A DATA STRUCTURE FOR A SPATIAL INFORMATION SYSTEM

Robert M. Haralick
Department of Electrical Engineering
Department of Computer Science
Virginia Polytechnic Institute
and State University
Blacksburg, Virginia 24061
and
Linda G. Shapiro
Department of Computer Science
Virginia Polytechnic Institute
and State University
Blacksburg, Virginia 24061

I. Introduction

We consider maps to be a visual representation of spatial data. We call the formal organizational structure by which we may represent spatial data in the computer a spatial data structure. In this paper we give a definition of spatial data structure and some examples illustrating its use in raster format data, in vector format data, and in procedures which do inferential reasoning with spatial data. The structure is rich, flexible, and efficient enough to logically store any of the spatial information in maps, line drawings, region adjacency graphs, and other geographic entities that we might desire to represent.

II. A Spatial Data Structure

An atom is a unit of data that will not be further broken down. Integers and character strings are common

examples of atoms. An attribute-value table A/V is a set of pairs A/V = {(a,v) | a is an attribute and v is the value associated with attribute a}. Both a and v may be atoms or more complex structures. For example, in an attribute-value table associated with a structure representing a person, the attribute AGE would have a numeric value, and the attribute MOTHER might have as its value a structure representing another person.

A spatial data structure D is a set D = {R1,...,RK} of relations. Each relation Rk has a dimension Nk and a sequence of domain sets S(1,k),...,S(Nk,k). That is for each k = 1,...,K, Rk $\subseteq$ S(1,k) X ... X S(Nk,k). The elements of the domain sets may be atoms or spatial data structures. Since the spatial data structure is defined in terms of relations whose elements may themselves be spatial data structures, we call it a recursive structure. This indicates 1) that the spatial data structure is defined with a recursive definition (and not that the information stored in it is infinitely recursive), and 2) that it will often be possible to describe operations on the structure by simple recursive algorithms.

A spatial data structure represents a geographic entity. The entity might be as simple as a point or as complex as a whole map. An entity has global properties, component parts, and related geographic entities. Each spatial data structure will have one distinguished binary relation containing the global properties of the entity that the structure represents. The distinguished relation is an attribute-value table and will generally be referred to as the A/V relation. When a geographic entity is made up of parts, we may need to know how the parts are organized. Or, we may wish to store a list of other geographic entities that are in a particular relation to the one we are describing. Such a list is just a unary relation, and the interrelationships among parts are n-ary relations.

For example, we may represent the state of Virginia by a spatial data structure. In this case, the A/V relation would contain global attributes of the state such as population, area, boundary, major crop, and so on. The values of most of these attributes (population, area, major crop) are atoms. The value of the boundary attribute is a spatial data structure defining the boundary.

One obvious division of the state is into counties. A list of counties could be included as one of the relations, or it might be more valuable to store the counties in a region adjacency relation, a binary relation associating each region (county) with every other region (county) that neighbors it. Counties, of course, would also be represented by spatial data structures.

Some other geographic entities that are related to a state are its highways, railroads, lakes, rivers, and mountains. Some of these entities will be wholly contained in the state and others will cross its boundaries. One way to represent this phenomenon is to use a binary relation where the first element of each pair is a geographic entity, and the second element is a code indicating whether the entity is wholly contained in the state. The spatial data structures representing the geographic entities themselves would contain more specific information about their locations. Figure 1 illustrates a simplified spatial data structure containing an attribute-value table, a county adjacency relation, and a lakes relation for the state of Virginia.

