Using Rasters Approximations For Processing of Spatial Joins

GERALDO ZIMBRÃO
JANO MOREIRA DE SOUZA
Systems and Engineering Program
Graduate School of Engineering
Universidade Federal do Rio de Janeiro
Caixa Postal 68511, CEP: 21945-970, Rio de Janeiro, Brazil,
email: zimbrao@cos.ufrj.br, 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. Generally, an approximation can identify only positive and inconclusive answers, or only negative and inconclusive answers. In that way, two complementary approximations are required for filter construction. This article presents a polygon approximation for spatial join processing which we call four-colors raster signature (4CRS), and its variants. Differently from other proposals, this approximation has the advantage of being complete. One can use this approximation to identify the three possible classes of answers when examining candidate pairs of polygons. 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.

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), uses spatial databases generally constructed upon relational databases, and need especially 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 [SQL395,SAIF94].

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. [BRINK94] 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 so as 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 final 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, an 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 [VEEN95]: “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 (e.g., lakes) or it may be disconnected (e.g., 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 [BRINK94] 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, originating in the joins found in 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.

2.2 Test Data

Our test data is composed of several sets of polygons containing up to five thousand elements each. We have used data from various origins such as some municipalities in European countries, American counties and Brazilian municipalities [IBGE96]. Typically, the average number of points per polygon in each data set ranges from 22 to 96 (see Table 1). As suggested in [BRINK94] we have generated the outer data sets shifting the original polygons by random displacements of ***x*** and ***y*** coordinates.

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

Table 1 - # of points and vertices in each data set

3. Related Works

Recently it has been observed that the amount of research work related to the subject treated here is increasing, although it is still too early to say that it is abundant. As we have mentioned, [BRINK94] presents a multi-step spatial query processor (**MSQP**). In other works by the same group comparisons are made amongst the approximations used [BRINK93b] and on Polygon complexity [BRINK95]. In [VEEN95] one can also find different approximations for spatial objects. In [GÜNT93] a general model to estimate the cost of spatial joins is presented. Finally, [PATE96], [MING95,96], [BERC96] and [BRINK93b,96] present algorithms to perform the selection of candidate polygon pairs. Any one of these algorithms can be chosen for use as the initial step to be taken previous to the application of our algorithm.

3.1 The Spatial Join Processor

In a frequently mentioned reference, [BRINK94] presents the Multi-step Spatial Query Processor (**MSQP**), designed to perform spatial joins. In such processor, a spatial join is divided in three steps, that we describe later, in detail.

The spatial join processor defined by [BRINK94] allows for a modular structuring of the processing of spatial joins that had no precedent. Our work fits to this structure so as to be able to replace (with some advantage) one of the modules used by **MSQP**. Other works already mentioned (Partition Based Merge Join[PATE96], Spatial Hash Joins, [MING96]) can also be adapted to match **MSQP**, implementing some of its steps. We now proceed to describe the **MSQP** processor in detail:

3.1.1 MSQP Step 1: Candidate Selection

Firstly, instead of making an exact representation of the polygons, the Minimum Bounding Rectangle (MBR) is used as an approximation to compute an approximate spatial join. This step returns what is called a candidate set, since it contains all the elements of the answer as well as some others originating in the Cartesian product which do not intersect one another;

3.1.2 MSQP Step 2: Filter Application

In the second step, approximations that are more accurate than MBR are used for refining the answer, so as to eliminate some false candidates. Moreover, positive answers can also be identified by means of such approximations, with no need to access the exact geometrical representation of the spatial objects. *Our work deals primarily with this step;*.

3.1.3 MSQP Step 3: Exact Geometrical Comparison

All the remaining pairs of candidates are eventually examined at this stage. This step requires access to the exact geometrical representation of the spatial objects, and it is frequently the most time-consuming step. Nevertheless, the amount of time spent in this step can be reduced using better approximations in the previous stage.

4. The Raster Approximation

A spatial access method organizes the objects in a spatial structure accordingly with a geometrical key [BRINK94]. Due to its simplicity, **MBR** is the most popular geometrical 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, real world cartographic objects are very poorly approximated by MBRs.

The current approximation techniques can be divided into three classes [BRINK93b]: *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 approximations 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.

