Quadtree Region Representation in Cartography: Experimental Results

Hanan Samet
Azriel Rosenfeld
Clifford Shaffer
Robert E. Webber
Computer Science Department and
Center for Automation Research
University of Maryland
College Park, MD 20742

ABSTRACT

A study was conducted of the use of quadtrees to encode region data in a cartographic application. Programs were written to perform a variety of tasks on a sample data base. A summary of the efficiency of this representation is presented.

1 Introduction

An experimental study of the use of quadtree region representations in cartography is currently underway. Quadtree encodings are used for a database containing three maps of different features of a small region in Northern California. These maps describe a floodplain, elevation contours, and landuse classes for the region. The data were hand digitized to 400x450 pixels. Labels were associated with each of the pixels in the resulting regions. The regions were encoded using a 512x512 grid. Programs have been written to perform various analysis and manipulation tasks on the quadtree-encoded regions. This paper summarizes data obtained in the first stage of the study. It deals with the compactness of the encodings and the efficiency of the programs. Further details about the study can be found in [4].

The quadtree representation of regions, first proposed by Klinger [3], has been widely studied in the past several years. Numerous algorithms have been developed for constructing compact quadtree representations, converting between them and other region representations, converting between them and other region representations, computing region properties from them, and computing the quadtree representations of Boolean combinations of regions from those of the given regions. For recent overviews of this literature see [6]

2 Results

In order to be able to handle cartographic data we had to slightly modify the definition of a quadtree. In particular, we decided to use a different label for each region in the floodplain (3), each landuse class (35), and 100 foot elevation contour (11). This means that our quadtree is no longer binary but instead is multicolored. Thus leaf nodes now have a range of values. This did not prove to be a problem. Nevertheless, certain operations, such as union and intersection are not defined in terms of multicolor quadtrees, and thus they had to be modified. In particular, they are defined in our system in terms of binary quadtrees by considering one of the colors as BLACK (usually the color of the object of interest) and the other colors to be WHITE. Note that boundaries between landuse classes and contour elevations are treated as lines of zero width and do not appear explicitly in our quadtrees.

In order to evaluate the quadtree building process, a separate quadtree was constructed for each landuse class, elevation range, and floodplain region. Quadtree building was done "bottom up," by scanning the image and merging nodes when all the nodes in a 2x2 block are found to be of the same type, as described in [5]. The number of "nodes created" ranged from about 1700 to about 14,500 (still only a small fraction of the 2 ** 18 potential pixels in the 512x512 grid), while the number of nodes in the final quadtrees ranged from about 130 to about 13,000. The difference between the number of nodes created and the number of nodes in the final quadtree reflects the fact that merging took place. To see the effect of the compaction resulting from use of a quadtree, the number of nodes should be compared with the total number of pixels in a 512x512 grid - i.e., 262,144. Of course, quadtree nodes take up more space than pixels; however, there exist encodings of quadtrees which use no more than 2 bits per node [2].

The quadtrees were stored on disk in files that contained lists of the node types met in a preorder traversal of the quadtree. Some of the simpler algorithms can manipulate these files directly. Others read the file into core where it is expanded to include pointers to father nodes and son nodes.

A large number of quadtree algorithms were tested on the database. In particular, we were interested in obtaining timing information. This was difficult to do in the sense that precise machine execution times are hard to obtain. Thus we also measured machine independent concepts such as the number of nodes visited by the various algorithms. We were especially interested in comparing the costs of a number of different neighbor

Webber
00695d3

computation techniques since they form the heart of many basic operations. This comparison was done by measuring the connected component labeling process [7] as it was performed for each region type (land use class, elevation range). In particular, we compared the neighbor finding techniques of Samet [8] which make use of a common ancestor; those of Hunter and Steiglitz [1] which make use of "ropes" (links between nodes of the same size that are spatially adjacent); and a new technique [4] which is based on modifying the quadtree traversal algorithm to pass the neighbors of each subtree's root as parameters. This required more time than using ropes (because of the need to pass the parameters), but required 40% less memory. As expected, the method of ropes was the most efficient time-wise (i.e., an average cost of 1.5 nodes being visited). However, it had the extra storage cost for the links. Interestingly, the neighbor finding techniques of [8] resulted in 3.5 nodes being visited on the average. This vindicated the theoretical analysis and the model of node configuration which served as its basis.

Programs were also written to compute the area, perimeter, coordinates of the centroid and the enclosing upright rectangle for each of the connected components. Other programs included point in polygon determination and union and intersection of regions. A special case of intersecction termed "window" was also developed and tested. It intersects a given region (defined by a quadtree) and a given square window (of size a power of two). This is done by finding the smallest subtree of the given quadtree that contains the window. If it coincides with the window, then the subtree is returned; if not, then the window is split into quadrants and each of them is processed analogously with respect to the current subtree. This program proved to be rather costly since it involved building a new quadtree for a subregion of the map specified by the boundaries of the window.

Various quadtree display techniques were attempted. In particular, a quadtree truncation method was devised which attempts to approximate the map without displaying the quadtree nodes at the deepest levels. Instead of uniformly treating GRAY nodes lower than a certain level as BLACK or WHITE, they were set to the appropriate color based on their weighted average - i.e., BLACK if the majority of the constituent pixels is BLACK and WHITE if vice-versa. The results showed a gentle degradation when quadtree truncation was compared at different levels. More importantly, truncation showed a significant reduction in the number of nodes in the tree with relatively minor loss of accuracy.

In general, it should be noted that displaying quadtree files in much faster than pixel-based files on many common CRT devices that allow specification of an entire block to be of one color. The leaves of the quadtree indicate a convenient patitioning of a map into monochrome blocks. Since it is not necessary to store the quadtree in core in order to display it, simple hardware could be designed that displayed quadtree files directly.

3 Conclusion

The experiments described in this paper provide a quantitative assessment of the efficiency of quadtrees as a means of representing regions in a geographic information system and confirmed their viability.

References

1. Hunter, G. M. and Steiglitz, K., Operations on images using quadtrees, <u>IEEE Transactions on Pattern Analysis and Machine Intelligence</u> <u>1</u>, 1979, 145-153.

2. Kawaguchi, E. and Endo, T., On a method of binary picture representation and its application to data compression, <u>IEEE Transactions on Pattern Analysis and Machine Intelligence</u> <u>2</u>, 1980, 27-35.

3. Klinger, A., Patterns and search statistics in <u>Optimizing Methods in Statistics</u>, J. S. Rustagi, Ed., Academic Press, New York, 1971.

4. Rosenfeld, A., Samet, H., Shaffer, C., and Webber, R., E., Application of hierarchical data structures to geographic information systems, Computer Science TR-1197, University of Maryland, College Park, MD, June 1982.

5. Samet, H., Region representation: quadtrees from binary arrays, <u>Computer Graphics and Image Processing</u> <u>13</u>, 1980, 88-93.

6. Samet, H. and Rosenfeld, A., Quadtree structures for image processing, <u>Proceedings of the Fifth International Conference on Pattern Recognition</u>, Miami Beach, December 1980, 815-818.

7. Samet, H., Connected component labeling using quadtrees, <u>Journal of the ACM</u> <u>28</u>, July 1981, 487-501.

8. Samet, H., Neighbor finding techniques for images represented by quadtrees, <u>Computer Graphics and Image Processing</u> <u>18</u>, 1982, 37-57.

The support of the U.S. Army Engineer Topographic Laboratories under Contract DAAK70-81-C-0059 is gratefully acknowledged, as is the help of Janet Salzman in preparing this paper.