A SURVEY OF HIERARCHICAL PARTITIONING METHODS FOR VECTOR IMAGES

Valerian T Noronha[1]

Abstract Vector image data formats have some unique functional capabilities, but spatial objects are difficult to retrieve on a two-dimensional key. This article examines a number of strategies for speeding up the object retrieval process by partitioning an image into manageable subsets. The emphasis is on solutions that can be implemented on microcomputers, for image data bases up to about 10 Mb.

Introduction

Irregularly shaped spatial objects in two or more dimensions, stored in uni-dimensional computer file address space, pose special problems in retrieval. The tesselated data model skirts the problem by viewing objects as part of a universal phenomenon, varying only in degree, and imposing regular, readily addressable partitions on the universe. It is therefore efficient in performing tasks involving complex data retrieval and comparison (for example, analytical overlay). Vector methods on the other hand, despite several unique functional capabilities (for example, in the area of topological relations and network analysis), are so hindered by object retrieval problems that their performance on several tasks is unacceptably slow. This is particularly evident in operations such as selective plotting, point-in-polygon enquiry and overlay, where parcels of space must be examined serially.

Take the query, "In which country is Ottawa (45°25′N, 75°43′W)?" Given a traditional vector data base of international outlines in closed-polygon spherical coordinates, this operation requires the sequential examination of every polygon in the file, including obvious non-candidates such as Cuba and Cyprus. Equally nonsensical operations are required in, say, plotting a selected portion of the world, for example the Middle East. Objects from Alaska to Antarctica are blithely read, passed to the coordinate transformation and clipping routines, and eventually found to lie outside the display window. Retrieval time is dependent on the number of objects in the file, whereas it should logically depend only on the number of objects retrieved.

The BRP

The minimum bounding rectangle (MBR) is a simple device often used to eliminate unnecessary geometric computations. Being aligned with the coordinate axes, the MBR is defined by just four real numbers: either the positive diagonal of the rectangle, or its centre and dimensions. In the example above, the point-in-polygon routine to locate Ottawa could eliminate Cuba, Cyprus and every country other than the United States (part of which lies north of the 45th parallel) by trivial rejection of the MBR of each of those countries. The query could thus be solved with about 150 quick rectangle checks and 2 complete point-in-polygon tests.

The MBR is easily incorporated into a file structure by prefixing each object with the bounding rectangle — hence the Bounding Rectangle Prefix, or BRP. The four additional words required to describe each object are usually not a major storage overhead, given the subjectivity of manual digitizing. They could add substantially to the size of images composed largely of short strings, such as networks or sparsely digitized files, but there are techniques to reduce the overhead (Noronha, 1988).

It is difficult to understand why BRP'd structures are often not employed in commercial systems. There is little programming skill or effort required in computing and testing the bounding rectangles, yet the benefits in performance are considerable. In the area of microcomputer-based mapping, virtually every vendor boasts a "proprietary" image file format. Informal tests by the author indicate that these are usually just binary transcriptions of length-encoded coordinate strings, as in Figure 1. The only additional information

[1] Presented at the Third International Symposium on Spatial Data Handling, Sydney, 1988. At the time of writing, Dr Noronha was Assistant Professor at the University of Alberta. He can now be reached at *noronha@dgrc.ca*

in the binary file is usually a header record containing the global MBR, which permits instantaneous determination of plotting scale.

"Label", n

$X_1, Y_1, \quad X_2, Y_2, \quad \ldots \quad X_n, Y_n$

Figure 1. Spatial object in popular "IMA" image file structure

$X_{min}, Y_{min}, X_{max}, Y_{max}$, n, "Label"

$X_1, Y_1, \quad X_2, Y_2, \quad \ldots \quad X_n, Y_n$

Figure 2. BRP'd file format

A corresponding BRP'd format is shown in Figure 2. For each object, the first record (the MBR) is read; if the rectangle can be trivially rejected, a dummy word is read from the coordinate record and the program proceeds to the next object. This technique requires no elaborate file structuring. Its effectiveness depends on the size of the image file (*M*), the number of objects retrieved (*r*) and the length of each object.

