DATA STRUCTURES AND ALGORITHMS FOR RASTER DATA PROCESSING

Albert L. Zobrist
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91103

I. Introduction

The digital image raster data structure is proving to be a useful representation for geographic information from the viewpoints of both storage and processing. Several components of this trend can be identified: the development of scanners such as Landsat for remote sensing of the Earth's surface, the development of a raster representation for terrain (elevation) data, the spread of image processing technology, and increased sophistication in application areas. The raster data structure is familiar to planners as a fine grid cell system, but current emphasis is upon ultra-large or ultra-fine raster processing made possible by image processing technology.

The terms storage and processing imply to a computer scientist the need for investigation of data structures and algorithms. The goals of this investigation are to

---

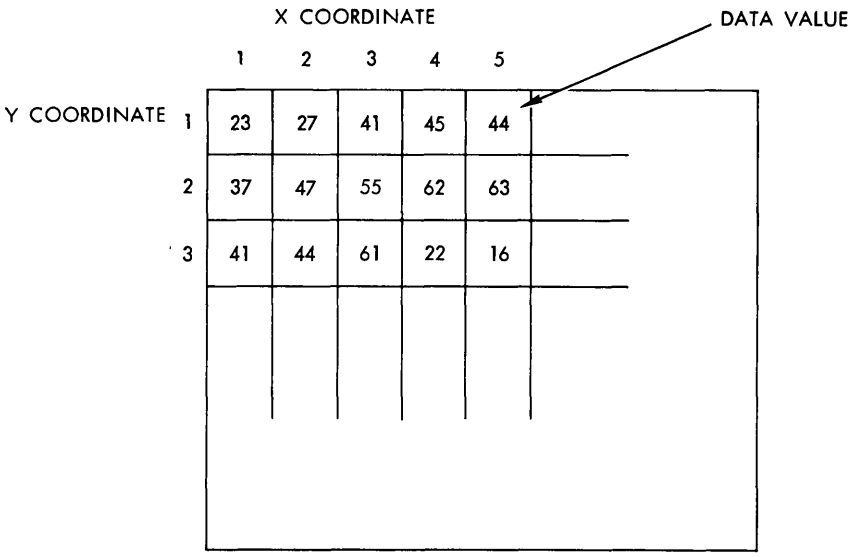
This paper presents one phase of research conducted at the Jet Propulsion Laboratory, California Institute of Technology, under NAS 7-100, sponsored by the National Aeronautics and Space Administration

outline the general algorithmic needs for raster data processing, identify some of the hard problems to be solved, develop general approaches and specific solutions to these problems, and come to an understanding of the potentialities and limitations of computer processing in this area. Computer science has developed a methodology for a study of this sort, but a key consideration here is the specialization to geographic information processing. In order to demonstrate that algorithmic methods research is necessary, this report concentrates on some of the crucial elements of the Image Based Information System (IBIS) developed at JPL (Bryant and Zobrist, 1977).

There is a need to bring to bear the analytical approach of a computer scientist to algorithmic problems. To give a simple example, an alphabetized list of names can be searched more rapidly than the same list unalphabetized. For a list of length n, the former search takes on the order of log(n) probes, denoted O(log n) whereas the latter takes on the order of n probes, O(n). It does not matter that the latter search averages 1/2 n probes because if n is one million, log (n) is twenty and 1/2 n is 500,000. A proper study gives a clear definition of the data structure, the algorithm (in terms of primitive operations), and an analysis of the execution time in the average case or worst case.

