A RASTER-ENCODED POLYGON DATA STRUCTURE FOR LAND RESOURCE ANALYSIS APPLICATIONS

Stephan W. Miller
Department of Geography and Anthropology
Louisiana State University
Baton Rouge, LA 70804

ABSTRACT

A prototype structure for areal data sets is discussed in the context of providing efficient data storage, retrieval, and analysis as required by land resource analysis applications. The prototype is a "raster-encoded polygon format" and is derived from a run-length raster format in which each run has multiple attributes associated with it. A transformation program, involving the first phase of the raster-to-vector conversion process, associates each run with a parent polygon entity. The total size of each entity, and information concerning its adjacency to other entities, are recorded in an entity table along with all attribute values for the entity. This results in a reduction of total volume of the run-length raster file, and simplified subsequent manipulation and retrieval operations based upon attribute values of the data. The underlying raster structure retains its advantages for overlaying new data sets and complex retrieval operations. Since the parent polygon area and adjacency relationships are identified, queries based upon threshold size or adjacency can now be addressed with what is essentially a raster data structure.

INTRODUCTION

Land resource analysis involves the evaluation of the capabilities and constraints of the physical characteristics of a region, usually conducted in support of some decision-making process of public or private agencies. The evaluation produces quantitative information and supporting graphics from maps and supplemental information contained in inventories of land resource characteristics. Geographic information systems have been developed to enhance the evaluation of land resources, as well as for conducting other forms of spatial analysis. The distinguishing characteristic for land resource analysis applications, however, is that diverse data sets (particularly areal data sets) must be integrated in order to develop satisfactory solutions to the questions being addressed (Knapp, 1980). This, in turn, has significant ramifications for the manner in which spatial data is organized for use in a geographic information system.

Discussion of spatial data organization have centered on differences between raster and vector data structures. For purposes of inventory, and less complex retrieval and analysis operations, vector formats are preferred for areal data (Tomlinson and Boyle, 1981). When data from several sources must be merged, however, as is frequently the case for land resource analysis, vector formats are not advantageous because of difficulties which most systems have in performing polygon overlay (Tomlinson and Boyle, 1981). These difficulties affect the ability to merge data initially, to retrieve data from irregular areas of interest in a region, and to modify and update data sets. Raster formats have been recognized as advantageous for overlay and many retrieval and analysis functions. The problem of data storage, retrieval, and analysis for large data sets resulting from a high resolution raster scan

can be ameliorated by using compression techniques. A variant of run-length compression has been described previously and used successfully to merge and retrieve areal spatial data (Miller, 1980; 1981). This technique uses a group of adjacent columns of grid cell data along a row (i.e., a run) as a spatial unit (see Figure 1).

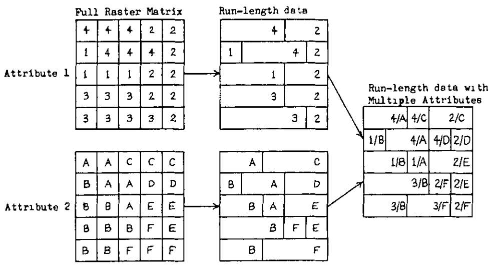
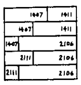


Figure 1. Derivation of run-length data with multiple attributes.

In spite of the advantages offered by compact raster processing, it is unable to fulfill the requirements for certain types of query. Queries based upon size and adjacency of composite polygon areas, for example, cannot be addressed since the basic spatial unit corresponds only to a horizontal portion of a polygon. Vector formats or, as Knapp observes, "entity based data structures," can much more readily respond to such queries (Knapp, 1979). The realization that alternate formats offer advantages for different operations has led to increased interest in the problem of raster-to-vector and vector-to-raster operations (Nichols, 1981). The ability to interchange formats is certainly appealing and it offers a solution to problems of storage, retrieval, and analysis for land resource applications. Yet the cost of such transformations, particularly the raster-to-vector transformation, can be large. Once accomplished, it is likely that the data will subsequently need to be transformed back into raster format for any complex retrieval and analysis operation. In light of this, a case can be made for developing a hybrid raster format which takes on certain characteristics of polygonal data structures. A prototype for such a data structure, suitable for land resource analysis operations, is presented in this paper.

