Appeared in the *GeoSpatial Visual Analytics: Geographical ormation Processing and Visual Analytics for Environmental Security*, pp. 31–43. Springer Business Science Media, Berlin, 2009.

SORTING SPATIAL DATA BY SPATIAL OCCUPANCY [*]

Hanan Samet
Center for Automation Research
Institute for Advanced Computer Studies
Computer Science Department
University of Maryland
College Park, Maryland 20742, USA
*(*hjs@cs.umd.edu*)*
www.cs.umd.edu/˜hjs

Abstract. The increasing popularity of web-based mapping services such as Microsoft Virtual Earth and Google Maps/Earth has led to a dramatic increase in awareness of the importance of location as a component of data for the purposes of further processing as a means of enhancing the value of the nonspatial data and of visualization. Both of these purposes inevitably involve searching. The efficiency of searching is dependent on the extent to which the underlying data is sorted. The sorting is encapsulated by the data structure known as an index that is used to represent the spatial data thereby making it more accessible. The traditional role of the indexes is to sort the data, which means that they order the data. However, since generally no ordering exists in dimensions greater than 1 without a transformation of the data to one dimension, the role of the sort process is one of differentiating between the data and what is usually done is to sort the spatial objects with respect to the space that they occupy. The resulting ordering should be implicit rather than explicit so that the data need not be resorted (i.e., the index need not be rebuilt) when the queries change. The indexes are said to order the space and the characteristics of such indexes are explored further.

Key words: Spatial indexing; Sorting; Geometric data structures.

1. Introduction

The increasing popularity of web-based mapping services such as Microsoft Virtual Earth and Google Maps/Earth has led to a dramatic increase in awareness of the importance of location as a component of data for the purposes of further processing as a means of enhancing the value of the nonspatial data and of visualization. Both of these purposes inevitably involve searching. The efficiency of searching is dependent on the extent to which the underlying

[*] This work was supported in part by the National Science Foundation under Grants CCF-05-15241, IIS-0713501 and IIS-0812377, Microsoft Research, and NVIDIA.

data is sorted. The conventional definition of the term *sort* is that it is a verb meaning:

1. To put in a certain place or rank according to kind, class, or nature
2. To arrange according to characteristics.

The sorting is encapsulated by the data structure that is used to represent the spatial data thereby making it more accessible. In fact, the term *access structure* or *index* is often used as an alternative to the term *data structure* in order to emphasize the importance of the connection to sorting.

The notion of sorting is not new to visualization applications. One of the earliest examples is the work of Warnock who, in a pair of reports that serve as landmarks in the computer graphics literature (Warnock, 1968; Warnock, 1969), described the implementation of hidden-line and hidden-surface elimination algorithms using a recursive decomposition of the picture area. The picture area is repeatedly subdivided into rectangles that are successively smaller while it is searched for areas that are sufficiently simple to be displayed. It should be clear that the determination of what part of the picture area is hidden or not is equivalent to sorting the picture area with respect to the position of the viewer. This distinction is also present in back-to-front and front-to-back display algorithms. These algorithms form the rationale for the BSP tree representation (Fuchs et al., 1983; Fuchs et al., 1980) which facilitates visibility calculations of scenes with respect to a viewer as an alternative to the z-buffer algorithm which makes use of a frame buffer and a z buffer to keep track of the objects that it has already processed. The advantage of using a visibility ordering over the z-buffer algorithm is that there is no need to compute or compare the z values. Sorting is also used to accelerate ray tracing by speeding up the process of finding ray-object intersections (e.g., (Glassner, 1984; Samet, 1989b; Samet, 1989a)).

Notwithstanding the above definition, sorting usually implies the existence of an ordering. Orderings are fine for one-dimensional data. For example, in the case of individuals we can sort them by their weight, and given an individual such as Bill, we can use the ordering to find the person closest in weight to Bill. Similarly, we can use the same ordering to also find the person closest in weight to John. Unfortunately, in two dimensions and higher, such a solution does not always work. In particular, suppose we sort all of the cities in the US by their distance from Chicago. This is fine for finding the closest city to Chicago, say with population greater than 200,000. However, we cannot use the same ordering to find the closest city to New York, say with population greater than 200,000, without resorting the cities.

