VASCO: Visualizing and Animating Spatial Constructs and Operations*

František Brabec
Computer Science Department, Center for Automation Research and Institute for Advanced Computer Studies, University of Maryland
College Park, Maryland 20742, USA
brabec@cs.umd.edu

Hanan Samet
Computer Science Department, Center for Automation Research and Institute for Advanced Computer Studies, University of Maryland
College Park, Maryland 20742, USA
hjs@cs.umd.edu

Cemal Yilmaz
Computer Science Department, Center for Automation Research and Institute for Advanced Computer Studies, University of Maryland
College Park, Maryland 20742, USA
cyilmaz@cs.umd.edu

ABSTRACT

A video is used to demonstrate a set of spatial index JAVA™ applets that enable users on the worldwide web to experiment with a number of variants of the quadtree spatial data structure for different spatial data types, and, most importantly, enable them to see in an animated manner how a number of basic search operations are executed for them. The spatial data types are points, line segments, rectangles, and regions. The search operations are the window query (i.e., a spatial range query) and a nearest neighbor query that enables ranking spatial objects in the order of their distance from a given query object. The representations and algorithms are visualized and animated in a consistent manner using the same primitives and colors so that the differences between the effects of the representations can be easily understood. The video demonstrates the PR quadtree, PM1 quadtree, and R-tree. The applets can be found at `www.cs.umd.edu/~hjs/quadtree/`.

Categories and Subject Descriptors

E.1 [**Data Structures**]:

General Terms

Algorithms

Keywords

quadtrees, k-d trees, R-trees, visualization, nearest neighbor algorithms

1. INTRODUCTION

The representation of spatial data is an important issue in a wide variety of applications including computer graphics, computational geometry, computer vision, pattern recognition, and geographic information systems (GIS). There are two principal methods of representing spatial data. The first is based on a decomposition of the underlying space into disjoint blocks so that a subset of the objects are associated with each block. This subset is often defined by placing a bound on the number of objects that can be associated with each block (termed a *stopping condition*). The drawback of this disjoint method is that when the objects have extent (e.g., lines, rectangles, regions, and any other non-point objects), then an object may be associated with more than one block. The alternative is to use an object hierarchy that aggregates objects into groups based on proximity and then use proximity to further aggregate the groups. The drawback of this approach is that it results in a non-disjoint decomposition of 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 work was supported in part by the National Science Foundation under Grants IRI-9712715, EIA-99-00268, IIS-00-86162, and EIA-00-91474.

SoCG'03, June 8–10, 2003, San Diego, California, USA.
ACM 1-58113-663-3/03/0006.

2. VIDEO

In this video we demonstrate the use of VASCO, a system for Visualizing and Animating Spatial Constructs and Operations. We show in a comparative manner a number of hierarchical spatial data structures that are examples of the above representation techniques. In order to be able to visualize these representations, we restrict our data domain to two dimensions and hence we only look at points, lines, rectangles, and regions. However, the representations can be used for higher dimensional data. The video demonstrates the highlights of JAVA applets that can be found at: `www.cs.umd.edu/~hjs/quadtree/`.

For the methods based on a disjoint decomposition of the underlying space and disjoint hierarchies, we subdivide the representations into four classes depending on the underlying data type. All of these representations are based on a recursive decomposition of the underlying space and thus are variants of quadtrees and k-d trees (e.g., [5, 6]). For point data, VASCO includes the point quadtree, PR quadtree, MX quadtree, k-d tree, PR k-d tree, PMR quadtree, PMR k-d tree, bucket PR quadtree, and bucket PR k-d tree. For line data, VASCO includes the PM1, PM2, PM3, PMR, and

bucket PM quadtrees. For rectangle data, VASCO includes the MX-CIF quadtree, rect quadtree, PMR rect quadtree, PMR k-d tree, and bucket rect quadtree. For region data, VASCO includes the region quadtree. The video shows explicitly how the data structure is built for a PR quadtree for points, and for a PM1 quadtree for lines.

It also shows the decompositions for the displayed data set that result from the use of each of the remaining data structures. VASCO permits the user to delete data although we do not show this in the video.