Figure 3 shows how retrieval time varies with r and M, using files with and without BRPs. It is based on tests with a 700 Kb file (M=1300) on an IBM PC-AT. Without BRP checking, plotting requires 93 seconds (r=4) to 106 seconds (r=1300). A BRP test reduces plotting time to about 6 seconds (r=4).

These observations, albeit crude, dramatically illustrate two issues. First, clearly, there are substantial advantages to be gained from BRPs, since regular retrieval times are dependent on *M* rather than *r* (the horizontal line in Figure 3), and are unacceptable for files containing more than a handful of objects. The second issue is the basis of the remainder of this paper: while BRP testing reduces retrieval time substantially, 1.5 seconds per object is well in excess of the documented 0.1 second capability of the hardware, and this disparity increases with *M* .

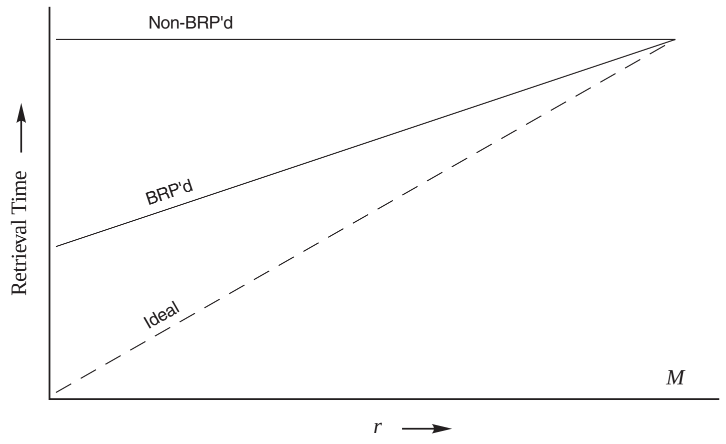

Figure 3. Plotting time with and without BRPs

The BRP, it must be emphasized, is an interim, easily-programmed solution, in which retrieval time is dependent on both *M* and *r*. As *M* increases beyond a few hundred objects, superior techniques, in which time depends exclusively on *r*, are required.

Multi-Level BRPs: Spatially Partitioned Images

The discussion above has introduced MBRs at two levels: first, in defining global coordinate limits to determine plotting scale, and secondly as BRPs around each object. One may define additional levels of MBRs in image files, creating hierarchical storage structures.

Returning to the Ottawa example, rather than testing the BRPs of each country in the world, it would clearly be more appropriate to examine whole continents, rather than countries, in an initial scan. That would eliminate Europe, Africa, Asia and Australia. America would test positive, forcing a lower-level examination. Most American countries would be rejected on the basis of their BRPs; only Canada and the United States would require complete point-in-polygon tests. The query, when solved using this simple hierarchical search, requires about 25 rectangle tests and 2 point-in-polygon checks.

If large sections of the data base are to be ignored in the search process, the data must be sorted into spatially coherent units, and placed in random-access storage media. Retrieval of a section of the data is then a matter of specifying the storage address of the required record. This is the essence of a Spatially Partitioned Image File (SPIF). It can be implemented satisfactorily on microcomputers using random-access disc files.

Requirements of a Spatial Data Structure

The example above uses familiar political boundaries to illustrate a hierarchical organization of space. However, in situations where recognizable geographic partitions do not exist, such as in natural vegetation or soil maps, determination of partitions is subjective and non-reproducible. From the standpoint of operational efficiency, there is a need for an *optimal* partitioning of the data base. This raises the need for a theory of spatial partitioning for organizing image data bases. It would be appropriate to begin with a formal treatment of data structures and retrieval queries.

Queries to the data base may be categorized by the nature of the query and the type of object to be retrieved. The retrieval key may be spatial (based on the location of the object, for example, "Show all cities in Canada") or aspatial (based on other attributes of the object, such as "Show all cities with populations over 1 million"). This paper is concerned only with spatial keys.

Queries and structures for points

A simple spatial query is: Given a set S of points (X,Y), is a given point P(x,y) contained in the set? If the points are assumed to lie along a horizontal line, the data are best stored in ascending order of x-coordinate, and structured in a binary search tree (Knuth, 1973). When the points lie in 2-dimensional space, the binary search must be performed on two dimensions. This requires that the points be sorted with their y-coordinates as the secondary sort key. The search in the first dimension, say X-, reduces the number of candidates to those with an abscissa exactly equal to x.

