ADVANCED DATA STRUCTURES FOR THE INTERPRETATION OF IMAGE AND CARTOGRAPHIC DATA IN GEO-BASED INFORMATION SYSTEMS

final report
Grant No. NAG 5-369

submitted to
NASA Goddard Space Flight Center

by
Dr. Donna J. Peuquet
Dept. of Geography
University of California at Santa Barbara
Santa Barbara, California 93106

(NASA-CR-176559) ADVANCED DATA STRUCTURES FOR THE INTERPRETATION OF IMAGE AND CARTOGRAPHIC DATA IN GEO-BASED INFORMATION SYSTEMS Final Report (California Univ.) 84 p HC A05/MF A01 CSCL 08B G3/43

N86-20933

Unclas
05558

ADVANCED DATA STRUCTURES FOR THE INTERPRETATION OF IMAGE AND CARTOGRAPHIC DATA IN GEO-BASED INFORMATION SYSTEMS

final report
Grant No. NAG 5-369

submitted to
NASA Goddard Space Flight Center

by
Dr. Donna J. Peuquet
Dept. of Geography
University of California at Santa Barbara
Santa Barbara, California 93106

INTRODUCTION

This document represents the final report for the project entitled "Advanced Data Structures for the Interpretation of Image and Cartographic Data in Geo-Based Information Systems" (award no. NAG 5-369). This report is necessitated by the move of the Principal Investigator to The Pennsylvania State University and describes work still in progress. A proposal to continue this research and complete current work elements at PSU is currently in preparation.

Overview

The purpose of this project was to investigate the use of new methods to improve the flexibility and overall performance of very large, heterogeneous databases.

This project commenced in 1983 to study new methods of representing spatial data that would be suitable for very large heterogeneous database applications. The original project envisaged cooperative empirical comparison of two newly developed hybrid data structures, one being the Vaster structure developed by the Principal Investigator, and the other being the Topological Grid structure developed at NASA/GSFC.

The objective was to test whether such hybrid structures represent an improvement in performance over other previously known structure types, and to quantify and compare the performance characteristics of the Vaster and Topological Grid structures.

Two problems related to designing any empirical test which would be meaningful and valid soon became apparent in this work; 1) No theoretical framework of spatial data structures then existed to use as a reference for evaluating the selection of the specific data structures chosen for testing among the possible alternatives, existing and potential. 2) The primitive tasks to be performed for the comparison were found to be in some cases difficult to define in that they tend to be context dependent. This is attributable to the fuzzy nature of spatial entities and spatial relationships.

This finding also had two far-reaching implications; 1) It indicated a significant gap in the necessary knowledge for designing and implementing any integrated spatial database or geographic information system with predictable results. 2) Conventional algorithmic approaches for spatial data handling are inadequate for accommodating the required variety of applications or for providing the required level of efficiency.

As a result of these preliminary findings, a review of existing spatial data structures was performed and an overall comparative framework developed. This work was documented in the report entitled "Spatial Data Models: A Conceptual Framework and Comprehensive Review".

In addressing the problem of algorithmic approaches for handling large, heterogeneous databases, it was readily apparent that Artificial Intelligence techniques hold much promise as tools for providing the efficiency, flexibility and robustness required for the spatial databases envisaged by NASA. This use of AI techniques in a geographic/earth-sciences database context has only begun to be explored. As a result of this finding, a major extension of the original work in this area was begun in the latest phase of the project.

This expanded work was performed in conjunction with an overall research effort that was also funded by the U.S. Geological Survey, the National Science Foundation and Digital Equipment Corporation to design and build a knowledge-based geographic information system.

The specific task associated with this overall research effort is to explore methodologies that will allow the following GIS performance requirements to be satisfied within a single, unified environment:

1) the ability to process large, multi-layered, heterogeneous databases;

2) the ability to query such databases about the existence, locations and properties of complex spatial objects;

3) a level of efficiency in responding to queries that allows the system to be tailored easily to accommodate a variety of applications.

The achievement of these requirements imply the following capabilities within a GIS:

1) the ability to answer a wide range of complex queries posed by the scientist concerning phenomena that may not be explicitly encoded in the database;

2) the use of knowledge-based, non-exhaustive search to limit and control the level of database retrieval needed to answer queries;

3) the use of an extremely efficient and robust database architecture; and

4) the ability to inductively 'learn' new information regarding spatial objects and the relationships between those objects.

The system currently under construction, called KBGIS II, is based upon the design concepts and overall capabilities demonstrated in a simple, 'proof-of-concept' system called 'KBGIS I'. This system was completed in late 1983 with funding from the U.S. Geological Survey and Digital Equipment Corporation.

Construction of KBGIS II began in mid-1985 with added support from NASA/GSFC. This enabled a specific emphasis on innovative techniques for spatial data representation and the use of AI techniques for search and retrieval in a very large, heterogeneous spatial database.

In the latest phase of the NASA/GSFC grant, the work elements specified were the following:

1) The development of high-level search heuristics which incorporate:

   a) high-level descriptive knowledge concerning the general characteristics of specific types of objects and where they are likely to geographically occur; and

   b) knowledge concerning what information is contained in the system, the overall functional characteristics of the system and where specific classes of information are stored.

2) The development of efficient, non-exhaustive Spatial Search procedures including:

   a) the detection of complex feature presence or absence through the use of "spectral signatures;" and

   b) the use of spatial spectra and other heuristics for guiding hierarchical search by indicating directions for prioritizing search.

3) The development of object-based search procedures, for both general and domain-specific heruristics. This would include:

   a) object retrieval based on full, or complete, object descriptions; and

   b) object retrieval using rules of inference for objects when only partial object descriptions or descriptions of "similar" objects are available.

4) Investigation of methods to increase the storage and performance efficiency of the database.

In terms of the architecture of KBGIS II, these elements are manifested as a dual object/spatial database and a three-tier hierarchical search subsystem. The nature and current status of these components are briefly reviewed below.

SYSTEM COMPONENTS

Spatial Language

All spatial objects in KBGIS II are expressed in terms of a language which was developed as part of the work supported by the USGS. In this language, a spatial object is a set of pixels having various properties:

1. Properties assignable to individual pixels, such as landuse, ground cover class or elevation. These are termed PPROPs.

2. Properties assignable to a group of pixels, or to individual spatial entities (e.g., a lake or a road). Examples of such properties include size, shape and orientation. These are termed GPROPs.

3. Properties assignable to two groups of pixels or to two individual spatial entities. These are relationship properties, termed RPROPs, and include distance, relative direction and containment.

Using these three classes of properties and the logical connectives /\ (and), V (or) and ~ (not), one may represent arbitrary spatial objects. For example, using infix notation to denote properties and "obji" to represent the ith spatial subobject, we may represent a city as a residential annulus (obj1) of a given size surrounding a commercial core (obj2) of a given size as follows:

```
((el (LU ?obj1 commercial))
 (el (AREA ?obj1 (30 40)))
 (el (LU ?obj2 residential))
 (el (AREA ?obj2 (50 60)))
 (el (CONTAINS ?obj2 ?obj1) t ) ).
```

In this language, a specific spatial object is defined when each obji in the object description is bound to a specific set of spatial indices (i.e., when all predicates "el" takes on the value "true"). Some fuzziness may be introduced into object descriptions by allowing "el" to take on a range of values.

The queries in the system are the fundamental, inverse queries:

(find locations <number of instances> <spatial object> <spatial window>)

and

(find objects <spatial window> <class of object>)

In the first type of query, the spatial object is defined in terms of the language, and is satisfied when sets of spatial locations are correctly bound to obji for each i.

The spatial object language and the queries described above are similar to those in some other spatial database and image processing systems (see for example Ballard and Brown, 1982, pp. 335-340).

Locational Database

a) General Description

The Locational Database of the system contains, for the most part, information on the spatial distribution of relatively primitive PPROP's, although some high-level (i.e., derived) spatial objects are also stored in part of this database. The overall conceptual structure employed for the Locational Database is the linear quadtree structure. This type of structure has been discussed extensively in the literature (Peuquet, 1984; Mark, 1984; Abel and Smith, 1984). In the area quadtree model used in KBGIS II, a given geographic area is recursively subdivided into four quadrants, as shown in Figure 1. This subdivision of space translates functionally into a regular, balanced hierarchy of degree 4. Any entry at any hierarchical level in the Locational Database is directly addressable utilizing a recursive spatial addressing scheme shown graphically in Figure 2. Each separate type of data, or coverage, can be viewed as being arranged in a separate quadtree with all quadtrees spatially registered in a common coordinate reference system.
A detailed description of the logical structure of the Locational Database is given in Appendix A.

The Locational Database significantly extends the conventional linear quadtree concept to include various types of higher-level information at the nodes of the location tree. Each node in the location tree is structured as a three-dimensional frame (see Figure 3a). One slot is allocated for each data layer in the database. An additional slot is allocated for some data pertinent to the node as a whole. Each layer (slot) in turn is a frame which contains the following slots: a slot that is used to store the data value of the layer at that node; a slot for storing information describing how data for that layer are spatially distributed; a slot for storing the names of domain-specific functions (i.e., system software modules) that can be used to guide search; and a slot for storing temporary flags used during search.

Each of these slots may also contain subunits, called facets, which allow the storage of multiple pieces of information within any given slot. For example, the Search Tag slot has several facets to allow multiple searches to be executed in parallel. The Distribution Parameters slot may also contain more than one statistic for describing various aspects of spatial distribution. This is particularly helpful for dynamically prioritizing the various regions of the image represented in lower levels of the location tree for ordering candidate areas to be searched at each

stage of the process. The data value slot for higher-level (i.e., non pixel-level) nodes can also have more than one facet to allow storage of generalized values that have been computed from different inheritance rules needed for different applications.

Overall, this frame structure has several advantages:

1. It allows any combination of logical layers in the database to be searched simultaneously.

2. It allows the use of heuristic search guided by higher-level knowledge stored within the Location Tree itself to dramatically improve search efficiency.

3. It increases the overall flexibility and robustness of the database for representing spatial information.

4. It allows compatibility with the higher-level, LISP-based portions of the system and a total integration of AI and spatial database techniques, while minimizing the overhead of the C-LISP interface.

5. The Locational Database is the logical dual of the Spatial Object Knowledge database, thus allowing a cleaner overall system design.

b) Current Status

This revised Locational Database design is now fully implemented. Test data for a selected geographical area has also been loaded into the Locational Database. It is felt that this current design not only contains the necessary added elements required by the search process, but that it also represents a dramatic improvement in flexibility and performance efficiency.

Object Knowledge Base

a) General Description

The Spatial Object Knowledge Database stores the definition of all objects, whether user-defined or learned by the system. Spatial objects are hierarchically defined. At the lowest level, a spatial object is a set of pixels characterized by a set of elemental data values such as ground cover and elevation. Higher level spatial objects consist of sets of hierarchically-lower spatial objects with connecting relations (RPROP's) between them. The Spatial Object Knowledge Base thus serves to define implicitly objects that are not directly stored in the spatial database.

As noted above, spatial objects are represented formally in the spatial object language. For each member of an important subset of object definitions there is a frame data structure in the object knowledge base,

that is similar in structure to the Location Tree frame. An illustration of the data structure used in the object knowledge base is given in Figure 3b. This knowledge base is easily expandable by the system user.

The defined by slot contains a list of subobjects that together define the current object.

The heuristics slot contains information that is characteristic of that object and can be used to guide the location tree search when the user desires to locate some examples in a certain area or to find what objects exist in a certain area. An example of such a heuristic might involve the knowledge that an object of class A is frequently found close to objects of class B, which have a frequent occurrence. Such heuristic knowledge is accessible to the rules in the rule-based control system.

The "examples" slot contains the geographical coordinates, expressed as location tree coordinates, of selected known examples of a given spatial object. If more than a few examples are known, then the system limits stored examples to those that are frequently requested or are expensive to retrieve. Expensive objects tend to be those which are complex in that they are defined in terms of many other, more primitive objects. These limiting criteria are needed in order to prevent needless redundancy with the Location Tree Database and to achieve an optimal balance between economical storage and rapid query response.

The information in any slot in the Object Knowledge Database can be added, modified or deleted explicitly by the user by means of a spatial object knowledge base editor. Furthermore, information may be inserted into any slot (except Examples) by the inductive learning phase of the system. Autonomous modification of the Examples slot can be updated with new examples as queries are answered using a specialized set of redundancy-limiting rules, as outlined above.

b) Current Status

The revised and expanded implementation of the data structures for spatial objects are now complete. A few spatial objects have been defined and implemented in the Object Knowledge Database.

Search

The primary task in the latest phase of the NASA/GSFC grant was to design and implement a powerful, heuristic search facility which will use high-level knowledge to guide search and use the hierarchical nature of the storage structures to maximum degree. The design objective of this facility is to avoid search in unlikely portions of the database through the use of a priori information. Maximally efficient search is the counterpart to the flexible and robust database structure, and is essential to enable any GIS to handle a very large, heterogeneous geographic database.

A single, integrated control structure which can efficiently handle both object-based search and locational search efficiently was designed. Since heuristic locational search was viewed as the more complex and less explored form of search, this was chosen first for detailed study and implementation. A general overview of the search control structure is given below. This is followed by a description of the spatial search process that employs this structure.

a) General Description

The approach used for the control of search over the input data sets involves the joint application of five principles:

1. The use of hierarchical decomposition in both data structures and in search procedures applied to the data structures.

2. The use of best-first search procedures, in which domain-specific knowledge is used to reduce as much as possible the sets of spatial indices explored in satisfying queries.

3. The use of a constraint satisfaction approach to query satisfaction.

4. The use of recursion.

5. The use of dynamic updating of the system's knowledge base in response to query satisfaction.

Although these five principles have been employed in other research, the particular combination that has been employed in the construction of KBGIS II appears to be unique. In particular, it is one of the few known control structures that combines a range of AI and spatial database techniques in answering queries about complex spatial objects from a combination of large databases and knowledge bases.

When a query is entered by a system user, it is parsed and checked for syntactic correctness, and the user is prompted for any necessary modifications. The object of the query is then transformed into a semantic network representation in which the links represent RPROP relations between the subobjects of the query (i.e., constraints) that must be satisfied. The network is then augmented with heuristic knowledge and the subobjects at the nodes are ordered. A recursive, constraint satisfaction procedure is then applied to the nodes in the designated order. Search first occurs in the Object Knowledge Base for stored examples of specific subobjects that satisfy the relational and spatial constraints. If satisfaction of the query cannot be accomplished by this lookup procedure, new procedures are invoked (recursively) to search the Locational Database. When a query is ultimately satisfied by a search of the Locational Database that is considered 'expensive', the examples found of the 'expensive' object or

subobject are stored in the Object Knowledge Base for use in future queries.

The recursive hierarchical structure of the Locational Database is employed to maximum degree for search of the Locational Database. The major control of query satisfaction is also hierarchical, being performed via three levels of control:

1. High-level search on the semantic network derived from the Object Knowledge Base.

2. Mid-level search using high-level spatial knowledge to select heuristics for low-level search and avoid redundant search.

3. Low-level search on the Locational Database.

The use of best-first, knowledge-based search is intended to reduce the amount of search required to satisfy a query.

b. The Spatial Search Process

1) High-Level Search

The search process takes a syntactically correct query as input. The query is first transformed into a semantic net. The net is implemented as a set of structures. These structures are of two kinds, one representing objects (named the subobject structure), and the other representing the RPROPs (named the relation structure). The RPROPs conceptually function as the relational links between subobjects, and the nodes in the semantic net.

The process of transformation consists of analyzing each conjunctive term in the query and either creating a new instance of a structure, or storing information in the appropriate slot of an existing structure. Each instance of the subobject structure represents an object that is a subobject of the query or a subobject in the definition of some higher object in the Object Knowledge Base. The subobject structure slots are shown in Figure 4.

At the end of the initial transformation of the query into a semantic net the type, relations, PPROP and GPROP slots of the subobject structures created are filled. Global lists of all the subobjects (nodes) and all the relations (arcs) in this primary net are stored in a short term memory area. The rules attached to the heuristics slot in the object data structure are then used to augment the primary net by creating new subobject and relation structures.

Next, the subobjects in the semantic network representation of the query object are prioritized to determine the order in which the individual subobjects and their associated RPROP's (i.e., constraints) will be passed to the lower levels of the search process for resolution. Prioritization

for this constraint satisfaction process is accomplished using two factors:

1) Prerequisite object(s) needed to satisfy RPROPs.

2) The level of constraint (i.e., estimated amount of areal elimination) imposed by the RPROP.

Levels of constraint for RPROPs in this context are determined through the use of the distribution knowledge stored in the high levels of the Locational Database. From this knowledge, information on the typical size of the object, form of the distribution (e.g., degree of clustering) and areal concentrations can be derived for any search window and for any layer or combination of layers. This distribution knowledge is computed and saved in the Locational Database at the time of initial data input and is thus always present.

The estimated computation cost of spatial search for a given object or subobject is calculated primarily on the basis of the estimated total number of location tree nodes which must be accessed given the stored distribution knowledge. The prioritization procedure aims at maximally utilizing all available information to minimize the total system effort used in satisfying a query in anticipation of a Locational Database search. This information includes object-oriented knowledge, locationally-oriented knowledge and knowledge concerning the performance characteristics of the system itself.

The first step in the search procedure attempts to satisfy all the unary predicates (GPROPs) that apply. Processing consists of comparing stored GPROP information against desired GPROP information and matching the ranges of function values. If no information on the desired GPROP is available for a particular example, it is pushed to the end of the queue of example locations.

The constraint satisfaction checking is accomplished by backtracking through the space of examples. Subobjects are instantiated in decreasing order of priority. The backtracking develops a partial solution based on the application of the RPROPs to the found examples. The advantage of backtracking is that it yields the best partial path using all examples. While in a conventional constraint satisfaction problem it might be desirable to apply arc and path consistency algorithms, in the present case the problem is not a closed one; the domain of each variable can be expanded through spatial search. The set of partial solutions developed is stored in a best first manner.

The k most advanced solutions using previously known or found examples are extracted from the results of the backtracking, where k is an adjustable parameter. This list of partial solutions is iterated through as follows. For each partial solution (best first) the locations of the known examples are bound to the curloc slots of the appropriate primary subobjects. The curloc slots of subobjects that are to be searched is set to nil. The primary subobjects to be searched are stored in a global list. The value of each primary relation is set to TRUE or FALSE depending on the

solution being investigated. The primary relations that are FALSE are also stored in a global list. These two global lists are passed to the mid-level search module, which then initiates a constrained search restricted to a subarea within the search window on the Locational Database. This operates on the subnet defined by the set of currently FALSE relations, and the associated subobjects. The definitions of each subobject to be searched and the heuristic information that pertains to it are available to the mid- and low-level search.

Upon the conclusion of the Locational Database search, control is transferred back to the high-level search module. If the additional instantiations found via locational search do not yet complete satisfaction of the query, the next of the k partial solutions is instantiated. If a solution has been found (i.e., all constraints are satisfied), the query has been answered and the solution is returned.

When the list of partial solutions is exhausted, constraint satisfaction is reapplied to the net and the procedure repeats itself until a solution has been found or a predetermined stopping criterion is satisfied. The reapplication of the constraint satisfaction is based on examples of specific subobjects which have been added by the spatial search module during the course of the k preceding unsuccessful searches to the exampleloc slot of structures.