RASTER-ENCODED POLYGONS

An early line-printer mapping routine, developed originally by the Bureau of the Census, was entitled CMAP and it is an example of the raster-encoded polygon (REP) data structure (Monmonier, 1973). For CMAP, thematic mapping units, such as states, counties or census tracts, were encoded in a run format. The administrative codes for the mapping unit were associated with each run segment and a table of values (see Figure 2). From the raw values in the table (e.g., percentage of owner-occupied houses), a class interval number and associated overprint characters could be assigned. More recently, designers of the Domestic Information Display System (DIDS) coined the term "raster-encoded polygon" and employed a data structure somewhat akin to that of CMAP. Instead of a line-printer, DIDS uses color graphics display terminals that have been developed for the field of image processing (Dalton, et al, 1980). Refinements added to the data structure allow the data to be zoomed in selective increments without loss of significant boundaries.



Census Unit	Value	Class	Symbols
1407	92%	5	O V A X
1411	78%	4	O V X
2106	42%	2	O /
2111	58%	3	O X

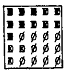
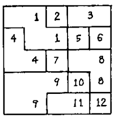


Figure 2. Line printer mapping of raster-encoded polygon data.

As with most applications involving run-length data sets to date, DIDS dealt with a single coverage, namely, administrative boundaries (i.e., state, county, and selected municipal and census tract boundaries). Data compression ratios of 10:1 and better were observed under these conditions. For natural resource applications, however, multiple coverages are needed. Yet what has generally not been appreciated is the fact that data compression ratios in the range of 4:1 to 6:1 and better have been observed for multi-layered data sets involving up to nine coverages (Miller, 1980). What is suggested, then, is that a multi-layered compressed data set as depicted in Figure 1 can be transformed into a raster-encoded polygon data set (see Figure 3). The polygon entities derived from this transformation correspond to mapping units as defined for CMAP and DIDS. Instead of a single attribute and code, however, codes for multiple attributes are associated with each entity in a table.



ENTITY TABLE

Entity	Attributes		Area	MinLine	MaxLine	MinElem	MaxElem	Pointer-Adjacency Table
1	4	A	4	1	2	1	2	5
2	4	C	1	1	1	3	3	8
3	2	C	2	1	1	4	5	12
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
12	2	F	1	5	5	5	5	(last element)

ADJACENCY TABLE

Entity	Length	
null	3	
2	2	
5	1	
4	3	
7	1	Last edge for entity 1
null	1	
1	2	
3	1	Last edge for entity 2
null	3	
2	1	
5	1	
6	1	Last edge for entity 3
⋮	⋮	
null	2	
8	1	
11	1	Last edge for entity 12

Figure 3. Entity and adjacency tables of a raster-encoded polygon data.

TRANSFORMATION OF COMPRESSED RUN-FORMAT DATA INTO RASTER-ENCODED POLYGONS

The transformation of compressed run-length records involves at least two distinct steps assuming the data currently exist as single overlay files. The data must be merged and an efficient method of performing this task on sequential lists of run-length records has been previously described (Miller, 1981). The algorithm compares successive ending-column locations looking for the minimum for runs along a row or scan line on two or more coverages. Pointers to the attribute list provide the attribute information, while the minimum ending-column determines the ending column location of the composite run. Secondly, any redundancy which exists in the composite data should be eliminated. The routine for deriving composite records which is described above will ensure that no redundancy exists in the initial composite data set. If, however, only a few attributes are chosen for subsequent processing, further compression of the data may be possible. Any redundancy should be removed in order to simplify the transformation of the data to the raster-encoded polygon structure.

The transformation of compressed raster data into a raster-encoded polygon structure builds upon the work of other researchers. Rosenfield describes an algorithm for segmentation--the assignment of individual raster elements to a significant entity (Rosenfield, 1979). Nichols adapted the algorithm suggested by Rosenfield and developed an effective raster-to-vector conversion method (Nichols, 1981). The first phase of the raster-to-vector conversion involves labelling and edge processing. For a complete raster-to-vector algorithm, subsequent phases are needed to link edges into chains; to weed and smooth redundant coordinates in the chains; and, finally, to piece together the chains needed to define polygons. The labelling and edge processing phase, however, is considerable less involved than subsequent phases and a strong argument can be made for delaying any further processing since this phase provides the data needed for land resource applications.

