quality and the compression rate was stronger: the better the approximation quality, the larger the compression rate.

Table 5 - Average size (bytes) of compressed signatures (Brazil)

	4CRS (250 Cells)	**4CRS (350 Cells)**	**4CRS (550 Cells)**	**4CRS (750 Cells)**
Original size	35.8	50.2	77.0	104.5
Compressed size	19.7	25.44	33.8	42.5

4.5 Dealing with Approximations of Different Sizes

As we have already explained, the approximations will be part of the polygon keys, and this means that they will be stored in the R*-Tree index, in the same way [BRI94] does. This is a reasonable approach, since both the calculations of a raster approximation and the calculation of 5C & ER-EL approximation are relatively slow operations. Furthermore, one can expect to compute these approximations one time, to store and to use them many times later.

In order to deal with the variable sizes of the approximations without spending much space (which would increase the number of disk access operations) we must group a significant number of raster approximations in one bucket. The choice of this number must take into account the average key size. We must also pay attention to the possible occurrence of an overflow. The average compact key size can be determined experimentally for each limit ***p*** of cells. Adding to it a safety margin can then reduce the number of overflows to admissible limits. Figure 10 shows the distribution for 4CRS key size when the limit ***p*** is set to 750. Furthermore, as we can see in Figure 10b, less than 80% (actually, 77,19%) of the 4CRS keys are smaller than 56 bytes, which is the size of 5-C plus ER, and 99,88% are smaller than 80 bytes, which is the size of 5-C plus ER-EL [BRI94, BRI93b].

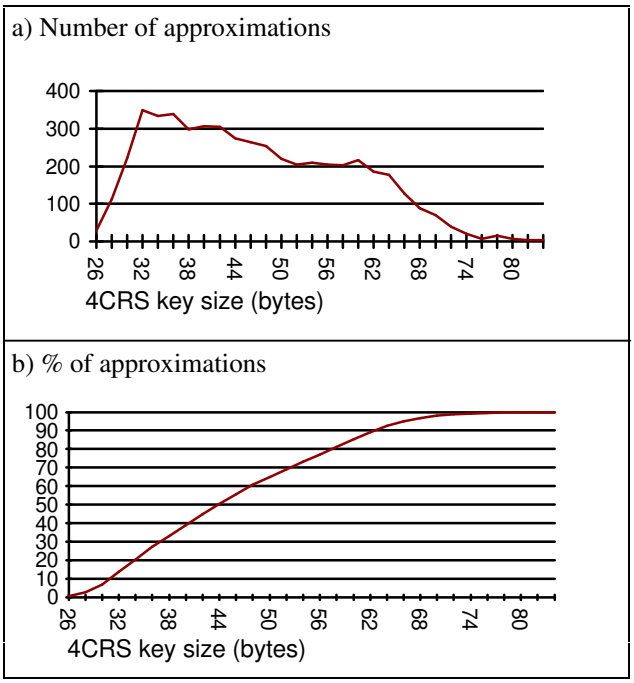


Figure 10 - a) Distribution of approximations/key size; b) Cumulative distribution.

The spatial data structure used as an index in the processor's first step must be adapted to deal with buckets and overflows, so that we can solve the problem of the variable-sized keys. It is important to notice that, in general, these structures already work with buckets [SAM90] to access the disk efficiently. Moreover, the buckets used in the R*-Tree family of data structures usually have space utilization rate between 50% and 100%. This means that the adaptation to be made in such structures is not very difficult as we ourselves have been able to verify adapting R*-Trees.

5 Experimental Results

5.1 Test Data

Our test data is composed of several sets of polygons containing up to forty-five thousand polygons. We have used data from various origins such as some municipalities in European countries, American counties and Brazilian municipalities [IBG96]. Typically, the average number of points per polygon in each data set ranges from 22 to 96 (Table 6). As suggested in [BRI94] we have generated the other data set (named *data set'*) shifting the original polygons by random displacements of x and y coordinates, and then performing the spatial join using *data set* × *data set'*. The only exception to this rule is the join Brazil-A × Brazil-B. These data sets were generated as follows: data set Brazil was randomly expanded, shifted, rotated and replicated 9 times (Brazil-A) and 4 times (Brazil-B).

Table 6 - # of points and vertices in each data set

Data set	# Polygons	Average # of vertices
South America	228	54
Europe	249	96
South USA	496	22
Brazil	5081	80
Brazil-A	45.729	80
Brazil-B	20.324	80

5.2 Spatial Join Comparison

Now we present the average of some results obtained in our work, to allow for a comparison with the other approximations (Tables 7 and 8). To compute the averages, we have discarded the South of the USA results because of its unusual polygon uniformity. We compared 4CRS with two approximations: 5C-ER (56 bytes) and 5C-ER&EL (80 bytes). Also, we have omitted Brazil-A and Brazil-B results because these data sets are derived from Brazil data set.