2) Mid-Level Search

There is a link between the high-level, object-oriented search and the low-level, location-oriented search which we term the mid-level search. The role of the mid-level search is to invoke a low-level search of the Locational Database if information on specific object or subobject instances stored in the Object Knowledge Base is insufficient to satisfy the query, and to directly invoke the GPROP and RPROP operators.

The mid-level search utilizes a series of working memory cells, containing information on the current subobject being searched. Each cell contains the complete locational definition of each candidate instance for each active subobject. These candidate instances can be augmented or initially entered by the low-level search. RPROP constraints are tested by invoking the appropriate function on these cells.

As stated previously, the `object´ to be retrieved to satisfy a user query is defined in terms of a conjunction of subobjects with a certain number of GPROPs. Each subobject can itself be defined in terms of a conjunction of subobjects. Therefore, the definition of the main object can be nested several levels deep. For this reason, the mid-level search, including management of the cells, is implemented as a recursive function.

The mid-level search loads components of the prioritized semantic net created by the query processor for each subobject to be found into a working memory cell. This is done for each subobject, in turn, when it is raised to the head of the priority queue. Since subobjects may appear more than once in the definitions of a series of more complex objects, duplicate

subobjects are temporarily 'marked' to avoid redundant search. If no examples were passed down from the Object Knowledge Base or if known complete examples have already been discarded for not satisfying the constraints, then the mid-level search invokes the low-level search expert. This expert operates directly upon the Locational Database within the current search subwindow to retrieve complete locational definitions for new examples of a given subobject that has been defined as a conjunction of primitive objects. (A primitive object is defined in this context as corresponding to a single layer in the Locational Database.) The process employed by the low-level search expert is described below.

If full locational definitions of known examples are present, the appropriate GPROP operators are called to eliminate all candidate instances not satisfying the given GPROP constraints. Each of these GPROPs is implemented as a separate function. These are: shape, texture, size and distribution. If the subobject represents a conjunction of primitive objects which has been found through search of the Locational Database, these may have been already invoked at that level.

The appropriate PPROP operator is called to sequentially check each of these candidate subobject instances to find examples satisfying the given RPROP constraint. Each RPROP is also implemented as an individual function. All but one of these operate on the locational information contained in two cells. The RPROP functions are: 1)inside, outside (containment), 2) relative direction (8 possible - north, north-east, east, etc.), 3) adjacency (= next to, touching), 4) distance, 6) connectivity, 7) nearest neighbor.

3) Low-Level Search

The locational search for each individual subobject is designed to eliminate as much area as quickly as possible while minimizing the total number of accesses to the Locational Database. This is partially accomplished by performing a search on as many data layers as possible simultaneously (i.e., as each locational node is retrieved). Constraints which can be satisfied during direct search on the Locational Database include all GPROPs (potentially distributed over separate sub-objects).

When the low-level search is initially invoked by the searcher, a locational search queue for the entire query search window is initialized. Search is subsequently limited to within a given locational subwindow which represents the most likely spatial window containing a subobject instance or instances that can potentially satisfy the relevant component of the original query. This subwindow is determined by the high-level search.

When all instances are found within the given subwindow that satisfy the GPROP constraints, the state of the search in relation to the entire search window is saved and control is passed back to the mid-level search. If the instances found do not satisfy the RPROP constraints, control is passed back to the low-level search and the process resumes in a new subwindow.

The location tree search process is divided into two distinct tasks;

1. Find individual instances.

2. Find complete locational coverage of each instance.

Finding individual instances is performed with either a modified breadth-first or a modified depth-first strategy. Finding the complete locational coverage of each instance is performed either as soon as the first location of a new instance is performed either as soon as the first location of a new instance is found via a region growing procedure, or as a batch clustering procedure after all locations occupied by any of the instances are found.

The first step for search in each locational subwindow is to determine the initial pair of search heuristics (one for each of the two steps) to be used. This is done on the basis of;

1. The desired quantity, relative to the total expected number of instances within the search window (i.e., all or most vs. one or few).

2. The overall density of the distribution within the window.

3. The typical size of the desired object (small vs. large).

These are expressed in terms of a small set of rules used to determine the proper choice, and are evaluated based on the distribution information contained in the Locational Database.

If these constraints indicate a non-exhaustive search (e.g., find the largest object), a depth-first strategy is preferred in order to quickly narrow in on the desired instances. If, on the other hand, an exhaustive search is indicated (e.g., find all instances within the search window), the choice of heuristic would in this case be chosen so as to minimize the maximum size of the areal search queue during the location tree search. For example, if the object being sought is known to be 'large' and there are known to be few instances of such an object within the search window, then as soon as the first spatial index contained within such an object is found, that index is immediately passed on to a region growing procedure which finds the remaining areal coverage for that instance. This process is repeated until all desired objects are found.

If, on the other hand, the object being sought is 'small' and 'numerous' then all locations within the location tree which satisfy the constraints for the object being sought are gathered into a single list. This list is then passed to a clustering procedure which groups locations in the individual objects. These two situations represent simple examples of a rule-based procedure for selecting the appropriate combination of search heuristics and parameters to be used.

The order in which the areal quadrants (i.e., nodes of the location

tree) are to be searched for each subobject is dynamically adjusted during spatial search on the basis of global distribution information stored at all levels of the location tree except for the leaves. As this distribution information changes relative to scale (i.e., lower levels of the tree) and to that area of the window currently being searched, the heuristics themselves may also be modified to maximize search efficiency. It must be noted here that due to the nature of heuristics, the fastest possible method of search is not guaranteed for every instance.

4. Types of Spatial Search

The satisfaction of a locationally-based query will usually entail finding a number of objects within a given spatial search window. Each instance of a given object must satisfy the set of GPROPs that apply to that object. In addition, the set of objects found must satisfy all the RPROPs that apply.

The locational search in satisfying a component of a user query can be classed into three distinct types according to the amount of high-level knowledge which is available to guide the search in the form of constraints added to the original user query:

1. Search for known objects which are completely described in the object knowledge database.

2. Search for partially described objects.

3. Search for objects which are not described in the object knowledge database.

In the first class of search, all characteristics which represent high-level knowledge used to guide spatial search are known and stored in the object knowledge database. This includes properties (GPROPs) such as `typical` size, shape. It also includes examples of known locations. In some cases, the stored locational examples may provide enough instances to satisfy the relevant portion of the query. In this case, all search heuristics are bypassed and the process becomes one of direct retrieval.

In the second class of search, some information concerning the desired object is known. Such information includes stored knowledge which describes the characteristics of the class of objects to which the desired object belongs. This general knowledge is used in heuristics very early in the search process to more quickly narrow the spatial areas to be searched. Stored information may also include knowledge of known spatial associations with specific other objects that are fully described. In this case, the search for unknown objects is anchored around the locations of these known objects within the search window.

In the last class of search, the spatial characteristics of the desired object are completely unknown. This relies solely on the lower-level heuristics which use the distribution information of individual

Locational Database data layers to limit and prioritize the spatial areas to be searched. This would obviously be the slowest type of search since it has the least amount of knowledge for guidance. This is, however, the most frequent type of search used during the early phases of system use until higher-level knowledge learned from previous queries is accumulated in the object knowledge database.

c) Current Status

The structure of the overall search control process which incorporates all of the features originally envisioned has been designed and a preliminary test version has been implemented.

Basic principles for using spatial spectral information in guiding spatial search, as well as a method of representing this information within the current database design has also been developed. This is described in a separate report attached as Appendix B.

SUMMARY OF CURRENT STATUS

In light of the abbreviated time of work, it must be noted here that most of the work elements that were specified for the current phase of the grant were cumulative in nature; i.e., commencement of productive work in some areas required completion of major portions of other work elements. Both components of the database (Object Knowledge Base and the Locational Database) were completely redesigned and rebuilt. The overall search control structure, which is intended to handle both object- and locationally-oriented queries with equal ease and efficiency, was designed and a preliminary version was implemented. Building upon this control structure, empirical investigation of the use of this structure for answering locationally-based queries was conducted.

Two factors combined to slow the anticipated pace of this work: First, the VAX/Common Lisp environment proved to be insufficient for a large, multidimensional database. Much time and effort was spent on devising 'clever', low-level methods for stretching the data handling capacity of this environment. Second, new algorithms for many 'standard' spatial data manipulation procedures had to be developed so that they were efficient and compatible with other procedures in this significantly different and innovative database/search design.

BIBLIOGRAPHY

1. Abel, D.J. and J.L. Smith, 1983. "A Data Structure and Algorithm Based on a Linear Key for a Rectangle Retrieval Problem." Computer Vision, Graphics and Image Processing, Vol. 24, pp. 1-13.

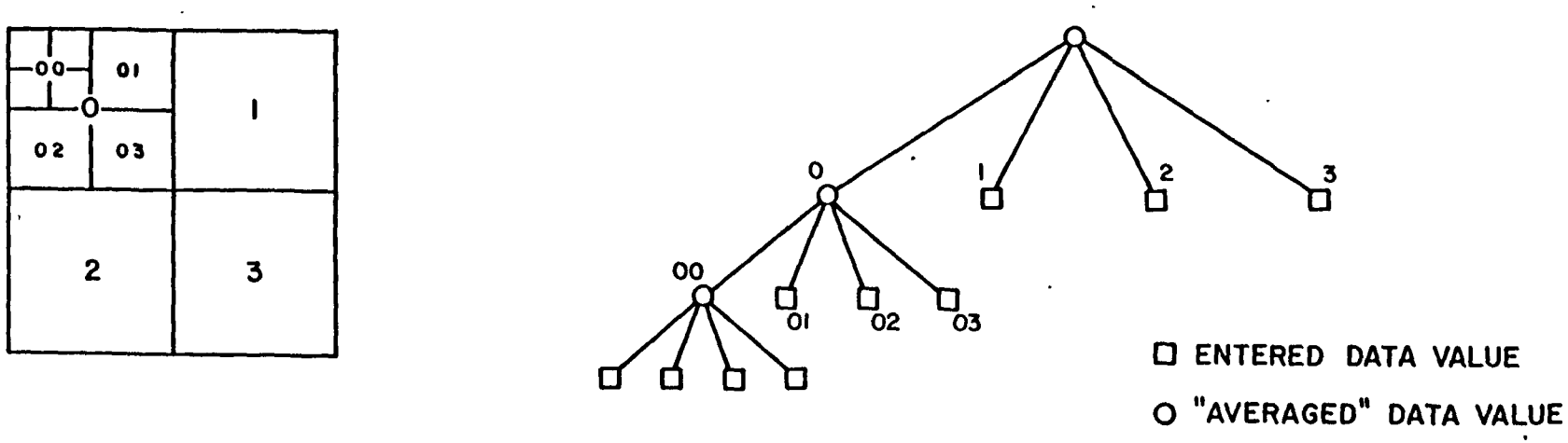
2. Ballard, D.H. and Brown, C.M., 1982. Computer Vision. Prentice Hall, Englewood Cliffs, New Jersey.

3. Mark, D. and J.P. Lauzon, 1984. "Linear Quadtrees for Geographic Information Systems," Proceedings, International Symposium on Spatial Data Handling, Zurich, Switzerland, Vol. 2, pp. 412-430.

4. Nagao, M., 1984. "Control Strategies in Pattern Analysis," Pattern Recognition, 17, pp. 45-56.

5. Peuquet, D., 1984. "Data Structures for a Knowledge-Based Geographic Information System," Proceedings, International Symposium on Spatial Data Handling, Zurich, Switzerland, Vol. 2, pp. 372-391.

6. Samet, H., 1984. "The Quadtree and Related Hierarchical Data Structures," ACM Computing Surveys, Vol. 16, No. 2, pp. 187-260.



- ONE QUADTREE FOR EACH VARIABLE
- DIRECT ADDRESSING

Figure 1: Logical Structure of Location Tree Database

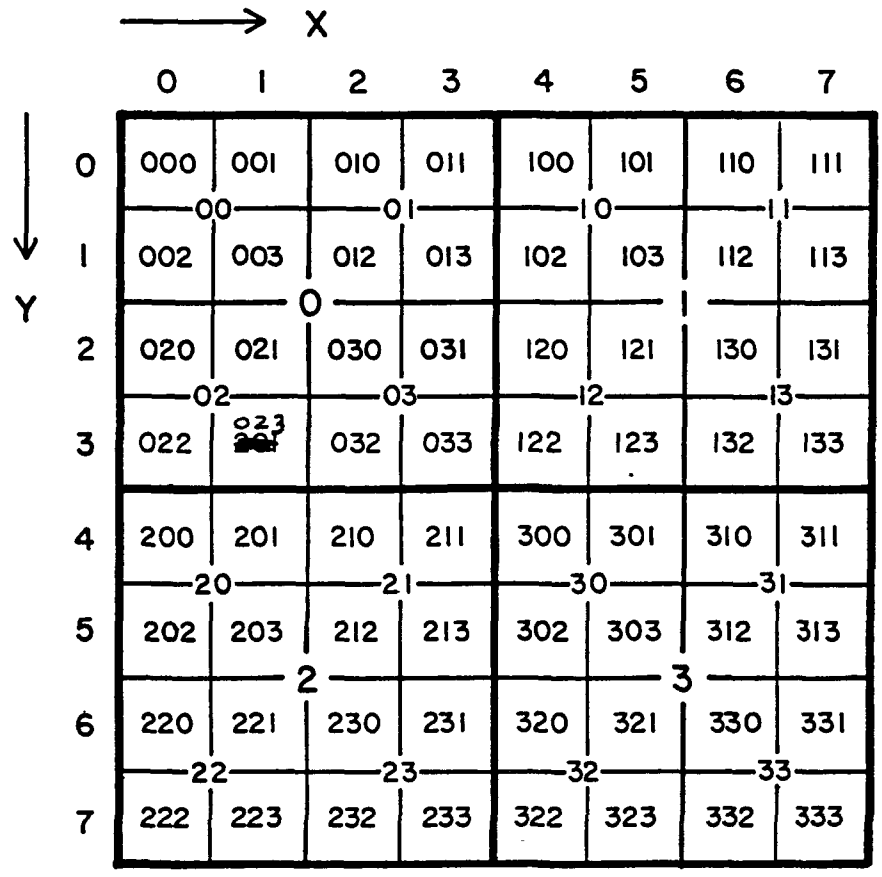


Figure 2: The Hierarchical Quadtree Drressing Scheme

LOCATION NAME :

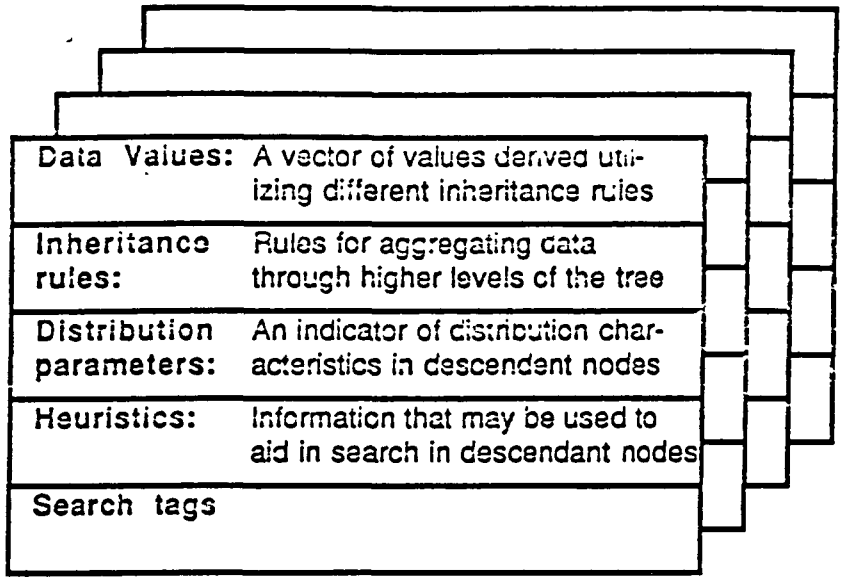


Figure 3a: Location Tree Node Structure

OBJECT NAME =

CLASS:	Name of the class of spatial objects to which this particular object belongs
DEFINES:	A list of spatial objects, each member of which is defined in terms of this particular object
DEFINEDBY:	A disjunctive normal form definition of the object in the rule language
COMPLEXITY:	A measure of the cost of search for the object
HEURISTICS:	Pointers to low level algorithms that can search for instances of the object, possibly without recourse to its definition
EXAMPLES:	Known examples of the object in the location tree database. Spatially indexed using a discrimination net

Figure 3b: Object Knowledge Base Node Structure

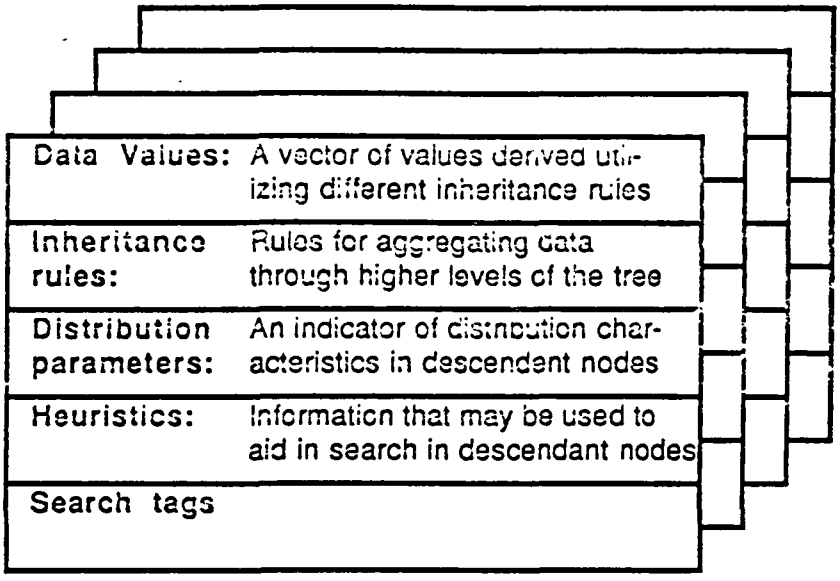
SUBOBJECT (NODE) NAME =

TYPE:	Name of the object that the node represents
SUBOBJECTS:	Names of nodes used in the hierarchical expansion of this particular node
RELATIONS:	Names of binary and ternary relations taking this node as one of their arguments
AUXOBJECTS:	Names of nodes representing auxiliary objects linked to this node through heuristic relations
AUXRELATIONS:	Names of heuristic relations
GPROPS:	Description of the unary constraints that this node must satisfy
PPROPS:	Pprop information, for primitive objects
EXAMPLELOC:	Locations of known/searched examples of the node within the current window
CURLOC:	Specific example that is part of the current solution being developed

Figure 4:

Subobject structure in semantic net

LOCATION NAME :



OBJECT NAME =

CLASS:	Name of the class of spatial objects to which this particular object belongs
DEFINES:	A list of spatial objects, each member of which is defined in terms of this particular object
DEFINEDBY:	A disjunctive normal form definition of the object in the rule language
COMPLEXITY:	A measure of the cost of search for the object
HEURISTICS:	Pointers to low level algorithms that can search for instances of the object, possibly without recourse to its definition
EXAMPLES:	Known examples of the object in the location tree database. Spatially indexed using a discrimination net

APPENDIX A

DATA STRUCTURES FOR A KNOWLEDGE-BASED GEOGRAPHIC INFORMATION SYSTEM

Donna J. Peuquet
Department of Geography
University of California at Santa Barbara
Santa Barbara, California 93106
U.S.A.