The corresponding *orthogonal range query* is: which of the points in S lie *between* $P_1(x_1,y_1)$ and $P_2(x_2,y_2)$. In one dimension ($y_1=y_2$), this is a search for points that project on to a specified line segment, and is solved by a range search: a simple extension of the binary search technique. In two dimensions, P_1 and P_2 define a rectangular query area. A range search is performed on each dimension. Now, however, the candidates remaining at the end of the X-search have several different abscissas, and must be re-sorted by ordinate to enable the Y-search. Clearly, there are difficulties in organizing a data base to suit regular binary searches on both dimensions.

Examples of more complex point queries may be to list all points in S that lie within an irregular polygon; to list all points in S that lie within a distance d of P(x,y); or to find the points in S that are nearest to or furthest from P(x,y). They are essentially variants of the orthogonal range search.

A number of techniques have been developed to store point sets in hierarchical spatial structures. These include the point quadtree (Finkel & Bentley, 1974), essentially a multi-dimensional binary search tree (Knuth, 1973); the k-d tree (Bentley, 1975) and others (Burton, 1978; Shamos and Bentley, 1978; Bentley and Friedman, 1979; Samet, 1984). Most of these are extensions of well known 'region'-quadtree methods. They are reviewed by Samet (1984).

Queries for lines and areas

The coordinate string or polyline (Figure 1) is a generic structure for storing features other than points. It accommodates both closed and open polygons, can be complexed to describe islands and lakes, and can be supplemented with topological information to represent networks. As indicated above, a string may initially be summarized by its bounding rectangle.

As with point data, there are two principal classes of spatial queries: given a set R of rectangles, find all objects that contain the point P(x,y); and find all objects that intersect a search window $Q(x_1,y_1,x_2,y_2)$. Again, there are more complex queries, such as finding all objects that intersect an irregular polygon, finding the nearest or furthest object from a point or other object, etc. In the special case where the objects are disjoint polygons (e.g. political provinces in a country), and the query object is a point or small rectangle (i.e. the point inclusion problem), a 'hit' terminates the search; this is not true in the case of range searches. Note that to secure a hit, it is necessary to test the object, not just its MBR.

Structures For Lines And Areas

Characteristics of tree structures

The concept of the tree is central to hierarchical data structuring. Dendrograms are used in several disciplines to organize descriptive taxonomies, and in statistics to illustrate data clusters. A number of branching structures are recognized in the context of geographic regionalization (Grigg, 1965; 1967).

Search trees differ from classification dendrograms in that they are designed specifically to optimize the search process. First, efficient traversal of a search tree relies on the difference in ordinal or interval values at each branch. Classification trees on the other hand may branch on the basis of a nominal variable; pruning of irrelevant branches is therefore less likely. Secondly, the form of a tree has a direct bearing on its efficacy in navigating a data set. Knuth (1973) and Aho et al (1983) discuss properties of trees that optimize access to data objects.

Two principal classes of tree-based rectangle organization methods have appeared in the computer science literature. Non-disjoint rectangle systems are intuitively appealing to the geographer, since they mimic — and can accommodate — geographic taxonomic trees, such as the political systems illustrated above. Disjoint structures are regular geometric decompositions of space, and depend little on the actual spatial configuration.

Hierarchical rectangle management

Geographic features on a given layer are typically disjoint, if coded in topologically correct form; however, their bounding rectangles might intersect. The following techniques handle sets of non-disjoint object-oriented rectangles.

R-tree

The R-tree (Guttman, 1984) — the R stands for rectangle, not range — is a geographically appealing and intuitive structure. The BRPs of spatial elements are aggregated into successively larger rectangles, each higher order module completely enclosing all its descendants (Figure 4) — as in the continent-country illustration earlier in this paper. Each MBR is allocated to one and only one higher order rectangle, although there may be more than one potential parent (for example, the US state of Maine is completely contained within both the Canadian and US MBRs). Guttman describes algorithms for initial construction of the tree, and for insertion and deletion of elements.