The problem is that for two dimensions and higher, the notion of an ordering does not exist unless a dominance relation holds (e.g., (Preparata and

Shamos, 1985))—that is, a point $a = \{a_i | 1 \leq i \leq d\}$ is said to dominate a point $b = \{b_i | 1 \leq i \leq d\}$ if $a_i \leq b_i, 1 \leq i \leq d$. Thus the only way to ensure the existence of an ordering is to linearize the data as can be done, for example, using a space-filling curve (e.g., (Sagan, 1994; Samet, 2006)). The problem with such an approach is that the ordering is explicit. Instead, what is needed is an implicit ordering so that we do not need to resort the data when, for example in our sample query, the reference point for the query changes (e.g., from Chicago to New York). Such an ordering is a natural byproduct when we sort objects by spatial occupancy, and is the subject of the remainder of this paper.

2. Methods Based on Spatial Occupancy

The indexing methods that are based on sorting the spatial objects by spatial occupancy essentially decompose the underlying space from which the data is drawn into regions called *buckets* in the spirit of classical hashing methods with the difference that the spatial indexing methods preserve order. In other words, objects in close proximity should be placed in the same bucket or at least in buckets that are close to each other in the sense of the order in which they would be accessed (i.e., retrieved from secondary storage in case of a false hit, etc.).

There are two principal methods of representing spatial data. The first is to use an object hierarchy that initially aggregates objects into groups based on their spatial proximity and then uses proximity to further aggregate the groups thereby forming a hierarchy. Note that the object hierarchy is not unique as it depends on the manner in which the objects were aggregated to form the hierarchy. Queries are facilitated by also associating a minimum bounding box with each object and group of objects as this enables a quick way to test if a point can possibly lie within the area spanned by the object or group of objects. A negative answer means that no further processing is required for the object or group, while a positive answer means that further tests must be performed. Thus the minimum bounding box serves to avoid wasting work. Data structures such as the R-tree (Guttman, 1984) and the R*-tree (Beckmann et al., 1990) illustrate the use of this method.

As an example of an R-tree, consider the collection of straight line segment objects given in Figure 1(a) shown embedded in a 4×4 grid. Figure 1(b) is an example of the object hierarchy induced by an R-tree for this collection. Figure 1(c) shows the spatial extent of the bounding rectangles of the nodes in Figure 1(a), with heavy lines denoting the bounding rectangles corresponding to the leaf nodes, and broken lines denoting the bounding rectangles corresponding to the subtrees rooted at the nonleaf nodes.

