

MANAGING SPATIAL OBJECTS WITH THE VMO-TREE

Weiping Yang and Christopher Gold

Chaire industrielle en géomatique Centre de recherche en géomatique
Université Laval Québec, Canada

Abstract

An object-oriented approach is described to model a set of 2D vector data, depicting points, lines, and polygons. These primitive spatial objects are embedded in Euclidean space and are topologically related with the dynamic Voronoi diagram. A *spatial object condensation* technique is used to partition the Voronoi diagram into smaller subsets. Each subset corresponds to a bounded subspace called a *condensed object* which can contain primitive and smaller condensed objects. Condensed objects are managed and manipulated by the *VMO-tree* (Voronoi based Map Object Tree), a novel data model describing relationships between condensed objects. The VMO-tree can be constructed by either top-down, bottom-up, or combined approaches. Spatial objects in the VMO-tree are presented in different levels of detail. Topological structures for every condensed object are integrated by the structure of the tree. Spatial searches can be performed vertically along edges of the tree or horizontally inside a topological structure of each node object. It is not required that all nodes of the tree are loaded in memory at one time. Any node object is an independent map and can be worked with separately. Each map sheet corresponds to a disk page and has a contiguous memory occupation.

1, Introduction

Contemporary mapping applications require that spatial data be handled with flexibility. This need can be imposed on all aspects of processing data, such as: viewing, storing, manipulating, analyzing, and communicating. Ideally, any interesting data can be organized as an operable unit or object, the change of focus back and forth amongst objects should be smooth and automatic, and the system should allow both outlined and detailed views. From the data structure point of view, these requirements imply that the spatial data be topologically structured in a hierarchical fashion, with varying levels of detail. Each node accommodates a region of arbitrary polygonal shape which again contains isolated points, lineal features, and smaller regions of different complexity, and the nodes at each level correspond to disk pages.

The paper describes an object-oriented dynamic data model satisfying the above needs. It first summarizes previous contributions to spatial data handling and sets the perspective view of this research. An alternative approach based on the kinematic Voronoi diagram [Gold 1990; Roos 1991; Gold et al. 1995] is then proposed. Disadvantages of the primitive Voronoi

diagram applied to large data sets are pointed out. They are overcome by introducing the *spatial object condensation* [Yang and Gold 1995] concepts. The major contribution of this paper, the VMO-tree class, is described as a dual structure managing condensed objects.

2. A review of spatial data structures

Realizing that it is the quantity and geometric complexity of spatial data that make the task of management and operation intricate, research over the past two decades has aimed to moderate the problem by decomposing a data set into smaller ones. This results in numerous geometric data structures which are broadly categorized into three families: trees (i.e. K-d tree [Bentley 1975], Quadtrees [Samet 84]); buckets (i.e. Grid Files [Nievergelt et al. 1984, EXCELL [Tamminen 1983)]; and linear orderings (i.e. Morton code [Morton 1966], Spiral order [Mark and Goodchild 1986]). Tree structures recursively decompose space. Depending on decomposition schemes used and types of geometric primitives concerned, trees can have different shapes and the contents of intermediate nodes vary as to whether geometric elements participate in the partition. Leaves of a tree can contain a single object or a bucket of multiple objects. Bucket structures are not hierarchical: space is flatly divided into rectangles of different size. Each rectangle has a location code as its address. The criteria guiding the divisions vary depending on the distribution of spatial objects and on the dynamics of a structure being used. The storage for indices to buckets is generally smaller than that for trees because each bucket contains more objects. Linear orderings transform a k-dimensional problem into a one-dimensional one. They originate from scanning and storing pixels of images into linear lists and are extended to index points and buckets. Other ordering schemes are designed to preserve spatial contiguity.

Classical geometric data structures are constructed for 0-cell objects. Variants of the original data structures have been developed to accommodate complex objects. For example, Matsuyama et al. [1988] modify the K-d tree to index axes-parallel rectangular blocks, where three types of geometric primitives are recorded. The PM Quadtree [Rosenberg 1985], is used to partition a polygonal map. This type of adapted data structure is still based on the space *decomposition*. Another type of tree structure is based on *object decomposition*, where the dividing criteria are oriented to objects themselves. This kind of data structure is suitable for representing polylines and polygons in different levels of detail. Nodes of trees record the result of a line generalization process, with lower ones containing partial objects of finer resolution. Examples of object based trees are the Strip tree [Ballard 1981], the Arc tree [Gunther and Wong 1989], and the BLG-trees [van Oosterom 1990]. Examples of data structures that support generic geometric objects include the R-tree [Guttman 1984], the R⁺-tree [Faloutsos et al. 1987], the Field-tree [Frank and Barrera 1989], the Cell-tree [Günther 1988], and the BANG files [Freeston 1987].