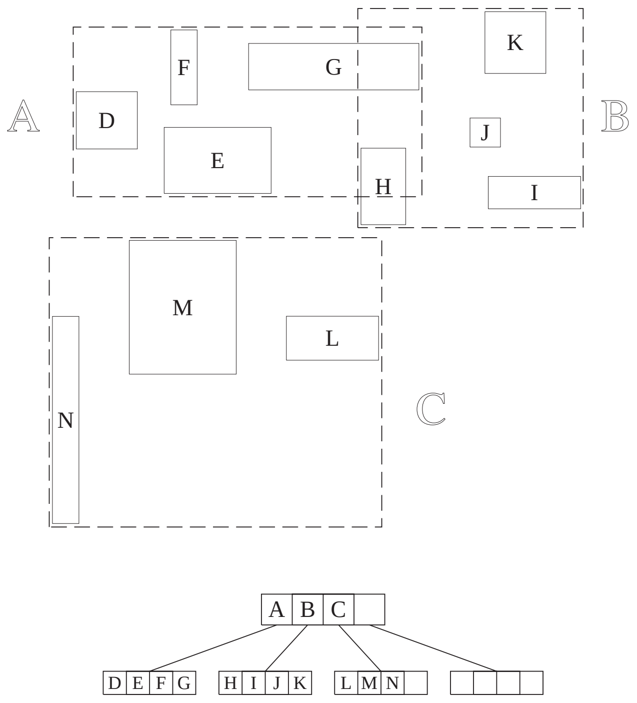

Figure 4. Rectangles organized in an R-tree (after Faloutsos et al, 1987)

There are two aspects to evaluating such a structure: first, the procedures for accessing a chosen object in the file, and second, the criteria for aggregating modules into higher order units. On the issue of access, it is clear from the definition of the R-tree that siblings at a node may intersect. Further, they contain varying degrees of irrelevant space. In a point or range search, therefore, a parent rectangle may test positive even if *none* of its sons contains the key. The accumulated wild goose chases takes away from the efficiency of the search.

Accordingly, the objective in aggregation is to minimize the chance that a rectangle tests positive when all its descendants test negative. This amounts to minimizing the excess area (the parent rectangle less the union of its sons) or "coverage," created by aggregation at a node. The pair of objects whose composite MBR would generate the most wasted coverage are therefore selected as poles for branching.

The effectiveness of the R-tree structure in a given spatial context depends on the "hierarchical partitionability" of the set of objects. Objects that are perfectly nested and whose MBRs are disjoint (as in some geometrically defined political units) lend themselves to efficient hierarchical organization.

Variants on the R-tree

A major problem with the R-tree is the intersection of siblings. A given object may be enclosed by more than one higher-order rectangle, although it is allocated to only one parent (as illustrated in the case of Maine, above); in a point or range search, therefore, even when the query point or rectangle is completely enclosed by a data rectangle, all siblings of the data rectangle must still be tested for possible intersection with the query area. A query rectangle completely contained in Maine, for example, would force all of Canada to be examined.

If it could have been assumed that siblings were disjoint, complete containment of the search rectangle would have obviated further examination of siblings. Disjointness is most useful at a high level in the tree, since it leads to trivial rejection of a larger portion of the data. An alternative to the wasted-space aggregation criterion of the R-tree would therefore be to minimize or eliminate intersection between rectangles at as high a level in the tree as possible. The packed R-tree (Roussopoulos and Leifker, 1985) organizes the tree to minimize intersection, while simultaneously trying to minimize coverage.

The R$^+$-tree (Faloutsos et al, 1987) eliminates rather than minimizes intersection. It identifies problem objects (such as G in Figure 4), assigning them to more than one parent, without clipping the object (Figure 5). Since non-leaf siblings in the tree are disjoint, the search process is more efficient. However, since leaf rectangles may now have more than one parent, they may be retrieved more than once in a given search. The retrieval routine must therefore maintain a list against which to check each object as it is fetched from the master data base. Although only leaf nodes are multiply referenced, this can be a problem when r is large.