4.1 Using Raster Signatures to Approximate Polygons

Among the results obtained by [BRINK94] two are particularly important for our work. The first result is that the step of exact geometrical comparison, on account of requiring the search of several objects in the disk, is the most time consuming of all. Thus, efforts must be made to enhance the filters so as to reduce the number of polygons that must be brought to memory. Secondly, about two thirds of the candidates selected through 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 has occurred in sets of random data originated from real data. In case this assumption is confirmed, it would be interesting to improve the

filters so as to enhance the detection of intersections during the course of the spatial query processor's second step.

Therefore, we propose an approximation to replace, within the frame of the proposed processor, the **5C** (conservative) and the **ER-EL** (progressive) approximations: 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:

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

Table 2 - Types of Cell in the Raster Approximation 4CRS

As an example, we have the polygons in Figure 1.

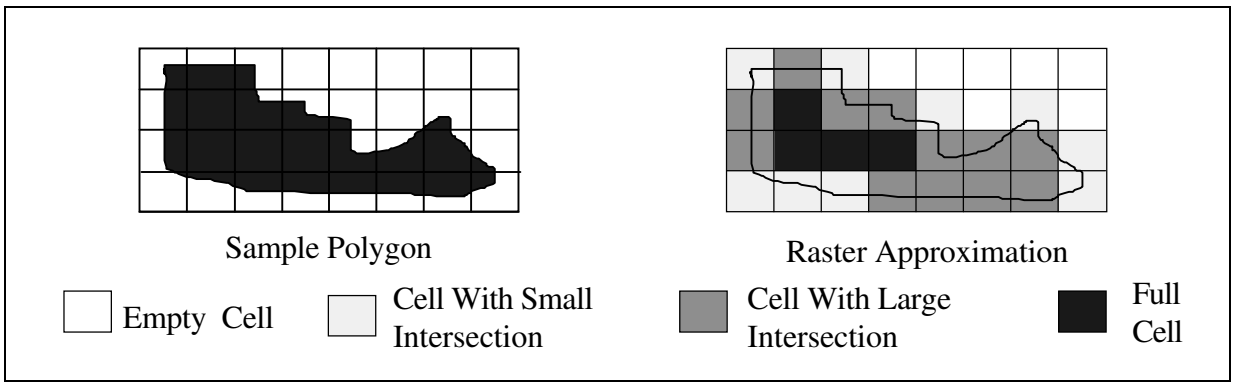


Figure 1 - Cells in a Raster Approximation 4CRS

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

	Empty Cell	Cell with Weak Intersection	Cell with Strong Intersection	Full Cell
Empty Cell	**Discarded**	**Discarded**	**Discarded**	**Discarded**
Cell with Weak Intersection	**Discarded**	*Candidate*	*Candidate*	**Accepted**
Cell with Strong Intersection	**Discarded**	*Candidate*	**Accepted**	**Accepted**
Full Cell	**Discarded**	**Accepted**	**Accepted**	**Accepted**