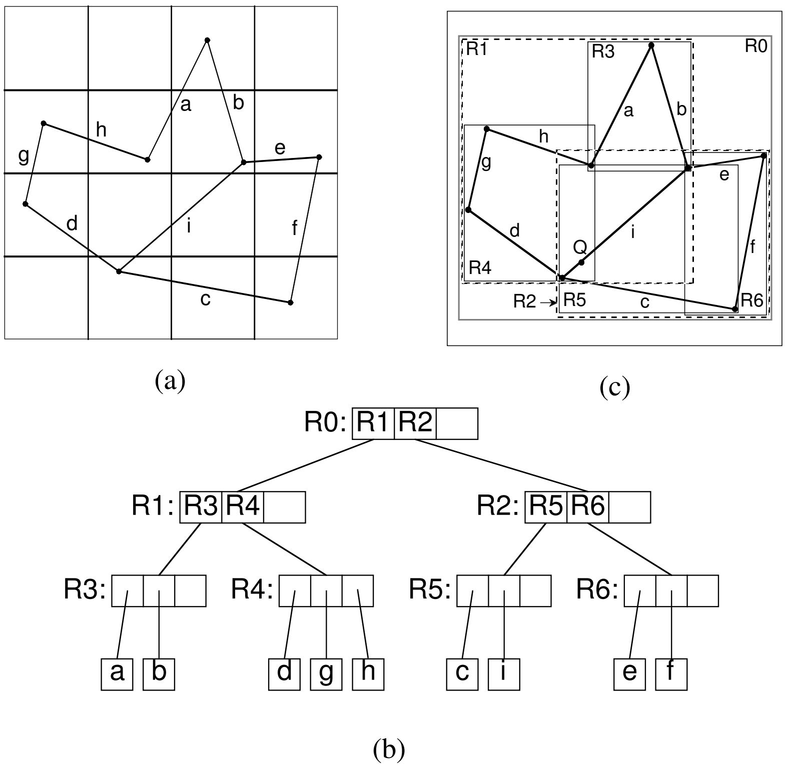

Figure 1 (a) Example collection of straight line segments embedded in a 4×4 grid, (b) the object hierarchy for the R-tree corresponding to the objects in (a), and (c) the spatial extent of the minimum bounding rectangles corresponding to the object hierarchy in (b). Notice that the leaf nodes in the (c) also store bounding rectangles although this is only shown for the nonleaf nodes.

The drawback of the object hierarchy approach is that from the perspective of a space decomposition method, the resulting hierarchy of bounding boxes leads to a non-disjoint decomposition of the underlying space. This means that if a search fails to find an object in one path starting at the root, then it is not necessarily the case that the object will not be found in another path starting at the root. This is the case in Figure 1(c) when we search for the line segment object that contains Q. In particular, we first visit nodes R1 and R4 unsuccessfully, and thus need to visit nodes R2 and R5 in order to find the correct line segment object i.

The second method is based on a recursive decomposition of the underlying space into disjoint blocks so that a subset of the objects are associated with each block. There are several ways to proceed. The first is to simply redefine the decomposition and aggregation associated with the object hierarchy method so that the minimum bounding rectangles are decomposed into disjoint rectangles, thereby also implicitly partitioning the underlying objects that they bound. In this case, the partition of the underlying space is heavily dependent on the data and is said to be at arbitrary positions. The k-d-B-tree (Robinson, 1981) and the R^+-tree (Sellis et al., 1987) are examples of such an approach.

The second way is to partition the underlying space at fixed positions so that all resulting cells are of uniform size, which is the case when using the uniform grid (e.g., (Knuth, 1998)), also the standard indexing method for maps. Figure 1(a) is an example of a 4×4 uniform grid in which a collection of straight line segments has been embedded. The drawback of the uniform grid is the possibility of a large number of empty or sparsely-filled cells when the objects are not uniformly distributed. This is resolved by making use of a variable resolution representation such as one of the quadtree variants (e.g., (Samet, 2006)) where the subset of the objects that are associated with the blocks are defined by placing an upper bound on the number of objects that can be associated with each block (termed a *stopping condition* for the recursive decomposition process). An alternative, as exemplified by the PK-tree (Samet, 2004; Wang et al., 1998), makes use of a lower bound on the number of objects that can be associated with each block (termed an *instantiation* or *aggregation* threshold).

Quadtrees (Hunter and Steiglitz, 1979; Klinger, 1971) and their three-dimensional octree analogs (Hunter, 1978; Meagher, 1982). have also been used widely for representing and operating on region data in two and three dimensions, respectively (e.g., (Samet, 1988)). In particular, algorithms have been devised for converting between them and numerous representations such as binary arrays (Samet, 1980a), boundary codes (Dyer et al., 1980; Samet, 1980b), rasters (Samet, 1981a; Samet, 1984; Shaffer and Samet, 1987), medial axis transforms (Samet, 1983; Samet, 1985), terrain models (Sivan and Samet, 1992), boundary models (Tamminen and Samet, 1984), constructive solid geometry (CSG) (Samet and Tamminen, 1985), as well as for many standard operations such as connected component labeling (Samet, 1981c), perimeters (Samet, 1981b), distance (Samet, 1982), image dilation (Ang et al., 1990), and computing Euler numbers (Dyer, 1980). Quadtrees and their variants are to be distinguished from pyramids (e.g., (Tanimoto and Pavlidis, 1975; Aref and Samet, 1990)) which are multiresolution data structures.