An algorithm has been developed to perform the labelling and edge-detection phase for multi-layered data in a run format. The first step in the algorithm suggested by Rosenfield requires that redundant elements along a row be eliminated. Since the redundant data has already been eliminated for the run format data, a slight amount of processing time is saved. A description of the transformation routine follows.

```
procedure   Build Entities
begin
            LoadScan (lscan, latt, nrls, eof)
            for i:=i,nrls do begin
                 lscan(i,3)=i
                 EdgesAcross (lscan,i,null, null);
                 Uptable;
            end;
            nent=nrls
            EdgesDown (lscan, nrls)
NextScan    LoadScan (cscan, catt, nrcs, eof)
            if (eof) then begin
                 for i:=i, nrls do
                      EdgesAcross (lscan,i,null,null);
                 end;
                 stop
```

```
          j=1
          for i=i,nrcs do
Att-Check      for K:=i,natt
                    if (catt(i,K)≠latt(j,k)) then
                         EdgesAcross (lscan,j,cscan,i)
                         go to J-Update
               end;
               if (cscan(i,3)=0) then
                    cscan(i,3)-lscan(j,3)
                    go to J-Update
               Write UF (lscan,j,cscan,i)
J-Update       if (cscan(i,2)=lscan(j,2)) then
                    j=j+1
                    go to Nent-Check
               if (cscan(i,2)<lscan(i,2)) then go to Nent-Check
                    j=j+1
                    go to Att-Check
Nent-Check     if (cscan(i,3)=0) then
                    nent=nent+1
                    cscan(i,3)=nent
                    Uptable
          end;
          EdgesDown (cscan,nrcs)
          Shiftscan (lscan, nrls,cscan, nrcs)
          go to Next-Scan
end;
```

For the previous or last scan line, *lscan(j,1-3)* respectively contain the beginning and end column position and the assigned entity code (i.e., a sequence number). The *jth* entry in *lscan* corresponds to the *jth* entry in *latt,* which holds code values for the *natt* attributes in the file. The *cscan* and *catt* arrays are similar to *lscan* and *latt,* except that they refer to the current scan line. Two values, *nrls* and *nrcs,* refer respectively to the number of records for the last and current scan.

The algorithm loads the first scan line of data into the *lscan* and *latt* arrays. Boundary lengths between the outer area (null) and the runs along the first scan line are then computed by the *EdgesAcross* function and written to the edges file. Function *UpTable* builds the initial version of the entity table. The current number of entities *(nent)* is set to the number of records in the initial scan line *(nrls).* Of course, the vertical, pixel-length boundaries between runs along a scan line must also be computed and the function *EdgesDown* performs this task.

Function *LoadScan* loads the series of run records for the next scan line. At the end-of-file, *EdgesAcross* must again be used to compute lengths between runs along the last scan line and the outer boundary. For all scan lines between the first and last, consecutive lines are examined in order to determine which entity number should be assigned to each run along the current scan line. For a run along the current scan line to be considered contiguous to a run from the last scan line, at least one pixel overlap must exist. For the contiguous runs, all attribute code values must match before *cscan (i,3)* can be assigned an entity sequence number. If any of the *natt* attributes does not match, the vertices, and length of the edges between the two runs must be computed and written to the edges file. Function *EdgesAcross* handles this task.

If *cscan (i,3)* is presently unassigned, it takes on the value of *lscan (j,3)*. If not, then the current run overlaps and matches more than one run from the previous line and function *Write UF* must write an edit request to an update file so that previously processed records can be assigned properly in a second pass. This is necessary since polygons coalesce as the processing proceeds from one scan line to the next. Assigned entity numbers, both in the entity table and in the edges file, must be updated. Since the entity table would be stored in memory, pointers can be updated. For the output raster file, and the edges file, however, a second pass is required to update previously assigned entity numbers (see Figure 4).

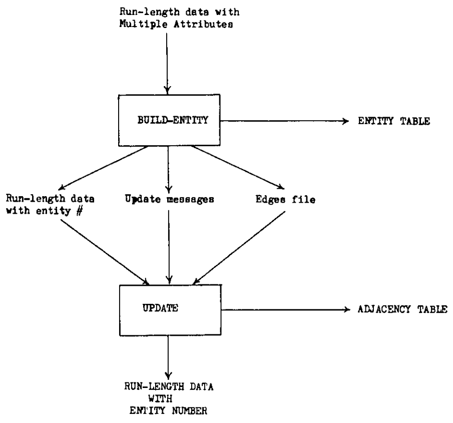

Run-length data with Multiple Attributes
BUILD-ENTITY
ENTITY TABLE
Run-length data with entity #
Update messages
Edges file
UPDATE
ADJACENCY TABLE
RUN-LENGTH DATA WITH ENTITY NUMBER

Figure 4. Flowchart of the Build-Entity Procedure

ADVANTAGES OF THE RASTER-ENCODED POLYGON DATA STRUCTURE

The raster-encoded polygon (REP) structure offers certain advantages for storage, manipulation, and retrieval over conventional forms of both raster or vector format data. With regard to storage, a run notation is maintained with the REP format. Instead of each run record carrying all associated attribute code values, however, only a single value, the index to the entity table, is associated with each run record. This reduces the volume of the raster data file over that of either conventional or compressed raster notations. Manipulative operations, especially those involving attribute-checking such as merge and dissolve, are perhaps most effectively carried out on topologically-structured vector data. Yet such operations can be done quite efficiently on the REP format since the entity table provides complete attribute and adjacency information. A pass through the entity table provides the index numbers which meet constraints for adjacent attributes and a pass through the data file provides all run segments associated with the index numbers.