The input data set of this step is the set of candidates, which is a set of pairs of polygons. The set of candidates was originated as a result of the step one - the step of MBRs join (in this case, we have used a R*-Tree). It is important to notice that no matters what method we use to compute step one, we will always obtain the same set of candidates. It is due to fact that we are computing MBR joins.

Table 7a describes the data sets in terms of intersecting polygons and MBR pairs of candidates. In the Tables 7b and 8, the first three columns show the classification of our filter for each pair of candidates. The column **Accepted** shows the percentage of candidate pairs that were identified as pairs of polygons that intersects each other. It is a true polygon intersection and it was discovered without having access to the exact representation of the polygons. The column **Rejected** shows the percentage of candidate pairs that were identified as pairs of polygons that do not overlaps. Again, we do not have to access the exact representation of the polygons in the disk in order to decide. The column **Candidates** shows the percentage of candidate pairs that our filter could not identify, and it will be the input data set of step three. The column **Rejected Identified** shows the percentage of pairs of polygons without intersection that were identified, and the column **Accepted Identified** shows the percentage of pairs of polygons with intersection that were identified by our filter.

Table 7a - % of pairs of intersecting polygons/MBR pair of candidates

	Brazil	South USA	Europe	South America
%of intersections	65.49%	82.21%	68.63%	67.86%

Table 7b - Performance of the filters (a) 4CRS-350, (b) 4CRS-750 and (c) 5-C/ER&EL (best results are bold face)

	Accepted	Rejected	Candidates (smaller is better)	Rejected Identified	Accepted Identified
Brazil					
(a)	57.23%	23.23%	19.55%	67.29%	87.39%
(b)	**59.96%**	**26.30%**	**13.74%**	**76.20%**	**91.56%**
(c)	37.15%	25.75%	37.10%	73.69%	57.10%
S. USA					
(a)	78.78%	12.17%	9.04%	68.41%	95.84%
(b)	**79.37%**	13.74%	**6.89%**	**77.01%**	**96.60%**
(c)	62.51%	**16.26%**	21.24%	80.22%	77.92%
Europe					
(a)	59.73%	20.49%	19.77%	65.33%	87.04%
(b)	**62.28%**	**23.48%**	**14.24%**	**74.85%**	**90.75%**
(c)	41.39%	22.48%	36.13%	70.36%	60.82%
S.America					
(a)	60.45%	21.22%	18.33%	66.02%	89.08%
(b)	**62.48%**	**24.18%**	**13.34%**	**74.88%**	**92.28%**
(c)	41.50%	24.49%	34.01%	72.58%	62.63%

Table 8 - Comparison of methods (average rate, without south USA)

	Accepted	Rejected	Candidates (smaller is better)	Rejected Identified	Accepted Identified
4CRS (350 points)	59.1%	21.6%	19.2%	66.2%	87.8%
4CRS (750 points)	**61.5%**	**24.6%**	**13.7%**	**75.2%**	**91.53%**
5-C & ER (our results)	23.5%	23.6%	52.8%	71.4%	35.2%
5-C & ER-EL (our results)	40.0%	23.6%	36.4%	71.4%	60.2%
5-C & ER (Europe A) [BRI94]	23%	23%	54%	66%	30%
5-C & ER-EL (Europe A) [BRI93b,BRI94]	40.5%	20.8%	40.84%	66.3%	59.2%

It is important to note that the column ***Candidates*** in Table 8 is the determinant factor of the problem. These numbers show the size of the input data set for the step 3, which dominates the total execution time of the query. In this way, smaller rates indicate that fewer polygon intersection tests must be done. As one can see, our average results (13.7% of intersecting MBRs) exhibit a shrinking rate of more than 60% when compared with the best results previously presented (**5-C/ER&EL**, 36.4% of intersecting MBRs). As we have mentioned early, the combination of 5-C plus ER-EL spends more bytes in the key than our 4CRS approach. So, one can expect at least the same I/O costs for our approach since we are using a smaller key.

Although [BRI94] has neglected the time spent on 5-C intersection tests because it does not affect significantly the overall time, we decided to show these times in Table 9 - mainly because it is the only part of our algorithm that could result in time increase. As one can see, 4CRS key decompression and cell comparison have spent nearly the same time that 5-C. This result reflects the fact that 4CRS only uses fast integer operations: bit shifting and logical operations like AND, XOR and OR. In contrast, 5C-ER&EL approximation uses some floating point operations or double precision integer multiplication and division. These times were measured by first measuring the execution time spent on performing only step 1 and subtracting it from the time spent on performing steps 1 and 2. We have to adopt this methodology because the spatial processor is not sequential: the execution of step 2 occurs interspersed with step 1. The test machine is a SUN Sparc2 with 32 Mb main memory. Although this test has been performed with only one of the data sets (the largest), the result of this test shows that its impact on the total time of the query is very small - less than 0.75% of total query time. One should note that the measured time includes the time spent on unpacking the approximations and comparing them.