While geometric data structures provide a facility for data storage and access, they cannot form the sole data structure for a spatial information system. In addition to managing and accessing geometric data, a spatial information system must be able to answer topological queries about topological relationships among spatial entities. Examples of the topological relationships include adjacency, containment, neighbourhood, and connectivity. Although it can be argued that geometric data structures preserve certain topological relationships in their subdivision, they are not designed to efficiently answer all topological queries. Quick response to topological queries is the purpose of using topological data structures in designing a spatial information system. Well known topological structures, as are widely used in current vector based commercial GISs, are based on the planer graph topological data model. Topological relationships of a given data set, representing points, lines, and polygons, can be calculated and assembled after all the data is present.

We are left with the situation that, on the one hand, research on geometric data structures is directed towards the hierarchical, representing objects in different levels of detail, and partitioning space or objects into disk pages; and on the other hand, widely used topological data structures are still based on a flat, indivisible planer graph model. In a hybrid system, where both geometric and topological data models are employed, complex links between the two must be built. It would be difficult to maintain the integrity of topological relationships if update occurs in geometric data structures.

Recent research on spatial data models tackles the problem by developing generic spatial data models with a hierarchical topology. Worboys [1993] proposes a canonical representation of areal objects which explicitly captures connectedness and region inclusion. Without providing an implementation structure, the paper formalizes the definition of classes of areal objects and operations upon these objects. Bertolotto and De Floriani [1994] report a HPEG (Hierarchical Plane Euclidean Graph) for a multiresolution representation of a region. The HPEG recursively decomposes regions into smaller ones. Region boundaries at one level are simplified by creating e-homotopys with line generation algorithms, where e specifies the radius of a band convoluted from a chain of edges. Each smaller region in HPEG is a PEG whose features are refined with smaller horizontal error. In order to support navigation in the hierarchy of PEGs, the boundary information of a PEG must be recorded and the links to the direct refinements of a PEG must be maintained. The HPEG provides a way of browsing a map at different levels of resolution. The data structure implementing the HPEG is based on the DCEL [Preparata and Shamos 1985], an encoding of an edge-oriented planar subdivision. It is not clear how a PEG be stored in a separate disk page and operated upon as an independent object in a dynamic environment.

Starting in Section 3, we propose an alternative implementation of a hierarchical topological data model. This data model is aimed at producing a set of meaningful map objects, such that each node of the hierarchy has both geometric definition for its embedded components and a complete topology for the components. In addition, each node corresponds to a disk page and

can be worked with in both linked and unlinked fashions. We believe that the ability to represent maps as manageable objects in both spatial and memory subspaces is a key feature in object-orientation.

3. The dynamic Voronoi diagram

Given a set of n ($n > 1$) objects $S = \{\text{points, line segments}\}$ in the R^2 Euclidean space, the dynamic Voronoi diagram, $V(S)$, is constructed incrementally [Gold 1990; Roos 1991; Gold et al. 1995] which partitions the plane into *Voronoi regions*, v , such that for two distinct objects $s_i, s_j \in S$, the dominance of s_i over s_j , is defined as the subset of the plane being at least as close to s_i as to s_j :

$$D(s_i, s_j) = \{x \in R^2 \mid \delta(x, s_i) \leq \delta(x, s_j)\},$$

for δ denoting the Euclidean distance function. $D(s_i, s_j)$ is a closed half plane bounded by the bisector of s_i and s_j , denoted by $B(s_i, s_j)$:

$$B(s_i, s_j) = \partial D(s_i, s_j) = \{x \in R^2 \mid \delta(x, s_i) = \delta(x, s_j)\}.$$

The Voronoi region $v(s_i)$ is the portion of the plane lying in all of the dominances of s_i over the remaining objects in S :

$$v(s_i) = \bigcap_{s_j \in S - \{s_i\}} D(s_i, s_j).$$

If two bisectors $B(p, q)$ and $B(p, r)$ (p, q , and $r \in S$) intersect, the intersection is called a *Voronoi point* and the bisector delimited by two consecutive Voronoi points is called a *Voronoi edge*. Two objects are *neighbours* if they share a common Voronoi edge (possibly extended to infinity). The Voronoi point is at an equal distance to at least three objects and is therefore the circumcentre of a circumcircle defined by these objects.

If S is a point set, $v(p)$ for $p \in S$ is a (possibly unbounded) convex region intersected by all half-planes containing p and delimited by bisectors between p and other members in S . This nice property does not hold for a general Voronoi region if S contains points and lines. The bisector between a point and a line is no longer a straight line but a locus of points bisecting them. While the definition of the dominance $D(p, q)$ still holds, the bound $B(p, q)$ may become a parabola. Figure 1 shows the Voronoi diagram (grey) of points and lines (dark).