The PM_1 quadtree (Hoel and Samet, 1991; Samet and Webber, 1985) (see also the related PMR quadtree (Nelson and Samet, 1986; Nelson and Samet, 1987)) is an example of a variable resolution representation for a collection of straight line segment objects such as the polygonal subdivision given in Figure 1(a). In this case, the stopping condition of its decomposition rule stipulates that partitioning occurs as long as a block contains more than one line segment unless the line segments are all incident at the same vertex which is also in the same block (e.g., Figure 2). A similar representation has been devised for three-dimensional images (e.g., (Ayala et al., 1985) and the references cited in (Samet, 2006)). The decomposition criteria are such that no node contains more than one face, edge, or vertex unless the faces all meet at the same vertex or are adjacent to the same edge.

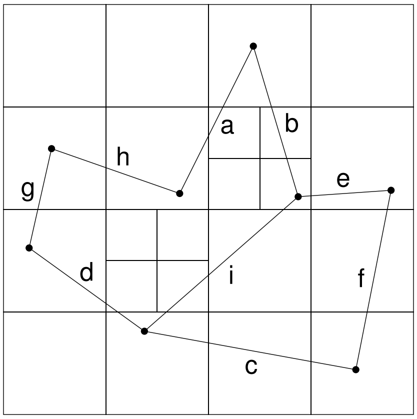

Figure 2 PM_1 quadtree for the collection of straight line segment objects of Figure 1(a).

The principal drawback of the disjoint method is that when the objects have extent (e.g., line segments, rectangles, and any other non-point objects), then an object may be associated with more than one block. This means that queries such as those that seek the length of all objects in a particular spatial region will have to remove duplicate objects before reporting the total length. Nevertheless, methods have been developed that avoid these duplicates by making use of the geometry of the type of the data that is being represented (e.g., (Aref and Samet, 1992; Aref and Samet, 1994; Dittrich and Seeger, 2000)). Note that the result of constraining the positions of the partitions means that there is a limit on the possible sizes of the resulting cells (e.g., a power of 2 in the case of a quadtree variant). However, this means that the underlying representation is good for operations between two different data sets (e.g., a spatial join (Hoel and Samet, 1995; Jacox and Samet, 2007)) as their representations are in registration (i.e., it is easy to correlate occupied and unoccupied space in the two data sets, which is not easy when the positions of the partitions are not constrained as is the case with methods rooted in representations based an object hierarchy even though the resulting decomposition of the underlying space is disjoint). For a recent empirical comparison of these representations with respect to multidimensional point data, see (Kim and Patel, 2007).

3. Example of the Utility of Sorting

As an example of the utility of sorting spatial data suppose that we want to determine the nearest object to a given point (i.e., a "pick" operation in computer graphics). In order to see how the search is facilitated by sorting the underlying data, consider the set of point objects A–F in Figure 3 which are stored in a PR quadtree (Orenstein, 1982; Samet, 1990b). The PR quadtree recursively decomposes the space in which a set of point objects lie into four equal-sized squares until each cell is empty or contains just one object (i.e., the objects are sorted into the cells which act like bins). The PR quadtree represents the underlying decomposition as a tree although our figure only illustrates the resulting decomposition of the underlying space into blocks (i.e., the leaf nodes/blocks of the PR quadtree).

The search must first determine the leaf that contains the location/object whose nearest neighboring object is sought (i.e., P in our example). Assuming a tree-based index, this is achieved by a top-down recursive algorithm. Initially, at each level of the recursion, we explore the subtree that contains P. Once the leaf node containing P has been found (i.e., 1), the distance from P to the nearest object in the leaf node is calculated (empty leaf nodes have a value of infinity). Next, we unwind the recursion so that at each level, we search the subtrees that represent regions overlapping a circle centered at P whose radius is the distance to the closest object that has been found so far. When more than one subtree must be searched, the subtrees representing regions nearer to P are searched before the subtrees that are farther away (since it is possible that an object in them might make it unnecessary to search the subtrees that are farther away).

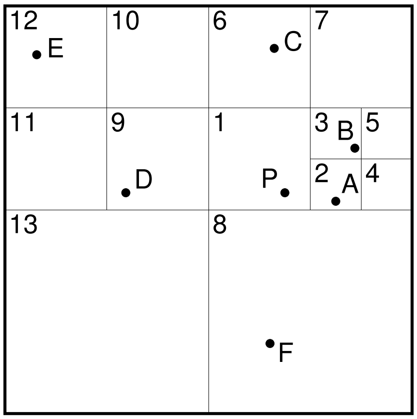

Figure 3 Example illustrating the neighboring object problem. P is the query object and the nearest object is represented by point A in node 2.