Abstract

There is a rapidly growing need to use geographic information systems (GIS) to manage extremely large databases containing data integrated from a number of imagery, cartographic and other sources for a variety of applications. Current GIS technology is, however, exhibiting severe shortcomings in meeting these performance demands.

The underlying cause of these current shortcomings of GIS is that geographic data possesses a number of special characteristics not shared with other types of two- or three-dimensional data which must be taken into account: First, natural geographic boundaries tend to be very convoluted and irregular. They subsequently do not lend themselves to compact definition or mathematical prediction. Geographic databases consequently tend to become extremely large. Second, the data in digital form tend to be incomplete, imprecise and error-prone due to both the context dependencies of the definitions of many geographic entities and relationships, and the characteristics of the data gathering process. Third, spatial relationships tend to be fuzzy or application-specific, and the number of possible spatial interrelationships is very large.

Spatial data structures or data models which take all of these characteristics into account currently do not exist. In the present paper, a database design is described as the basis of a prototype system currently under development which Artificial Intelligence techniques with recent developments in database and spatial data processing techniques to overcome these problems.

The architecture of this database, its advantages in utilizing AI techniques in a spatial context and its functionality within a running knowledge-based geographic information system are discussed.

Introduction

Problems with Current Geographic Information Systems

Many problems of geographic analysis require the storage, retrieval, and manipulation of databases that:

(1) contain very large amounts of geographic data;

(2) contain many diverse data types, such as real-valued functions (e.g., digitized elevation data, rainfall data), vector-valued functions (e.g., multispectral data), categorical-valued functions (e.g., land use data) and functions taking general symbolic values (e.g., place-name data);

(3) contain many layers of data of different types, interrelated by complex (implicit) relationships;

(4) are capable of solving a large array of different problems, some of which may not have been anticipated by the system designers;

(5) provide quick interactive response.

There has been recent recognition of the need to construct geographic information systems (GIS) that possess all of these capabilities. For example, a growing number of federal-level government agencies world-wide are currently attempting to build very large, integrated spatial databases for incorporation into geographic information systems which will, in turn, become the basic analytical and information tool within their respective organizations. The U.S. Geological Survey is envisioning a cartographic database containing all information from 55,000 7-1/2 minute map sheets covering the entire United States. A conservative estimate of the total digital data volume generated by these maps when digitized at cartographic precision is 10^{15} bits for a single data layer (e.g., topography or hydrology). NASA is currently planning to develop an integrated database system incorporating all Landsat and other spacecraft data for the earth, as well as outer space.

Current GIS have consistently exhibited severe problems with response times and storage volumes for data sets of any significant size, as well as with rigidity and narrowness in their range of applications. Existing GIS technology requires operation on a predefined data set with "built-in" objects and relationships. The integration of different types of geographic data, such as imagery and digital map data, is possible only under very special conditions.

These problems are related to three special characteristics of geographic data which are not shared with other types of two-

or three-dimensional data. First, geographic "objects" tend to be poorly defined and their boundaries tend to be convoluted and irregular. Hence, they do not lend themselves to compact definition and quickly become extremely large. Second, the data in digital form tend to be incomplete, imprecise and error-prone due to high variability in characteristics and overall quality of original archival documents and the characteristics of the digital capture process. Third, spatial relationships between objects tend to be imprecise or application-specific, and the number of possible spatial interrelationships is very large.

Potential Solutions

The current performance demands on GIS technology thus requires significant advances in three areas; to develop data models which represent geographic phenomena more efficiently and completely, develop procedures for searching complex geographic databases in an extremely flexible and efficient manner, and to develop methods for dealing with imprecision.

It is the aim of the research described in this paper to construct a prototype GIS which simultaneously addresses all of these areas by employing:

(1) a highly flexible, robust and efficient method for data representation and retrieval;

(2) heuristic search procedures;

(3) the capacity to learn.

Recent research in the area of spatial data models has revealed the significant potential of an entire class of data models new to the application of automated geographic data processing but which are based on well known geometric principles. These are recursive tessellation models which offer a much greater degree of flexibility and efficiency than grid or vector-type models currently used in GIS.

Heuristic search procedures involve the use of shortcuts to greatly increase data retrieval speed by eliminating large portions of the database from consideration at an early stage of the search process. Heuristics utilize built-in data about the data--i.e., knowledge about how the various data elements are distributed within the database.

The capacity to learn is the most powerful means of dealing with imprecision and uncertainty. This allows a system to provide answers in terms of "confidence margins" and to adapt to new kinds of queries and applications.

In order to deal with fuzzy entity definitions and fuzzy relationships, knowledge needs to be built into the database

system. This would obviate the current need to insert artificial precision into the data, and would allow a more natural data representation. This knowledge-base can also be used as a self-checking mechanism to detect and potentially correct data errors. Such a system would be able to answer user queries concerning the relationships of complex objects in a large landscape where the specific query or application cannot be anticipated by (i.e., built into) the system. This flexibility would not only greatly extend the types of data and range of application a single system could handle, it would extend the useful life of a system for a given application as needs grow and change over time. The system would also use its knowledge-base to effectively narrow down the problem's search space with respect to: (1) data retrieved from the large geographic database, and (2) activation of GIS operations on the data retrieved.

Besides overcoming a number of existing major problems in GIS, the application of the results of recent AI research on learning may have a profound impact on the field of geography as a whole, since learning procedures could be built into a GIS for the purpose of adapting to the imprecise and voluminous nature of geographic data as the system acquires knowledge about the phenomenon. This means that such a system can be used to acquire new knowledge concerning the nature of geographic phenomena and their interrelationships.

The prototype described here is designed to solve problems which are generally considered to require expertise, and which at the same time may require rapid search and manipulation on an extremely large and heterogeneous data space. Such systems are called "knowledge-based" because they are based on the use of stored facts and logical rules, including heuristics. The current prototype system under development at UCSB has been named the Knowledge Based Geographic Information System (KBGIS).

Knowledge-based systems to date are normally limited to either solving complex problems or to handling very large data volumes. However, the current research expects to show that database and spatial data processing techniques, though developed by independently evolving technologies, can be combined with AI techniques and applied in a complementary manner to enhance the capabilities of the individual techniques for application in GIS. Similarly, it is expected that this combination would yield a system with data volume capacities, levels of computational efficiency, and range of applications well beyond those which can be provided using traditional GIS approaches. The KBGIS is designed to demonstrate the feasibility and potential power of this new approach.

One major problem in doing this is that the methods of data representation in AI, though superficially similar to hierarchical representations in spatial data handling, differ significantly in their methods of implementation and manner of use. Since the data storage and retrieval methods employed are

the basis upon which the rest of any system is built, this was the first problem to be solved in building a knowledge-based geographic information system.

A brief overview of the system will be given in the next section. The remainder of this paper will present a detailed description of the database architecture and its functionality within a knowledge-based geographic information system.

Overview of the Knowledge-Based Geographic Information System

KBGIS is currently restricted to answering queries in a given domain, centering around the use of remote sensed data for natural resource problems. The current functional database includes topography, rainfall, land use and vegetation for a small test area.

Figure 1 shows the major components of KBGIS and how they are interrelated. The dominant feature of this design is the dualism of functions relating to; 1) spatial objects, and to 2) the distribution of those objects in space. These functions are logically distinct in the system design, with each group of functions and storage methods specifically tailored for maximum flexibility and efficiency in handling the two primary types of operation: one is to answer queries concerning objects at a specified location, the other is to answer queries concerning the location of specified objects.

Because of this duality, many equivalent procedures are implemented in both portions of the system, and much data are stored on both forms. Nevertheless, a basic assumption underlying this design is that much of the power and efficiency of the system is derived from this.

Under actual use conditions, it is expected that queries may often have both object-based and spatially-based components. The two portions of the system are therefore interrelated and designed to work together in answering individual queries.

Knowledge is built into the system in the form of rules and the procedures for using these rules. These rule-based procedures occur at several levels of the system for performing various specific functions, as will be seen below.

System Architecture

As shown near the top of Figure 1, the query parser acts as the translator between the user and the remainder of the system and functions in the same way as a query parser in any modern information system. The high-level user query language is

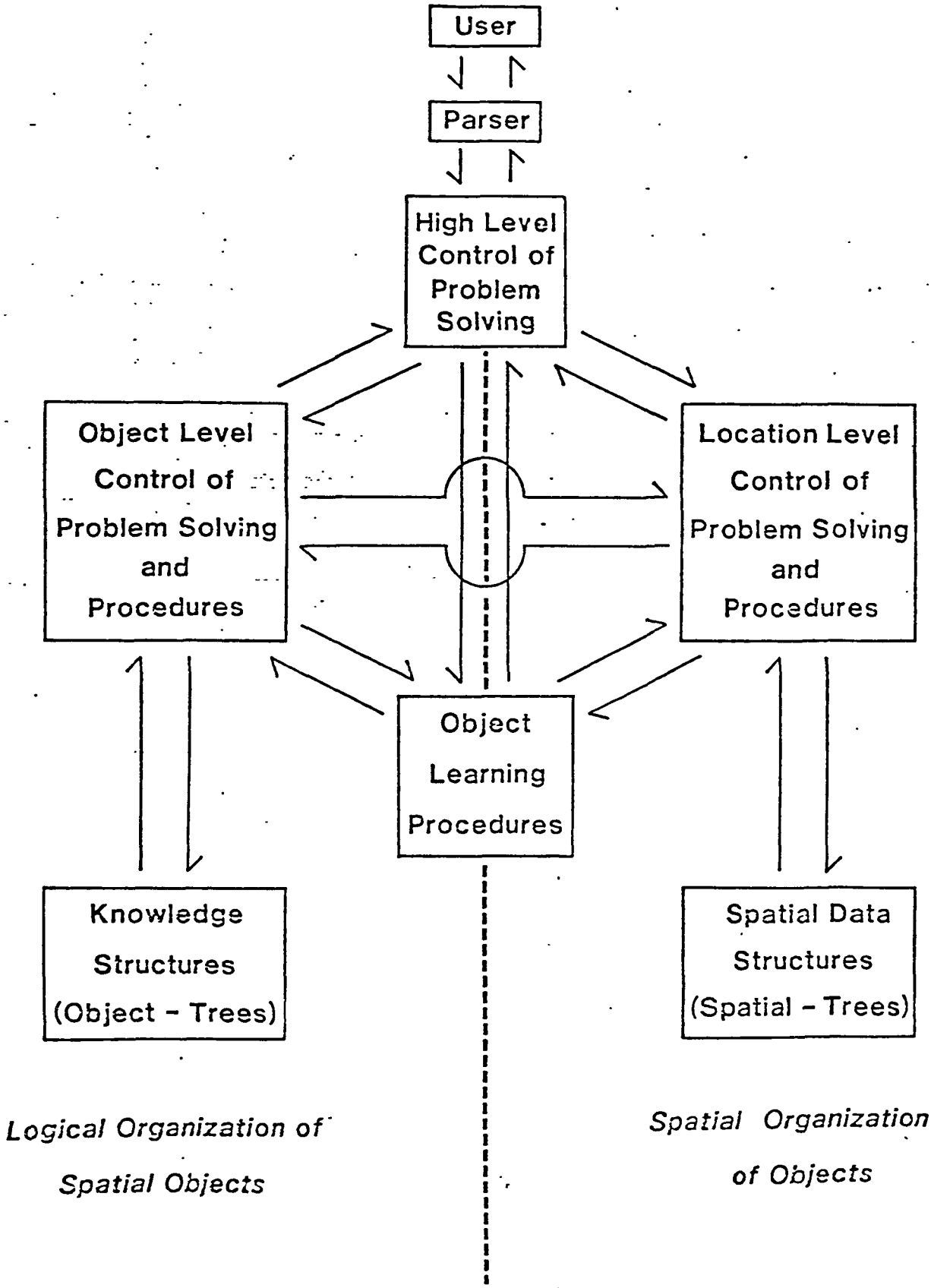


Figure 1. the KBGIS general architecture

translated into sequences of lower-level system commands.

The next level in the system is the high-level query control. This portion of the system contains knowledge about what capabilities are available within the rest of the system, what combinations of lower-level tasks are needed to perform higher-level complex tasks, and maintains statistics on how much time is required to execute these tasks. This knowledge is used by the high-level control to determine precisely what path through the system is to be used in answering each user query, such as, whether the object or spatial portion of the database must be used in answering a particular portion of the query. This knowledge is also used for automatic performance tuning by selection of the shortest-time path based on performance statistics from previous queries. This tuning mechanism may also detect system bottlenecks which can only be corrected by manual intervention to modify or add algorithms.

The next level of the system consists of two complementary components, one for the object and the other for the spatial portion of the system. These each contain object- and spatially-oriented problem solving rules,respectively. These rules are gradually accumulated and modified by the learning portion of the system. They are used by the system for various types of tasks, such as database search and retrieval.

These two components of the system also contain the libraries of low-level algorithms which are combined in various ways determined by the higher level control to answer queries. These low-level algorithms operate directly on the stored data.

The learning procedures by which the system is capable of acquiring new knowledge concerning the properties of data entities and the relationships between entities are contained in a separate portion of the system which operationally transcends the object/spatial distinction.

Learning can be accomplished either inductively or deductively. Inductive learning is user driven and is accomplished by the user supplying examples of new characteristics or relationships. Deductive learning is accomplished by the storage and accumulation of information gained from previous queries.

The KBGIS Database

Overview

The KBGIS is based upon a pair of hierarchical data models, one spatially-oriented and the other object oriented. The data are organized spatially in the form of quadtrees. Quadtrees have been discussed extensively by Samet and others (Samet, 1983; Klinger, 1971; Klinger and Dyer, 1976; Finkel and Bentley, 1974).

In the area quadtree model which is used in KBGIS, the geographic area is recursively divided into four quadrants, as shown in Figure 2. This subdivision of space translates functionally into a regular, balanced tree of degree 4

Each separate type of data, or coverage, is arranged in a separate quadtree, and all quadtrees are spatially registered, as shown in Figure 3. This arrangement has become known as a "forest" of quadtrees, and allows efficient search through multiple coverages for all data pertaining to a given location.

The regular nature of this data model allows a simple indexing scheme to be used for direct addressing to increase retrieval efficiency. This indexing scheme will be described in greater detail later in this paper.

The data are also organized by object in a tree structure. The particular data model used here is known as an and-or tree, a standard device used in artificial intelligence applications for problem solving by subdividing a problem into a series of smaller sub-problems. In the context of the KBGIS, the and-or tree, is used to record, in effect, a taxonomy of data objects. At each node in the and-or tree the names and properties of classes of data objects are stored. There are two kinds of links. The father-son links without a cross-link bar, as shown graphically in Figure 4, is an "or" relation. This indicates, for example, that either mature or immature orchard can be classes in a more general sense as "orchard." The father-son links shown graphically with a cross-link bar indicates an "and" relation. For example, "eroding orchard" in Figure 4 is composed of the combined set of "orchard" and "slope".

The objects in any individual Object Tree are functionally data values. These values may be nominal in nature; "county", "city", "orchard", etc. They may also represent elevation range or other data value classes which we would not normally associate with named geographic objects.

Each class of data (hydrology, topography, etc.) is not as distinctly organized into separate hierarchies as is the case with the quadtree portion of the database. Since there are often logical interrelationships between given classes of data, the Object Trees tend to be interlinked.

At the leaves of the Object Trees are the primative objects which cannot be further subdivided. With these objects are stored the spatial definitions of each individual occurrence within the database. This spatial definition is in the form of a list of quadtree addresses.

Because of the more irregular nature of the Object Tree, this portion of the database is implemented using pointers.

