AN OBJECT-BASED DYNAMIC SPATIAL MODEL, AND ITS APPLICATION IN THE DEVELOPMENT OF A USER-FRIENDLY DIGITIZING SYSTEM.

Christopher M. Gold
Dept. des Sciences Géodésiques et de Télédétection
Université Laval
Ste-Foy, Quebec, Canada G1K 7P4
Tel:(418)656-3308  Fax:(418)656-7411  E-Mail: CGOLD@LAVALVM1

Abstract

A possible set of requirements for a future GIS spatial data structure is that it should have no topological gaps in it at any time, that it should be locally updatable and hence permit real-time updating, editing and querying, that full object insertion and deletion should be supported (i.e. dynamism), that a primitive object on the map should equal one object in the data structure, that objects should be capable of being moved within space (and, by implication, time) while preserving topology, and that the underlying methods need not be restricted to only two dimensions. Such a system would thus be capable of being used interactively, either for data entry and editing, or for “what-if” simulation. As a preliminary platform for the concepts, a user-friendly digitizing system is being implemented that permits a high level of user interaction with the growing map, based on a dynamic spatial data structure - the Voronoi diagram.

To most users of geographic information systems the underlying spatial data structure linking the parts of the map together, along with the way the system designers imagined “space” to be constructed, appear to be of little direct relevance. Nevertheless, many of the apparently incomprehensible bugs/features of a GIS are directly related to the underlying spatial model used. Despite the fact that there may be possible “work-arounds” of the underlying limitations, this does not excuse a lack of understanding of the properties of a chosen spatial data structure and the underlying perception of space that was used in it (the spatial model). In this paper the importance of the spatial data structure being dynamic is emphasized, using the two-dimensional digitizing process as an example. Further discussion examines the benefits of the dynamic Voronoi spatial model for single-coverage problems, and then touches on multi-layer overlay processes and the extension to higher dimensions.

Introduction

Various workers (Peuquet, 1984, Gold, 1990b and others) have explored some of the assumptions involved in the raster and vector models of space used in most GISs. The author’s work has led him to study in detail a conceptually appealing alternative - the Voronoi diagram. Attractions include full dynamism, a space-covering tiling, an object-based approach and the simplification of various GIS algorithms.

A GIS based on such a spatial model would be capable of true interactive use. This primarily consists of “what-if?” queries, where a modification of the spatial component of the data would be immediately available for analysis. Many kinds of simulation fail in this category - both where a variety of modifications are made manually and evaluated (e.g. landscape architecture) or else where many alternatives are tested automatically by the
computer program (e.g. network analysis or forest harvesting). One basic application that falls in this category is, surprisingly enough, a fully-interactive topological digitizing system. Such a system does indeed primarily consist of what-if queries: if the line segment is good then accept it; if not then return a message or perform some corrective action.

The first digitizing systems were developed completely off-line, recording the coordinates and button events directly onto magnetic tape. This made map digitizing extraordinarily difficult, as there was no visible record of which lines had been entered. The development of independent desk-top computers permitted a form of user interaction. Here the line segments entered could be verified, and displayed on a screen. Nevertheless, with a few honourable exceptions no “simulation” was possible - that is, a line could not be entered and the effect on the data structure immediately examined. Full interaction was not available. This necessitated the “batch” mentality towards map digitizing that is still ingrained in many organizations. The existence of a dynamic spatial data structure could change this.

For a digitizing system to be fully interactive it must be capable of attempting to insert the operator’s requested line segment (or point) into the current topological network. If there are no problems, then fine. Otherwise (usually because of a collision or near collision with an existing map object) some form of response is required. If the proposed line segment crosses an existing segment, for example, either an intersection node must be formed or else the proposed segment must be rejected. If the cursor or “pen” approaches within some tolerance of another line segment or end-point, perhaps the two objects should be snapped together. If two objects are snapped together, perhaps one or else two polygons have been completed - does the user wish to confirm this, and name them?