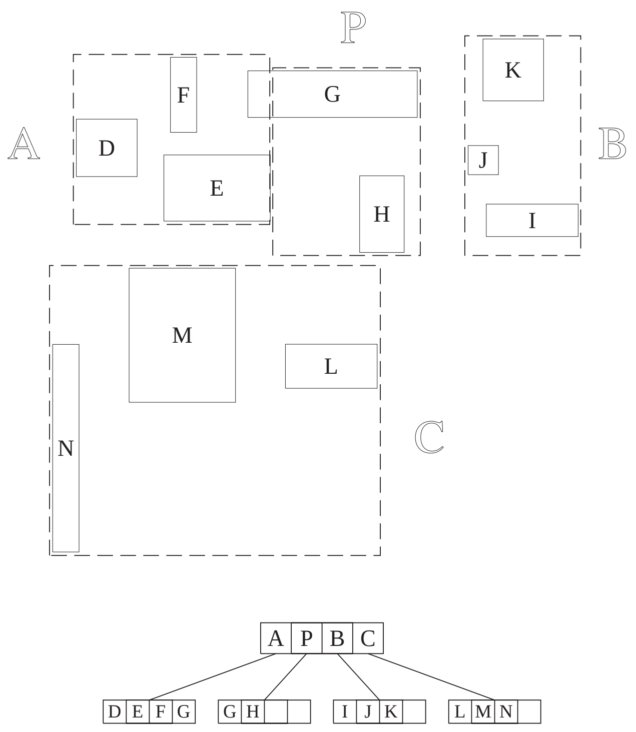

Figure 5. Rectangles organized in an R+ tree (after Faloutsos et al, 1987)

Quad-CIF quadtree

The quad-CIF quadtree (Kedem, 1982) offers an alternative method for managing sets of rectangles. The universe is recursively decomposed into quadrants until every rectangle is skewered (Figure 6). At each stage of subdivision, the data rectangles intersected by the division lines are assigned to the current subdivision node; thereafter they may not be assigned to descendants of that node (for example, the rectangle M in Figure 6 is stored, along with G and L, at the first subdivision node only; it is not associated with the lower order lines that intersect it). Queries to the data base are handled by variants of quadtree set operation algorithms (Hunter and Steiglitz, 1979).

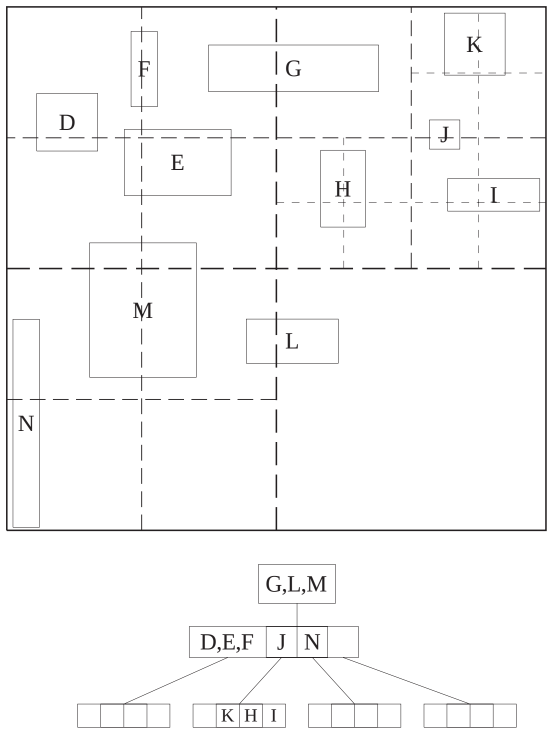

Figure 6. Rectangles organized in a Quad-CIF tree

Regular spatial decomposition

In this second class of trees, the universe is recursively decomposed with geometric regularity, with no reference to the data except to determine the outer limits of the coverage. Spatial objects are associated with those regular divisions, rather than with their custom-moulded BRPs (the quad-CIF tree seems to fit this definition too, but is really a hybrid).

Tesselated data have traditionally been stored in regularly decomposable forms, using quadtrees or complex tilings (Bell et al, 1983). Vector strings on the other hand pose special problems, since geographic objects simply do not respect arbitrary geometric boundaries. There are three solutions to this problem, as outlined below.

Fragmentation

The first is to clip the objects wherever they cross the grid lines, in much the same way as conventional map sheets divide space at neat lines. This is unacceptable in digital files, for two reasons. First, fragmentation of objects in this manner causes unsightly bleeding and delays in cartographic plotting as the pen is lifted and restored to the paper. Secondly, analytical queries to the data base may require that the fragments be reconstituted into whole objects. The PM family of quadtrees (Samet and Webber, 1985; Nelson and Samet, 1986), while it does involve fragmentation of objects, takes an approach to storage that is radically different from what is being discussed in this paper. The same can be said of the strip tree (Ballard, 1981) and vaster (Peuquet, 1983).