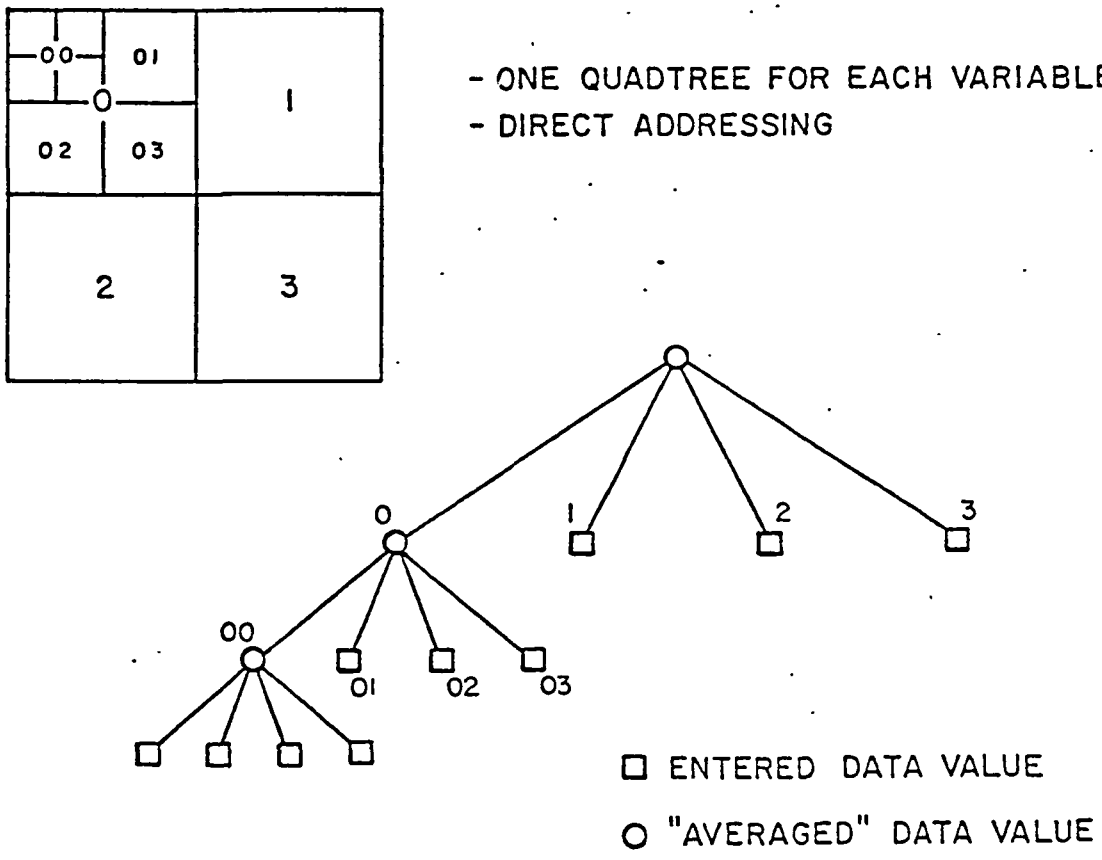


Figure 2. General Spatial Tree structure

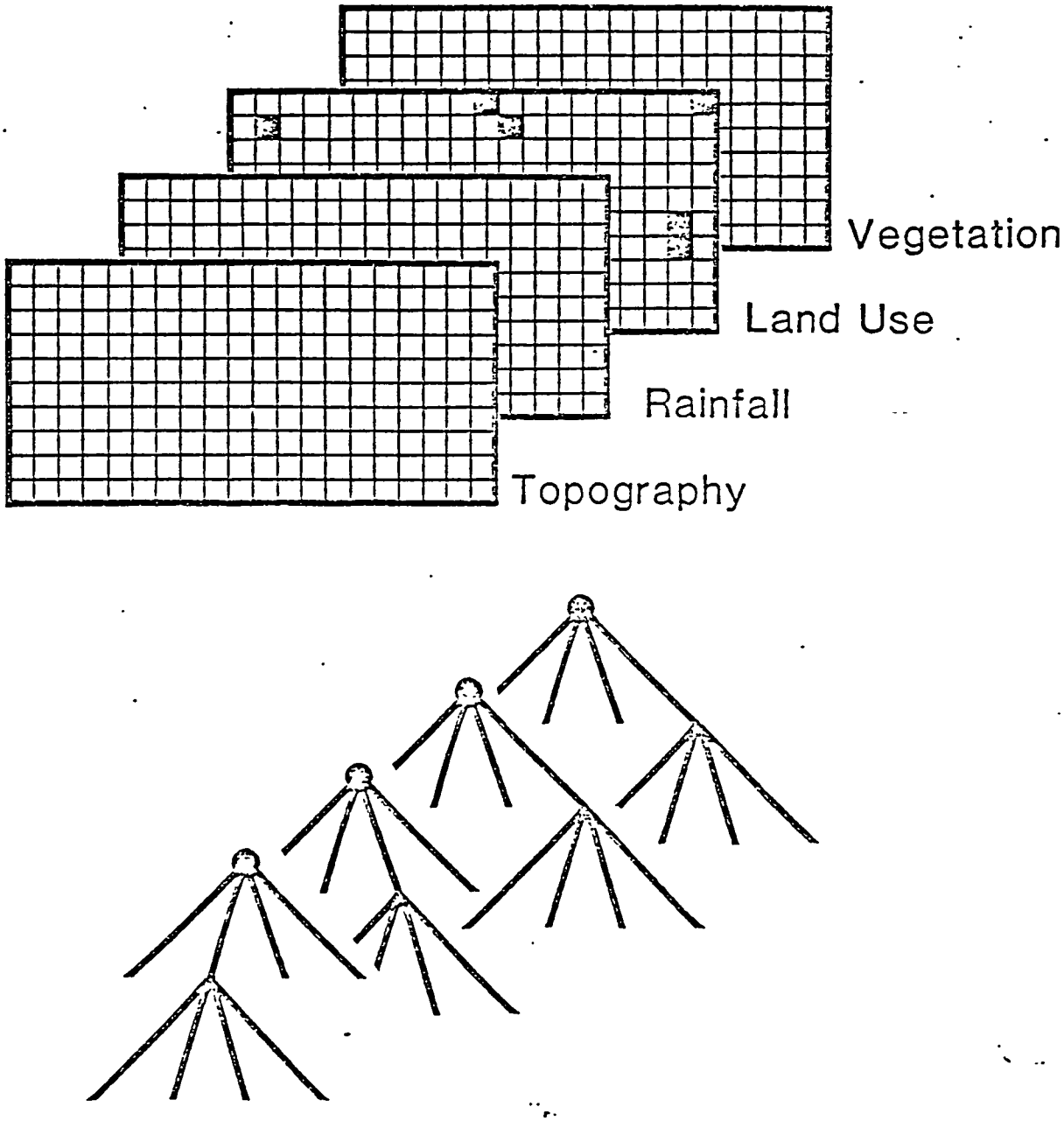


Figure 3. Spatially registered areal data coverages when transformed to quadtree format result in a forest of quadtrees

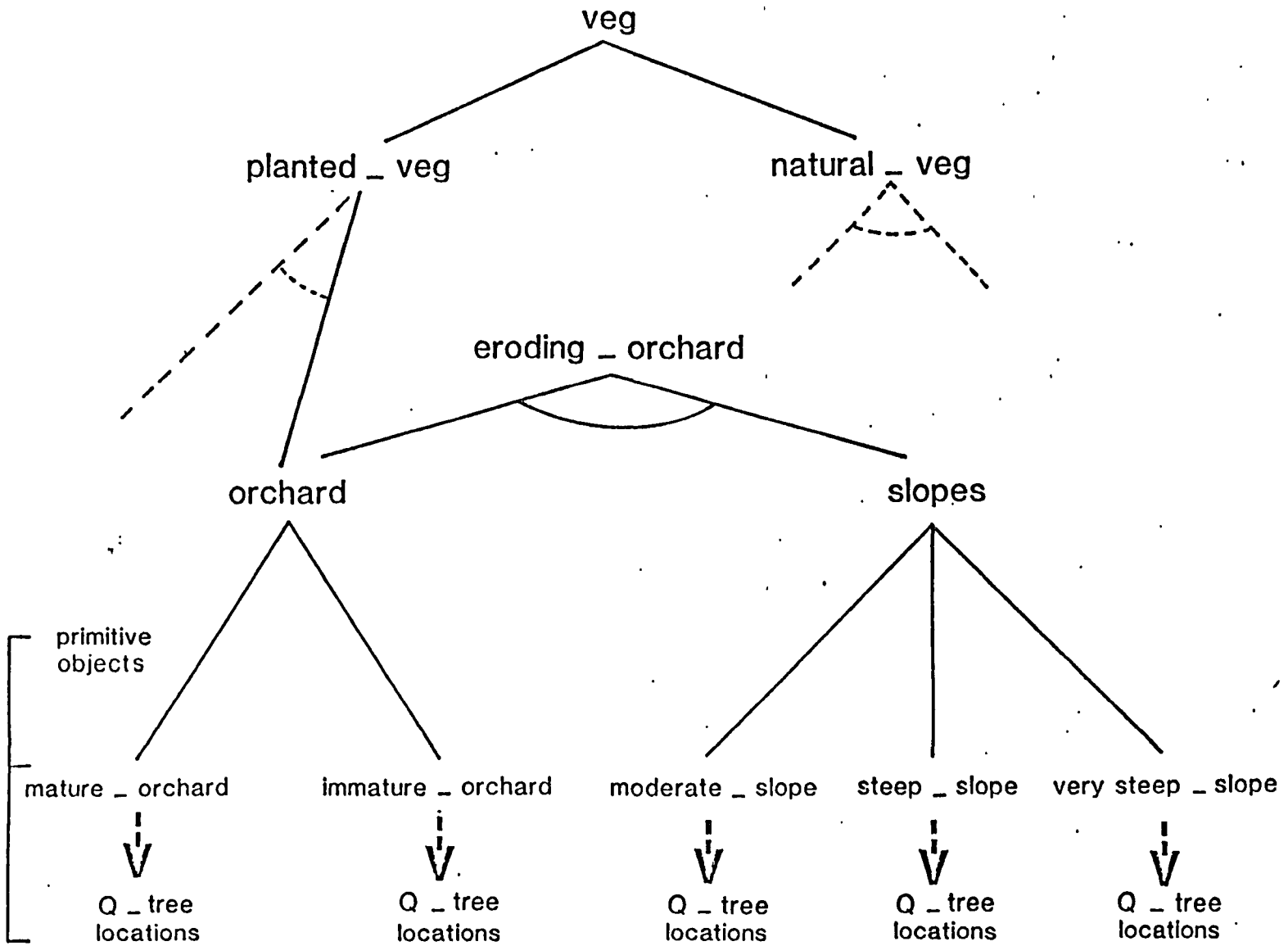


Figure 4. A sample Object Tree

Spatial Trees

The quadtree data model was chosen for use in the Knowledge Based GIS for a number of reasons:

First, Quadtrees are based on the square grid model and retain all of the inherent advantages of a regular tessellation. The foremost of these is its extreme flexibility in being able to portray the distribution of almost any type of phenomenon over space. This flexibility in part is derived from its simplicity and in part from the characteristic that all spatial interrelationships are implicitly and inherently present within the data model. The square grid is also compatible with a wide range of existing spatial data processing techniques, particularly those derived for the field of image processing.

Second, tree storage and search techniques is one of the more thoroughly researched and better understood topics in computer science. This includes techniques for implementation as a file structure, such as hashing, which would eliminate the need to store null cells while allowing rapid retrieval via direct addressing. In other words, familiarity with regular tree handling means that the implementation techniques are known and performance characteristics are predictable.

Third, in cartographic terms this is a variable scale scheme based on powers of 2. This means that scale changes between these built-in scales merely require retrieving stored data at a lower or higher level in the tree. Stored data a multiple scales also can be used to get around problems of automated map generalization. The obvious cost of these features, however, is increased storage volume.

A fourth factor is efficiency. At first glance, it would seem that the quadtree data model would be unsuitable for large-area spatial databases because of combinatorial explosion. However, it must be remembered that although the number of cells increases four times at each successive level in the hierarchy, the area of cells at each successive level decreases by one-fourth. This means that the individual cell size quickly becomes very small. Thus, 9 levels are needed to store 40 meter resolution pixel data at the lowest level of the quadtree for a geographic area equivalent to one 7 1/2 minute quadrangle map. This results in 57,334 leaf nodes and approximately 87,381 nodes in the whole tree. To store data for the entire surface of the earth at 40 meter resolution, only 21 quadtree levels are required. A full quadtree of this depth contains approximately 10^{12} nodes.

Fifth, the recursive subdivision facilitates physically distributed storage, and greatly facilitates browsing operations. This is a major advantage for very large databases. Windowing, particularly if designed to coincide with areas represented by quadtree cells, is also very efficient.

All of the above advantages hold for application of the quadtree data model in any large GIS. Nevertheless, the most significant advantage of quadtrees for a knowledge-based GIS is that their inherent hierarchical structure is compatible with standard AI problem solving techniques, which are in large part based on tree traversal.

Spatial Addressing

Because KBGIS is specifically designed to efficiently handle very large volumes of spatial data, the usual method of implementing a quadtree structure through the use of pointers is too inefficient. This is true both in terms of the extra space required to store the pointers and the time required to sequentially trace through a long sequence of pointers in order to retrieve any single data element.

KBGIS employs a direct addressing scheme which is derived from a locational code initially described by Morton [1966]. Although this code was based on a grid, it can be expressed in quadtree form as was noted by Torpf and Hertzog [1981].

The basis of the numeric locational key used in KBGIS is a recursive numbering scheme for four quadrants, as seen in Figure 5. Each successive level in the spatial decomposition requires one additional digit to indicate level. This scheme, at any given level, is the Morton matrix scheme. The only difference is that base-4 notation is used to give a better representation of the quadtree model. This results in physical storage of the quadtree nodes in breadth-first order, i.e., each complete level is stored in sequence, starting from the top level. The finest resolution layer, or leaves of the quadtree, are thus stored as the last level in each coverage file. This order maintains the level-specific nature of the addressing scheme.

This scheme has important characteristics which facilitate spatial retrieval (Mark and Lauzon, 1983). The numeric ordering of the quadtree cells generates the trace of an N-shaped Peano curve, as shown in Figure 6 (White, 1982). These Peano curves, also know as space-filling curves, track through space in such a way that N-dimensional space transformed into a line and vice-versa.

The three properties of Peano curves which are useful for spatial database applications are:

(1) The unbroken curve passes once through every locational element in the data space.

(2) Some of the spatial associativity of the scanned data space and the single dimension found by the scan are preserved. Most particularly, points close to each other on the curve tend to be close to each other in space, and vice versa.

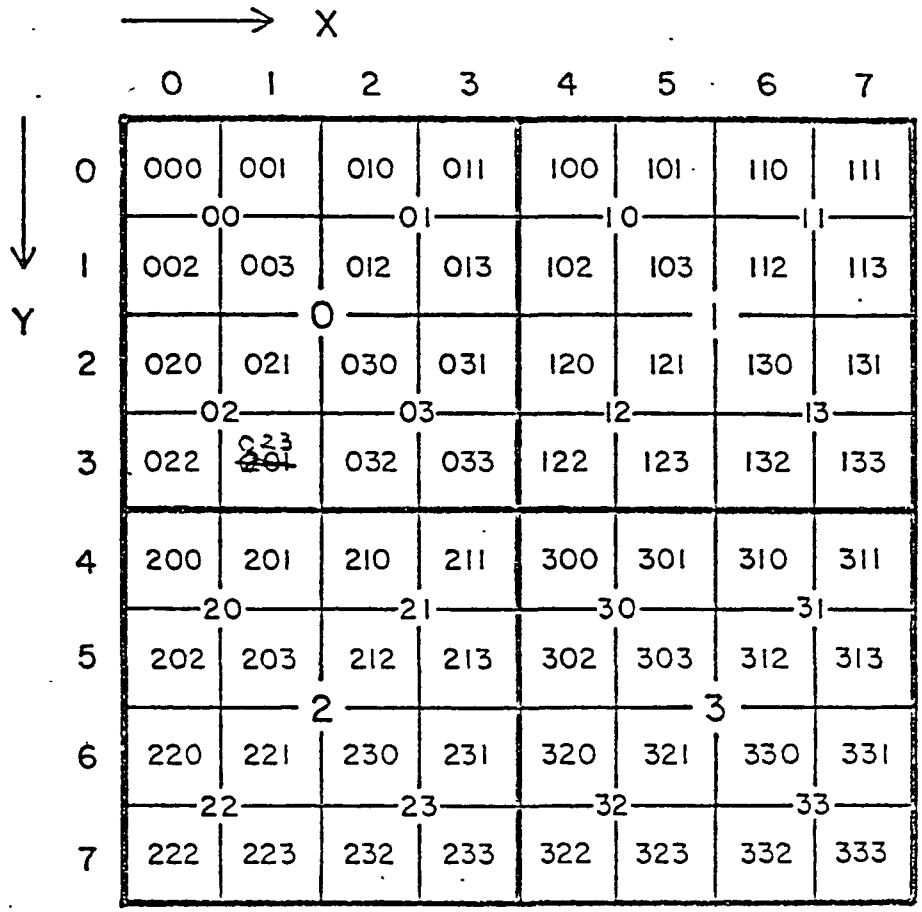


Figure 5a. The base-4 quadtree addressing scheme

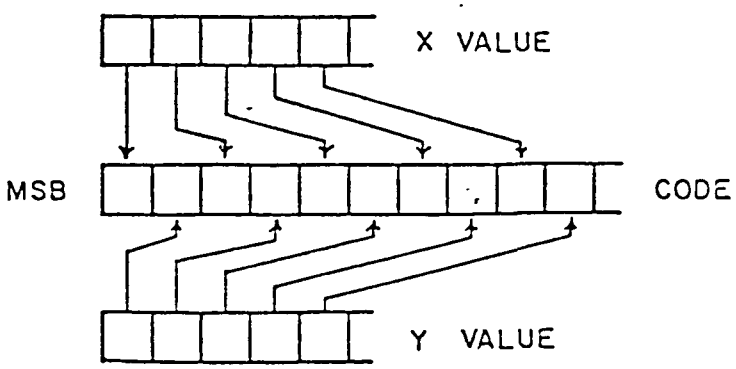


Figure 5b. Bitwise interlacing of x and y coordinates to form the address

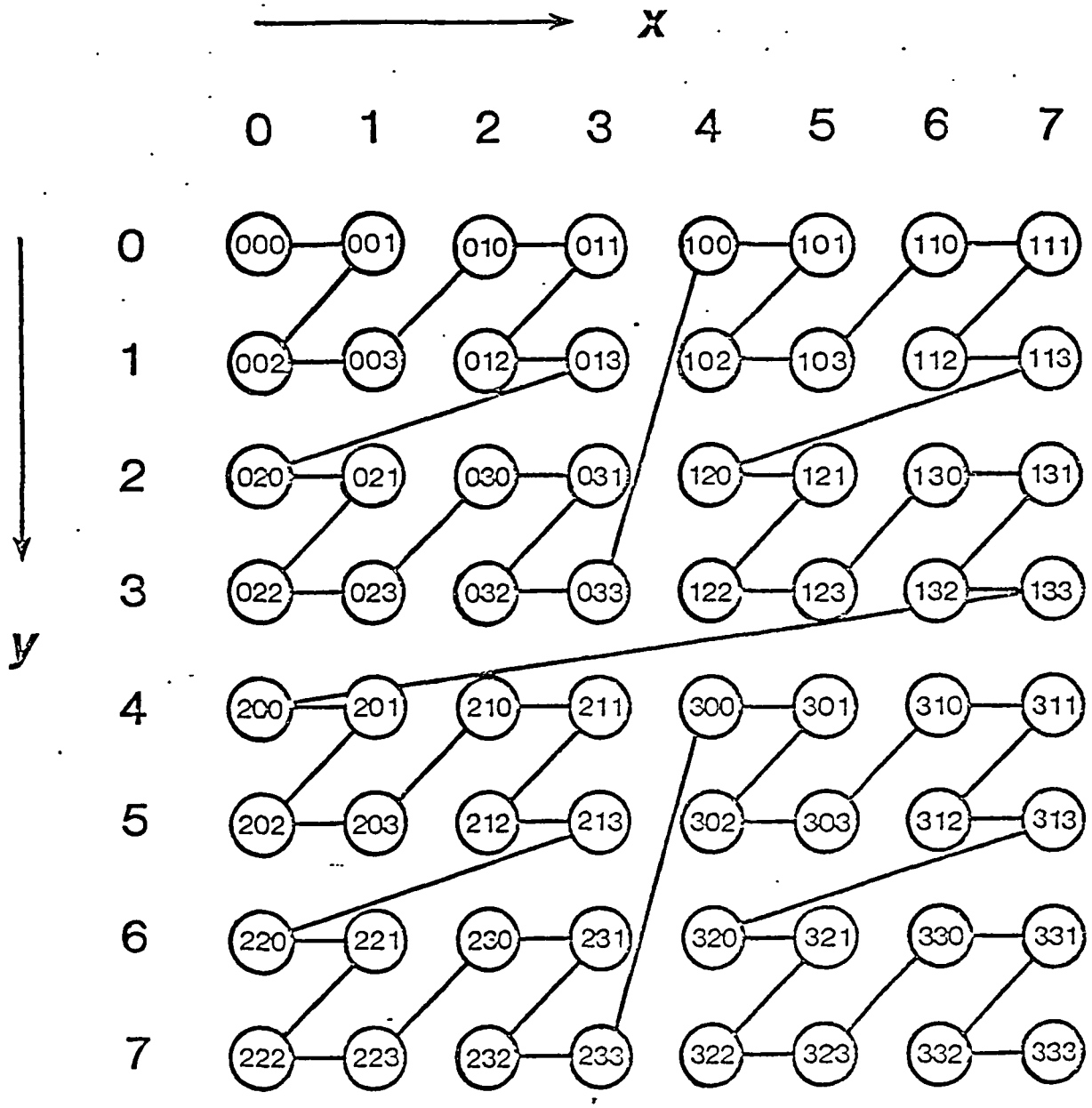


Figure 6. The relationship between the quadtree addressing sequence and and N-shaped Peano scan

(3) The curve acts as a transform to and from itself and N-dimensional space.

Obviously, the first property is essential in database applications to provide a unique search key.

The second property provides efficiency in retrieval for areally coherent data in consecutive or near by locations in storage. This is an advantage with geographic data since many phenomenon such as land use, and elevation tend to be spatially clustered.