3.1 The Delaunay triangulation

The Voronoi edge separating objects p and q indicates that p, q are neighbours. A straight line linking p, q can represent this relationship. For e Voronoi edges we can have the same number of straight lines. Each Voronoi

point corresponds to one triangle formed by three objects on the circumcircle¹. The union of these triangles is the dual structure of the Voronoi diagram, the Delaunay triangulation $D(S)$. Each straight line linking two objects is called a *Delaunay edge*. For $S = \{\text{points, line segments}\}$, the corresponding Delaunay edges between a line and a point can be graphically implemented by linking the point and the middle of the line segment. Figure 2 illustrates a Delaunay triangulation (dotted) for some lines and points (solid).



Figure 1 A Voronoi diagram of points and lines.

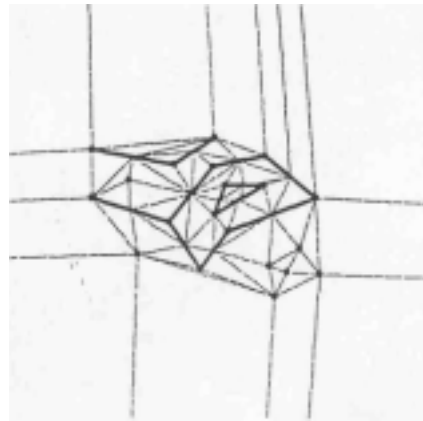


Figure 2 A Delaunay triangulation of points and lines.

Unlike its counterpart for a point set, the Delaunay triangulation for a set of points and lines is not necessarily *equiangular* [Sibson 1977]. It simply becomes a graph representing adjacency relationships between objects. Nevertheless, the circumcircle tangent to three vertex objects is object-free, i.e. none of the given objects in S is contained in its interior.

By saying so we assume that S is in general position, i.e. there are no cocircular cases and not all objects in S are collinear. These degenerate cases can be handled by some additional rules.

3.2 The memory space related problems

The *memory space* is referred to as the computer memory allocated to store objects and their topological structures. In our case, the topological structure is the Delaunay triangulation. Two problems can be observed in applying the dynamic Voronoi diagram as a fundamental data model for a GIS, within the framework of incremental construction. Firstly, compared with other geometric and topological data models used in most current GISs, the Voronoi diagram consumes more memory space. In addition to storing objects and their coordinates, the structure must maintain the triangle network. For a complicated polygonal map composed of hundreds of thousands of short lines, the size of the triangulation could be extremely large. Secondly, since the Voronoi diagram is generated incrementally, the input of objects does not necessarily follow a particular order. This may result in an instance of the data structure with poor spatial indexing. That is, objects and triangles that are close in space may be stored far apart in memory. This makes the ordering of the space according to some pattern impossible. It follows that no matter how big a map is, all the object and topological data must be loaded in memory in order to ensure the presence of the neighbourhood around any area of interest.

Large memory use with unnecessarily detailed objects and their relationships lowers the overall performance of the system and hinders operators from concentrating on more important properties of a data set. Managing spatial data in a non-splittable fashion fails to meet the requirements for modern data structures which must be dynamic, hierarchical, with varying levels of detail, and divisible into disk pages [van Oosterom 1993]. If a part of the Voronoi diagram of detailed objects can be taken away from the organizing structure and represented by some compound objects, the number of objects present in memory would be smaller. Again, if the separated compound object preserves its original topological partitions of the space, it can be loaded into memory at any time as an independent workspace. The whole space can then be managed seamlessly as if there is no partition. Furthermore, if scattered indices to objects and their local topological relationships in a memory subspace can be aggregated, they can be more efficiently managed and retrieved. These expectations can be achieved by introducing the concept of condensed spatial objects.

4. The spatial condensation technique

The spatial object condensation technique [Yang and Gold 1995] has two aspects. It firstly partitions the Voronoi diagram into subsets such that the topological structures for subsets of objects in the interior of partitioned subspaces are independent of each other. Topological relationships between partitioned subspaces are preserved in the mutual partitioning boundaries. Secondly, it separates partitioned subsets from the original memory space into different memory spaces which can be stored on separate disk pages.

During this process a transformation is applied to spatial objects and Delaunay triangles such that if they are close in space, they will be stored close by in the memory. This eliminates large gaps between indices of spatially contiguous triangles and objects in the original memory space.

The general procedure of condensing a spatial object starts with a predefined partitioning boundary around a cluster of primitive spatial objects, some of which form closed polygons. The partitioning boundaries are not necessarily required to be simple loops. They can be complex loops with other lines incident to them. It is assumed that there exists a heuristic approach that can identify desired clusters with closed boundaries to be partitioned. Besides line objects, map frames as well as constrained Delaunay edges also constitute boundaries for partitioning thematic polygons, map sheets, and clusters of points.