II. Data Structures

Until recently, the image raster format has been used primarily as a computer processable equivalent of a photo, with the value stored in each cell of the image representing a shade of grey or a color. But if the image is of a geographical area, then the value in the cell can be a datum for the area corresponding to that cell. A principle advantage of the image representation is that data for a geographical point can be accessed immediately by position in the image matrix. Figure 1 illustrates the calculation of memory address of the data value from a latitude-longitude pair. This assumes that the image is map projected. If the image is not map projected, correlative data analysis is still possible if several data sets have been registered so that the same position in each image matrix corresponds to the same geographic point. The image datatype appears to be a powerful and general representation for spatially distributed data, and the range



LOCATION FORMULAS:

$$X = \left[A \times LAT + B \times LONG + C \right] \text{ NEAREST INTEGER}$$

$$Y = \left[D \times LAT + E \times LONG + F \right] \text{ NEAREST INTEGER}$$

MEMORY ADDRESS = BASE + KX + Y

Figure 1. Image raster as a spatial data representation.

of uses can be divided into several broad categories.

Physical Analog. The pixel value represents a physical variable such as elevation, rainfall, smog density, etc.

District Identification. The pixel value is a numerical identifier for the district which includes that pixel area.

Class Identification. The pixel value is a numerical identifier for the land use or land cover, or for any other area classification scheme.

Tabular Pointer. The pixel value is a record pointer to a tabular record which applies to the pixel geographical area.

Point Identification. The pixel value identifies a point, or the nearest of a set of points, or the distance to the nearest of a set of points.

Line Identification. The pixel value identifies a line, or the nearest of a set of lines, or the distance to the nearest of a set of lines.

This range of uses requires that the system be able to handle images composed of words of varying length. For example, to identify census tracts in Los Angeles County requires 1500 different pixel values and elevation maps can require 15,000. This is more than the usual 256 grey levels handled by photographic image processing systems.

A negative aspect of image representation is the tremendous redundancy incurred in most cases. Most of the representations above have a more compact equivalent in graphics format. Several of our projects have involved over a billion bytes of data in image format. A second concern is with accuracy of raster representation versus polygon representation of areas, lines, or points, or of physical variables. For example, to reduce the error of approximation to a curved border by half, a graphics data set needs twice as many vectors but a raster needs four times as many cells.

III. Geographic Data Structures

A bit of wisdom that has developed in data base

management is that data relationships which are important to processing or retrieval must be represented in the storage structure in a way that makes the processing or retrieval efficient. For example, in a personnel data base that includes a "boss' identifier" field in each employee's record it would be easy to retrieve the upward chain of command for any employee but would be harder to construct a downward chain. The addition of two extra fields, "first subordinate identifier" and "coworker chain identifier", allows downward chains to be constructed rapidly. Geographic information systems pose this problem to a far greater extent because of the wealth of relationships inherent to spatial data and because of the difficulties with computer processing of spatial data.

The basic raster format for geographic data was presented in the last section, and a simple extension would be to construct a stack of raster planes over a geographic area. A usual requirement for efficient processing is that the rasters be spatially registered.

More complex structures are needed to link raster data with graphics or tabular data. Two methods of linking raster to tabular have been developed for the IBIS system (Figure 2).

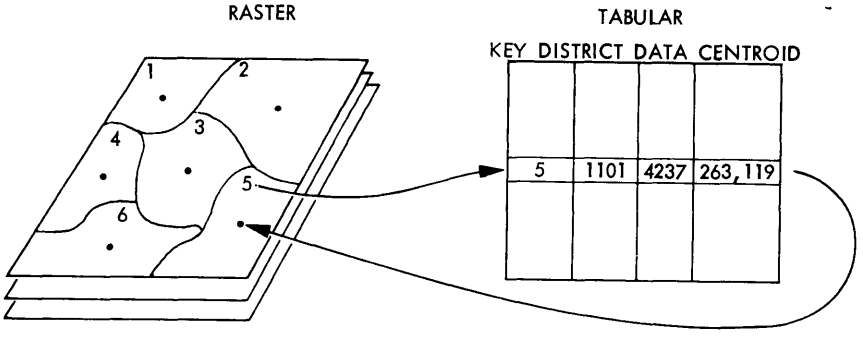


Figure 2. Illustration of two basic raster to tabular links.