Table 9 - Average time spent on main memory approximation comparisons (Brazil data set, 5081 polygons and 33.188 intersecting MBRs)

	Time (seconds)
5C-ER&EL	2.34
4CRS	2.48

Only the I/O-time of step three was measured: the CPU-time spent on this step was not included in the final results. The main reason is that [BRI94] has shown that the plane-sweep algorithm, that we used in our tests, can be replaced by their TR*-trees algorithm with impressive CPU performance gains and little I/O overhead. Also, due to modular structure of MSQP, we could use SID [HUA97a] in this step. By the way, shrinking the size of input data set of this step certainly results in a CPU and I/O cost reduction, no matter what algorithm is used for detecting polygon intersection. As we have mentioned, sometimes our approach leads to 60% of reduction of input data set of this step.

Finally, we have implemented a testbed for complete spatial join experiments in C++ on the same machine stated before. The testbed includes a MSQP

implementation as stated in [BRI93a] and [BRI94], and we adopt the same methodology they used to measure CPU and I/O costs. We used a R*-Tree to implement the MBR join of step one, and we have implemented all the optimizations suggested in [BRI93a] and [BRI94]. Table 10 shows our results. Both joins were performed using a buffer of 22 pages of 4Kb. The size of data set Brazil is 5,081, and this join produces 31,998 pairs of intersecting MBRs and 20,885 pairs of intersecting polygons. The size of data set Brazil-A is 20,324, and the size of Brazil-B is 45,729. This join produces 108,153 pairs of intersecting MBR, and 59,887 pairs of intersecting polygons.

Table 10 - Total join performance - 4CRS (750 points) and 5-C/ER&EL

	Number of exact inter-section tests		**Number of disk access (all steps)**		**Total execution time (seconds)**	
	5-C ER&EL	**4CRS**	**5-C ER&EL**	**4CRS**	**5-C ER&EL**	**4CRS**
Brazil x Brazil (shifted)	11,872	4,696	11,451	6,069	130.8	70.8
Brazil-A x Brazil-B	41,962	19,984	44,192	26,610	529.8	332.0

The column **Number of exact intersection tests** shows the size of input data set of step three, the most time-consuming step. These are the pair of candidates that the filter step could not identify as being a true polygon intersection or a false-intersection (a MBR intersection only). The column **Number of disk access** includes the sum of disk access of all steps. Of course, the size of the R*-tree and the size of the input date set of step three have a great influence on this number.

The size of the Brazilian municipalities varies a lot: there are municipalities in the Amazonian region that are larger than small countries, while other municipalities are very small. Moreover, the data sets of second join, Brazil-A and Brazil-B, have different average polygon areas, which lead to many scale changes. These joins show that scale changes have little performance impact in our algorithm. The performance gains were about 45% and 37%, respectively, when compared with the previous approach.

6 Conclusions and Future Works

This work has proposed a new approximation for the filtering of intersecting polygons in the process of making spatial joins. We achieved good results with the use of this new approach, specially when compared to state-of-art techniques.

The use of a raster approximation has shown to be advantageous over other methods used in the filtering step in the accomplishment of spatial joins processing. Even in the case in which the gain was smaller (using fewer cells, or on scale changing), we have obtained a reduction of 50% in the number of exact comparisons and in the cases in which we used a greater number of cells, the amount of reduction reached 60%. As we have seen, this is the most costly part of the spatial join processing since it requires the search and transfer of large objects from the disk to the main storage [BRI94, MIN96]. So, for any algorithm used in step one or step three , there will be a shrinking on size of the input data set of step three, resulting in smaller CPU and I/O costs, as our experimental results on Table 10 have shown. Also, when measuring total time spent on performing a spatial join, the 4CRS approximation outperforms other filters by up to 45%.

It is also important to notice that the raster approximation proposed here neither interferes in nor prevents the use of any approach that employs parallelism to process spatial joins. Also, our approximation does not interfere with new improvements in R*-Tree joining [HUA97b].

It should be investigated the use of raster approximations involving more colors, for example, eight colors. Although we pay the extra cost of having an additional bit for color (shape) representation, the relative abundance of full and empty cells over those with strong and weak intersection seems to indicate that, using the compression schedule proposed here, one should not expect a noticeable increase in the sizes of the maps. Moreover, the use of more colors will certainly decrease both the size of the indecision area and the information loss during the changes of scale. Promising results were obtained in a preliminary work on this [ZIM97].