Table 3 - Possible Outcome of Cell Matching: Only Three Inconclusive Cases

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 2

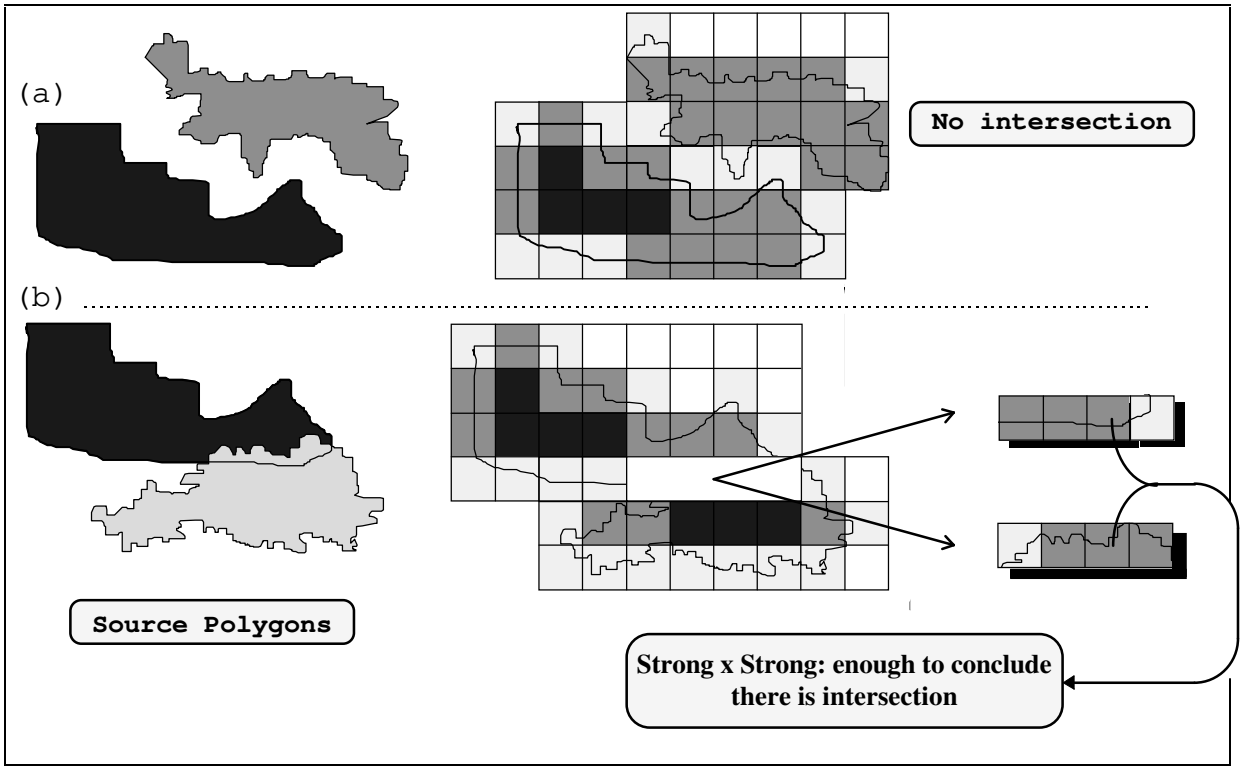


Figure 2 - (a) non-intersecting polygons, (b) intersecting polygons

In order to compare two approximations of distinct polygons, we must firstly assure that their cells have the same size and also 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 also that the beginning of each cell be a multiple of

the same power of two ($2^n a$). 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 3).

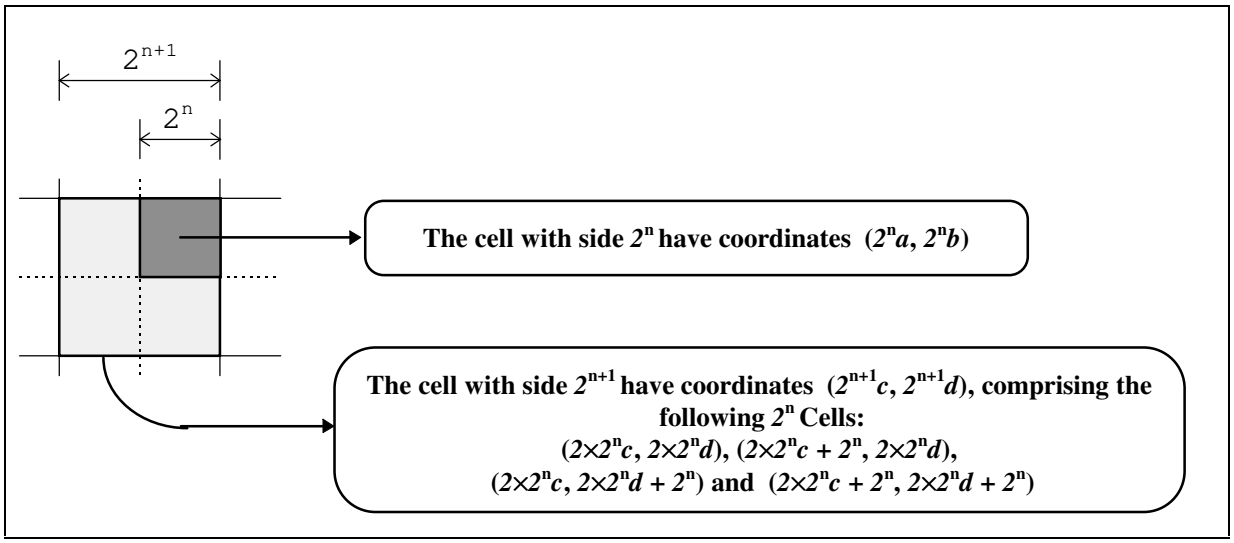


Figure 3 - Line up of Cell corners

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 4), it is accomplished through the grouping of 2^m cells, having in mind that the coordinates of the beginning of each cell is proportional to the length of its side.