The left part of the figure shows a stack of rasters over an area with the top raster representing districts by repeating a unique key in every cell over the tract area. Tabular data pertaining to a district can be found by locating the record containing its key. The reverse link uses a centroid (X,Y) in the tabular file to find the centroid pixel in the raster. Note that either set of links could be used to calculate and store the other. The key link is also effective in the reverse direction for chloroplethic mapping of a color legend developed in the tabular file.

Directly linked raster-graphical files have not been developed, but instead a rapid conversion from graphical to raster has been created so that all of the spatial data is placed in an image stack during analysis. Graphics data are allowed in the raw data base where they allow for economical capture, editing, and storage.

IV. Examples of Algorithms

The purpose here is not to give a complete cataloguing of image processing techniques, but to show a few of the algorithms that made the IBIS system possible.

The first problem to be attacked was rubber sheet registration of raster data sets. The difficulty involves the allocation of limited main storage to storage of a part of the raster input so that the raster output can be computed efficiently. Two previous methods did not work well for large or highly rotated input rasters (for example 3000 x 3000 rotated 11°). Method 1 stores a band of raster lines internally as shown in figure 3. Because of rotation an output line will only have a short intersection with this band. Therefore it is only possible to calculate an extremely large number of short output segments which must be written to disk and later reconstructed into the output raster. The later reconstruction involves excessive disk head motion for large cases. If the raster is n by n and available storage is fixed, then the length of calculated segments is $O(1/n)$, the number of cells is $O(n^2)$ hence the number of short segments is $O(n^3)$ and disk head motion will increase by this factor under a simple reconstruction scheme. Method 2 avoids the reconstruction of short segments by storing all of the input raster in the neighborhood of an output line. But

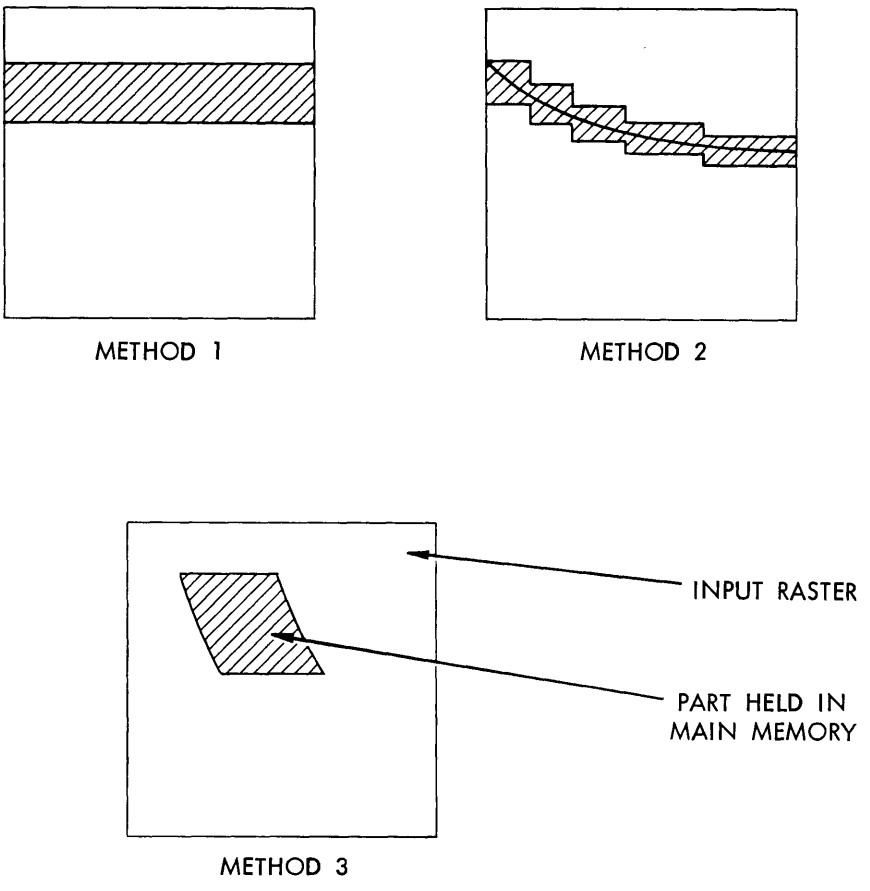


Figure 3. Diagram of allocation schemes for main memory during raster rubber sheet operation.