As a result of the third property, the address code represents a bitwise interlacing of the x and y cartesian coordinates at each level of resolution, as shown in Figure 5b. The quadtree address can thus be quickly calculated from any x-y coordinate by alternating bits from the binary representations of each coordinate. Conversely, the cartesian coordinate for any given quadtree address is easily calculated by performing the reverse process (Torpf and Herzog, 1981).

The properties of Peano curves have also been shown to have significant utility for a number of procedures derived from remote sensing applications which are used within KBGIS. These applications include histogram equalization adaptive thresholding and spatial spectrum techniques. The reason for this is that these and many other techniques for manipulating and analyzing image data are sequential and single channel operations, i.e., they are linear in nature. The property of preserving some of the spatial relationships within the Peano scan sequence allows for these procedures (Stevens, Lehan and Preston, 1983).

The most important consideration of Peano curves is that these known properties, and others which can be derived from them in this particular context, can be used to form the primary search heuristics used by the locational problem-solving portion of the system. These search heuristics are in the form of a series of rules which act as the basis of a dynamic decision-making process on how to most quickly retrieve the required quadtree nodes when specific addresses are not specified. This type of search is needed to answer an object-based query, such as "find the locations of all irrigated orchards", when this object has not yet been entered into the Object Trees. As such, the query is interpreted as a series of search constraints. (As a result of the query, "irrigated orchards" may also be entered into the Object Tree for use with future queries.)

The spatial data structure used in KBGIS is efficient for a number of other essential frequently used procedures in a GIS context. Some of these procedures are discussed below.

Data Retrieval Operations

Many spatial retrieval operations are performed via address

calculations. A tree traversal search which may require many access to the database to retrieve individual data items is thus avoided. The address calculations are performed using bitwise-interlaced arithmetic. For example, retrieval of the last neighbor cell of cell address 031 requires the following simple addition:

031 + 1 = 120

,or in binary notation;

```
 001101
+     1
--------
 011100
```

Retrieval of the North-West neighbor of the same cell requires a subtraction:

031 - 3 = 012

,or in binary notation;

```
 001101
-    11
--------
 000110
```

Geometric calculations such as point-to-point distance are performed in the standard manner, but using bitwise-interlaced arithmetic:
For example, cell distance between location 032 and 312:

312 - 032 =

```
110110
-001110
--------
 1 0 0      = 4_10 = ΔX
   1 0      = 2_10 = ΔY
```

$(100_2 - 10_2)^2 = 100_2 = 4_{10}$

The primary difference to be remembered here is that cell size is level-specific. All distance and area measurements must therefore be adjusted to a constant scale.

Rectangle retrieval problems, including range searching and containment are reduced to binary search problems involving binary code range shrinking. The range search problem has been described in detail by Torpf and Herzog [1981]. This involves recursively testing if a candidate address is within the

specified window. If not, the addresses for the descendant subtree are not tested. Thus, whole quadrants can be eliminated one at a time without any database access taking place. This address test is implemented as a bitwise comparison of the candidate quadtree code.

Empirical experiments performed by Torpf and Herzog indicate that search time complexity for range search is logarithmic with the number of records using this scheme. More specifically, the expected time for searching a single quadtree is $O(2(\log N + F))$, where N is the number of nodes in the tree, and F is the number of nodes found within the specified range.

Search Heuristics

Although point and rectangle retrieval retrieval are seen to be efficient and straightforward utilizing a bitwise interlaced scheme for direct addressing, it is often not suitable in the retrieval of geographic objects, which tend not to occur naturally in compact shapes. Geographic entities, instead, tend to have convoluted and highly irregular shapes. They may also have holes or be discontinuous, with isolated portions scattered throughout the data space. This is particularly true within the context of the KBGIS where an "object" is really a data value or range of values, such as "orchard" or "20-30 feet elevation."

The rules used by search heuristics are tailored to minimize search time for these types of occurrences. Spatial search heuristics are particularly important in the Spatial Trees because this is the larger portion of the database. Although Object Trees are initially manually entered into the system, these are rudimentary in nature. All other data are entered into the system as quadtrees. As the system is used it acquires knowledge about spatial objects and builds the Object Trees autonomously. Thus, in the system's early state, it must rely heavily on retrieving information from the Spatial Tree portion of the database. This means that queries such as "find the locations of all eroded orchards" can often not be aided by referencing the Object Tree portion of the database. Once an object-based query such as this is answered utilizing the spatially-based portion of the database, this information is stored in the Object Tree for use in future queries. In this way, the data are propogated in the Object Tree and object-based queries gradually become more efficient.

The objective of the search heuristics is to search very quickly through a very large database by using knowledge about how the data are spatially distributed in order to eliminate large portions of the database for consideration at an early stage of the search. In order to aid this process on the Spatial Trees, spatial statistics derived from the spatial spectra are used as additional inputs to the decision-making process. These spatial statistics are computed and stored at the nodes of the

Spatial Tree.

Spatial spectrum techniques were developed within the field of image processing and provide a compact method for describing spatial distributions. Thus, spatial spectra can be used to guide search procedures by indicating at a given quadtree level (i.e. scale) whether the distribution in the descendant nodes is clustered, scattered, and how large or small the individual objects tend to be.

Development of these heuristic search techniques are still in the formative stages, but it is expected that this feature will allow spatial search timing performance which are substantially better than methods utilizing conventional methods.

Summary

The integration of techniques from spatial database, Artificial intelligence and image processing show significant promise in overcoming simultaneously several of the major obstacles preventing geographic information systems from handling large, heterogeneous spatial databases in an efficient and flexible manner.

The Knowledge Based Geographic Information System is demonstrating that techniques from these independently developed fields can be integrated in a natural manner, and that the capabilities of one can be used to substantially enhance another.

The duality of the database design of KBGIS is in itself a significant step in overcoming some seemingly inherent shortcomings in current GIS.

Most existing GIS are based either on vector- or raster-type data storage structures. From a modeling perspective, the Object Trees and Spatial Trees used in KBGIS are a vector-type data model and a raster-type data model, respectively. The basic logical component of a vector model is a spatial entity or object. The spatial organization of these objects are explicitly stored as attributes of these objects. Conversely, the basic logical component of a raster-type model is a location in space. The existence of a given object at that location is explicitly stored as a locational attribute.

The use of both simultaneously, with carefully designed and controlled redundancy, would in itself greatly enhance the potential range of applications and flexibility in representing data and speed efficiency of any GIS.

APPENDIX B

THE DERIVATION AND APPLICATION OF SPATIAL SPECTRA DERIVED FROM QUADTREE-FORMATTED IMAGES

Zi-Tan Chen

Donna J. Peuquet

Geography Department

University of California

Santa Barbara, CA 93106

Abstract

An efficient technique is presented to generate and represent spatial distribution information for large volumes of spatial data. Quadtree spatial spectra (QTSS) are introduced and derived based on the quadtree data structure. In comparison with conventional transforms such as the Fast Fourier Transform and the Fast Walsh Transform, it is found that QTSS are much more efficient to calculate but there is also some information loss. QTSS generation and its uses in high-, low-pass, and band-pass filtering are also developed.

Introduction

Radiation spectra, consisting of tables or histograms of reflectance components distributed at different electromagnetic wavelengths within a digital image have been widely used in the fields of physics, chemistry and environmental remote sensing to identify objects. For example, vegetation, water and dry bare soil can be easily identified via their distinctive characteristic spectral signatures [10]. This is graphically shown in Figure 1. Here it can be seen that the radiation spectrum of vegetation has a high response at wavelength 1μm, while soil has a high response at wavelength 0.5μm.

Similarly, spatial spectra consisting of tables or histograms of spatial components distributed at different spatial wavelengths can be used to distinguish individual objects and to describe the spatial arrangement of those objects. In the example shown in Figure 2, a rectangular object and its 2-Dimensional Fourier transform are presented.

Spatial spectra have been widely used in digital image processing for filtering operations, such as smoothing and edge enhancement [8]. The basic transform function most commonly used for generating spatial spectra is the Fourier Transform. The Walsh Transform and the Hadmad transform are also used in some cases. These spatial transforms are well developed. There is no information loss during either forward or inverse transform procedures, which make them useful for image reconstruction in image processing applications. Although The Fast Fourier Transform (FFT) increases the speed of calculating the Fourier transform significantly, it is still too slow for frequent operations on

very large images and its 2-D spatial spectra requires almost as much storage space as the original image.

The purpose of the present paper is to describe a method to extract and represent information concerning the spatial distribution of objects within a digital image which is efficient enough for practical use in a very large data base context. To this end, we assume that such a method must have the following properties:

(1) Generating the new spatial spectra from a large volume of 2-D data must be significantly faster than the FFT.

(2) It must have a highly compact representation.

(3) It must retain a significant degree of spatial information and allow spatial filtering functions similar to other conventional transforms.

(4) A variety of functions based on interpreting the new spatial spectra must be simple and computationally efficient.

It is assumed that some information loss can be incurred, resulting in "approximations" for the sake of computational speed. In an effort to satisfy these requirements, we introduce the concept of Quadtree Spatial Spectra (QTSS). This is a quickly computed spatial spectrum statistic based on the quadtree data structure. In the following discussion, we will show that although it is not complete information preserving, as in the Fourier and Walsh transforms, it still retains a great deal of useful information about the original image.

A review of the Quadtree structure

The quadtree data structure has recently received considerable attention in the literature. It has significant advantages for geographic data storage and processing, and some of these are particularly advantageous for large volumes of data. The quadtree structure and associated algorithms have been discussed extensively by Samet and others [21] [9].

Samet provides an extensive review of this subject. In the Quadtree data structure, the geographic area is divided recursively into four quadrants. This subdivision of space translates functionally into a regular, balanced tree of degree 4. Only those quadrants which contain data need be stored. This simple form of quadtree data compaction is known as a linear quadtree [11]. As shown graphically in Fig.3, each square is a quadtree "node". A node with a larger square is at a higher hierarchical level of the quadtree, and vise versa. A sizeable repertoire of algorithms based on the quadtree structure comparable to those used in geographical data processing based on other data models have been developed. These algorithms include computation for geometric properties such as region area and centroid, set operations for the comparison of images [22]. connected component labeling and neighbor detection [18], distance transforms [19], image segmentation and data smoothing [16] and edge enhancement [15]. Algorithms for the conversion of data from raster storage to quadtree format have been developed and refined [5] and the storage efficiency examined [6] and enhanced by using linear quadtrees [7]. Quadtree-based geographic information systems have also been proposed [20] and such structures

have been presented [14] [12]. The possibility of using artificial intelligence to improve a very large quadtree-based GIS has also been considered [3] [4] [25].

Description of the Quadtree Spatial Spectra(QTSS)

Quad trees as a hierarchical data structure allow the storage of spatial distribution information which varies between levels of the hierarchy on a variable scale scheme based on powers of 2. Spatial information at each of these scales can be regularly organized as a quad tree spatial spectrum (QTSS).

We first assume that a raster image has been converted to its quad tree representation. A $2^N * 2^N$ binary raster image can be represented by its quad tree by successively subdividing it into four quadrants until all pixels within a quadrant are homogeneous [9]. Figure 4a shows a simple binary image. Figure 4b is its corresponding quadtree, shown graphically and figure 4c shows the same structure represented as a tree. Large cells, which are at higher levels in the hierarchy correspond to long wavelength components in the spatial spectrum.

For construction of a quadtree spatial spectra, we assume is that a "black" (i.e., occupied) quadtree node adds a component with a spatial wavelength equal to the edge length of this node. The additive value of this component is proportional to the size of the node. Thus, this spectra has power-of-two intervals on the wavelength axis, corresponding to the size increments of the quadtree nodes through levels of the hierarchy. The component strength at a wavelength, which

equals the edge-length of a quadtree node at level i, is proportional to the number of pixels (i.e. lowest level cells) within all those black nodes at level i. A table or a histogram containing a set of these wavelength components is defined as a Quadtree Spatial Spectra (QTSS) of the image.

In the quadtree spatial spectra shown in Figure 4d, note that there are two curves. The broken line represents the sum of nodes at each wavelength interval. The solid line is the quadtree spatial spectra curve. We first explain the broken curve. If an image can be represented by a quadtree with L levels, Its QTSS has N elements, and it can be represented as a set whose length is N. For instance, the image in Fig.4a can be represented as a quadtree (Fig.4b), which has four levels. Its QTSS has length 4. In the notation used in the reminder of the paper, the root node is defined at the level 0. There is no black node at level 0, so we put 0 as the first member. There is one black node at level 1 and at level 2, so the second and third members are set as 1, respectively. There are 14 nodes at level 4, we put 14 as the fourth member. This set is represented as:

(0 1 1 14)

This set is represented by a broken curve in Fig.4d. Then we multiply these node numbers by the total number of pixels (i.e. leaf-level cells) represented by a quadtree node at the corresponding level. The result is shown by the solid curve. A node at level 1 has 16 pixels and there is only one node, The solid curve thus has 16 at level 1. At level 2, there is also one node, but the solid curve has value 4 here, because each node has 4 pixels. At the level 3, there are 14 nodes and they are single pixels, so the solid curve has value 14. We define the solid

curve as the QTSS of this image, and represent it as following:

(0 16 4 14)

The Mathematical Derivation of QTSS

1. General Mathematical Formulas for Image Linear Transforms

An original scene $g(x,y)$ can be viewed mathematically as a continuous surface. Suppose a non-linear operator $SQ\{\ \}$ samples and quantizes an image into a digital image $[G]$, so that

$$[G] = SQ\{\ g(x,y)\ \} \tag{1}$$

A general separable linear transformation on an image matrix $[G]$, may be written in the form

$$[\alpha] = [U]^t[G][V] \tag{2}$$

Consider unitary transform operators $[U]$,$[V]$ such that

$$\begin{aligned} [U][U]^t &= I \\ [V][V]^t &= I \end{aligned} \tag{3}$$

Thus,

$$[G] = [U][\alpha][V]^t \tag{4}$$

Let's rewrite

$$\begin{aligned} [U] &= [\ u_1\, u_2\, u_3 \cdots u_i \cdots u_N\ ] \\ [V] &= [\ v_1\ v_2\ v_3 \cdots v_i \cdots v_N\ ] \end{aligned} \tag{5}$$

Then

$$[G] = [\ u_1 \ldots u_N\ ]\,[\ \alpha\ ] \begin{bmatrix} v_1 \\ \vdots \\ v_N \end{bmatrix} \tag{6}$$

If the matrix α is written as a sum

$$[\alpha] = \begin{bmatrix} \alpha_{11} & 0 & . & 0 \\ 0 & 0 & . & . \\ . & . & . & . \\ 0 & . & . & 0 \end{bmatrix} + \begin{bmatrix} 0 & \alpha_{12} & . & 0 \\ 0 & 0 & . & . \\ . & . & . & . \\ 0 & . & . & 0 \end{bmatrix} + \cdots \tag{7}$$

Then

$$[G] = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_{ij}\, u_i\, v_j^t \tag{8}$$

The Fourier transform, Walsh transform, Hadamard transform and discrete cosine transforms, can each be expressed in the above forms.

2. Function q(i, j) and set Q(i)

The definition of function q(i,j) at domain [0,T] is:

$$q(i,j) = \begin{cases} 1 & \text{if } j\frac{T}{2^i} \le x \le (j+1)\frac{T}{2^i} \quad \text{for} \quad j<2^i \\ 0 & \textit{else} \end{cases} \tag{9}$$

Where i is the level of the functions, j is the position sequence number from left to right, T is the period. For each level i, the total number of j is 2^i. For example, the q(0,0) has only j=0, the 1st level has j=0 and 1, so on.

$$q(0,0) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \qquad q(1,0) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad q(1,1) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

We can define a set for all q(i,j) at same level i, called Q(i):

$$Q(i) = [\, q(i,0)\, q(i,1) \cdots q(i,2^i-1)\,] \tag{10}$$

The pattern of function q(i,j) at top three levels for T=16 are as follows:

3. Matrix Description

The set $Q(i)$ can be described as a matrix. For instance, for T=8, Q(2) is:

$$Q(2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Similarly, if we substitute Q for U in equation (5):

$$\begin{aligned} [U] &= [u_1\, u_2 \cdots u_i \cdots u_N] \\ &= [\, Q[0]\, Q[1]\, Q[2]\, Q[3] \cdots \,] \\ &= [\, q(0,0)\, q(1,0)\, q(1,1)\, q(2,0) \cdots \,] \end{aligned} \tag{11}$$

In matrix format T=8 would be:

$$[U] = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

4. Digital Image linear description in q(i,j)

Now we can describe a digital image applying the above formulas. Note that all q(i, j) within each Q(i) are orthogonal to each other. Thus, an image can be serially described by the following:

$$\begin{aligned} [G] &= Q(0)\,[\,\alpha_0\,]\,Q(0)^t + \bar{G}_0 \\ &= Q(0)\,[\,\alpha_0\,]\,Q(0)^t + Q(1)\,[\,\alpha_1\,]\,Q(1)^t + \bar{G}_1 \\ &= Q(0)\,[\,\alpha_0\,]\,Q(0)^t + Q(1)\,[\,\alpha_1\,]\,Q(1)^t + Q(2)\,[\,\alpha_2\,]\,Q(2)^t + \bar{G}_2 \\ &= \cdots\cdots \end{aligned} \tag{12}$$

where $\bar{G}_i$ is the redundancy of the image from the i_{th} level approximation term

and α_i is a matrix coefficient for the i_{th} term. We can recursively execute the procedure until arriving the bottom of the quad tree, where the nodes are pixels. The redundancy at this level is 0,

$$[G] = \sum_{i=0}^{l} Q(i) [\ \alpha_i\ ] Q(i)^t = \sum_{i=0}^{l} \sum_{j=0}^{2^i} \alpha_{ij}\ Q(i,j)\ Q(i,j)^t \tag{13}$$

where l is the maximum level number at the quad tree. α_{ij} is the coefficient. For deriving the last step, we apply the orthogonal property of all function q(i,j) with same level i. When the values of all pixels in the image are either "1" or "0", the image is called a binary image. (i.e., α_{ij} is either 0 or 1).

From a mathematical point of view, two notes are interesting here according to definitions of orthogonal and complete. First, a group of q(i, j) with identical i is orthogonal, but they are not complete. For instance, q(2, 0), q(2, 1), q(2, 2) and q(2, 3) are orthogonal. Second, total groups of q(i, j) with all i are not orthogonal. Nevertheless, equation (12) can represent an image precisely. Both orthogonal and complete definitions are from Beauchamp, 1984.

5. QTSS formulas

As mentioned above, all functions q(i,j) at the same level i are orthogonal. Figure 5 shows all patterns at the top three levels of $q(i_1, j_1)\ q(i_2, j_2)^t$.

The size of the individual quadtree cell varies with its level in the tree. The number of squares in the whole image is 4^i. Thus, α_{ij} is the coefficient for existence of a particular quadtree cell and we use the term

$$\sum_{j=0}^{2^i} \alpha_{ij}$$

to denote the component of the spatial spectrum at level i. Then, a set consisting of these members at all levels can be expressed as:

$$[\ \sum_{j=0}^{4^1}\alpha_{0j}\quad \sum_{j=0}^{4^1}\alpha_{1j}\quad \cdots\quad \sum_{j=0}^{4^i}\alpha_{ij}\quad \cdots\ ] \tag{14}$$

If the image is a binary image, where α_{ij} is 0 or 1, each member $\sum_{j=0}^{4^i}\alpha_{ij}$ is the number of squares at i_{th} level. Figure 4 shows a simple generation of QTSS directly from a quad tree pattern.