In order to provide a more user-friendly digitizing system of this type there must be an underlying dynamic spatial data structure. “Dynamic”, when not merely a salesman’s pitch, has two related meanings here. The word is often related to movement, suggesting that the data structure can be maintained as objects move in space - which is true. A more precise meaning comes from computing science, where a dynamic index structure (e.g. a B-tree) may be fully maintained whatever transactions (object insertions or deletions) take place. This is often not true: deletions, in particular, may not be possible without rebuilding a significant portion of the whole tree. If the structure is to be dynamic, and respond rapidly to any transactions, the updating procedure must be well defined and local in scope. The relation of this concept to dynamic as in spatial movement, and with real-time maintenance of a spatial data structure, is clear.

**Spatial DataStructures**

There are two common spatial models: the raster, where space-filling square tiles are given attributes associating them with map objects, and the vector, where the primary objects are line segments. Why are most (vector) spatial data structures not dynamic? Surely it would be desirable if they were? The answer involves the underlying spatial model.

A common assumption in the design of a vector GIS is that boundaries (arcs) initially may be created unrelated to each other. In order to provide the adjacency linkages
that must be maintained for subsequent queries, they terminate in nodes, which may connect to other boundaries. If the initial situation is a set of digitized lines, the usual way to find if and where they could connect to other lines and nodes is to test all the possible intersections of line segments. This can be a formidable task, involving much computing time, even without the extensive cleanup required by limited-precision digitizing: e.g. snapping to tolerances; dangling line ends: near-parallel lines that cross. Many algorithms used to create the required linkages treat it as a destructive task: i.e. “yesterday’s” map is broken into its original line segments, the updates are added and the map re-built by intersection testing. This approach is by its very nature a global operation hence by no means instantaneous, and thus not dynamic.

Even without further analysis, it appears clear that the “lines are connected if we can find an intersection” approach would have great difficulty in keeping up with even a slow operator if global intersection algorithms are used. This leaves two alternatives: either the line intersection approach must become a local, and thus rapid and dynamic, method; or the line intersection model must be re-thought (Gold, 1992a).

**The Voronoi Spatial Data Model**

This model was developed on the underlying assumption that map objects were not necessarily connected -- the initial problem was interpolation from arbitrarily distributed data (Gold, 1989). Spatial ordering and neighbourhood relationships were just as necessary when selecting “adjacent” data points for averaging as in the case of supposedly-connected polygons. How can an order or adjacency be imposed on unconnected objects?

One solution, to this and a growing number of related problems, turned out to be the Voronoi diagram, which is a tiling of the plane that assigns each possible map location to its nearest data object. The result, for point data, is a set of convex polygons (“bubbles”), each with a data point as its nucleus (Figure 1). The common boundaries between bubbles indicate when they are adjacent, and this set of adjacencies forms the Delaunay triangulation, which is frequently seen in TIN terrain models. From its very definition, this triangulation is unique for any particular data set (except in the degenerate case of more than three points on an empty circle). Since the triangulation is the network of all adjacency relationships, where adjacency means that two “bubbles” have a common boundary, the Delaunay triangulation is the spatial data structure for the Voronoi spatial model. Each map object has an associated bubble, or tile, and the collection of these bubbles tile the map area completely.

The definition of “nearest” for assigning any location on the map to a specific map object or tile is not restricted to

![Figure 1 The Voronoi regions of a simple point data set, showing the Delaunay triangulation](image)
point data: it is completely general. (Aurenhammer, 1991) reviews Voronoi methods well. This writer’s work has concentrated on dynamic Voronoi diagrams of points plus line segments. To achieve this, however, the concept of “navigating” a map must be examined.

Map Navigation

If, at all times, the spatial adjacency structure of the map is to be retained, then it should be possible to take a data point and steer it through the network -- without breaking the network of triangles that express the spatial relationships of the tiles or bubbles making up the map. (The utility of this process will be seen shortly.) If one visualizes the situation as a set of contiguous bubbles, then it is easy to imagine that one of them could be moved as a local process while disarranging only its immediate neighbours. (It is this property that guarantees the purely local update of the Delaunay triangulation.) The details of this process have been described elsewhere (Gold, 1990b, 1992c). Others (Devijver and Dekesel, 1982), (Gowda et al. 1983), (Kao et al. 1991) have described various aspects of dynamic Voronoi diagram maintenance.