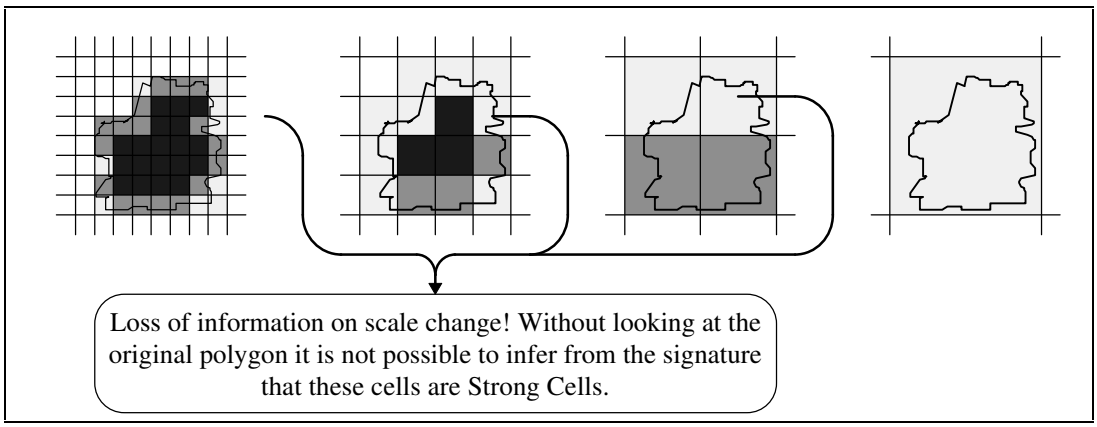


Figure 4 - Change of Scale

To perform this scaling, we must use a pessimistic approach. t 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 cells, 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 final result is Weak; otherwise, it is Strong.*

The raster approximation combines in one only 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 5), 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 concept of Approximation Quality, defined in [BRINK93b].