The QTSS method uses a set of regular squares, with a fixed series of sizes and positions. So far, the above derivation is not constrained to binary images. (i.e., α_{ij} does not have to be either 1 or 0). This means that QTSS is applicable to gray tone images.

Nevertheless, we will only discuss binary images in the remainder of this paper for the sake of clarity. It is more traditional to think in terms of the pixel concept instead of variable sized cells. A weight, which is the size (number of pixels) of nodes at different levels, is therefore multiplied by each term to improve interpretation of the spectra.

$$S_i = \left(\frac{T^2}{4^i}\right)\sum_{j=0}^{2^i}\alpha_{ij} \tag{15}$$

When this spectrum is normalized, the Quad Tree Spatial Spectra is generated.

$$QTSS = \frac{1}{\sum_{k=0}^{i} S_k}\,[\ S_0,\ S_1,\ S_2,\ \cdots,\ S_i\ ] \tag{16}$$

Efficiency Comparison with the Fast Fourier Transform

The QTSS and other transforms are all described in general mathematical forms (8) and (12). The Fourier, Walsh and other linear transforms have perfect orthogonal functions for whole wavelength domain. Their transforms are complete. As a result, their spectra in the wavelength domain include all spatial distribution information. Since the information amount in both spatial and frequency domains are the same, they can be used as bidirectional transforms, forward or inverse, without any loss of information, at least from the theoretical point of view. Based on this property, these transforms play an important role in image restoration as high-pass and low-pass frequency filtering procedures [8]. However, these transforms require a large amount of CPU time, especially for large 2-D data. For a 2-D image with size N*N, the FFT shortens execution time from $O(N^4)$ down to $O((N\ Log_2 N)^2)$. They are thus not suitable for frequent use in many geographic information system environments because of their computational complexity and subsequent time requirements.

Conversely, QTSS do not retain complete information of the original image. When we derived formula (12), each member only has the sum of the coefficients at the i_{th} level without recording their positions. We can therefore not reverse the QTSS procedure to restore the original image as is the case for the other transforms discussed above.

For an image with N*N pixels, the computational complexity of generating QTSS is proportional to $4/3N^2$, or $O(N^2)$. A simple explanation of this is as follows. Suppose at a level L, there are four quadtree nodes, $node_L[i]$, i=1, 2, 3, 4 .

They belong to one $node_{L-1}[j]$, at level $L-1$. A procedure for calculating their contribution to QTSS is:

```
begin
   initial j = 0
      begin
           for (i=1, 2, 3, 4) do
           if ( node_L[i] is 'black' )
           then j plus 1
      end

      begin
           if j = 4
           then QTSS ( L ) plus 4
           assign node_{L-1}[k] 'black'.
      end
end
```

This subroutine has 1.5*4 simple additions in the worst case, or we can say, the number of additions is less than proportional to the number of nodes. For an image with N*N pixels, the number of additions at the lowest level is less than proportional to $\frac{N^2}{4}$, and at the next lowest level, the number of additions is less than proportional to $\frac{N^2}{16}$, and so on. Thus the amount of total additions from leaves to the root of the quadtree is:

$$N^2\left(1+\frac{1}{4}+\frac{1}{16}+\frac{1}{64}+\cdots\right)=\frac{4}{3}N^2$$

This means that for larger images, there is an increasingly more obvious time saving. When an image is over 1024*1024 pixels, the CPU time spend for QTSS is less than one percent of the time required by the Fast Fourier transform. Thus, if all 2-D data were constructed as quad trees in a GIS, the utilization of QTSS will save a considerable amount of CPU time as compared to other transform

methods.

Spatial Distribution Knowledge Interpreted from QTSS

QTSS can directly describe the spatial distribution properties, such as density, degree of clustering and approximate location, of any phenomenon represented in a single data layer. Three graphic examples are shown in Figures 6, 7, and 8, which were extracted from the landuse data layer of our test data set representing an area adjacent to the Black Hills in southwestern South Dakota. Each figure includes a quad tree image and its corresponding QTSS histogram. In Figure 6a, the quadtree image of land covered by evergreen forest is seen to be predominately clustered in a few large areas. Much of the coverage for these large areas are represented by correspondingly large (i.e., high-level) quadtree nodes. The histogram of these data in Figure 6b shows a dominance of long wavelength components, with the peak located at level 5. Figure 7a is the rangeland component of the same landuse layer, showing a mixed distribution — most are clustered in smaller areas and some are small or fragmented areas. Figure 7b is its QTSS. Its wavelength component distribution seems relatively even, covering a wide wavelength domain. Its peak locates at level 8. Figure 8a is the urban component of the land use layer. Most of its pixels are scattered. A few of them are merged into small blocks. Figure 8b shows a strong short wavelength components in the QTSS. Its peak locates at level 9.

1. Use of QTSS for spatial filtering

QTSS can be utilized for "high-pass", "low-pass" and "band-pass" filters, similar to Fourier and Walsh transforms. In the wavelength domain, short wavelength components emphasize boundary information, while long wavelength components correspond to the interior areas of spatial objects. For example,

the edges of irregularly shaped objects tend to be represented by many small (i.e., low-levels) quadtree nodes. This effect has been experimentally mentioned in [17].

A filtering operation in the quadtree environment is relatively fast and simple. Those levels, whose nodes have edge length within the given omitted wavelength range are simply removed.

Edge enhancement can thus be performed as a high-pass filtering operation on data represented in quadtree form. Figure 9 shows a high-pass filtered result from the image in Fig.6, and its corresponding QTSS. A high-pass filter removed those nodes at level 0 ~ 10. All remaining nodes have wavelength equal to one pixel (the side-length of a node at level 11). Figure 8b shows an example of the edge enhancement effect resulting from this operation.

Conversely, Figure 10 shows two results from the same image after low-pass filtering. In Fig.10a, those nodes at levels 1 ~ 9 are kept; In Fig.10b, those nodes at levels 1 ~ 8 are kept. We don't see much difference between them visually, nor do we see much difference between the filtered images and the original, showing that most of the information, as it relates to total spatial area and overall pattern, has been preserved. The total volume of the data, however, has been greatly reduced due to that small instancery are eliminated and the remaining larger instances can be recorded very simply. Table 1 shows a comparison of precision and storage of those low-pass results. Here the precision is defined as a ratio of the number of remaining pixels over the number of total pixels in the original image. Assume that the space required by the original image are 100%. If we use the low-passed (1~10) levels to approximate the original image (1~11), the storage space is 48.4%, and still keeps 98.36% of the original pixels. This means that we save over half of the original storage space at a loss of less than 2% of the pixels. If we use a low-pass filter to remove (10~11)

and keep (1~9), the storage space is only 22.5% of the original space at a loss less than 5% of the pixels.

2. Scene context information generating from QTSS

Previous spatial analyses in digital image processing can be classified according to their active spatial wavelength bands. For the case of discussion, we use the width of a pixel as the spatial wavelength unit. Multi-radiational spectral analysis works on the single pixel level (≤ 1). Filtering involves several to dozens of pixels ($1 \sim 10$). Texture is still a local property and it depends on lower level spatial information, usually concerning from several to hundreds of pixels ($1 \sim 100$) [23]. Hence these processes only concern with the shorter wavelength portion of spatial spectra.

Contextual analysis depends on spatial information at a wide spatial wavelength range, but much emphasis is given on longer wavelength components of the spatial spectra. It is useful to employ QTSS in contextual analysis because it provides a convenient way to acquire, represent, and manipulate spatial information at a wide range in wavelength, from one pixel to a whole scene.

Generally speaking, two kinds of contextual descriptions are considered useful, and can be generated from QTSS:

a. the overall pattern of the distribution of objects. This could be scattered evenly, clustered, or a mixture of the two.
b. locational concentration relative to quadrants or subquadrants, such as eastern, western, northern or southern.

A brief description of the generation of these contextual information is following:

(1). distribution pattern --- measuring degree of clustering by QTSS

The spatial pattern of a distribution is essentially what spatial spectra

represent, and various distributional measures can thus be easily generated directly from QTSS. A quantitative representation the spatial pattern is explored here based on QTSS. Define $CLUS(L)$ as a set derived from Eq(17):

$$CLUS(L) = [\ C(0),\ C(1),\ .....,\ C(L),\ ....,\ C(i)\ ] \tag{17}$$

where

$$C(L) = \frac{\sum_{k=i}^{L} S_k}{\sum_{k=0}^{i} S_k} \tag{18}$$

where S has been defined in Eq.(15).

The interpretation of C(L) is the probability of a pixel clustering into blocks with the size larger than a node at the L_{th} quadtree level. In Figure 11, three CLUS(L) have been drawn from the QTSS in Fig. 6b, 7b, and 8b. We see, all C(0) are 1.0, and the CLUS(L) is monotonically decreasing from C(0)=1 to C(i)=0. Different positions of the three curves show different clustering behaviors: the one which is closer to the vertical axis shows stronger clustering.

If we wish to represent these same data at 85% accuracy, we simply draw a horizontal line at 0.85 and omit data above this level. We then know that for the evergreen forest data, level 9 ~ 11 can be omitted in this operation. However, for rangeland data, we only can omit levels 10 and 11. For residential data, we must use all of the original data.

(2). Locational concentration

One QTSS of the whole image is not enough for interpretation of locational concentrations. Storing QTSS at each quadrant and subquadrant within one image, we can know more detail about the image. Figure 12 shows the QTSS of Fig.7a, and four QTSS at its each quadrants. We can call this a Quadtree Spatial Spectra Tree (QTSST). Table 2 shows the corresponding five QTSS for residential landuse of Rapid City, South Dakota. The first entry represents the root, the

other four are for the four quadrants, respectively, where the $QTSS_p$(L, N) is the QTSS of phenomenon P of the N_{th} node at the L level. The QTSS at the root shows that there are residential areas in the image. From the QTSSs from the four quadrants, it can be seen that most of the residential areas are in quadrants 2 and 4, with a few in quadrant 1 and none in quadrant 3.

3. Complex phenomena represented by Multiple-layers

A complex phenomenon is defined as being represented by two or more separate data layers (i.e., "elementary" phenomenon) of the image. We can also analyze the combined features of the QTSSs of those elementary phenomena to derive information concerning complex phenomenon. We can call the combined QTSST a Quadtree Spatial Spectra Forest (QTSSF).

For example, an urban area can be characterized as possessing a particular amount of urban residential, industrial and transportation land use areas in characteristic distributional patterns. Then the three QTSST corresponding to these can be used to detect urban areas within a given scene, see Figure 13.

Concluding Remarks

In this paper, we have introduced a new form of spatial spectra, Quadtree Spatial Spectra (QTSS), which is based on the quadtree spatial data model. The primary advantage of QTSS is that it is significantly faster to calculate than the fast Fourier Transform. QTSS has a computational complexity of $O(N^2)$, where N is the total number of pixels, as opposed to $O((1/Log_2 N)^2)$ for FFT. Although QTSS is not completely information preserving as are the Fourier and Walsh transforms, they can still be used in a similar manner for describing and analyzing spatial distributions.

Based on the comparative computational speed of QTSS, it seems particu-

larly suitable for use in a large-scale geographic information systems context.

References

[1] K.G. Beauchamp, "Applications of Walsh and related functions, with an introduction to sequency theory", London, Academic Press, 1984.

[2] N.S. Chang and K.S. Fu, "Picture query languages for pictorial data-base systems", Computer Vol.14, No.11, pp.23-33, 1981.

[3] Z.T. Chen, "Quadtree Spatial Spectrum: Its generation and Applications", Proceedings of the International Symposium on Spatial Data Handling, Zurich, pp.208-237, 1984.

[4] Z.T. Chen and D. Peuquet, "Quadtree Spatial Spectra Guides: A Fast Spatial Heuristic Search in a Large GIS", Proceedings of AUTO-CARTO 7, Washington D.C., pp.75-83, 1985.

[5] C.R. Dyer, A. Rosenfeld and H. Samet, "Region representation: boundary codes from quad trees", Communications of the ACM, pp.171-179, 1980.

[6] C.R. Dyer, "The space efficiency of Quadtrees", Computer Graphics and Image Processing, 19, pp.335-348, 1982.

[7] I. Gargantini, "An effective way to represent Quadtrees", Communications of the ACM, Vol.25, No.12, (Dec), pp. 905-910, 1982.

[8] R.C. Gonzalez and P. Wintz, "Digital image processing", Addison-Wesley Pub. Inc. 1977.

[9] A. Klinger and C.R. Dyer, "Experiments on picture representation using regular decomposition", Computer Graphics and Image Processing, 5, pp. 68-105, 1976.

[10] J. Lintz and D.S. Simonett, (eds.), "Remote Sensing of Environment", Addison-Wesley, Reading, Mass., 1976.

[11] D.M. Mark and D.J. Abel, "Linear Quadtrees from Vector Representations of Polygons", IEEE Trans. Pattern Anal. Mach. Intell. Vol.PAMI-7, No.3, pp.344-349, 1985.

[12] D.M. Mark and J.P. Lauzon, "Approaches for quadtree-based geographic information systems at continental or global scales", AUTO-CARTO 7 proceedings, Washington D.C. pp. 355-364, 1985.

[13] H. Moellering and W.R. Tobler, "Geographical Variances", in Geographical Analysis, Vol.1, No.2, pp34-49, 1969.

[14] D. Peuquet, "Data structure for a knowledge-based Geographic Information System", Proceedings of the International Symposium on spatial data handling, Zurich, Vol.2, pp.372-391, 1984.

[15] S. Ranade, "Use of Quadtrees for Edge Enhancement", IEEE Trans. on System, Man and Cybernetics, Vol. SMC-11, No.5, May, pp.370-373, 1981.

[16] S. Ranade and M. Shneier, "Using Quadtrees to smooth images", IEEE Trans. on System, Man and Cybernetics, Vol. SMC-11, No.5, May, pp.373-376, 1981.

[17] S. Ranade, A. Rosenfeld and H. Samet, "Shape approximation using Quadtrees", in Pattern recognition, Vol.15, No.1, pp.31-40, 1982.

[18] H. Samet, "Connected Component Labeling Using Quadtrees", J. ACM, Vol.28, No.3, pp.487-501, 1981.

[19] H. Samet, "Distance Transform for Images Represented by Quadtrees", IEEE Trans. Pattern Anal. Mach. Intell. Vol.PAMI-4, pp.298-303, 1982.

[20] H. Samet, "Hierarchical data structures for representing geographical information", in Computer science TR-1208, University of Maryland, College Park, MD, 1982.

[21] H. Samet, "The Quadtree and Related Hierarchical data structures", ACM Computing Surveys, Vol.16, No.2, June, pp.187-260, 1984.

[22] H. Samet and M. Tamminen, "Computing Geometric Properties of Images Represented by Linear Quadtrees", IEEE trans. Pattern Anal. Mach. Intell., Vol.PAMI-7, No.2, 1985.

[23] E.H.H. Shih and R.A. Schowengerdt, "Classification of Arid Geographic Surfaces Using Landsat Spectral and Textural Features", in Photogrammetric Engineering and Remote Sensing, Vol.49, No.3, pp. 337-347, 1983.

[24] R.L. Shelton and J.E. Estes, "Remote Sensing and Geographic Information System: An Unrealized Potential", in Geo-Processing, No.1, pp.395-420, 1981.

[25] T.R. Smith and M. Pazner, "Knowledge-based control of search and learning in a large-scale GIS", Proceedings of the International Symposium on spatial data handling, Zurich, Vol.2, pp.498-519, 1984.

[26] W.R. Tobler, "Spectral Analysis of Spatial Series", Fourth Annual Conference on Urban Planning Information System and Programs, Berkeley, 1966.

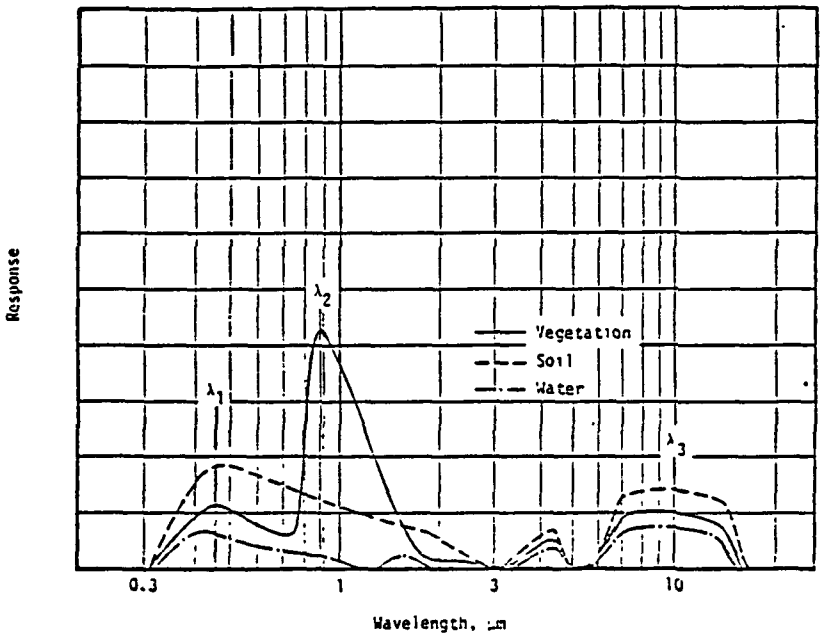
[27] W.R. Tobler, "Geographical Filter and their Inverses", in Geographical Analysis, Vol.1, No.3, pp234-253, 1969.

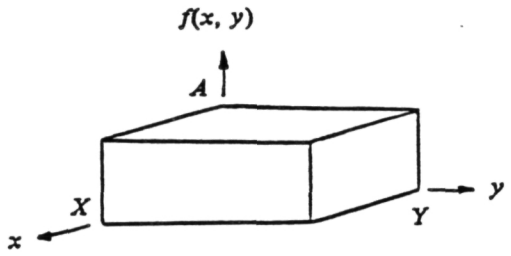
[28] R.F. Tomlinson and A.R. Boyle, "The state of development of systems for handling Natural resources inventory data", on Cartographica, Vol.18, No.4,

pp.65-95, 1981.

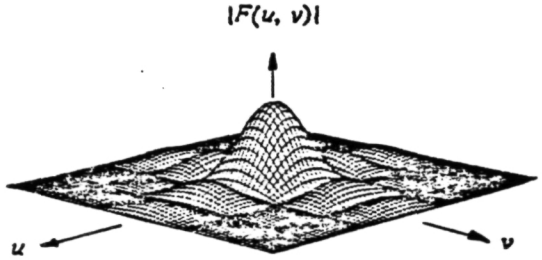
List of Figures and Tables

Figure 1. A histogram of radiation spectra for three geographic phenomena (Landgrebe 1972).