If we examine any triangulation, and focus on any particular data point P that we wish to move, there is a set of N immediately adjacent triangles all having P as a vertex, as in Figure 2. (The value of N averages six for a point data set, excluding boundary conditions.) There are therefore N neighbouring vertices to point P. Each adjacent triangle has one immediate exterior neighbouring triangle, which has two vertices in common with the adjacent triangle set, and one new vertex.

When P is moved a small distance, it modifies the shape of the boundaries between its Voronoi region and those of its neighbours. These boundaries are represented as the triangle edges connected to P in the dual triangulation. While modified, all the boundaries still exist, and hence the dual triangulation remains unchanged. For a somewhat larger perturbation of P, however, its Voronoi bubble will either touch another that it did not touch before or else it will separate from another Voronoi bubble to which it was previously adjacent. In the first case, as seen in the figure, two of the bubbles adjacent to P that were adjacent to each other no longer are - but P and a previously exterior bubble Q now touch. In terms of the triangulation, an adjacent/exterior triangle pair has been replaced by two adjacent triangles, simply by switching their common diagonal. In terms of the data points, Q has now become a Voronoi neighbour to P.

The second case, where P separates from a bubble to which

---

**Figure 2** Voronoi regions and triangulation around (a) a moving point P, and (b) P after it enters the circumcircle of the triangle containing point Q.
it was previously adjacent, is a direct reversal of the first case - requiring the switching of two adjacent triangles to form an adjacent/exterior pair. The condition under which the switch takes place is, in the first case, when P moves into the circumcircle of the exterior triangle and, in the second case, when P moves out of the circumcircle of the potential new exterior triangle (which would be made up of three data points adjacent to P) (Gold, 1990a). The circumcircle criterion follows from one definition of the Delaunay triangulation - that all its circumcircles are empty, with no data points in their interior.

Apart from the ability to move a point, a few other simple operations are required. Points are created by splitting an existing point and its associated region (see Figure 3), which is then moved to its final destination. Points are deleted in the reverse process. Lines are drawn by letting the trail of a moving point accumulate all the spatial adjacency relationships held by the point in its travels (see Figure 4), and deleted by having the moving point retrace its path. There is a close resemblance to turtle geometry, as all commands consist of selecting a turtle from one of the existing data points, and then passing it messages to perform the desired operation; both turtle and Voronoi geometry are object-oriented at the lowest level. Figure 5 shows the Voronoi regions of the line segments forming a polygon set, constructed using the Voronoi moving-point approach, together with the equivalent dual triangulation.

The Digitizing Process

All of these operations are local, and rapid. Visualizing the moving point as the cursor of a digitizer, line segments and points may be constructed as they are defined and inserted directly into the data structure. Since the moving point is functionally equivalent to a pen or cursor, including pen up/down equivalents, usage is almost intuitive. The result of the foregoing spatial model is a “bubble map”, with each point or line segment being the nucleus of a tile or bubble.

At any moment in time the “pen” has its own bubble, and hence its own set of neighbours. Examination of this neighbour set

Figure 3 One central polygon P and the associated triangulation (a), and (b) the resulting triangulation when polygon P is split into two.

Figure 4 The point data set, Voronoi regions and triangulation of Figure 1 with a single fine segment added.
permits ready navigation of the “obstacles” already in its path, including previous parts of the current line segment. Thus too-close approach to a neighbouring object may, depending upon the application or situation, trigger an intersection operation, a snapping-to operation or a rejection operation whereby a collision-avoidance mechanism prevents pen coordinates from intersecting an existing object. The Voronoi criterion does not permit two objects to be at the same location, and the triangle switching process must take account of this. Figure 6 shows the generalized levels of degeneracy of the spatial model (Voronoi regions) and the data structure (Delaunay triangles) as a map object C collides with another object B. The dual triangle has two levels of degeneracy as routine triangle-pair switches (= Voronoi edge switches) are made during the collision process. The first level - “Andorra” - shows point B’s region enclosed by the regions of A and C. This is possible in the Voronoi model - e.g. a point between two line segments. The final level of degeneracy - “Lesotho” - after one more switch operation has B entirely surrounded by the region of C. A triangulation may thus be used to represent an island polygon, but islands do not occur in the simple Euclidean unweighted Voronoi diagram. Thus a test to anticipate any triangle having two identical vertices will suffice to detect collision conditions. Some automatic or user-selected procedure may then be invoked to take the appropriate action - evasion, cancellation or connection.