Multiple referencing

A second solution is to follow the object through the various grid cells, and to maintain multiple references to it. This results in a multi-parent structure as in R$^+$-trees, and the same criticisms apply, although information provided confidentially to the author indicates that at least one commercial GIS follows this approach.

Multi-level storage

The third strategy would be to associate objects with several levels in the tree, that is, with interior nodes as well as leaf nodes, as in quad-CIF trees. In the Field Tree (Frank, 1983), each object is stored in a subdivision large enough to accommodate it without intersection (the decomposition and tree are identical to the quad-CIF structure, Figure 6, up to this point in the discussion).

Given the nature of tree searches, the object should ideally be associated with the lowest order node possible. There will unfortunately be objects which, although small enough to be accommodated at a low level, are intersected by higher-order subdivision lines purely because of their position. For example, a small circular object at the centre of the universe is contained in all four principal quadrants, no matter how finely the space is subdivided. Such "orphans" clutter the higher levels of storage, degrading efficiency.

The Field Tree incorporates an ingenious device to overcome the problem of orphans. Each level of subdivision is offset relative to the parent (Figure 7), so that some of the smaller orphans can be accommodated in lower-level cells.

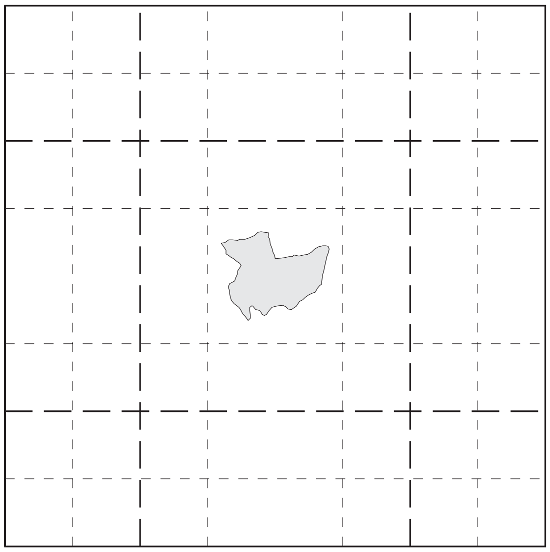

Figure 7. Field tree, showing an orphan adopted by translated lower-order field

Concluding Remarks

In devising a practical data structure, aside from retrieval time (average, best- and worst-cases), one must consider storage space requirements and the pre-processing needed to structure the data. If pre-processing is complex and protracted, one must consider dynamic procedures for inserting and deleting data records without having to repeat the entire preparatory procedure each time the data are changed. Additional factors that may be considered are

- efficiency of principal analytical algorithms (point-in-polygon, etc) when the data are stored in the required structure
- simplicity of retrieval/pre-processing algorithms, from the point of view of the programmer
- flexibility of the data structure

These criteria are self-evident, with the possible exception of the last. A general-purpose hierarchical structuring of space should ideally recognize and accommodate partitioning criteria other than those related to computing efficiency. First, there are well established geographic hierarchies in existence (political boundaries are an example). It may not always be acceptable, for instance, to assign the north eastern United States to Canada, in the interests of optimizing a tree. Whether for irrational emotional reasons or as a matter of practical convenience, the user should ideally be able to override a machine-determined partitioning. Second, there may be potential for a system that adapts itself to observed patterns of access to data, optimizing best-case rather than average-case retrieval. Adjustments would be required in the structure of the tree, or in the partitioning criteria.

Of the structures above, the generic R-tree rectangle management system is the only one that offers this flexibility. This is probably what makes it intuitively appealing, since one instinctively associates it with familiar political hierarchies. In other respects, however, the R-tree is not so attractive. Initial construction of the tree is expensive and slow — the algorithms described by Guttman are in fact heuristics. Since partitioning is irregular, the coordinates of rectangles corresponding to interior nodes in the tree must be stored; and access is suboptimal for reasons indicated earlier.