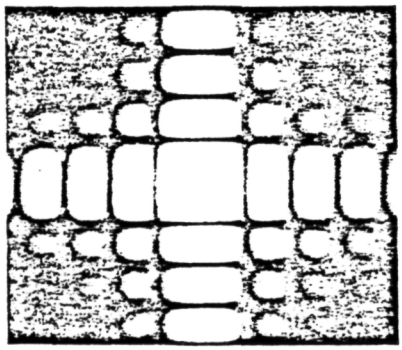




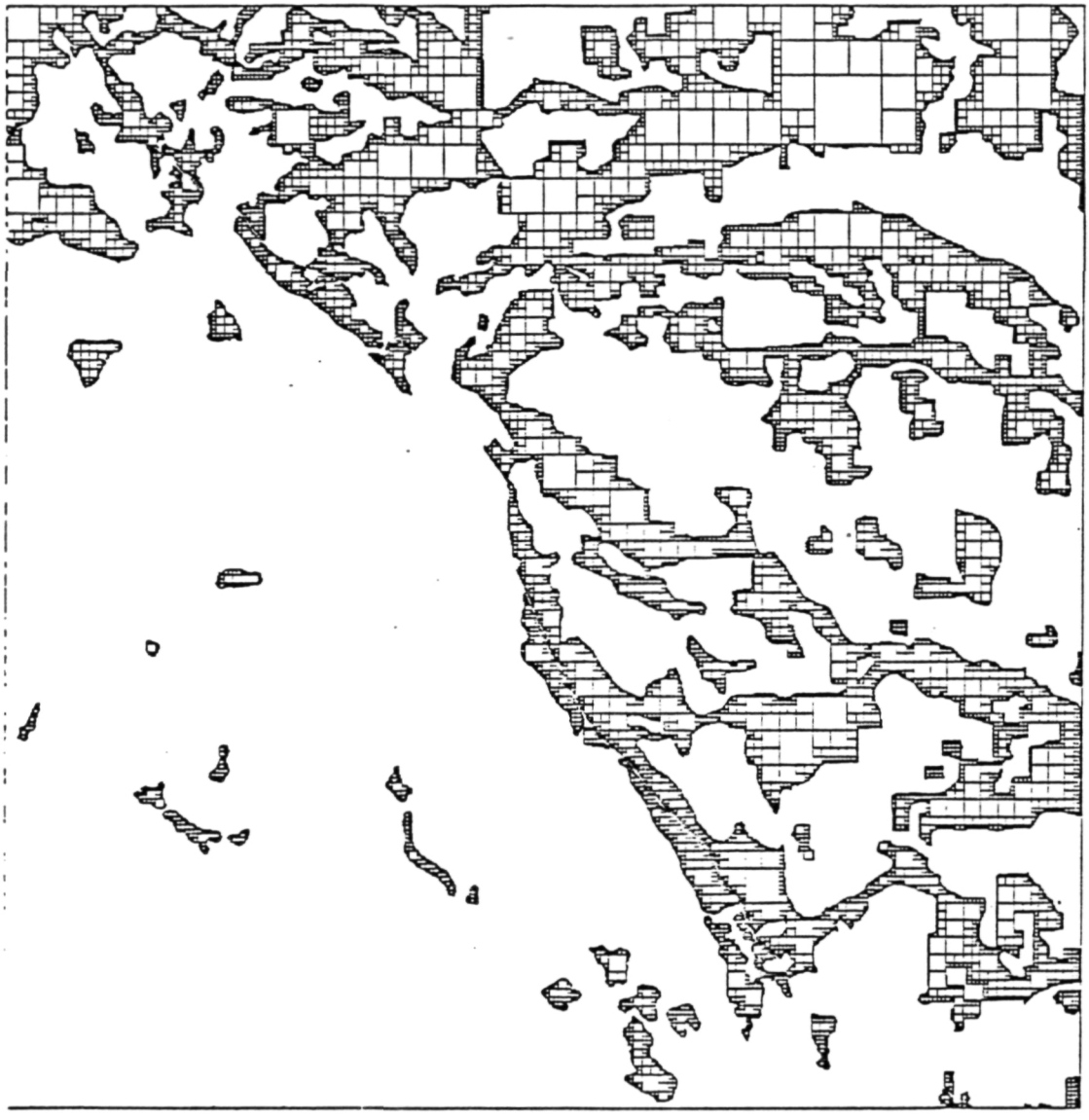
(a)



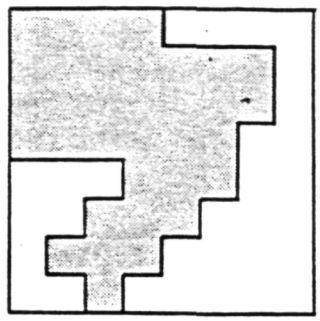
(b)



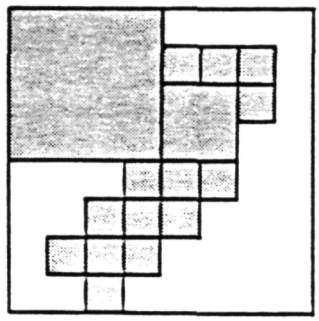
(c)



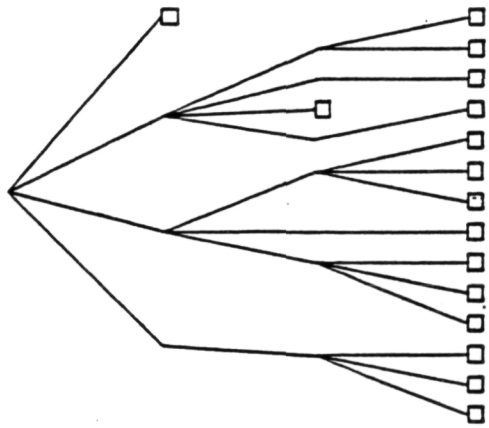
a) An image



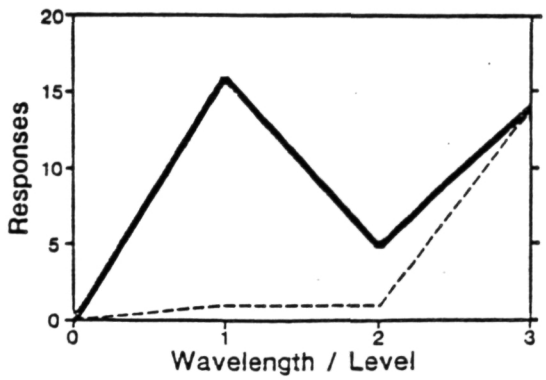
b) Its graphic quadtree representation



c) Its representation in tree form



d) Its Quadtree Spatial Spectrum



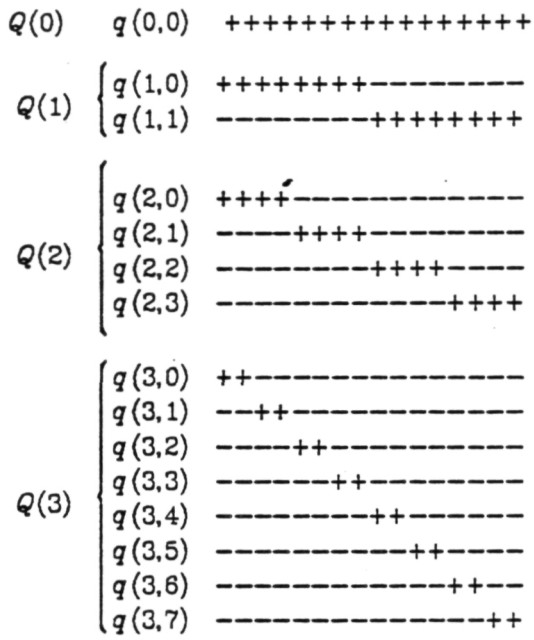
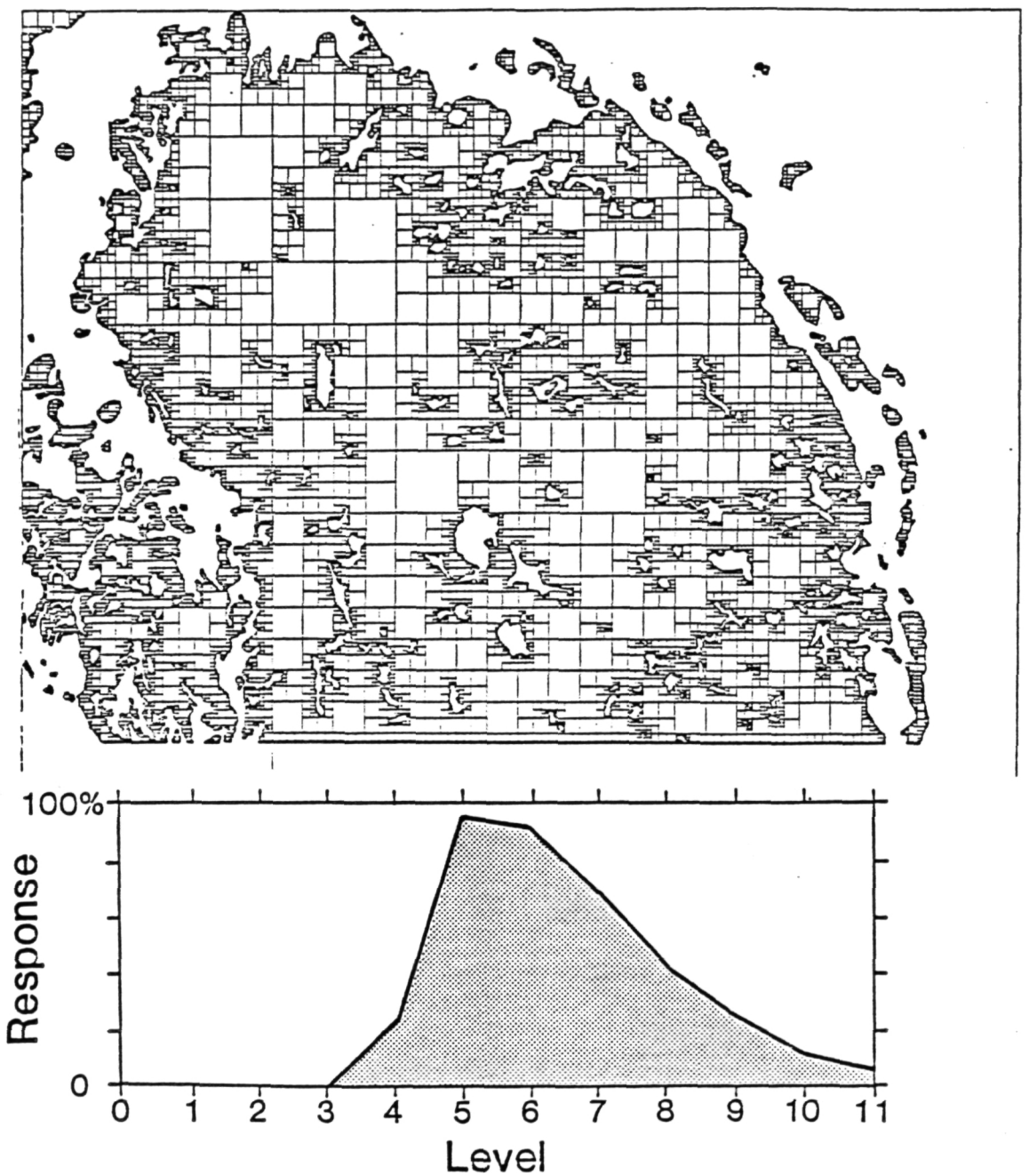
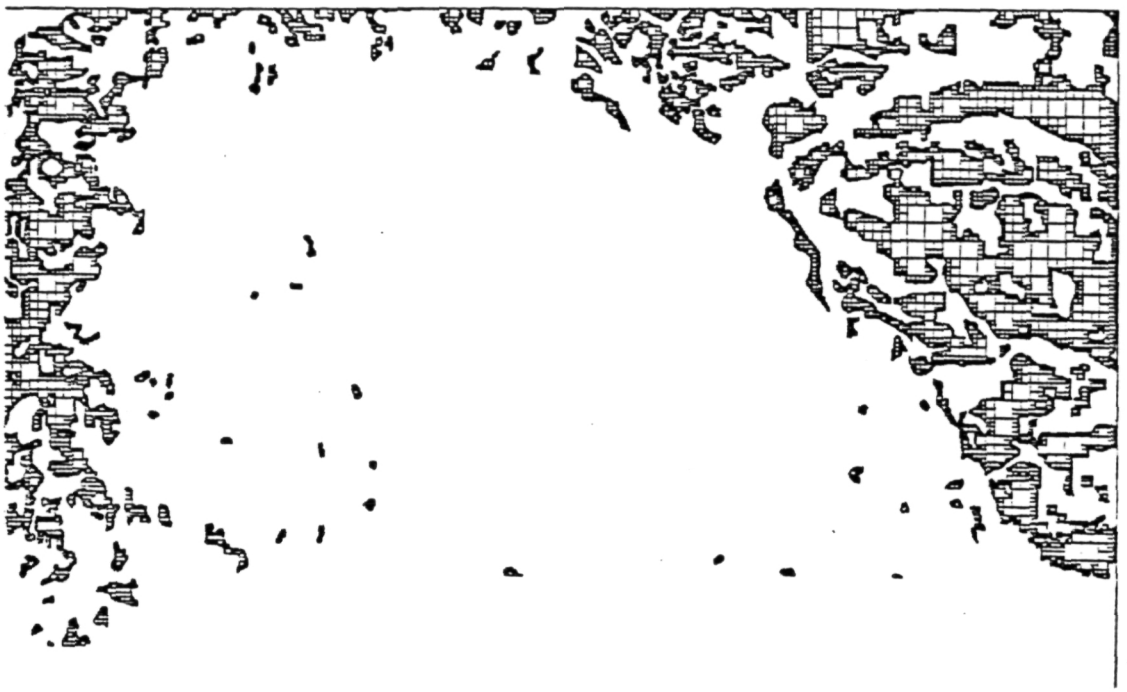
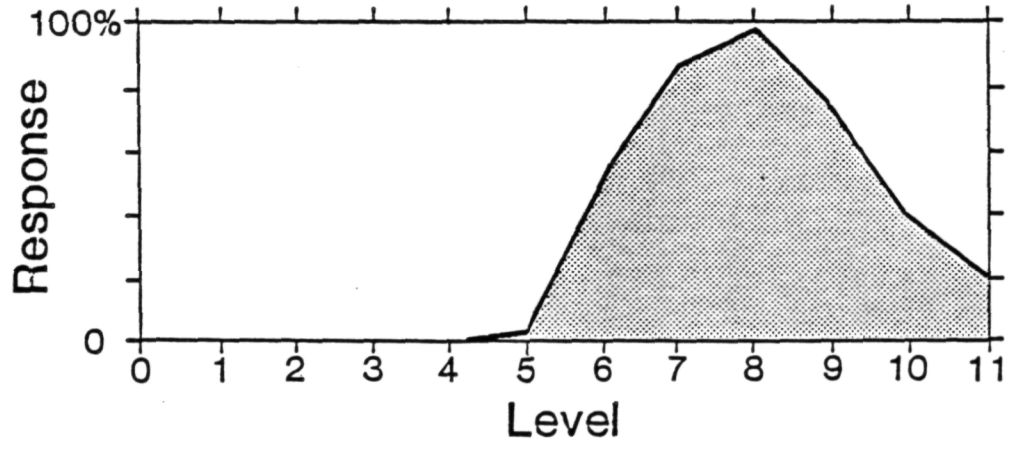
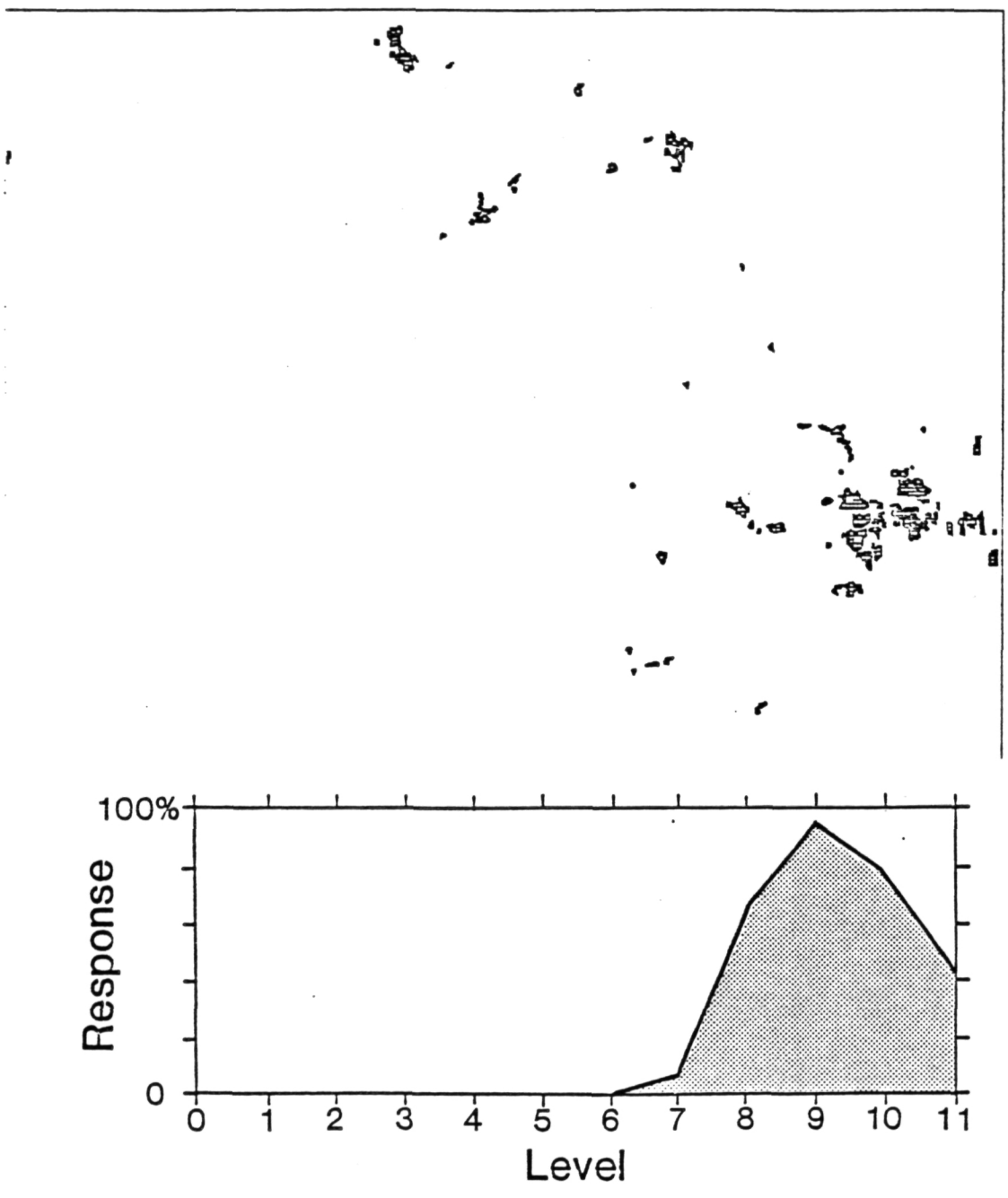