Comparison to Other Geographic Data Structures

The spatial data structure defined in this paper can easily store the spatial data used in well known vector based spatial systems such as the Canada Geographic Information System (Tomlinson, 1967), the U.S. Census DIME files (Cooke and Maxfield, 1967), and POLYVRT (Harvard Laboratory for Computer Graphics and Spatial Analysis, 1974). We illustrate this by defining a system with similar characteristics. In our system, a point is an atom consisting of an ordered pair (X,Y) where X represents latitude and Y longitude. A chain C is a spatial data structure C = {A/VC, LP}. LP is an ordered list (unary relation) of points that define the chain. The attribute-value table A/VC of a chain contains the attributes LEFT_POLYGON, RIGHT_POLYGON, NEXT_CHAIN_LEFT, and NEXT_CHAIN_RIGHT whose values correspond to the pointers in Tomlinson's system. The attribute-value table can also contain such global information as the length of the chain or a function to be used to interpolate between the points.

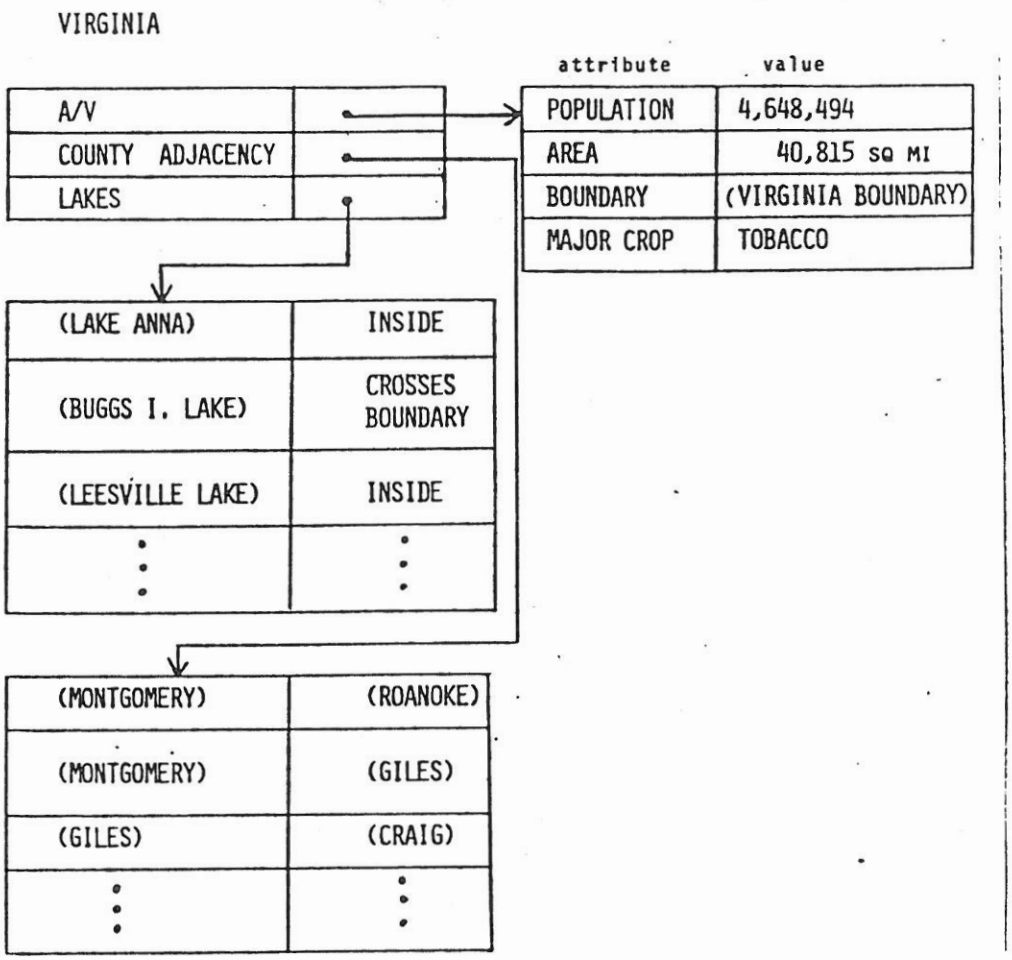


Figure 1 illustrates a spatial data structure representing the state of Virginia. The A/V relation is an attribute-value table. The COUNTY ADJACENCY relation contains pairs of adjacent counties. The LAKES relation contains pairs consisting of a lake and an indication of whether it lies inside or on the boundary of the state.