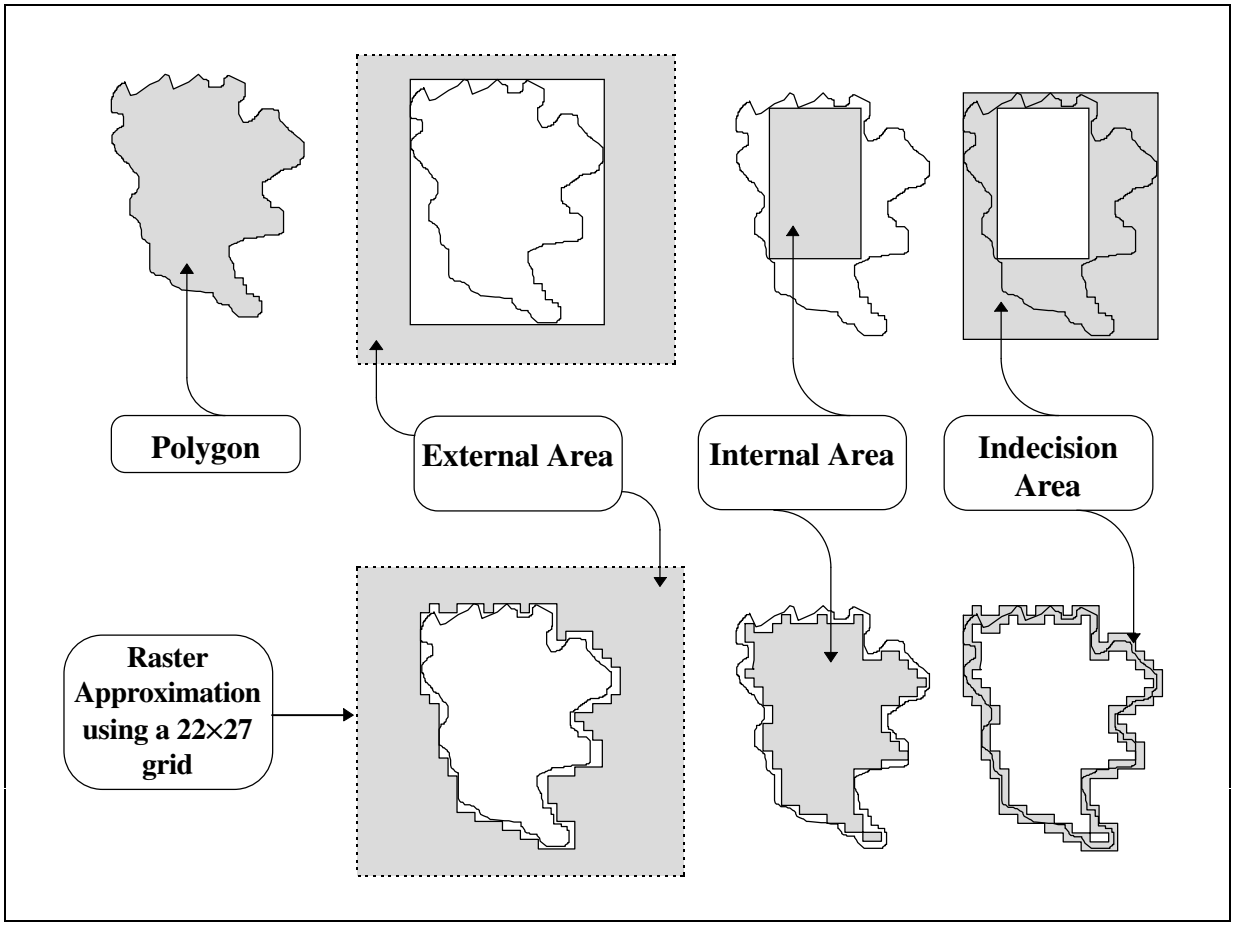


Figure 5 - Indecision area relatively to the object's area

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, as the following tables show (Tables 4-5). 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.

Data Set	4CRS (250 cells)	4CRS (350 cells)	4CRS (550 cells)	4CRS (750 cells)
Brazil	136.6%	131.5%	126.37%	122.8%
South USA	124.9%	121.6%	115.7%	113.9%
Europe	134.1%	129.5%	124.6%	121.8%
South America	134.3%	129.1%	124.4%	120.1%

Table 4 - Results for 4CRS Raster Approximation

Data Set	Convex Hull	5 Corner	MBR
Europe	125%	133%	193%

Table 5- Results from [BRINK93b]

Moreover, the analysis of our data set, originating in the real world, has shown that, considering a resolution of 750 cells, the cells present the following approximate distribution: 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%.

We will not proof 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 6, originate at least one full cell or one empty cell, and at most three cells having weak or strong intersection;
3. Hence, for each increment ***d*** in scale, we have (estimated values):
   a) $3^d(P + M) + 4r$ - remainder: cells with weak or strong intersection (at most).

b) 4^d (V + C) + remainder: full or empty cells (at least).

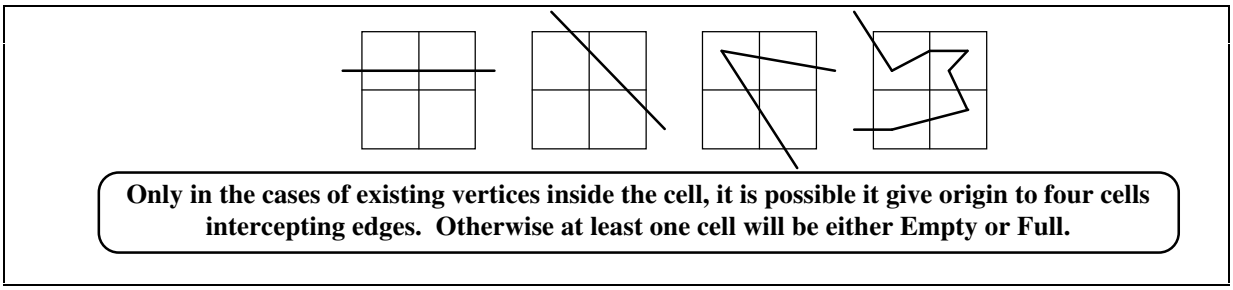


Figure 6 - Different cases of cell splitting

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 which dominates the ratio (**a**)/(**b**), i.e. $(3/4)^n$.

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 exception 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.

[BRINK95] presents a study of the polygon complexity and [GAED96] 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.

4.3 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.

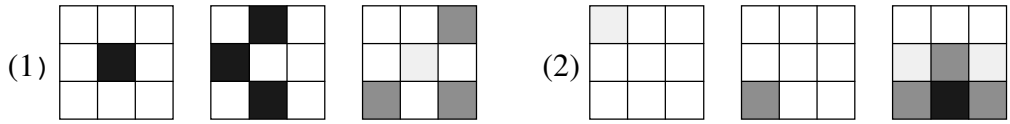
The raster approximation compression allows for a larger information density and makes it possible to use larger number of cells which, as we have seen, improves the filtering process and the decision ability in step 2. Nevertheless, this approach makes evident a problem that we have not faced before: the fact that the approximations will be of different sizes. One must have in mind, though, that this problems 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 from (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.