Figure 5 Triangle degeneracy levels during a collision, and the implied region topology. Triangle ordering is shown at the right.
Immediate response to potential collisions would thus permit immediate updating of data entry errors - e.g. unwanted line crossings and dangles. Another type of error - gaps, formed when a line segment is terminated just before it reaches another object, is handled in the previously described navigation process by checking the neighbours to the moving point, or “pen”. If the pen moves to within some tolerance of another object the new line can be “snapped” to that object, completing the desired link. Automatic polygon checking can then be performed if desired. This would certainly speed up the normal digitizing process, as the human preference is to complete a portion of a map while one is thinking about, and understands, that region. Temporal skipping (waiting for the rebuild operation) is even worse than spatial skipping (looking back and forth between map, screen, keyboard, etc.).

The underlying Voronoi spatial model thus translates to a new spatial data structure, which has properties distinctly different from earlier spatial models. This leads to a prototype application (a digitizing system) which has different properties from previous systems, and thus needs a different program architecture.

System Operation

Rather than give a formal specification of the system components, in order to save space we will describe the actions following a user action in the prototype digitizing system. A description of the relevant components will be given as they are invoked. It is assumed that the user is in the middle of digitizing a polygon set, and is adding a new line segment. Here the cursor refers to the screen icon controlled by the mouse; and the pen refers to the moving point, existing as an object within the spatial data structure, that follows the cursor. The spatial data structure is a Delaunay triangulation (or, alternatively, one could think of the dual Voronoi diagram) of a single map coverage.

Figure 6 Voronoi regions and Delaunay triangulation of a simple polygon map composed of points and line segments.
The user clicks on the appropriate mouse button, and the x,y coordinates are captured. One button is used to specify that the movement is performed with the pen “up” (i.e. move to a new location), and another specifies that the pen is “down” (i.e. a line segment is to be drawn).

In the first case a lateral search is performed through the network, starting from the last-used triangle or map object, and ending with the closest map object to the cursor’s current location (i.e. the point or line segment whose Voronoi region contains the specified x,y location). If the closest object is a line segment, one of its end-points is used. This is like the select procedure used by many CAD systems. A new point is split from this starting point and moved to the desired x,y location, using the triangle switch procedure.

In the second case the starting point has already been defined, and a new pen is split from the starting point, with a trailing line segment object, like an elastic band, between the two points. The point is moved to the desired x,y location as before - but the line segment, as the locus of the moving pen, remains as a permanent map object.

In each of these cases a point object is first located and then a message passed to it. There are nine possible messages (not all of them useful): three split options, combined with three merge options. If no split is requested the starting point is moved to the destination location; if single split then a new point is split off and moved (the “pen up” case); and if double split then both a point and a line segment are created (the “pen down” case). If no merge is requested then the pen remains at the destination location; if single merge is in effect then the pen is merged with an object at the destination location; and if double merge is in effect then the pen has moved back along an existing line, erasing it, and both the pen and the rudimentary line segment are merged with the other end point of the line segment. Table 1 shows the nine operations and how they are invoked.