In our example, the order in which the nodes are visited is given by their labels. We visit the brothers of the node 1 containing the query point P (and all remaining nodes at each level) in the order of the minimum distance from P to their borders (i.e., SE, NW, and NE for node 1). Therefore, as we unwind for the first time, we visit the eastern brother of node 1 and its subtrees (nodes 2 and 3 followed by nodes 4 and 5), node 6, and node 7. Note that once we have visited node 2, there is no need to visit node 4 since node 2 contains A. However, we must still visit node 3 containing point B (closer than A), but now there is no need to visit node 5. Similarly, there is no need to visit nodes 6 and 7 as they are too far away given our knowledge of A. Unwinding one more level reveals that due to the distance between P and A, we must visit node 8 as it could contain a point that is closer to P than A; however, there is no need to visit nodes 9, 10, 11, 12, and 13.

4. Concluding Remarks

An overview has been given of the rationale for sorting spatial objects in order to be able to index them thereby facilitating a number of operations involving search in the multidimensional domain. A distinction has been made between spatial objects that could be represented by traditional methods that have been applied to point data and those that have extent thereby rendering the traditional methods inapplicable. In our examples, the sorting supported operations that involve proximity measured in terms of as “the crow flies”. However, these representations can also be used to support proximity in a graph such as a road network (e.g., (Sankaranarayanan et al., 2005; Samet et al., 2008)).

The functioning of these various spatial sorting methods can be experienced by trying VASCO (Brabec and Samet, 1998a; Brabec and Samet, 1998b; Brabec and Samet, 2000; Brabec et al., 2003), a system for Visualizing and Animating Spatial Constructs and Operations. VASCO consists of a set of spatial index JAVA^{TM} (e.g., (Arnold and Gosling, 1996)) applets that enable users on the worldwide web to experiment with a number of hierarchical representations (e.g., (Samet, 1990a; Samet, 1990b; Samet, 2006)) for different spatial data types, and see animations of how they support a number of search queries (e.g., nearest neighbor and range queries). The VASCO system can be found at `http://www.cs.umd.edu/~hjs/quadtree/`. For an example of their use in a spatial database/geographic information system (GIS), see the SAND Spatial Browser (Brabec and Samet, 2007; Esperança and Samet, 2002; Samet et al., 2003) and the QUILT system (Shaffer et al., 1990). Such systems find use in a number of alternative application domains (e.g., digital government (Marchionini et al., 2003)).

References

Ang, C.-H., Samet, H., and Shaffer, C. A. (1990) A new region expansion for quadtrees, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12**, 682–686, Also see *Proceedings of the Third International Symposium on Spatial Data Handling*, pages 19–37, Sydney, Australia, August 1988.

Aref, W. G. and Samet, H. (1990) Efficient processing of window queries in the pyramid data structure, In *Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, Nashville, TN, pp. 265–272, Also in *Proceedings of the Fifth Brazilian Symposium on Databases*, pages 15–26, Rio de Janeiro, Brazil, April 1990.

Aref, W. G. and Samet, H. (1992) Uniquely reporting spatial objects: yet another operation for comparing spatial data structures, In *Proceedings of the 5th International Symposium on Spatial Data Handling*, Charleston, SC, pp. 178–189.

Aref, W. G. and Samet, H. (1994) Hashing by proximity to process duplicates in spatial databases, In *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM)*, Gaithersburg, MD, pp. 347–354.

Arnold, K. and Gosling, J. (1996) *The JAVA™ Programming Language*, Reading, MA, Addison-Wesley.

Ayala, D., Brunet, P., Juan, R., and Navazo, I. (1985) Object representation by means of nonminimal division quadtrees and octrees, *ACM Transactions on Graphics* **4**, 41–59.

Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. (1990) The R*-tree: an efficient and robust access method for points and rectangles, In *Proceedings of the ACM SIGMOD Conference*, Atlantic City, NJ, pp. 322–331.