because the stored input area is not a band, the input file must be reread as many times as there are plateaus in the lower part of the stored area. The thickness of the stored area is $O(1/n)$ and the length of the line is $O(n)$ hence the number of rereads of the input data set is $O(n^2)$. Since the amount of data is $O(n^2)$ the disk head motion will increase by $O(n^4)$. The new

method developed computes a uniform vertical band of optimal width in the output by storing a corresponding swath of input. The output segment width is independent of n hence disk motion depends on the number of times the input has to be read which is O(n) times amount of data yielding $O(n^3)$. The reconstruction stage is $O(n^2)$. Thus method 2 is unsuitable for large cases and methods 1 and 3 appear the same. A closer analysis reveals that method 1 forces a head motion for each segment for large cases whereas method 3 has O(n) sequential passes over the data which minimizes head motion and rotation latency. Methods 1 and 3 are implemented in the VICAR routines LGEOM and MGEOM respectively, and method 2 was reported by H.K. Ramapriyan (Ramapriyan, 1977).

The need for rubber sheet registration created a second problem. Rubber sheet programs require a uniform rectangular grid of control points for efficient computation of the lateral distortion. However, geographic control points are distributed in a haphazard manner and some means of converting these into a uniform grid was needed. This problem can be viewed as one of fitting a distortion surface to the geographic control points and evaluating that surface at the uniform grid points. A number of surface fitting methods are known, and among these the "finite element method" seemed to be most appropriate for the kinds of distortions evident in Landsat. This method involves triangulating the geographic control points and letting the distortion surface value over a triangle be the linear interpolation of the values at the corners. Of the many known triangulation methods (Lawson, 1977) the greedy triangulation, which inserts edges in a shortest first order was chosen through experimentation with small cases. The problem was that the greedy algorithm consumed $O(n^3)$ time for a case with n points because there are $O(n^2)$ candidate edges and most must be tested against O(n) edges which have already been placed in the triangulation. The addition of a simple rejection test to the algorithm cut the average case to $O(n^2)$ time. If a candidate edge links points p to q then it can be rejected if p or q are part of a triangle which crosses the candidate line (Manacher and Zobrist, 1979).

Format conversion of graphics to raster format brought out three interesting algorithmic problems. First,

the conversion of large numbers of vectors to marked cells corresponding to the vector location needs a method more efficient than solving equations of intersection for each cell versus each vector. Fortunately, an old method of controlling a digital plotter (Bresenham, 1965) was adaptable to plotting the vectors into the raster. For each vector, a starting raster cell is located, and an iterative procedure steps to adjacent or diagonally adjacent cells according to the direction of the vector. Main storage was conserved by operating on a horizontal band of the raster, plotting the intersections of the vectors with the band, writing the band to secondary storage, and then operating on the next band. Let storage be a fixed resource and consider a case with m vectors to be scribed into an n by n raster. The band thickness is O(1/n) hence the number of bands is $O(n^2)$ and the number of lineband intersection calculations is $O(m\ n^2)$. An easy speedup would involve chopping the lines on all band boundaries at once the sorting the line segments into piles, one pile for each band. Then the operation is $O(m \log m + n^2)$. The plotting step is obviously $O(m\ n^2)$. Without the speedup, the IBIS routine POLYSCRB is able to plot 40000 lines into a 3600 by 3600 raster in about 10 minutes computer time on an IBM 360/65.