4.3.1 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 different possibilities for the cells, we need 18 bits to code each of them. Nevertheless, some kinds of cell are highly improbable of occurring, while some others are very frequent (Figure 7). 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 50% 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 7 - Example of cell groups

Our compression algorithm is basically an adaptation of Huffman's, in which the set of groups has been partitioned in four classes. The tree is constructed once, and it is based on the analysis of a large set of data. 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 trees reached as much as 15 Kbytes and the compression rates were about 40% of the original size. The compressed keys size are smaller than those in [BRINK94] (Table 6). 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.

	4CRS (250 Cells)	4CRS (350 Cells)	4CRS (550 Cells)	4CRS (750 Cells)
Bytes	19.7	25.44	33.8	42.5

Table 6 - Average size of compressed signatures

4.4 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 index, in the same way [BRINK94] does. This is a reasonable approach, since both the calculation 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. Furthermore, as we can see in Figure 8b, about 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 [BRINK94,BRINK93b].

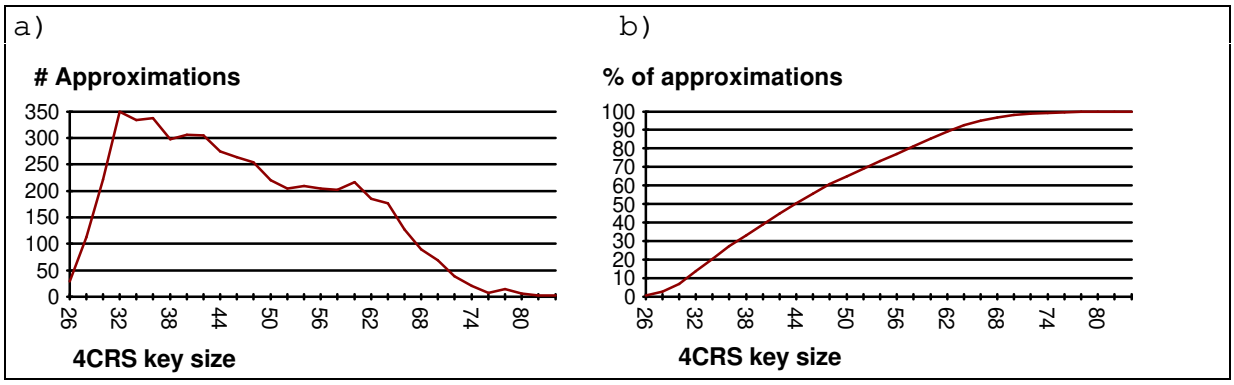


Figure 8 - a) Distribution of approximations/key size; b) Accumulative distribution.

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 [SAME90] 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.

4.5 Results

We now present some average results obtained in our work, to allow for a comparison with the other approximations (Tables 7 and 8). We have discarded the South USA results because of its unusual polygon uniformity.

Data Set (%intersecting)		Accepted	Rejected	Candidates (smaller is better)	Rejected Identified	Accepted Identified
Brazil	(a)	57.23%	23.23%	19.55%	67.29%	87.39%
(65.49%)	(b)	**59.96%**	**26.30%**	**13.74%**	**76.20%**	**91.56%**
	(c)	21.45%	25.52%	53.03%	72.97%	32.98%
South USA	(a)	78.78%	12.17%	9.04%	68.41%	95.84%
(82.21%)	(b)	**79.37%**	13.74%	**6.89%**	77.01%	**96.60%**
	(c)	51.92%	**14.62%**	33.46%	**81.03%**	63.35%
Europe	(a)	59.73%	20.49%	19.77%	65.33%	87.04%
(68.63%)	(b)	**62.28%**	**23.48%**	**14.24%**	**74.85%**	**90.75%**
	(c)	24.56%	22.45%	52.98%	70.51%	36.04%
South America	(a)	60.45%	21.22%	18.33%	66.02%	89.08%
(67.86%)	(b)	**62.48%**	**24.18%**	**13.34%**	**74.88%**	**92.28%**
	(c)	24.49%	23.01%	52.50%	70.91%	36.26%