A polygon P is a relatively simple spatial data structure P = {A/VP}. In this case, the attribute-value table contains the attributes FIRST_CHAIN and POSITION. The value of FIRST_CHAIN is the first chain of the polygon, and the value of POSITION is LEFT or RIGHT depending on whether the polygon lies to the left or to the right of the first chain. Such global attributes as AREA and CENTROID can also be stored in the attribute-value table.

Figure 2 illustrates this structure for a simple 'map' of two regions P1 and P2. In this example, we have chosen the chains to be the longest sequence of points that have exactly one region to their right and one region to their left. The directions of the chains

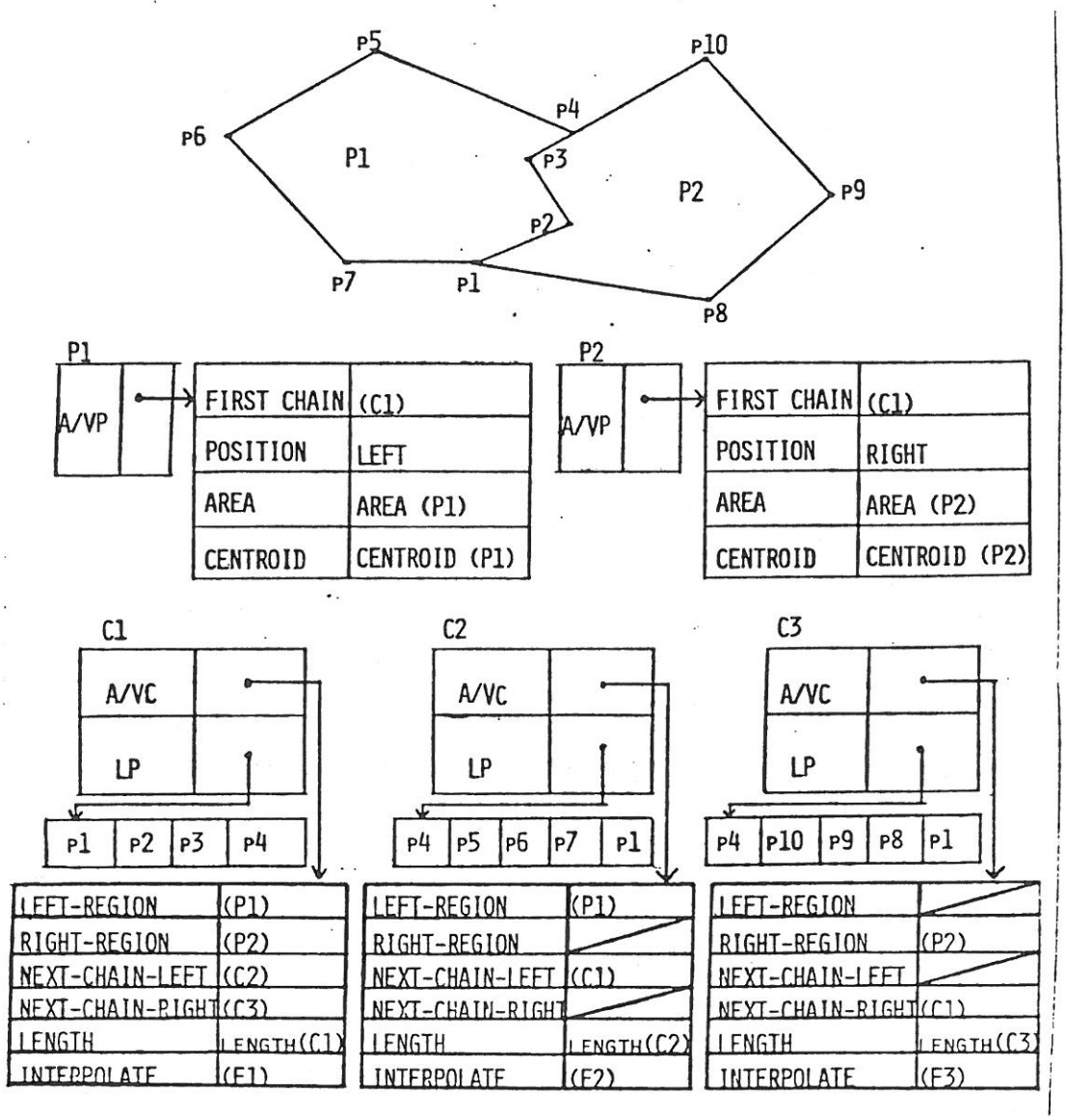


Figure 2 illustrates a spatial data structure for a simple map that encompasses the structures used in the Canada Geographic Information System, the DIME system, and the POLYVRT system.

were chosen arbitrarily. We do not mean to suggest that the points in a chain be stored sequentially as (X,Y) condinates. Instead, we are leaving the physical storage mechanism open. In some applications, storing differences or using Freeman chain codes (Freeman, 1974) might be appropriate. In other applications, storing the chain in a parametric functional form might be appropriate. Regardless of the physical form of storage, the structural aspect of the representation is the same.

Because the spatial data structure is a recursive structure, it can naturally handle the hierarchy of a region and its holes. We define the boundary of a region as follows. A boundary is a polygon plus a (possibly empty) list of boundaries of interior polygons. Thus a boundary can be represented by a spatial data