The difficulty of partitioning a subset of topological structures lies in its computer implementation. Most topological data structures explicitly encode certain adjacent or incident relationships of one object to others. For example, the DCEL based data structures contain boundary and co-boundary information. Each edge structure encodes polygons on both sides. This is also true for the Delaunay triangulation. In our implementation, indices of adjacent triangles are encoded in each triangle. Especially, two adjacent triangles, sharing a common edge collinear with the partitioning boundary, contain indices referring to triangles in another subspace. Two subspaces are therefore topologically linked through the two *bordering triangles*. In order to partition a topological structure into two independent subsets, the topological linkage has to be cut off. Since a bordering triangle is in the interior of a subspace, it is not stable in a dynamic environment. *Critical triangles* inside a partitioning boundary around both subspaces are developed to prevent triangles in the interior of a partitioned subspace from having indices referring to triangles in the exterior of the subspace. Only critical triangles have indices representing triangles in other subsets. Critical triangles will never be changed and are stable. Their indices can be modified so that they represent critical triangles belonging to the same subspace as the boundary, Figure 3 illustrates critical triangles inside a line object boundary in subspace S_i . The arrows depict indices referring triangles in the exterior of S_i . Figure 4 shows the result of modifying these indices to refer to triangles in S_i .

The next step of the condensation technique is to identify objects and their topological structure in one subspace. Once identified, they can be written into a newly allocated memory space and later removed from the original one. Unfortunately, due to an incremental construction, indices for one subset of the topological structure may not be contiguous. This makes sequential access to the subset impossible. A spatial traversal has to be employed. In our implementation, we use the flood fill algorithm to traverse the topological space in a partial order. For each triangle and object visited, instead of copying its original index to the new memory space, a new consecutive index number is assigned to it. This eliminates large gaps in memory space between two spatially adjacent objects.

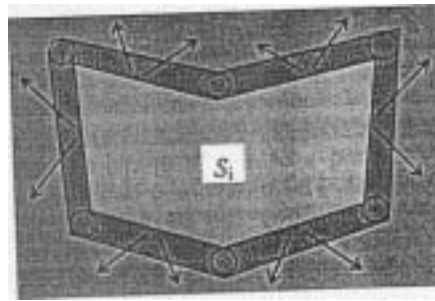


Figure 3 Indices In critical triangles referring to triangles in the exterior of S_i .

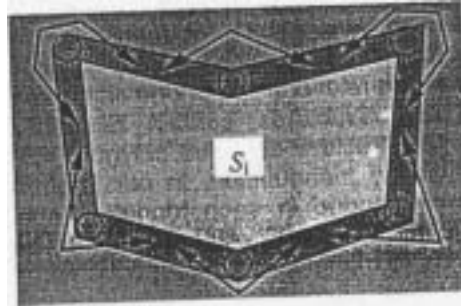


Figure 4 Modifying indices to refer to triangles on the boundary of S_i .

As an example, five clusters of complex objects are partitioned (Figure 5) from the map in Figure 1. Each partitioned object is a map of the same type as its ancestor. It can be worked with independently. Even smaller objects can be created from it. As is illustrated in Figure 6, three more objects are created from the object labelled "C". It can be shown that map modification and spatial queries in the interior of a partitioned subspace X_i are *self contained*. That is, no subsets of the topological structure in the exterior of X_i have to be involved in these operations.

5. The VMO-tree

The spatial object condensation creates complex polygonal objects. Each object resides in a separate memory space and can be saved on a storage medium. The portion of its memory space in the original map will then be released. In this section, we describe the relationship between the new object and the one from which it is created. A new class named *VMO-tree* (Voronoi based Map Object Tree) is introduced. The VMO-tree is a graph representation of the relationships of map objects. The prime roles of the VMO-tree are managing complex objects and serving as a platform for transactions upon these objects.

5.1 The data model for the VMO-tree class

The objects created by condensation naturally form a hierarchical relationship where the object from which a partition is made is the *parent* and the newly partitioned object is the *child*. A parent can have a number of children and each child itself can have children. When a complex polygon is condensed, the partitioning boundary is duplicated in both parent and child objects. At the parent level, the subspace $Y \subset R^2$ enclosed by the boundary is a simple polygon with all detail suppressed. Topologically, the subspace Y is equivalent to a closed disc while the subspace $X \subset R^2$, enclosing Y , is open. If we attach a label to the internal boundary of the parent, the enclosed subspace represents the condensed object. At the child level, the detail is expanded and the external side of the boundary has the same label as that of its parent.

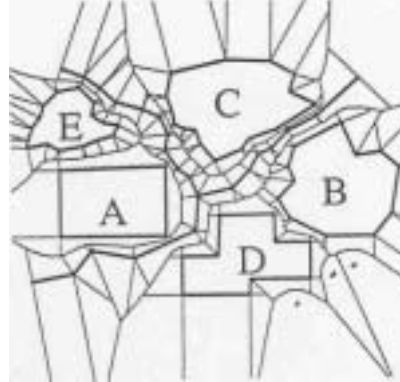


Figure 5 A partition of the map In Figure 1.