For the methods based on a non-disjoint decomposition of space we examine the R-tree [1]. In this case, the nature of the underlying data type is not of importance and we just examine rectangle data although VASCO can handle R-tree representations of point and line data as well. There are many variants of R-trees, where the distinguishing feature is the manner in which an overflowing node is split during insertion, and the manner in which the objects that comprise it are subsequently aggregated. The video shows a number of methods of dealing with this situation and displays the resulting aggregations at an intermediate level of decomposition for a number of the overflow techniques for a given data set.

VASCO provides a tool for comparing the space decompositions and object aggregations resulting from these different representations by demonstrating how they handle insertion and deletion of objects. A move operation is also provided to show the sensitivity of each representation to the movement of an object as well as the constituent types that make up the object. For example, in the case of a line segment object, we can also see the effect of moving one of its vertices. Similarly, for a rectangle object, we can also see the effect of moving one of its vertices or one of its edges.

In addition, VASCO provides a way to visualize in an animated way how the different representations can be used to perform a number of operations. The first is finding the nearest neighbors to a user-specified query object which can be a point, rectangle, polygon, polyline, or a sector. We illustrate an incremental nearest neighbor algorithm [2, 3] which means that the data objects that satisfy the query are returned and displayed one-by-one. When this algorithm is run to completion, it provides a full ranking of the data objects in terms of their distance from the query object. The advantage of the incremental algorithm over traditional methods that compute the k nearest neighbors (e.g., [4]) is that with these methods, if instead we want the $k+1$ nearest neighbors, then we must compute all $k+1$ neighbors again. In contrast, using our algorithm, the $k+1^{st}$ neighbor is found by simply continuing the search from the point immediately after having found the k nearest neighbors.

The incremental nearest neighbor algorithm makes use of a priority queue where the queue elements are the blocks of the underlying data structure as well as the objects themselves. The priority queue is ordered on the basis of the distance of its elements from the location of the query object which is a point in the example shown in the video. The algorithm works in a top-down manner in the sense that as elements are removed from the queue, they are checked to see if they correspond to blocks that are not at the lowest level of the hierarchy (i.e., nonleaf nodes). If this is the case, then their immediate descendants (i.e., the children) are inserted in the queue ordered according to their distance from the query object. Otherwise, the objects that they contain are inserted into the queue ordered according to their distance from the query object. If the element e that has been removed from the queue is a data object, then e is reported as the next nearest neighbor of the query object.

The VASCO system also enables the performance of overlap and within queries. The overlap query is equivalent to a window query where the query object serves as the window. It is frequently used in database operations to retrieve all spatial objects within a given subarea of the underlying space. The within query is equivalent to a corridor or buffer in a geographic information system (GIS). In this case, the operation results in retrieving all spatial objects within a given distance of the query object. Both of the algorithms are implemented in such a way that they report the resulting objects incrementally — that is, in increasing order of their distance from the query object. Again, as in the nearest neighbor operation, the query object is specified by the user and is either a point, rectangle, polygon, polyline, or a sector. The animation proceeds in the same manner as in the incremental nearest neighbor algorithm with the same color conventions.

VASCO also includes the region quadtree. Operations are provided for inserting and deleting pixels and blocks. Two variants of the move operation are provided: one that copies a pixel or block (thereby acting like a union operation), while the other one overwrites the pixel or block at the new location. Finally, operations are provided for converting a quadtree to an array, raster and chain code representation.

3. REFERENCES

[1] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD Conference*, pages 47–57, Boston, MA, June 1984.

[2] G. R. Hjaltason and H. Samet. Ranking in spatial databases. In *Advances in Spatial Databases — Fourth International Symposium, SSD'95*, M. J. Egenhofer and J. R. Herring, eds., pages 83–95, Portland, ME, August 1995. Also Springer-Verlag Lecture Notes in Computer Science 951.

[3] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM Transactions on Database Systems*, 24(2):265–318, June 1999. Also Computer Science TR-3919, University of Maryland, College Park, MD.

[4] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *Proceedings of the ACM SIGMOD Conference*, pages 71–79, San Jose, CA, May 1995.

[5] H. Samet. *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS.* Addison-Wesley, Reading, MA, 1990.

[6] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, Reading, MA, 1990.