structure B = {A/VB, LB} where A/VB contains the attributes FIRST_CHAIN and POSITION, and the unary relation LB is a list of boundaries. As before, FIRST_CHAIN is the first chain of the polygon, and POSITION indicates whether the bounded region lies to the left or the right of the first chain.

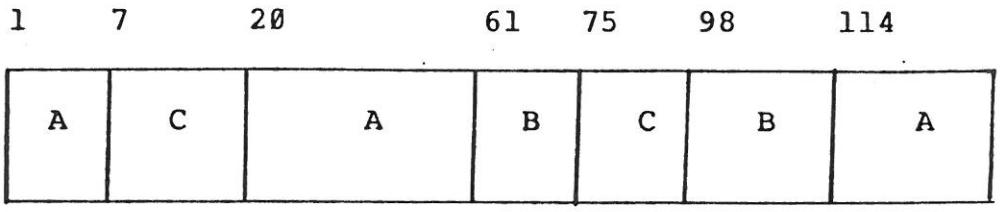
When LB is empty, B is a polygon or simple boundary. When LB is not empty, then B has holes in it. Each of these holes is also a boundary, so it may also have holes. Thus this spatial data structure handles the hierarchical polygonal data structures. Other hierachic structures such as Edwards, Durfee, and Coleman's (1977) and Brassel's hierarchically organized spatial data base of Thiessen polygons can be handled similarly by our spatial data structure.

In the triangle data structure (Gold, 1976; Males, 1977), each triangle points to its vertices and adjacent triangles. Thus a triangle is a spatial data structure T = {LV, AT} where LV is a list of three vertices, and AT is a list of adjacent triangles. Each vertex V is a spatial data structure V = {A/VV} where the attribute-value table A/VV contains the attributes SLOPE, ELEVATION, and other information.

The systems just described all store their geographic data in vector form. Vector form is only one form of spatial data. It represents areas by their boundaries and has the advantage of a very compact representation.

Raster or grid form is another form of spatial data. In this form areas are represented by the grid cells that cover them. The advantage of raster format data is the simplicity of performing certain tasks such as map overlay. Our spatial data structure can handle raster form data just as easily as it handles the vector form.