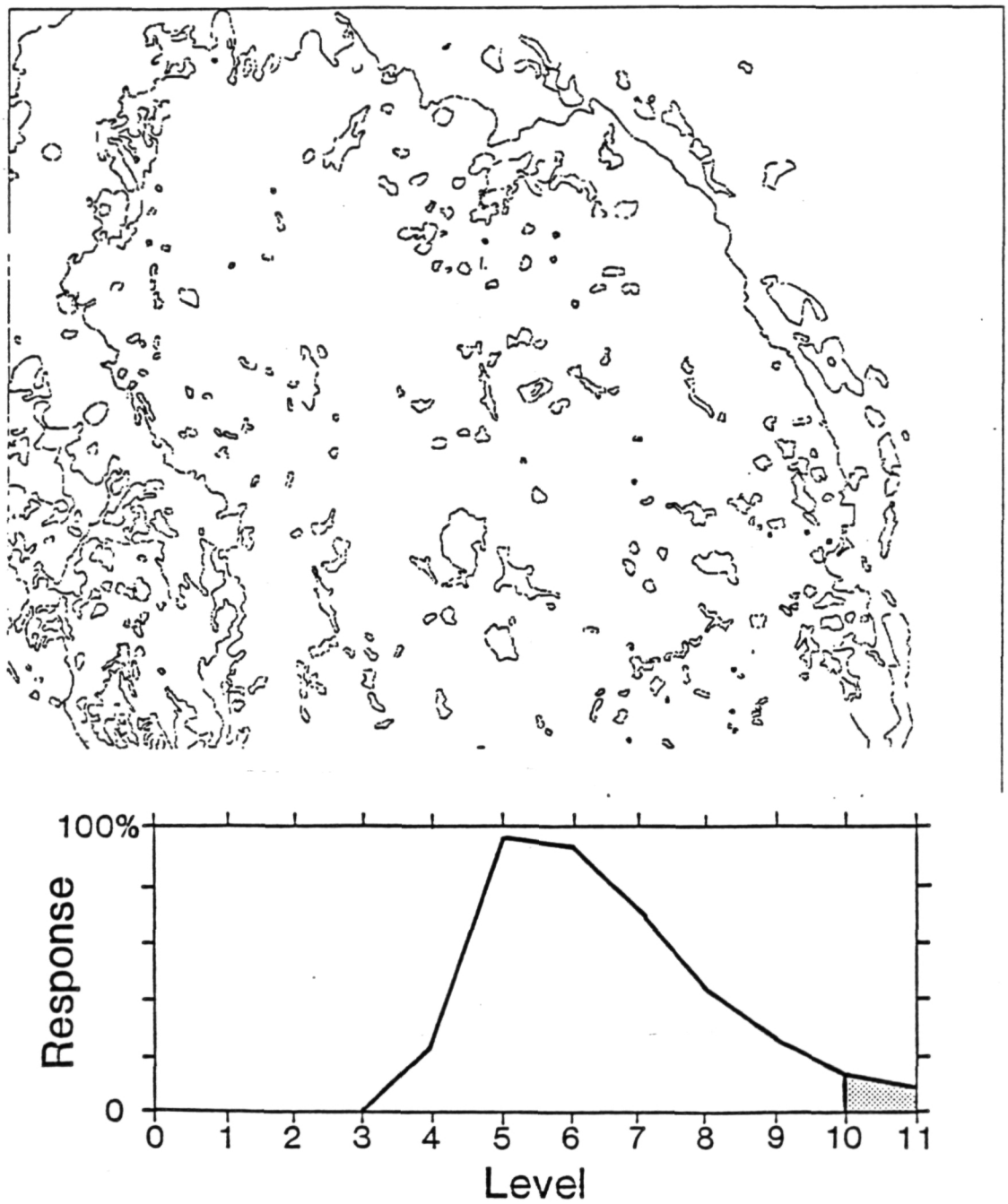
Figure 5. Patterns of functions Q(i) and q(i.j) at the top three levels

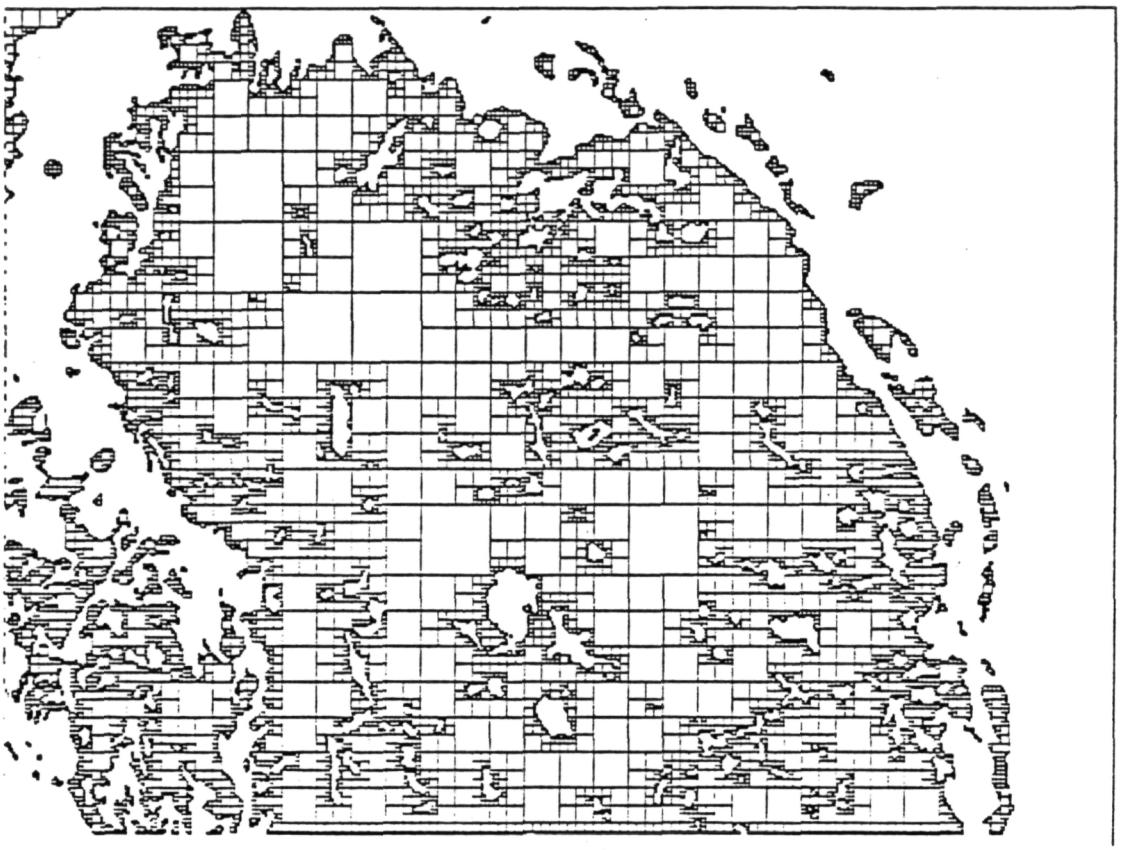

100%
Response
0
0 1 2 3 4 5 6 7 8 9 10 11
Level

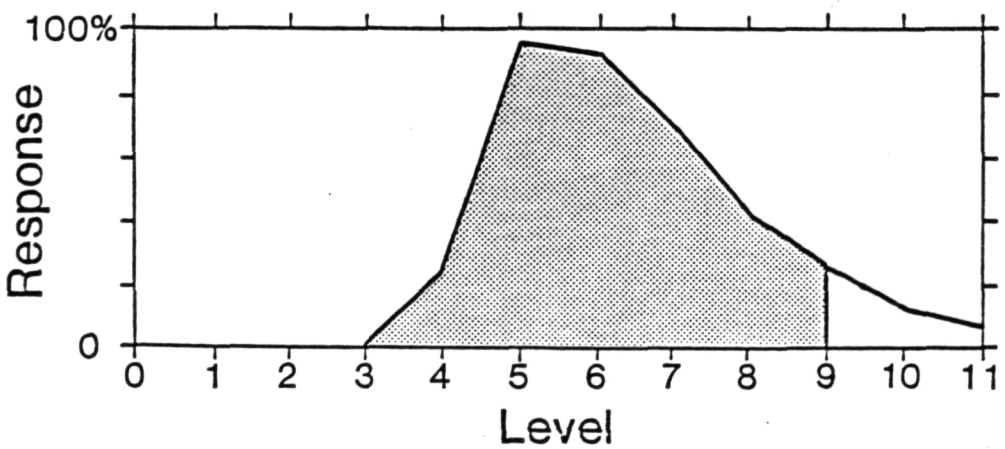


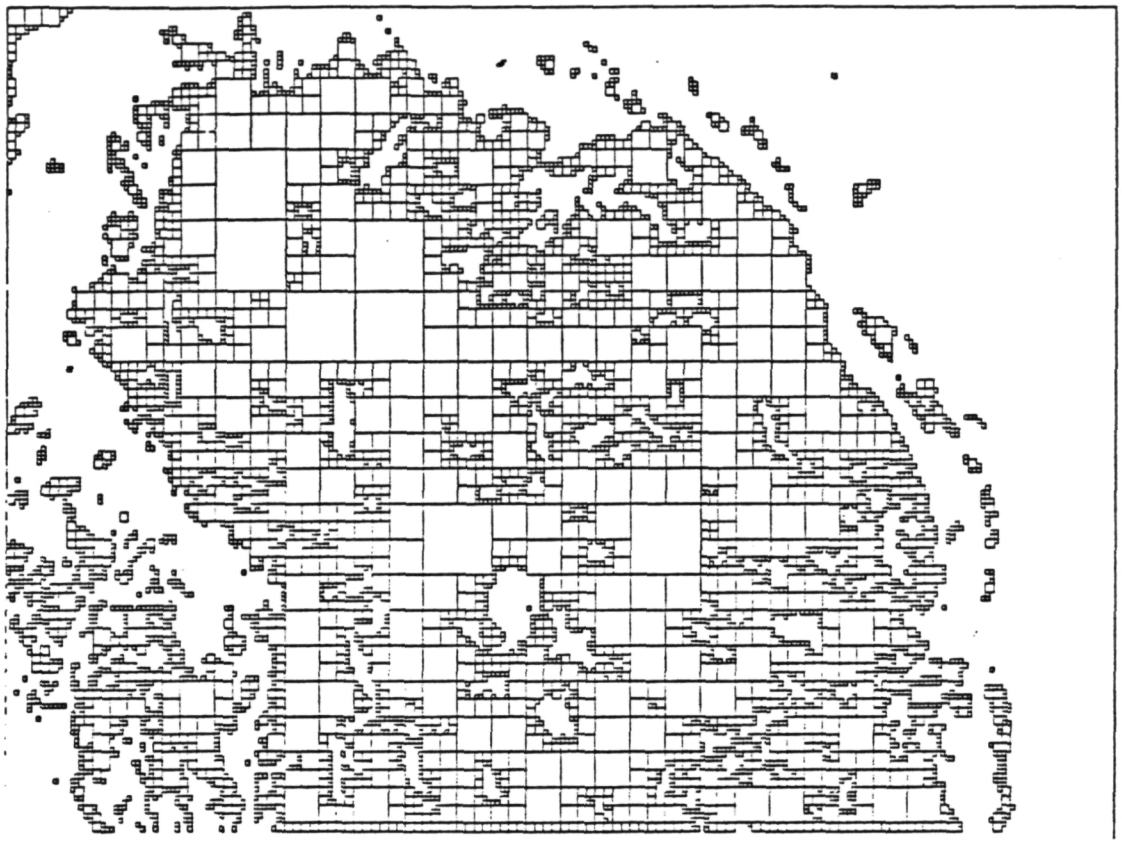



100%
Response
0
0
1
2
3
4
5
6
7
8
9
10
11
Level


100%
Response
0
0 1 2 3 4 5 6 7 8 9 10 11
Level







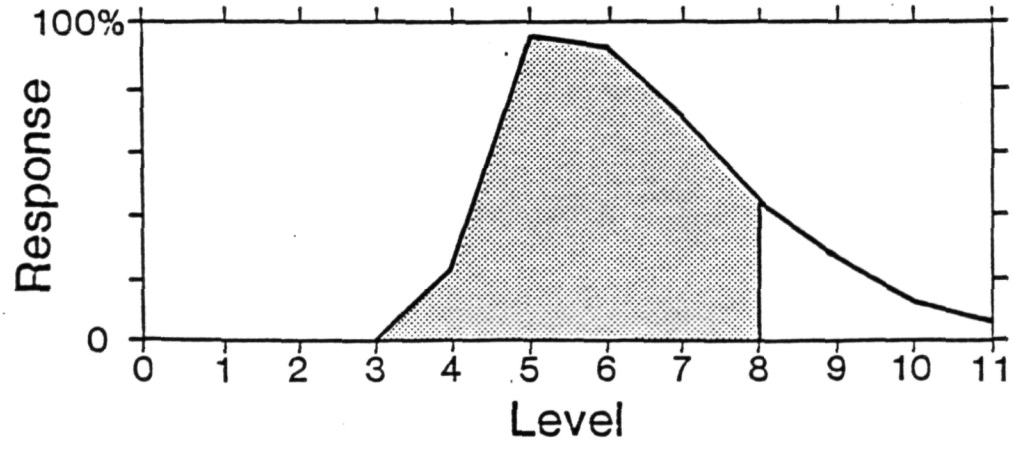


Table 1. List of precision and storage of
Low-pass results for the everegreen data

levels	nodes	storage	precision	pixels
1 ~ 11	57890	100.0%	100.0%	1828649
1 ~ 10	27991	48.4%	98.36%	1798756
1 ~ 9	13000	22.5%	95.08%	1738768
1 ~ 8	5229	9.0%	88.29%	1614528

Table 2. An example of the five QTSS (in Fig.12) of residential area

$QTSS_{lu11}(0, 0)$ = (0 0 0 0 0 0 2 82 472 1456 3311)

$QTSS_{lu11}(1, 1)$ = (0 0 0 0 0 0 6 64 252 525)

$QTSS_{lu11}(1, 2)$ = (0 0 0 0 0 1 42 214 692 1729)

$QTSS_{lu11}(1, 3)$ = (0 0 0 0 0 0 0 0 0 0)

$QTSS_{lu11}(1, 4)$ = (0 0 0 0 0 1 34 194 512 1057)

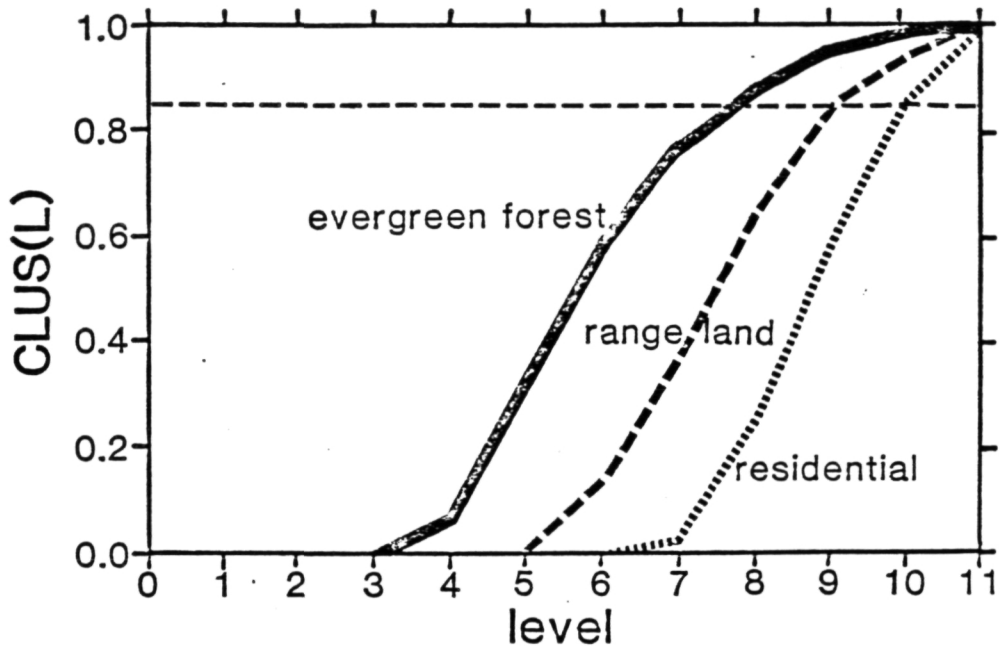
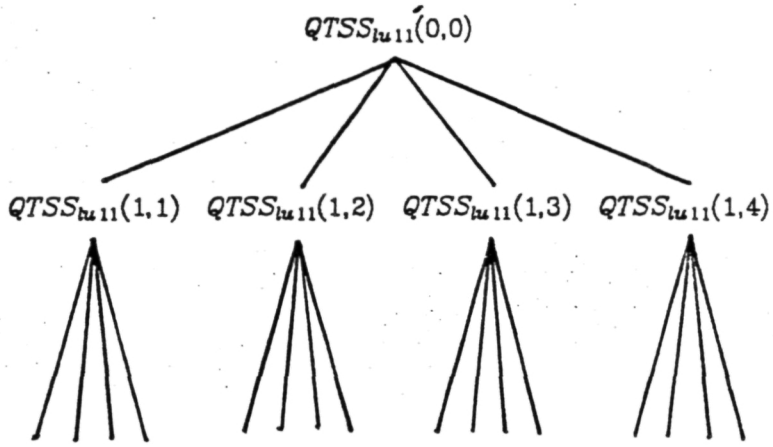
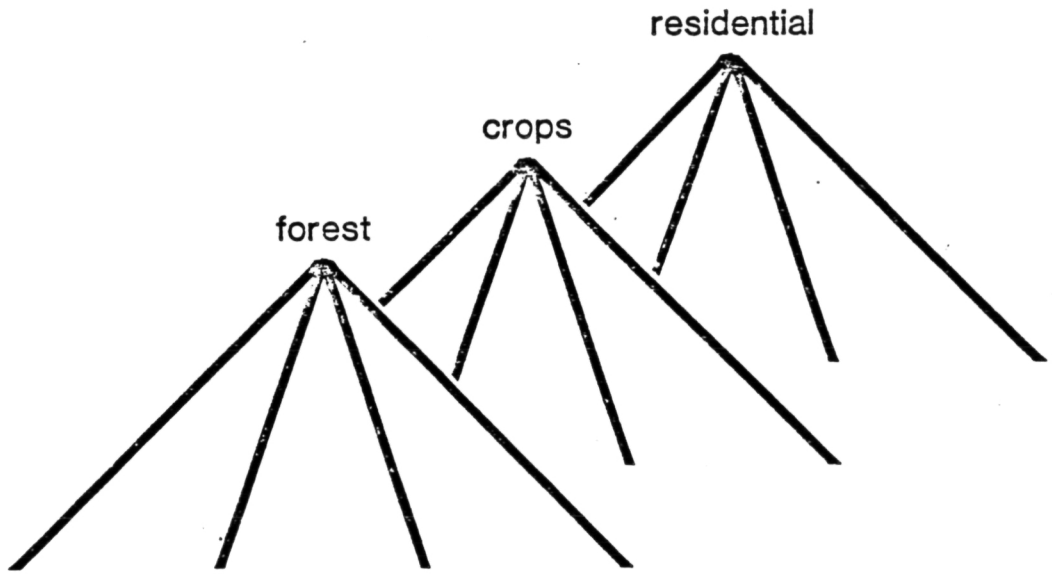

1.0
0.8
0.6
0.4
0.2
0.0
CLUS(L)
evergreen forest
range land
residential
0
1
2
3
4
5
6
7
8
9
10
11
level

Figure 12. The structure of a QTSS tree
(only the top two levels of the QTSS tree are shown here)




residential
crops
forest