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

	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	23.5%	23.6%	52.8%	71.4%	35.2%
5-C & ER (Europe A) [BRINK94]	23%	23%	54%	66%	30%
5-C & ER-EL (Europe A) [BRINK93b,94]	40.5%	20.8%	40.84%	66.3%	59.2%

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

Its important to note that the column ***Candidates*** in Table 8 is the determinant factor of the problem. These numbers shows the size of the input data set for the step 3, which dominates the total execution time. In this way, smaller rates indicates fewer polygon intersection tests to be done. As one can see, our average results (13.7% of intersecting

MBRs) exhibits an shrinking rate of more than 65% when compared against the best results previously presented (the last row, 40.84% of intersecting MBRs).

Our experiments have focused on the filter step. 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 using a smaller key. In this way, we focus on results obtained in the main memory part of our algorithm: the key decompression and cell comparison.

Although [BRINK94] 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 has 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, 5-C polygon use some floating point operations or double precision integer multiplications and divisions. These times were measured by first measuring the execution time spent on step 1 and subtracting it from the time spent on steps 1 and 2. We have to adopt this methodology because the spatial processor is not sequencial: the execution of step 2 occurs interspersed with step 1. The test machine is a SUN Sparc2 with 32 Mb main memory.

	Time (msec)
5C & ER	2340
4CRS	2480

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

One can expect a little increase for 5-C & ER-EL times. The time spent on step 3 was not measured, since [BRINK94] 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. 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 this factor reached 65% of reduction.

5. Variations of 4CRS

Many variations of 4CRS approach can be proposed to improve the overall performance of spatial join evaluation. In this work, we propose only two improvements: using more colors and using different kinds of intersection cells.

5.1 Using more colors (8CRS)

We can use more than four colors in a raster approximation in order to represent the amount of intersecting areas. Thus, depending on the polygon intersecting area, cells will be classified in a more accurate way. In this way, the next shows a plausible distribution using eight colors (3 bits).

Cell Type	Amount of Intersecting Area
Full	100%
Strong 3	between 83,33% e 100%
Strong 2	between 66,66% e 83,33%
Strong 1	between 50% e 66,66%
Weak 3	between 33,33% e 50%
Weak 2	between 16,66% e 33,33%
Weak 1	between 0% e 16,66%
Empty	0%

Table 10 - Kinds of Cells

Next table shows all the possible superposition of cells:

Cell Type	Empty	Weak 1	Weak 2	Weak 3	Strong 1	Strong 2	Strong 3	Full
Empty	Rejected	Rejected	Rejected	Rejected	Rejected	Rejected	Rejected	Rejected
Weak 1	Rejected	*Candidate*	*Candidate*	*Candidate*	*Candidate*	*Candidate*	*Candidate*	**Accepted**
Weak2	Rejected	*Candidate*	*Candidate*	*Candidate*	*Candidate*	*Candidate*	**Accepted**	**Accepted**
Weak 3	Rejected	*Candidate*	*Candidate*	*Candidate*	*Candidate*	**Accepted**	**Accepted**	**Accepted**
Strong 1	Rejected	*Candidate*	*Candidate*	*Candidate*	**Accepted**	**Accepted**	**Accepted**	**Accepted**
Strong 2	Rejected	*Candidate*	*Candidate*	**Accepted**	**Accepted**	**Accepted**	**Accepted**	**Accepted**
Strong 3	Rejected	*Candidate*	**Accepted**	**Accepted**	**Accepted**	**Accepted**	**Accepted**	**Accepted**
Full	Rejected	**Accepted**	**Accepted**	**Accepted**	**Accepted**	**Accepted**	**Accepted**	**Accepted**

Table 11 - Possible Outcome of Cell Matching

One can note that, in comparison to 4CRS approach, the new colors used have changed only the classification of cells that were formerly Strong and Weak. The amount of Full and Empty cells remains the same in both approaches, as expected. Thus, we conclude that only the indecision area changes must be compared. Using the 4CRS approach 25% of superposition cell cases were accepted in the indecision area. Now, using the 8CRS approach 42,66% of superposition cell cases were accepted in the indecision area, an even better result.

However, one can observe that in the indecision area, both 4CRS and 8CRS approaches evaluates to accept or inconclusive, never to reject. Therefore, by using eight colors we can expect an improvement only in the rate of identification of accepted cases. The rate of identification of rejected cases will remain the same. As in our approach the former rate is already the better rate, the overall improvement on spatial join performing is marginal. Thus, we have decided to omit those results.