Brabec, F. and Samet, H. (1998)a The VASCO R-tree JAVA™ Applet, In Y. Ioannidis and W. Klas (eds.), *Visual Database Systems (VDB4). Proceedings of the IFIP TC2//WG2.6 Fourth Working Conference on Visual Database Systems*, L'Aquila, Italy, pp. 147–153, Chapman and Hall.

Brabec, F. and Samet, H. (1998)b Visualizing and animating R-trees and spatial operations in spatial databases on the worldwide web, In Y. Ioannidis and W. Klas (eds.), *Visual Database Systems (VDB4). Proceedings of the IFIP TC2//WG2.6 Fourth Working Conference on Visual Database Systems*, L'Aquila, Italy, pp. 123–140, Chapman and Hall.

Brabec, F. and Samet, H. (2000) Visualizing and animating search operations on quadtrees on the worldwide web, In K. Kedem and M. Katz (eds.), *Proceedings of the 16th European Workshop on Computational Geometry*, Eilat, Israel, pp. 70–76.

Brabec, F. and Samet, H. (2007) Client-based spatial browsing on the world wide web, *IEEE Internet Computing* **11**, 52–59.

Brabec, F., Samet, H., and Yilmaz, C. (2003) VASCO: visualizing and animating spatial constructs and operations, In *Proceedings of the 19th Annual Symposium on Computational Geometry*, San Diego, CA, pp. 374–375.

Dittrich, J.-P. and Seeger, B. (2000) Data redundancy and duplicate detection in spatial join processing, In *Proceedings of the 16th IEEE International Conference on Data Engineering*, San Diego, CA, pp. 535–546.

Dyer, C. R. (1980) Computing the Euler number of an image from its quadtree, *Computer Graphics and Image Processing* **13**, 270–276, Also University of Maryland Computer Science Technical Report TR–769, May 1979.

Dyer, C. R., Rosenfeld, A., and Samet, H. (1980) Region representation: boundary codes from quadtrees, *Communications of the ACM* **23**, 171–179, Also University of Maryland Computer Science Technical Report TR–732, February 1979.

Esperança, C. and Samet, H. (2002) Experience with SAND/Tcl: a scripting tool for spatial databases, *Journal of Visual Languages and Computing* **13**, 229–255.

Fuchs, H., Abram, G. D., and Grant, E. D. (1983) Near real-time shaded display of rigid objects, *Computer Graphics* **17**, 65–72, Also in *Proceedings of the SIGGRAPH'83 Conference*, Boston, July 1983.

Fuchs, H., Kedem, Z. M., and Naylor, B. F. (1980) On visible surface generation by a priori tree structures, *Computer Graphics* **14**, 124–133, Also in *Proceedings of the SIGGRAPH'80 Conference*, Seattle, WA, July 1980.

Glassner, A. S. (1984) Space subdivision for fast ray tracing, *IEEE Computer Graphics and Applications* **4**, 15–22.

Guttman, A. (1984) R-trees: a dynamic index structure for spatial searching, In *Proceedings of the ACM SIGMOD Conference*, Boston, pp. 47–57.

Hoel, E. G. and Samet, H. (1991) Efficient processing of spatial queries in line segment databases, In O. Günther and H.-J. Schek (eds.), *Advances in Spatial Databases—2nd Symposium, SSD'91*, Zurich, Switzerland, pp. 237–256.

Hoel, E. G. and Samet, H. (1995) Benchmarking spatial join operations with spatial output, In U. Dayal, P. M. D. Gray, and S. Nishio (eds.), *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB)*, Zurich, Switzerland, pp. 606–618.

Hunter, G. M. (1978) Efficient computation and data structures for graphics, Ph.D. thesis, Department of Electrical Engineering and Computer Science, Princeton University, Princeton, NJ.

Hunter, G. M. and Steiglitz, K. (1979) Operations on images using quad trees, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **1**, 145–153.

Jacox, E. and Samet, H. (2007) Spatial join techniques, *ACM Transactions on Database Systems* **32**, 7, Also an expanded version in University of Maryland Computer Science Technical Report TR–4730, June 2005.

Kim, Y. J. and Patel, J. M. (2007) Rethinking choices for multi-dimensional point indexing: making the case for the often ignored quadtree, In *Proceedings of the Third Biennial Conference on Innovative Data Systems Research (CIDR 2007)*, Asilomar, CA, pp. 281–291.