The evaluation of the data structures and partitioning algorithms presented above is descriptive rather than rigidly analytical. There is limited scope for formal comparisons other than best- and worst-case performance estimates, since spatial data sets have unique properties. There is a need for effective spatial statistics to determine the appropriateness of various data structures in a given configuration.

Finally, notwithstanding the theoretical distinctions, all the methods described above offer dramatic reductions in object retrieval time. Most of them are easily programmed; pre-processing is usually quick, and retrieval times are almost linear in r. Images up to 10 Mb — a substantial portion of World Data Bank II — can now be processed at speeds that may be considered convenient even on microcomputers.

Acknowledgements

This research was supported by grant #A4191 from the National Sciences and Engineering Research Council of Canada. I am grateful to Andrew Frank, Toni Guttman, Donna Peuquet, Nick Roussopoulos and Hanan Samet for their input.

References

Aho A V, Hopcroft J E, Ullman J D, 1983. *Data Structures and Algorithms* (Addison Wesley)

Ballard D H, 1981. "Strip trees: a hierarchical representation for curves" *ACM Communications* 24 (5): 310-321; Corrigendum, 25: 213

Bell S B M, Diaz B M, Holroyd F, Jackson M J, 1983. "Spatially referenced methods of processing raster and vector data" *Image and Vision Computing* 1 (4): 211-220

Bentley J L, 1975. "Multidimensional binary search trees used for associative searching" *ACM Communications* 18 (9): 509-517

Bentley J L, Friedman J H, 1979. "Data structures for range searching" *Computing Surveys* 11 (4): 397-409

Burton W, 1978. "Efficient retrieval of geographical information on the basis of location" In Dutton G (Ed) *First International Advanced Study Symposium on Topological Data Structures for Geographic Information Systems* (Harvard Papers on GIS, Cambridge MA)

Faloutsos C, Sellis T, Roussopoulos N, 1987. "Analysis of object oriented spatial access methods" *ACM SIGMOD* 426-439

Finkel R A, Bentley J L, 1974. "Quad trees: a data structure for retrieval on composite keys" *Acta Informatica* 4: 1-9

Frank A U, 1983. *Storage Methods for Space Related Data: the Field Tree* (Institut für Geodäsie und Photogrammetrie, ETH, Zurich. Bericht Nr 71)

Grigg D B, 1965. "The logic of regional systems" *Annals, Assn of American Geographers* 55: 465-491

Grigg D B, 1967. "Regions, models and classes" In Chorley R J, Haggett P (Eds) *Integrated Models in Geography* (Methuen, London)

Guttman A, 1984. "R-trees: a dynamic index structure for spatial searching" *ACM SIGMOD* 47-57

Hunter G M, Steiglitz K, 1979. "Operations on images using quad trees" *IEEE, Pattern Analysis and Machine Intelligence* 1: 145-153

Kedem G, 1982. "The quad-CIF tree: a data structure for hierarchical on-line algorithms" *Proceedings, Design Automation Conference, Las Vegas* 352-357

Knuth D E, 1973. *The Art of Computer Programming. Vol 3: Sorting and Searching* (Addison Wesley)

Nelson R C, Samet H, 1986. "A consistent hierarchical representation for vector data" *ACM SIGGRAPH* 20 (4): 197-206

Noronha V T, 1988. "On ordering spaghetti: the BRP and other strategies" Presented at annual meeting of Association of American Geographers, Phoenix AZ

Peuquet D J, 1983. "A hybrid structure for the storage and manipulation of very large spatial data sets" *Computer Vision, Graphics and Image Processing* 24 (1): 14-27

Roussopoulos N, Leifker D, 1985. "Direct spatial search on pictorial databases using packed R-trees" *ACM SIGMOD* 17-31

Samet H, 1984. "The quadtree and related hierarchical data structures" *Computing Surveys* 16 (2): 187-260

Samet H, Webber R E, 1985. "Storing a collection of polygons using quadtrees" *ACM Transactions on Graphics* 4 (3): 182-222

Shamos M I, Bentley J L, 1978. "Optimal algorithms for structuring geographic data" In Dutton G (Ed) *First International Advanced Study Symposium on Topological Data Structures for Geographic Information Systems* (Harvard Papers on GIS, Cambridge MA)