Perhaps the chief advantage of this format, however, lies in the fact that subsequent merging and retrieval operations can be carried out efficiently on the REP format, since the basic raster organization of the data is maintained. Windowing operations are clearly more easily carried out in raster format for both regular and irregular windows.

For example, the difficulties associated with polygon overlay make it problematic to define and retrieve vector data from irregular window areas such as corridor search zones around convoluted stream segments. As long as these problems remain and these types of query remain important to land resource analysis, raster formats such as those implicit in REP will continue to be advantageous and preferred for storage of areal data in a spatial data base management system.

SUMMARY

The design and development of effective spatial data base systems for land resource applications requires an understanding of the importance of the polygon overlay capability. Of course, the problem can be dealt with in one of several ways. First, it is possible for well-defined research management problems to limit the number of overlays involved. The concept of "integrated terrain unit mapping" advanced by Dangermond (1979) avoids the problem of polygon overlay, at least for the initial task of creating the data base. Also, there are methods of polygon overlay which work, though few, if any, seem to work on sizeable data sets.

None of these solutions seem acceptable for most land resource analysis applications in which data requirements are diverse, data sets may be added incrementally, and the data base needs to cover large geographic areas at scales and resolution which result in complex vector data set. Resorting to a vector-to-raster conversion to facilitate overlay, and then from raster-to-vector for storage, seems a bit short-sighted. Study areas seldom coincide with sheet boundaries and the outlines of prospective study areas will usually not have been anticipated. This means that it must be possible to retrieve data routinely for highly irregular search areas. Only raster data formats and particularly compressed raster data sets, can address these types of query efficiently. On the other hand, vector formats for polygon data are truly advantageous for searches involving threshold sizes and adjacency. By transforming the compressed raster data into an entity-based data structure such as REP, it will be possible to efficiently address these types of query.

These advantages, coupled with advances in raster-based computer hardware, seem to indicate that the REP data structure holds much promise for land resource applications. Others have demonstrated, for example, that the hardware required to generate video signals from run-length encoded records is straightforward (Newman and Sproull, 1979). Color or grey-tone values for a look-up table could be generated easily from the entity table. These advances in data structure, coupled with less expensive micro-based hardware systems, hold much promise for geographic information systems developments in the 1980s.

REFERENCES CITED

Dangermond, Jack, 1979, "A Case Study of the Zulia Regional Planning Study, Describing Work Completed" in Second International Symposium on Topological Data Structures for Geographic Information Systems, Geoffrey Dutton, ed.

Knapp, E., et al, 1979, Spatial Data Integration I: Concepts, Requirements, and Current Capabilities, CSC/TM-7916241, Computer Science Corporation, Silver Spring, Maryland, pp. A-1 to A-11.

Miller, Stephan W., 1980, "A Compact Raster Format for Handling Spatial Data," Proceedings of the Fall Technical Meeting of the American Congress on Surveying and Mapping and the American Society of Photogrammetry, pp. CO-4-A-1 to CD4-8-18.

Miller, Stephan W., 1981, "Compressing Interpreted Satellite Imagery for Geographic Information Systems Applications over Extensive Regions," Proceedings of the Pecora VII Symposium, pp. 341-357.

Monmonier, Mark S., 1975, "Internally Stored Scan Lines to Increase the Efficiency of Small Memory Line Printer Mapping Programs," American Cartographer, Vol. 2, No. 2, pp. 145-153.

Nichols, David A., 1981, "Conversion of Raster Coded Images to Polygonal Data Structures," Proceedings of the Pecora VII Symposium, pp. 508-515.

Rosenfield, Azriel, 1978, "Extraction of Topological Information from Digital Images," in First International Symposium on Topological Data Structures for Geographic Information Systems, Geoffrey Dutton, ed.

Tomlinson, R.F. and Boyle, A.R., 1981, "The State of Development of Systems for Handling Natural Resources Inventory Data," Cartographica, Vol. 18, No. 4, pp. 65-95.