Consider, for example, storing an entire map of regions in a run length encoded form of raster grid cell data. Shown below is one row of such map data.



In the row shown, there are seven intervals, each one of which belongs to one of three regions: A, B, or C. Each interval is specified by a beginning pixel, an ending pixel, and an interval label. By grouping together all intervals of the same label we may represent this row by the following table.

		Internal List Name	
A:	(1,6),(20,60),(114,130)	IL47A	Partition
B:	(61,74),(98,113)	IL52D	List
C:	(7,19),(75,97)	IL36E	PL81B

In this case, the row points to the partition list (Merrill, 1973) PL81B which contains the interval lists IL47A, IL52D, and IL36E, each of which contains a set of intervals for some region in the row.

Grid cell data structures explicitely represent areas. We will call the entities employing this representaion 'map areas'. Thus, the entity 'map area' is a spatial data structure MA = {A/VMA,PLR}, where the attribute-value table A/VMA has the attribute THEME with values such as 'soil type' or 'land use'. PLR is the partition list binary relation. It consists of a set of ordered (row,partition list) pairs. The entity

'partition list' is a spatial data structure PL = {A/VPL,ILS}, where the attribute-value table A/VPL has the attribute ROW whose value is the number of the row being divided up by the partition list. ILS is the set of interval lists composing the partition PL. Finally, the entity 'interval list' is a spatial data structure IL = {A/VIL,HS}, where the attribute-value table A/VIL contains the attributes NAME and ROW. The attribute NAME takes on a value which is the name of the region to which the intervals in IL belong. The attribute ROW has as its value the row number. HS is the ordered list of horizontal strips (intervals) in the interval list. Each strip in HS is an ordered pair whose first component is the beginning pixel and whose second component is the ending pixel of the strip.

All of the fast geometric distance, region editing, point in polygon, and line intersection algorithms in vector format or in raster format have data structure requirements that are easily and naturally fulfilled by the general spatial data structure. In the next section we illustrate that the general spatial data structure can handle the data for inferential reasoning required by the intelligent query capability that we desire a spatial information system to have.

III. Design of a Spatial Information System

We are currently involved in the design and implementation of an experimental spatial information system using the spatial data structures concept of Section II. The system will answer user queries and solve problems presented to it in a subset of English. In this section, we describe some of the important features of the proposed system.

Major Data Structures

The spatial data structure is the primitive or building block of the system. A finite number of spatial data structure types will be allowed. For instance, the system might include spatial data structures representing the high-level entities states, cities, counties, highways, rivers, lakes, and mountains and the lower-level entities boundaries, simple boundaries, and chains. Thus the system might contain a spatial data structure whose name is MONTGOMERY and whose type is COUNTY.

For each type of spatial data structure, the system will keep a prototype structure. The prototype will indicate what attributes are found in the attribute-value relation of this type of spatial data structures and what relations besides the A/V relation comprise the data structure. Similarly a finite number of relation types will be allowed, and the system will keep prototypes of the allowable relations. Thus the STATE prototype might indicate that all spatial data structures of type STATE have a COUNTY_ADJACENCY relation. The COUNTY_ADJACENCY prototype would indicate that this is a binary relation and that both components of each pair in the relation are spatial data structures of type COUNTY. (See Figure 3 in the section entitled Primitive Operations.) A user query might involve a specific spatial data structure or a specific type of spatial data structure. For fast access in either case, the system will include a spatial data structure name dictionary that maps a name to a spatial data structure and a spatial data structure type dictionary that maps a type to a list of all spatial data structures of that type. Similarly a relation type dictionary will map a relation type to a list of all relations of that type. We are planning to implement relations as relational trees (Shapiro, 1979). Note that all of these structures can be represented by spatial data structures, unifying the whole system.