Figure 6 A partition of object C in Figure 5.

In what follows, four aspects of the data model for the VMO-tree class will be identified. They are namely: components of the class; data structures representing components and their relationships; operations on components; and constraints over operations.

5.1.1 The Components of the VMO-Tree Class

Definition 1: The *outline image* is the partitioning boundary left on the subspace X from which the space partitioning occurs. The *object outline* is the partitioning boundary duplicated on the partitioned subspace V whose extent is confined in the enclosure of Y .

Definition 2: The class *map* is a set $M = (P, L, R, A)$, where P is a class of points, L a class of lines including outline images of condensed objects, R a class of regions with closed boundaries, and A a class of condensed objects. Within the four subclasses, the point class is the most primitive one, the other three classes are aggregated from relatively simpler ones. For example, the compositions of R and A are $R = (P, L, R_s)$ with R_s denoting smaller regions covered by R , and $A = (P, L, R, A_s)$ with A_s denoting smaller condensed objects in A , respectively. Since A and M have the same composition of subclasses, they belong to the same map class. Therefore, a condensed object is also called a map in a recursive way. Both R and A represent areal objects. They differ in that the components in an instance $r \in R$ are present in a map instance containing r , while the components in an instance $a \in A$ are dropped from the memory space for a map instance. Only the outline image of a is preserved in the memory of a map instance. A map may not have all classes of objects presented. The following combinations are valid cases: 1) $M = (P, L, R, \phi) = (P, L, R)$, 2) $M = (P, \phi, \phi, \phi) = (P)$, 3) $M = (P, L, \phi, \phi) = (P, L)$, and 4) $M = (\phi, \phi, \phi, \phi) = \phi$. A map with no components is called a *null map or null object*.

Definition 3: An *object embedding* of a map object is a layout of its composing objects in R^2 Euclidean space. For a single map, the object embedding disallows general intersection between objects, following the concept of the single *valued vector map* [Molenaar 1989]. This ensures the planarity of the map. The object embedding can be dynamic. That is, the components of the map object can be changed. However, any change must be confined to the extent of the object outline.

Definition 4: A *topological embedding* of a map object is a realization of topological relationships over the object embedding of the map. The topological relationships are specified in the topological data structures. Note that the topological embedding of a map object does not apply to the interior of condensed objects in the map because the object embeddings of these objects are suppressed. However, the outline image of a condensed object is topologically embedded in the map object. The topological embedding is also dynamic, due to the dynamic property of the object embedding.

5.1.2 The VMO-Tree Structure

Definition 5: A VMO-tree is a graph $G = (N, E)$ with node set N and edge set E . The node set N is of the map class M and each node has a label. For any pair $\langle x, y \rangle \in N$, an edge $e \in E$ exists between x and y if the object embedding of x contains the outline image of y and y has both object and topological embeddings, and both of them are confined in its object outline duplication. The node x is called a parent, and the node y is a child.

The construction of a VMO-tree can be either top-down, bottom-up, or a combination of both. There are two scenarios when the top-down strategy is used. In either case, the root node is first created. One scenario of the top-down approach condenses a null object for each smaller area such that the whole study region is completely partitioned by the outline images of these null objects. These null objects are added into the VMO-tree at level 1. Each null object can then be worked on independently and even smaller objects are added in the tree at lower levels (Figure 7).

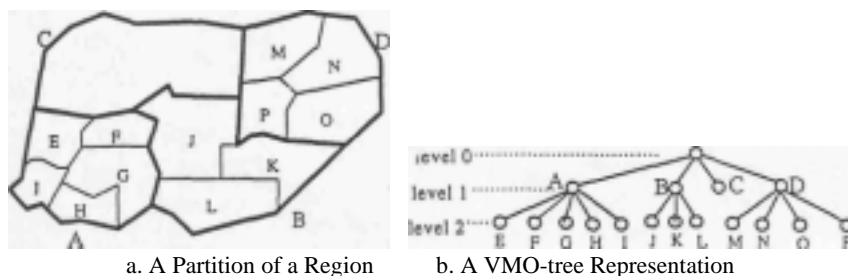


Figure 7 The top-down partition and its VMO-tree.

Another scenario of the top-down construction works with the root map and adds components in it. Some heuristic approach can be taken to aid the decision on when and where a cluster of components can be condensed. Whenever a new object is dynamically condensed, the node representing the object is inserted into the tree.

The bottom-up strategy works in a reverse way where the smallest distinct objects are first composed as maps separately. There are also two ways of constructing the tree. The first one creates the root and inserts into it all the object nodes from the first step. These objects will be at level 1. This results in a flat tree where all children have the root as the parent. Then a generation process takes place which aggregates smaller objects into bigger ones. When a bigger object n aggregated from m smaller objects at level i is formed, the object n becomes the parent of the m objects. Note that the aggregation happens at level M . The new node representing object n will therefore be at level i and the m children are dropped to level $i + 1$. The m pointers of the original parent at level $i - 1$ will be released and the tree becomes narrower by $m - 1$ branches at level i . The generation process can be performed heuristically.