Finally, alternative algorithms for compression of 4CRS raster approximation could be investigated: the quadtree polygon decomposition, gif methods and lzw encoding. Although these algorithms are more complex than the one presented here, they could lead even to a better space utilization.

Acknowledges

We would like to thank to the brazilian reaserch agencies CAPES and CNPq for partially financing this work.

References

[BER96] S. Berchtold, D. A. Keim, H. P. Kriegel: “The X-Tree: An Index Structure for High-Dimensional Data”, In Proceedings of 22th International Conference on Very Large Data Bases, Bombay, India, 1996.

[BRI93a] T. Brinkhoff, H. P. Kriegel, and B. Seeger: “Efficient processing of Spatial Joins Using R-Trees”. In Proceedings of the 1993 ACM-SIGMOD Conference, Washington, DC, USA, May 1993.

[BRI93b] T. Brinkhoff, H. P. Kriegel, and R. Schneider: "Comparison of Approximations of Complex Objects Used for Approximation-based Query Processing in Spatial Database Systems". In Proceedings of 9th International Conference on Data Engineering, Vienna, Austria, 1993.

[BRI94] T. Brinkhoff, H. P. Kriegel, R. Schneider, and B. Seeger: "Multi-step Processing of Spatial Joins". In Proceedings of the 1994 ACM-SIGMOD Conference, Minneapolis, USA, May 1994.

[BRI95] T. Brinkhoff, H. P. Kriegel, R. Schneider, and A. Braun: "Measuring the Complexity of Polygonal Objects". In Proceedings of ACM International Workshop on Advances in Geographic Information Systems, Baltimore, MD, USA, December 1995.

[BRI96] T. Brinkhoff, H. P. Kriegel And B. Seeger: "Parallel Processing of Spatial Joins Using R-Trees", In Proceedings of 12th International Conference on Data Engineering, New Orleans, LA, USA, 1996.

[GAE96] V. Gaede and C. Faloutsos: "Analysis of n-dimensional Quadtrees Using the Hausdorff Fractal Dimension". In Proceedings of the 22nd VLDB Conference Mumbai (Bombay), India, 1996.

[GÜN93] O. Günther: "Efficient Computation of Spatial Joins". In Proceedings of 9th International Conference on Data Engineering, Vienna, Austria, 1993.

[HUA97a] Yun-Wu Huang, Matthew C. Jones, Elke A. Rundensteiner: "Improving Spatial Intersect Joins Using Symbolic Intersect Detection". Proceedings of the 5th International Symposium on Advances in Spatial Databases, SSD'97, Berlin, Germany, July 15-18, 1997.

[HUA97b] Yun-Wu Huang and Ning Jing: "Spatial Joins Using R-Trees: Breadth-First Traversal with Global Optimizations". Proceedings of the 23rd VLDB Conference, Athens, Greece, 1997.

[IBG96] (Brazilian Institute of Geography and Statistics) Fundação Instituto Brasileiro de Geografia e Estatística - IBGE: "Malha Municipal Digital do Brasil - 1994", Rio de Janeiro, 1996.

[MIN95] M. L. Lo and C. V. Ravishankar: "Generating Seeded Trees From Data Sets". In Proceedings of 4th International Symposium on Large Spatial Databases, Portland, ME, USA, August, 1995.

[MIN96] M. L. Lo and C. V. Ravishankar: "Spatial Hash-Joins". In Proceedings of the 1996 ACM-SIGMOD Conference, Montreal, Canada, June 1996.

[PAT96] J. M. Patel and D. J. DeWitt: "Partition Based Spatial-Merge Join". In Proceedings of the 1996 ACM-SIGMOD Conference, Montreal, Canada, June 1996.

[ROG86] David F. Rogers " Procedural Elements For Computer Graphics" - McGraw-Hill Book Company.

[SAI94] MELP Spatial Archive and Interchange Format (SAIF): Formal Definition Release 3.1 April 1994 Reference Series Volume 1 - Surveys and Resource Mapping Branch - Ministry of Environment, Lands and Parks (MELP) - Province of British Columbia - Canada.

[SAM90] H. Samet: "The Design and Analysis of Spatial Data Structures", Addison-Wesley Publishing Company, 1990.

[SQL95] X3H2-95-084/DBL:YOW-004, (ISO-ANSI Working Draft) Database Language SQL (SQL3), Jim Melton ed., March, 1995.

[VEE95] H. M. Veenhof, Peter M. G. Apers, Maurice A. W. Houtsma: "Optimization of Spatial Joins Using Filters". In Advances in Databases, 13th British National Conference on Databases, BNCOD 13, Manchester, United Kingdom, July 1995.

[ZIM97] G. Zimbrão and J. M. Souza: "Using Raster Approximations For Processing of Spatial Joins". COPPE/UFRJ - Brazil - Technical Report ES-442/97 (in English).