5.2 Using Special Cells

Another way to obtain better results is to improve the quality of approximations, by means of using special cells. These special cells should keep track of polygon intersecting area distribution inside the cell. Table 12 shows some examples. The cells Full, Empty, Strong and Weak were omitted because no modification in their definition was done.

Cell	Description
	Cell with intersection area only in the Northeast.
	Cell with intersection area only in the Southeast.
	Cell with intersection area only in the Northwest.
	Cell with intersection area only in the Southwest.

Cell	Description
	Cell with intersection area strongly in the Northeast.
	Cell with intersection area strongly in the Southeast.
	Cell with intersection area strongly in the Northwest.
	Cell with intersection area strongly in the Southwest.

Table 12 - Variations of cell format

Although in these approach we can evaluate to reject even in indecision area, the results achieved were just a little better than the results of 4CRS approach. The main reason for that is the necessity of many bits in order to have a reasonable number of different kinds of cells, that increase the space needs for storing the signature. Again, due to marginal improvement over the original method, we have decided to omit the results.

5.3 Using More Points

The use of a big amount of points is the best of all alternatives studied, able to lead to better results. So, it is necessary the development of more efficient compression strategies, in order to use a big amount of points without paying the price of excessively big keys. In the next table we can observe that there is a sub linear correspondence between the number of points and the signature size. In other words, when we increase the number of points, the size of the signature also increase, but not in the same reason - proportionally, the signature increase is smaller than the point increase. This good result can be achieved because, as we

pointed before, when the number of points grows, there is an increase of Full and Empty cells, leading to a decrease of indecision area in the signature. In this way, signature entropy tends to diminish, resulting in a better rate of compression.

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, and some variants of this approximation.

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), 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 65%. 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 [BRINK94,MING96]. So, for any algorithm used in step 3, there will be a shrinking on size of the input data set, resulting in smaller CPU and I/O costs.

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.

Finally, alternates 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, allowing the use of more points.

7. References

[BERC96] 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.

[BRINK93a] 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.

[BRINK93b] T. Brinkhoff, H. P. Kriegel, and R. Schneider: "Comparison of Approximations of Complex Objects Used for Approximation-based Query Processing in Saptial Databse Systems". In Proceedings of 9th International Conference on Data Engineering, Vienna, Austria, 1993.

[BRINK94] 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.

[BRINK95] 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 Informatin Systems, Baltimore, MD, USA, December 1995.

[BRINK96] 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.

[GAED96] Gaede, V. and Faloutsos, C.: "Analysis of n-dimensional Quadtrees Using the Hausdorff Fractal Dimension". In Proceedings of the 22nd VLDB Conference Mumbai(Bombay), India, 1996.

[GÜNT93] O. Günther: "Efficient Computation of Spatial Joins". In Proceedings of 9th International Conference on Data Engineering, Vienna, Austria, 1993.

[IBGE96] Fundação Instituto Brasileiro de Geografia e Estatística: "Malha Municipal Digital do Brasil - 1994", Rio de Janeiro, 1996.

[MING95] M. L. Lo and C. V. Ravishankar: "Generating Seede Trees From Data Sets". In Proceedings of 4th International Symposium on Large Spatial Databses, Portland, ME, USA, August, 1995.

[MING96] M. L. Lo and C. V. Ravishankar: "Spatial Hash-Joins". In Proceedings of the 1996 ACM-SIGMOD Conference, Montreal, Canada, June 1996.

[PATE96] J. M. Patel and D. J. DeWitt: "Partition Based Spatial-Merge Join". In Proceedings of the 1996 ACM-SIGMOD Conference, Montreal, Canada, June 1996.

[SAIF94] 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.

[SAME90] H. Samet: "The Design and Analysis of Spatial Data Structures", Addison-Wesley Publishhing Company, 1990.

[SQL395] X3H2-95-084/DBL:YOW-004, (ISO-ANSI Working Draft) Database Language SQL (SQL3),Jim Melton (ed.), March, 1995

[VEEN95] H. M. Veenhof, Peter M. G. Apers, Maurice A. W. Houtsma: "Optimisation of Spatial Joins Using Filters". In Advances in Databases, 13th British National Conference on Databases, BNCOD 13, Manchester, United Kingdom, July 1995.