The second method of bottom-up tree construction generalizes more complex objects without knowledge of the root. When an aggregate object is created, it serves temporarily as a root of a smaller tree. A forest can exist during the whole process. A bigger tree is composed by combining a number of smaller trees.

From a cartographer's point of view, the top-down approach works from smaller scale, larger sized objects towards larger scale, smaller objects. The

deeper down in the VMO-tree, the more details an object can expose. On the contrary, the bottom-up approach works from larger scale, smaller sized objects towards smaller scale, larger objects. The higher a node is in the VMO-tree, the more abstract it becomes.

The combined construction repeatedly uses partition and aggregation processes. The description is straightforward and is omitted from this paper. Inserting an existing condensed object into the VMO-tree and aggregating smaller condensed objects into a bigger one are compound operations each of which takes several steps. The major steps for insertion duplicate the outline image of the new object in the proper map object represented by the parent node and attach the new child object to the tree. For aggregating, the outline of the new object can have the same extent as that defined by the union of outlines of the smaller objects, or a bigger one, depending on the heuristic decision.

5.1.3 Constraints upon the VMO-Tree Construction The VMO-tree is a topological structure mapped from partitioning of another topological space. In order to make VMO-tree well defined, the constraints guiding partitions must be defined. Let $A = \{a_1, a_2, \dots, a_n\}$ be a finite set of condensed objects contained in a map object. A is the parent and a_i ($i = 1, 2, \dots, n$) are children. These constraints state:

Constraint 1: The number of condensed objects contained in one map object corresponds to the number of children under the node representing the map object.

Constraint 2: For $a_i, a_j \in A$, ($i \neq j$), $a_i^0 \cap a_j^0 = \emptyset$. That is, interior spaces of any two distinct condensed child objects must be disjoint at the same level. Overlapped or partially overlapped subspaces enclosed by outline images are disallowed. The non-empty intersection of distinct member of A may only happens on the outline images.

Constraint 3: $\bigcup_{a \in A} \text{outline}(a) \subset c \text{ outline}(A)$. That is, the union of outline images of

all children must be confined in the outline of the parent object.

5.2 Operations over the VMO-Tree

There are five categories of operations over the VMO-tree: building operations; set-theoretic operations; spatial queries; the operations linking object attributes; and operations for network communications. The first three categories are spatial operations. More formal and detail discussions about some of these operations can be found in Worboys [1992]. We are currently developing a prototype of the VMO-tree classes and the format for their operations. Detailed descriptions are in preparation. For now, the functionality of each operation group is list below.

The building operations insert and delete nodes from the tree. They can be activated either interactively from the graphics interface or from behind the scene where only the tree and the node concerned are given. In either case, the outline image of the inserted node must be embedded in the space

represented by the tree, or the embedded outline image must be removed from the space when deleting. The building operations also contain basic editing tools which allow modification over components of a node object,

The set-theoretic operations work on either single or multiple themes. These include calculating unions, intersections, and complements, with respect to given themes and objects. New instances of the VMO-tree can be generated by a set operation. An example of this calculates the intersection of two objects, each from a VMO-tree instance. A third instance of the VMO-tree can result if the calculation results in a non-empty subset of objects.

Spatial queries can be greatly facilitated by the VMO-tree, which provides a fast search along edges of the tree, and the topological embedding within each spatial object, which enables navigation and searches over components of a map object. This ability to search objects vertically and horizontally resembles that of the B⁺ tree.

How to store and manage attribute data of spatial objects is still an open question. The majority of GISs implementing a relational model typically adopt a hybrid architecture. Due to the special characteristics of spatial data, future options have two major directions: extending relational systems with specialized functionality for geo-information; or moving to radically new approaches, such as object-oriented systems [Worboys 1994]. The option of adding specialized functions to relational databases not only can speed up the development, but also conforms to the reality that most attribute data are maintained by relational databases.

The VMO-tree manages map objects, some of which may reside in or be accessed from remote sites. The object networking functions have to take care of the data security, integrity and concurrency controls, and dynamic communications. A client-server architecture can be designed to allow node objects of the VMO-tree to be linked to client applications.

6, The interface for the VMO-tree operations

One of the objectives of the spatial data handling system based on the VMO-tree is that any operations on spatial objects should be made easy. This can be achieved by designing a well defined visual interface. For example, the VMO-tree can be represented by a visual component, following the visual programming paradigm. The VMO-tree component responds to some events triggered by users, and the methods built in the VMO-tree class can be activated by events. Figures 8, 9 and 10 illustrate a prototype interface from which some typical operations can be facilitated.