<table>
<thead>
<tr>
<th>SPLIT OPTION</th>
<th>MERGE OPTION</th>
<th>ACTION</th>
<th>DESTINATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>split = 0</td>
<td>merge = 0</td>
<td>move point</td>
<td>x,y coords.</td>
</tr>
<tr>
<td>split = 0</td>
<td>merge = 1</td>
<td>delete point</td>
<td>point object</td>
</tr>
<tr>
<td>split = 0</td>
<td>merge = 2</td>
<td>delete line + one end point</td>
<td>line object, gives end point</td>
</tr>
<tr>
<td>split = 1</td>
<td>merge = 0</td>
<td>create point</td>
<td>x,y coords.</td>
</tr>
<tr>
<td>split = 1</td>
<td>merge = 1</td>
<td>‘comet’ (?)</td>
<td>point object</td>
</tr>
<tr>
<td>split = 1</td>
<td>merge = 2</td>
<td>delete line</td>
<td>line object, gives end point</td>
</tr>
<tr>
<td>split = 2</td>
<td>merge = 0</td>
<td>create line</td>
<td>x,y coords.</td>
</tr>
<tr>
<td>split = 2</td>
<td>merge = 1</td>
<td>create/snap line</td>
<td>point object</td>
</tr>
<tr>
<td>split = 2</td>
<td>merge = 2</td>
<td>redraw line (?)</td>
<td>line object, gives end point</td>
</tr>
</tbody>
</table>

Table 1. The nine messages to a Voronoi map object.
As the pen is itself an object in the data structure, it has a Voronoi “bubble” like all other map objects. It therefore has a small number of neighbours that can be checked for potential collisions in real-time as the pen is moved towards the location specified by the cursor. The move operation consists of triangle-pair switches, and the triangle-degeneracy test prevents invalid operations (e.g. two lines crossing without an intersection) by testing if a potential switch is valid. If it is not, a previously-determined remedial action occurs (e.g. ask the user if he wants to generate an intersection and continue; generate the connection and stop; or cancel the last line segment). If the pen movement terminates within a specified tolerance of another map object, the map user may be requested to state if a connection was intended. If so, then zero, one or two new polygons may have been formed. Again, at the user’s specification, the completion of these circuits may be tested and the user prompted as to whether they should be labelled immediately. (They may, of course, simply be intermediate polygons to be subdivided later.) The potential exists to shade or otherwise flag polygons as they are labelled, thus simplifying the validation of the map entry process. This option would not be used when digitizing other data types - road networks, etc. As with newly-completed lines, newly-completed polygons are immediately available for query.

The design of the Voronoi-based user-friendly digitizing system is not yet complete. In particular, the optimal form of user interaction is not yet determined. It is hoped that this paper has shown why - all the implications of a fully interactive spatial data structure are not yet understood. It appears clear that many of our assumptions as to how manual digitizing is performed, and the wider uses of such a data structure, must be reexamined.

Applications

In many cases, the generation of the Voronoi structure is simply the first step of the solution to the spatial problem. For example, the generation of a corridor or buffer zone around some target set of objects is a trivial operation given the map region which is closest to each map object - this is just the Voronoi diagram. The point-in-polygon problem also reduces to determining the Voronoi region containing the pointer (Gold, 1991). Various forms of interpolation are also directly derived from the same diagram (Gold, 1992b). Polygon merge is achieved using the previous digitizing techniques to delete the unwanted boundary, preserving the spatial structure. Polygon overlay may be performed in an incremental fashion using the same basic tools (Gold, 1992c).

Conclusions

While there is insufficient space in a paper of this type to enumerate all the potential benefits of this new approach, a few may be mentioned in conclusion. Because all data structure modifications are incremental, rather than batch, history is preserved between actions - and a temporal log may be kept, permitting the replay (or rewind) of the map building process. Point-in-polygon queries may often be more easily performed on a seamless spatial mesh, especially where several modifications are made in the same neighbourhood (the usual case). Because of the seamless nature of the data structure queries may be made at any stage, assisting in the digitizing and editing process - these queries may be of the nearest-neighbour or set-of-neighbours type, permitting various forms of interpolation (Gold, 1992b). Simulation, where spatial objects may be moved, inserted or
deleted during modelling, is readily achievable with the tools just described, and the approach generalizes to higher dimensions and other metrics.

Acknowledgements

The funding for this research was made possible by the foundation of an Industrial Research Chair in Geomatics, jointly funded by NSERC and the Association de l’Industrie Forestière du Quebec.

References