Primitive Operations

The spatial data structure is a relational structure. Because of this, the spatial database system shares many characteristics of relational database systems. In particular, all of the primitive operations used in relational database systems are applicable to the relations of a spatial data structure. We will use a small example database to motivate the use of these and other primitive operations.

Figure 3 illustrates a set of prototypes for spatial data structures and their relations that might be found in a spatial information system. The STATE prototype indicates that STATE is a type of spatial data structure having an A/V_STATE relation, a COUNTY_ADJACENCY relation, and a RIVERS relation. The A/V_STATE relation has four attributes: NAME, whose

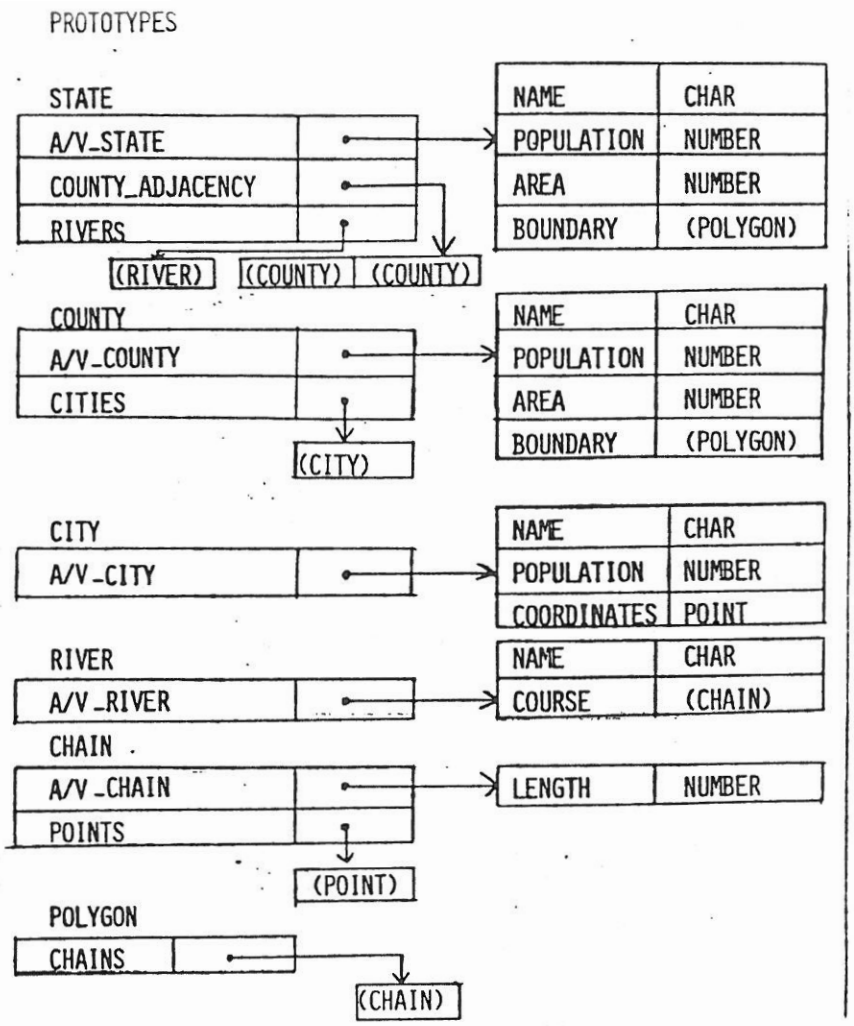


Figure 3 illustrates a set of prototypes for the spatial data structure types STATE, COUNTY, CITY, RIVER, CHAIN, and POLYGON and the relation types COUNTY ADJACENCY, RIVERS, CITIES, POINTS, and CHAINS.

value is a character string, POPULATION and AREA whose values are numbers, and BOUNDARY whose value is a spatial data structure of type POLYGON.