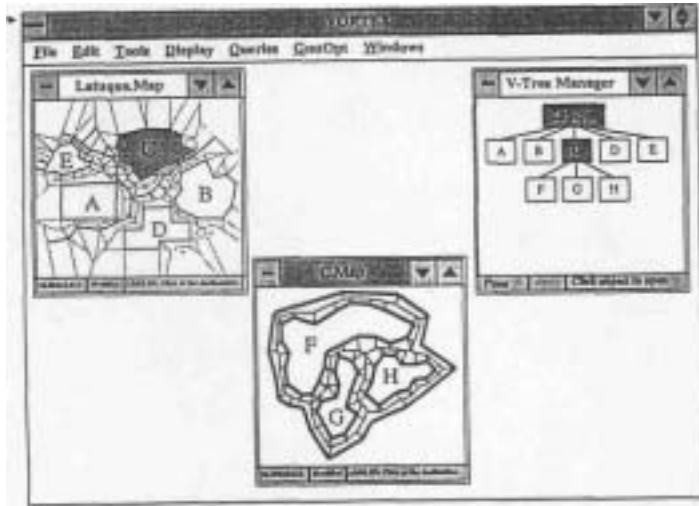


Figure 8 Accessing objects in the VMO-tree.

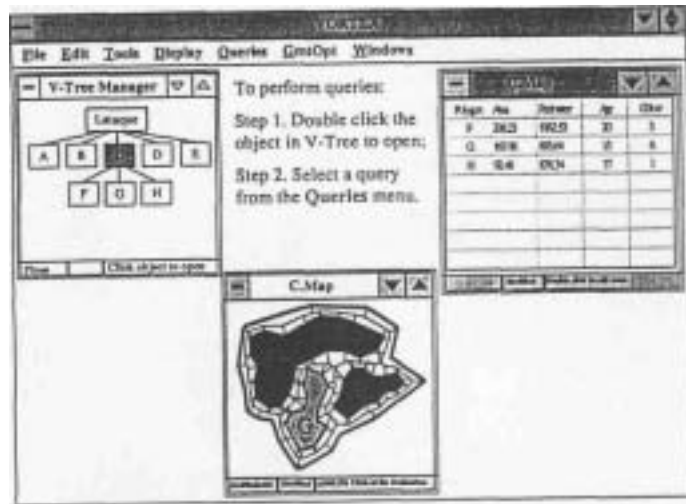


Figure 9 Viewing attributes of the VMO-tree objects.

7. Conclusions and future work

In this paper, an object-oriented spatial data model for managing maps is proposed. The objects are condensed from the primitive Voronoi diagram, each having a separate memory subspace corresponding to disk pages. Condensed objects are managed by an acyclic VMO-tree which is one-one correspondent to the map partitions. The VMO-tree can be built by either top-down subdivision, or bottom-up aggregation of map objects, or a combination of the two. The construction constraints ensures that the correspondent tree is well defined.

The current specifications of the data model are preliminary. The authors are developing detailed descriptions of relationships of spatial components, the operations, the boundary conditions for dynamic modifications, and the visual interface of the data model. The attribute data storage and networking issues be further studied in the near future.

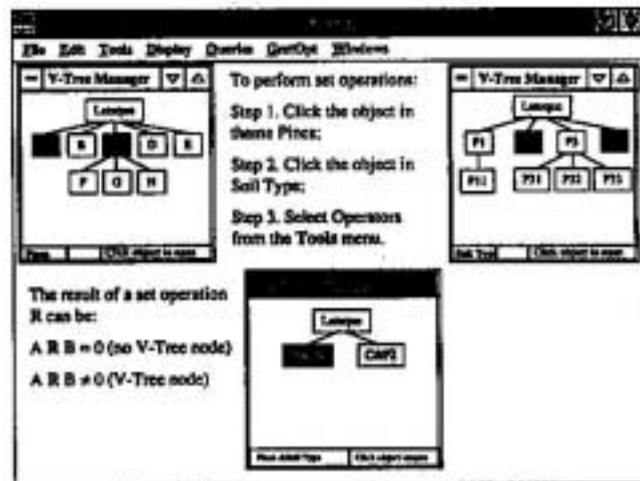


Figure 10 Set-theoretic operations through the VMO-tree.

Acknowledgements

The authors acknowledge the support by the Natural Sciences and Engineering Research

Council of Canada (NSERC) and the Association de l'Industrie Forestiere du Quebec (AIFQ).