In the case where district boundaries have been plotted into a raster, a useful operation is to identify the connected areas of non-boundary (Rosenfeld, 1976). Proceeding from the upper left corner, each non-boundary cell is assigned the same value as its upper or left neighbor, or a new value if the neighbors are both boundary cells. If the neighbors are different, an entry is made in a table of joins so that a subsequent pass can equate the joined regions. Only two lines of the raster have to be held in storage, but the table of joins must hold an entry for each pixel which has boundary cells above and left which is $O(n^2)$. The IBIS routine PAINT includes a small addition called the "zipper" which acts whenever a new region number on a single line encounters an old region number, the new region number is unzipped and replaced with the old region number. For typical geographic regions, the table of joins seems to need about 1.5 times as many entries as regions.

A useful trick for the display of districts is to color them using four colors so that no adjacent districts

are the same color. An algorithm for this would not be easy since it would "prove" the four color conjecture. The IBIS routine COLOR implements a five color algorithm which is $O(n^2)$ for n regions, and further, it can be used to attempt to produce a four coloring in $O(n^2)$ time (McLemore and Zobrist, 1977). In practice this routine has always succeeded with four colors.

Polygon overlay to produce a table of areas of intersections of two raster data planes amounts to a simple problem of counting the occurrences of cell pairs. However, if a table is set up for all possible cell pairs and one raster has range m and the other has range n, then the table needs space mn. However, m and n are often too large for this table to be stored in main memory. Consider the case where m traffic zones overlay n census tracts. If each zone touches on the average four tracts then only 4m storage is really needed. The standard solutions to this problem is to use a hash table to store the cell pair counts that actually occur (Knuth, 1973). The IBIS routine POLYOVLY uses a hash address function $H(C_1,C_2)$ of the cell value pair that is used as an address to store the count in the hash table. The values C_1 and C_2 are also stored with the count. If two different cell value pairs map to the same hash address (a collision), one of the counts is moved to a new location by a collision resolution scheme.

A similar technique was used in the IBIS routine F2 that performs mathematical operations on pairs of rasters. The user specifies an arbitrary operation by means of a Fortran like expression such as A + COS (A**B) and this expression is compiled and evaluated at each cell pair with the first value substituted for A and the second for B. A speedup is obtained by storing the cell pair and result in a hash table. However, there is little gain by resolving collisions since the collision resolution could take longer than reevaluating the expression. So collisions are simply allowed to overwrite previous results, and the behavior of this data structure is exactly like that of hardware cache memories. This data structure should probably be called a cache table.

V. Conclusion

Geographic information systems will require development

in a number of areas, such as system organization, data capture, man-machine interaction, modelling, standardization, etc., but it should be apparent from this report that some effort in algorithms research is needed as well.

References

Bresenham, J.E. (1965), "Algorithm for computer control of a digital plotter", IBM Systems Journal, vol. 4, no. 1, pp. 25-30.

Bryant, N.A. and Zobrist, A.L. (1977), "IBIS: a geographic information system based on digital image processing and image raster datatype", IEEE Trans. Geoscience Electronics, vol. GE-15, no. 3, pp. 152-159.

Knuth, D.E., (1973), The Art of Computer Programming, Volume 31, Sorting and Searching, Addison-Wesley, Reading, Mass., pp. 506-549.

Lawson, C.L. (1977) "Software for C' surface interpolation", Mathematical Software III, Academic Press, New York.

McLemore, B.D. and Zobrist, A.L. (1977), "A five color algorithm for planar maps", Second Caribbean Conference on Combinatorics and Computing, University of the West Indies, Cave Hill, Barbados.

Manacher, G.K. and Zobrist, A.L. (1979), "A fast, space-efficient average case algorithm for the "greedy" triangulation of a point set", SIAM Journal on Computing, (submitted).

Ramapriyan, H.K. (1977), "Data handling for the geometric correction of large images", IEEE Trans. Computers, vol. C-26, no. 11, pp. 1163-1167.

Rosenfeld, A. (1977), "Extraction of topological information from digital images", An Advanced Study Symposium on Topological Data Structures for Geographic Information Systems, Laboratory for Computer Graphics and Spatial Analysis, Harvard University, Cambridge.