Klinger, A. (1971) Patterns and search statistics, In J. S. Rustagi (ed.), *Optimizing Methods in Statistics*, New York, , Academic Press, pp. 303–337.

Knuth, D. E. (1998) *The Art of Computer Programming: Sorting and Searching*, Vol. 3, Reading, MA, Addison-Wesley, second edition.

Marchionini, G., Samet, H., and Brandt, L. (2003) Introduction to the digital government special issue, *Communications of the ACM* **46**, 24–27.

Meagher, D. (1982) Geometric modeling using octree encoding, *Computer Graphics and Image Processing* **19**, 129–147.

Nelson, R. C. and Samet, H. (1986) A Consistent Hierarchical Representation For Vector Data, *Computer Graphics* **20**, 197–206, Also in *Proceedings of the SIGGRAPH'86 Conference*, Dallas, TX, August 1986.

Nelson, R. C. and Samet, H. (1987) A population analysis for hierarchical data structures, In *Proceedings of the ACM SIGMOD Conference*, San Francisco, pp. 270–277.

Orenstein, J. A. (1982) Multidimensional tries used for associative searching, *Information Processing Letters* **14**, 150–157.

Preparata, F. P. and Shamos, M. I. (1985) *Computational Geometry: An Introduction*, New York, Springer-Verlag.

Robinson, J. T. (1981) The K-D-B-tree: a search structure for large multidimensional dynamic indexes, In *Proceedings of the ACM SIGMOD Conference*, Ann Arbor, MI, pp. 10–18.

Sagan, H. (1994) *Space-Filling Curves*, New York, Springer-Verlag.

Samet, H. (1980)a Region representation: quadtrees from binary arrays, *Computer Graphics and Image Processing* **13**, 88–93, Also University of Maryland Computer Science Technical Report TR–767, May 1979.

Samet, H. (1980)b Region representation: quadtrees from boundary codes, *Communications of the ACM* **23**, 163–170, Also University of Maryland Computer Science Technical Report TR–741, March 1979.

Samet, H. (1981)a An algorithm for converting rasters to quadtrees, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **3**, 93–95, Also University of Maryland Computer Science Technical Report TR–766, May 1979.

Samet, H. (1981)b Computing perimeters of images represented by quadtrees, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **3**, 683–687, Also University of Maryland Computer Science Technical Report TR–755, April 1979.

Samet, H. (1981)c Connected component labeling using quadtrees, *Journal of the ACM* **28**, 487–501, Also University of Maryland Computer Science Technical Report TR–756, April 1979.

Samet, H. (1982) Distance transform for images represented by quadtrees, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **4**, 298–303, Also University of Maryland Computer Science Technical Report TR–780, July 1979.

Samet, H. (1983) A quadtree medial axis transform, *Communications of the ACM* **26**, 680–693, Also see CORRIGENDUM, *Communications of the ACM*, 27(2):151, February 1984 and University of Maryland Computer Science Technical Report TR–803, August 1979.

Samet, H. (1984) Algorithms for the conversion of quadtrees to rasters, *Computer Vision, Graphics, and Image Processing* **26**, 1–16, Also University of Maryland Computer Science Technical Report TR–979, November 1980.

Samet, H. (1985) Reconstruction of quadtrees from quadtree medial axis transforms, *Computer Vision, Graphics, and Image Processing* **29**, 311–328, Also University of Maryland Computer Science Technical Report TR–1224, October 1982.

Samet, H. (1988) An overview of quadtrees, octrees, and related hierarchical data structures, In R. A. Earnshaw (ed.), *Theoretical Foundations of Computer Graphics and CAD*, vol. 40 of NATO ASI Series F: Computer and System Sciences, Berlin, West Germany, Springer-Verlag, pp. 51–68.

Samet, H. (1989)a Implementing ray tracing with octrees and neighbor finding, *Computers & Graphics* **13**, 445–460, Also University of Maryland Computer Science Technical Report TR–2204, February 1989.

Samet, H. (1989)b Neighbor finding in images represented by octrees, *Computer Vision, Graphics, and Image Processing* **46**, 367–386, Also University of Maryland Computer Science Technical Report TR–1968, January 1988.