References

- Ballard, D.H. 1981. Strip Trees: A Hierarchical Representation for Curves. Communications of the ACM, 24(5), 310-21.
- Bentley, J.L. 1975. Multidimensional Binary Search Trees Used for Associative Searching. Communications of the ACM, 18(9), 509-17.
- Bertolotto, M. and L. De Floriani 1994. Multiresolution Topological Maps. in Molenaar, M. and S. de Hoop (eds.), Advanced Geographic Data Modelling: Spatial Data Modelling and Query Languages for 2D and 3D Applications. Publications on Geodesy, New Series No. 40, Delft, Netherlands, 179-90.
- Faloutsos, C., T. Sillis, and N. Roussopoulos 1987. Analysis of Object Oriented Spatial Access Methods. ACM SIGMOD, 16(3), 426-39.
- Frank, A. and R. Ban-era 1989. The Field-tree: A Data Structure for Geographic Information System, in Symp. on the Design and Implementation of Large Spatial Databases, Santa Barbara, California, Berlin, Springer-Verlag, 29-44.
- Freeston, M.W. 1987. The BANG File: A New Kind of Grid File. in Proc. ACM SIGMOD Conf., San Francisco, California, 260-69.
- Gold, C.M. 1990. Spatial Data Structures: the Extension from One to Two Dimensions, in: L.F. Pau (ed.), Mapping and Spatial Modelling for Navigation, NATO ASI Series, F, No. 65, Springer, Berlin, 11-39.
- Gold, C.M., P.P. Remmele, and T. Roos 1995. Voronoi Diagrams of Line Segments Made Easy. In Proc. 7th Canadian Conference on Computational Geometry, Québec Canada, 223-28.

- Günther, O. 1988. Efficient Structures for Geometric Data Management. Lecture Notes Computer Science No. 337. Springer-Verlag, Berlin.
- Günther, O. and E. Wong, 1989. The Arc Tree: An Approximation Scheme to Represent Arbitrary Curved Shapes. In Litwin, W. and H.-J. Schek (eds.), Foundations of Data Organization and Algorithms. Paris, France, Lecture Notes in Computer Science No. 367, Springer-Verlag, 354-70.
- Guttman, A. 1984. R-Trees: A Dynamic Index Structure for Spatial Searching. ACM, SIGMOM, 13,47-57.
- Mark, D.M. and M.F. Goodchild 1986. On the Ordering of Two-Dimensional Space Introduction and Relation to Tesseral Principles, in Diaz, B. and S.B. M. Bell (eds.) Proc. of the Tesseral Workshop No. 2, (Swindon: Natural Environment Research Council), 179-92.
- Matsuyama, T., L.V. Hao, and M. Nagao 1984. A File Organization for Geographic Information Systems Based on Spatial Proximity. Computer Vision, Graphics and Image Processing, 26, 303-18.
- Molenaar, M. 1989. Single Valued Vector Maps - A Concept in Geographic Information 3 Systems. Geo-Informationssysteme, (2)1,18-26.
- Morton, G.M. 1966. A Computer Oriented Geodetic Data Base, and a New Technique in File Structuring. Unpublished report, IBM, Canada Ltd
- Nievergelt, J., H. Hinterberger and K.C. Sevcik 1984. The Grid File: An Adaptable, Symmetric Multikey File Structure. ACM Trans. on Database Systems, 9(1), 38-71.
- Preparata, F. and M. Shamos 1985. Computational Geometry: An Introduction. Springer-Verlag, New York.
- Roos, T. 1991. Dynamic Voronoi Diagrams. PhD Thesis, Universitat Wurzburg, Switzerland.
- Rosenberg, J.B. 1985. Geographical Data Structures Compared: A Study of Data Structures Supporting Region Queries. IEEE Trans. on Computer Aided Design, 4(1), 53-67.
- Samet, H. 1984. The Quadtree and Related Hierarchical Data Structures. Computing Surveys, 16(2), 187-260.
- Sibson, R. 1977. Locally Equiangular Triangulations. Comput. J. 21, 243-45.
- Tamminen, M. 1983. Performance Analysis of Cell Based Geometric File Organizations Computer Vision, Graphics, and Image Processing, 24(2), 160-81.
- van Oosterom, P. 1993. Reactive Data Structures for Geographic Information System Oxford University Press.
- Worboys, M.F. 1994. Object-Oriented Approaches to Geo-Reference Information. Int. J. Geographic Information Systems, (8)4, 385-99.
- Worboys, M.F. and P. Bofakos 1993. A Canonical Model for a Class of Areal Spatial Objects. in Abel, D. and Ooi, B. C. (eds.), Advances in Spatial Databases, Proc. of 3rd International Symposium, SSD'93, Lecture Notes in Computer Science No. 692, Springer-Verlag, 36-52.
- Worboys, M.F. 1992. A Generic Model for Planar Geographic Objects. Int. J. Geographic Information Systems, (6)5, 353-72.
- Yang, W. and C.M. Gold 1995. Dynamic Spatial Object Condensation Based on the Voronoi Diagram. In Proc. 4th International Symposium of LIESMARS, Wuhan, China, 134-45,

Contact address

Weiping Yang and Christopher Gold
 Chaire industrielle en geomatique
 Centre de recherche en geomatique
 Universite Laval
 Quebec, Qc
 Canada G1K7P4
 E-mail: weiping@gmt.ulav.ca