The COUNTY_ADJACENCY relation is a binary relation, and each member of each pair is a spatial data structure of type COUNTY. The RIVERS relation is a unary relation, and each element is a spatial data structure of type RIVER. The other prototypes convey similar information.

The following questions are possible queries to a spatial information system having the prototypes of Figure 3. Under each question, we suggest a sequence of operations that might be performed to answer the query.

1) What cities are in state X ?

A. Locate state X.
B. Perform a projection operation on COUNTY ADJACENCY(X) to obtain a list of counties.
C. For each county Y in the list
For each city C in CITIES(Y)
1. Look up N = NAME(C).
2. Add N to the relation being created.

2) What cities lie on rivers in state X?

A. Locate state X.
B. Perform a projection operation on COUNTY ADJACENCY(X) to obtain a list of counties.
C. For each county Y in the list
For each city C in CITIES(Y)
For each river R in RIVERS(x)
if
POINT_CHAIN_DISTANCE(COORDINATES(C), COURSE(R))=Ø
then add C to the relation being created.

3) What counties in state X does river R flow through?

A. Locate state X.
B. Perform a projection operation on COUNTY ADJACENCY(X) to obtain a list of counties.
C. For each county Y in the list
if CHAIN_INTERSECTS_POLYGON(COURSE(R), BOUNDARY(Y))
then add Y to the relation being created.

From these and other sample queries, we find the following operations are necessary: projection in the relational database sense, selection in the relational database sense, intersection or join in the relational

databases sense, look up the value of an attribute, call on geometric or distance functions, create a list, add elements to a list, comparison, determine if an N-tuple is a member of a relation, create a new relation, and create a new spatial data structure.

IV. Summary

We have defined a general spatial data structure to be used as the building block in a spatial information system. The structure is general enough to represent all the data structures used in previous systems without changing their logical structures and can handle vector or raster data with equal use. We have discussed the major data structures needed for a spatial information system. Finally we have presented a set of possible queries, described the sequence of operations needed to answer them, and used these sequences to motivate the primitive operations required for the spatial information system.

REFERENCES

1. Brassel, K., "A Topological Data Structure for Multi-Element Map Processing," An Advanced Study Symposium on Topological Data Structure for Geographic Information Sustems, Harvard University, Cambridge, Massachusetts, October 1977.

2. Cooke, D. and W. Maxfield, "The Development of a Geographic Base File and Its Uses for Mapping," Proceedings of URISA, Garden City, Long Island, September 1967.

3. Edwards, R.L., R. Durfee, and P. Coleman, "Definition of a Hierarchical Polygonal Data Structure and the Associated Conversion of a Georgraphic Base File from Boundary Segment Format," An Advanced Study Symposium on Topological Data Structure for Geographic Information Systems, Harvard University, Cambridge, Massachusetts, October 1977.

4. Freeman, H., "Computer Processing of Line Drawing Images," Computing Surveys, Vol. 6, No. 1, March 1974, pp. 57-97.

5. Gold, C., "Triangular Element Data Structures," Users Applications Symposium Proceedings, The University of Alberta Computing Services, Edmonton, Alberta, Canada, 1976.

6. Laboratory for Computer Graphics, "POLYVERT: A Program to Convert Geographic Base Files," Harvard University, Cambridge, Massachusetts, 1974.

7. Males, R., "ADAPT - A Spatial Data Structure for Use with Planning and Design Models," An Advanced Study Symposium of Topological Data Structures for Geographic Information Systems, Harvard University, Cambridge, Massachusetts, October 1977.

8. Merrill, R., "Representation of Contours and Regions for Efficient Computer Search", CACM, Vol. 16, 1973, pp. 69-82.

9. Shapiro, L. G., "Data Structures for Picture Processing" to appear in Computer Graphics and Image Processing, 1979.

10. Tomlinson, R., "A Geographic Information System for Regional Planning," Land Evaluation (Stewart, ed.), McMillian of Australia, Sydney, Australia, 1968.