Samet, H. (1990)a *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*, Reading, MA, Addison-Wesley.

Samet, H. (1990)b *The Design and Analysis of Spatial Data Structures*, Reading, MA, Addison-Wesley.

Samet, H. (2004) Decoupling partitioning and grouping: overcoming shortcomings of spatial indexing with bucketing, *ACM Transactions on Database Systems* **29**, 789–830, Also University of Maryland Computer Science Technical Report TR–4523, August 2003.

Samet, H. (2006) *Foundations of Multidimensional and Metric Data Structures*, San Francisco, Morgan-Kaufmann.

Samet, H., Alborzi, H., Brabec, F., Esperança, C., Hjaltason, G. R., Morgan, F., and

Tanin, E. (2003) Use of the SAND spatial browser for digital government applications, *Communications of the ACM* **46**, 63–66.

Samet, H., Sankaranarayanan, J., and Alborzi, H. (2008) Scalable network distance browsing in spatial databases, In *Proceedings of the ACM SIGMOD Conference*, Vancouver, Canada, pp. 43–54, Also University of Maryland Computer Science Technical Report TR–4865, April 2007 (SIGMOD 2008 Best Paper Award).

Samet, H. and Tamminen, M. (1985) Bintrees, CSG trees, and time, *Computer Graphics* **19**, 121–130, Also in *Proceedings of the SIGGRAPH'85 Conference*, San Francisco, July 1985.

Samet, H. and Webber, R. E. (1985) Storing a collection of polygons using quadtrees, *ACM Transactions on Graphics* **4**, 182–222, Also see *Proceedings of Computer Vision and Pattern Recognition'83*, pages 127–132, Washington, DC, June 1983 and University of Maryland Computer Science Technical Report TR–1372, February 1984.

Sankaranarayanan, J., Alborzi, H., and Samet, H. (2005) Efficient query processing on spatial networks, In *Proceedings of the 13th ACM International Symposium on Advances in Geographic Information Systems*, Bremen, Germany, pp. 200–209.

Sellis, T., Roussopoulos, N., and Faloutsos, C. (1987) The R^+-tree: a dynamic index for multi-dimensional objects, In P. M. Stocker and W. Kent (eds.), *Proceedings of the 13th International Conference on Very Large Databases (VLDB)*, Brighton, United Kingdom, pp. 71–79, Also University of Maryland Computer Science Technical Report TR–1795, 1987.

Shaffer, C. A. and Samet, H. (1987) Optimal quadtree construction algorithms, *Computer Vision, Graphics, and Image Processing* **37**, 402–419.

Shaffer, C. A., Samet, H., and Nelson, R. C. (1990) QUILT: a geographic information system based on quadtrees, *International Journal of Geographical Information Systems* **4**, 103–131, Also University of Maryland Computer Science Technical Report TR–1885.1, July 1987.

Sivan, R. and Samet, H. (1992) Algorithms for constructing quadtree surface maps, In *Proceedings of the 5th International Symposium on Spatial Data Handling*, Vol. 1, Charleston, SC, pp. 361–370.

Tamminen, M. and Samet, H. (1984) Efficient octree conversion by connectivity labeling, *Computer Graphics* **18**, 43–51, Also in *Proceedings of the SIGGRAPH'84 Conference*, Minneapolis, MN, July 1984.

Tanimoto, S. L. and Pavlidis, T. (1975) A hierarchical data structure for picture processing, *Computer Graphics and Image Processing* **4**, 104–119.

Wang, W., Yang, J., and Muntz, R. (1998) PK-tree: a spatial index structure for high dimensional point data, In K. Tanaka and S. Ghandeharizadeh (eds.), *Proceedings of the 5th International Conference on Foundations of Data Organization and Algorithms (FODO)*, Kobe, Japan, pp. 27–36, Also University of California at Los Angeles Computer Science Technical Report 980032, September 1998.

Warnock, J. E. (1968) A hidden line algorithm for halftone picture representation, Computer Science Technical Report TR 4–5, University of Utah, Salt Lake City, UT.

Warnock, J. E. (1969) A hidden surface algorithm for computer generated half tone pictures, Computer Science Technical Report TR 4–15, University of Utah, Salt Lake City, UT.