ABSTRACT

It is possible to eliminate some of the traditional problems with GIS implementation by changing the model of space used. The Voronoi spatial model has been used as the basis of a set of core GIS or spatial operations, with encouraging results. It has been implemented in a PC environment so as to leave as much flexibility as possible for the designer of any particular application. Examples include polygon dissolve, buffer zone, overlay, interpolation and robot navigation problems.

1. INTRODUCTION: INTERACTIVE DSS AND THE SPATIAL MODEL.

Approaches to spatial data modelling and queries have often been limited by the model of space being used, the batch nature of spatial relationship generation, and the subsequent static view of spatial data. Due to the absence of true graphical interaction of the user with the “map” (e.g. via a cursor or mouse) spatial queries have also been considered to be static operations.

In the effort to achieve a basic interactive spatial decision support system, in this case for forestry applications, it became necessary to re-evaluate the assumptions on which the traditional raster and vector spatial models were based - and in particular the definition and preservation of the adjacency relationships between objects. In brief, the raster model does preserve adjacency relationships through the implicit tiling of the map space, but does not relate directly to objects embedded in that space - thus it is more useful for representing field information, e.g. elevation or temperature. The vector model assumes a polygonal tiling of space, but it must be constructed by the laborious specification of boundaries and nodes. It may thus be used to specify objects, but their spatial relationships are dependent on the detection of line intersections. The vector model thus is not appropriate for spatially unconnected or non-polygonal features, and forms a functional topology only after all errors are corrected and additional containment relations are specified.

2. ADVANTAGES OF THE VORONOI SPATIAL MODEL.

The Voronoi model (see the survey in Aurenhammer 1991) has the complete tiling property of the raster model, but the tiles are based on the proximal region around each object (e.g. a point or a line segment in the current implementation, as in Figure 1). Static methods for constructing the Voronoi diagram for both points and line segments are known (e.g. Fortune, 1986), but fully dynamic local methods are new. (Roos, 1990 has produced point movement algorithms.) Here the unambiguous
definition of spatial adjacency as being a property of two polygons with a common boundary may be preserved under object movement, insertion or deletion, as the Voronoi tessellation is space-filling.

Figure 1. The Voronoi tessellation of a set of points, line segments and polygons.

2.1 Local updating

This gives a fully local means of preserving “topology”, allowing fully interactive map update in response to the user’s graphical actions (see Gold, 1992). It also permits the mixing of many types of data - fully connected polygons, connected hydrography, discrete points and line segments - within the same overlay. Each layer is the adjacency specification of a selected set of objects (expressed as the dual Delaunay triangulation, with pointers between adjacent triangles, and to the objects forming triangle vertices). The preservation of the Voronoi structure as a point moves is achieved by detecting “topological events” (Roos, 1990) and then switching triangle pairs.

2.2 Multiple layers single objects

There are three record types stored in the Voronoi system: coordinate records (containing x,y, etc.); Delaunay triangles (containing pointers to the three adjacent triangles and to the three map objects forming the vertices); and the map objects themselves, which may be points or half-line segments (and have pointers, as required, to the coordinate records and the matching half-line). As there are no direct pointers associated with map objects, (all topology being contained within the Delaunay triangle records) the same object may be inserted into several layers simultaneously. Thus a map layer consists of a pointer to any triangle within a particular mesh that may be followed to the desired location or map object. Different layers, being defined solely as the set of neighbourhood relationships (triangles), may include the same map object. This eliminates object duplication in the attribute database, as well as differing coordinate representations of the same object in various layers - eliminating the consequent sliver polygons when overlaid.
2.3 Representation of fields and objects simultaneously

Because all objects have a proximal zone, the “area-stealing” interpolation model may be used to estimate field values (e.g. elevation) at intermediate locations (see Gold, 1989). These map objects may be of any form, thus permitting precise interpolation between points, line segments or complete polygons, with any specified level of continuity, eliminating the distinction between object and field data. Figure 2 shows interpolation to a line segment.

![Figure 2. Interpolation to a single “high” line segment surrounded by low data points.](image)

2.4 Choice of coordinate systems

The simple (nearest-object) Voronoi diagram may be implemented in various dimensions and metrics. While currently only implemented in two dimensions in the Euclidean metric, the same approach may be used in various others. Work is in progress to implement the operations described here on the sphere.

2.5 The interactive map for interactive decision support, and its interface

The underlying design described here has permitted the development of prototypes of new interactive spatial decision support tools. The dynamic spatial data structure supports incremental construction and editing of the map. This has the particular advantage that in specifying previously-defined objects to connect, for example, it is necessary merely to select them by “pointing” within their Voronoi zones, followed by the appropriate command mapped to the cursor buttons. (This “proximal query” is basic to all interaction, and is equivalent to normal human gestures for specifying objects.)

2.6 Incremental actions and recorded history

A map is built up of a sequence of incremental actions, as with a paper map, and may subsequently be updated in the same fashion by adding or deleting points or line segments. This allows the creation of a historical “tape recording” of the map construction that may be replayed to any desired date. This gives a basic level of temporal query - the whole map may be played forwards or
backwards to a particular date, or a particular point location may be watched and have all state changes reported. Further research is under way to expand these forms of spatio-temporal query.

3. SYSTEM DESIGN

3.1 The object-oriented approach to spatial objects

All construction commands are naturally “object-oriented” in that an object is selected and the appropriate command passed to it as a message. There are nine such commands (three ways of splitting a new point from an old and three merge options, along with point movement - see Gold 1992). For example, we may move an existing point (no split, no merge), or create a new line segment (split a new point and trailing line segment from a pre-existing point, and then move the point to the desired location). Deletion follows the reverse sequence from creation. Figure 3 illustrates the nine basic commands.

One of nine messages is sent to the designated point object:

![Diagram of Voronoi construction commands]

Figure 3. The nine basic Voronoi construction commands.

3.2 Change

Change is treated as the addition or deletion of map objects over time, or the movement of one or more points over time. as with robot simulation or fluid modelling. Lines are created as the locus of a moving point, and point movement in the Voronoi spatial model is treated as a sequence of “topological events”, or triangle-pair switches. Roos (1990) worked with simultaneously moving points, and the same approach may be used, one point at a time, together with the nine split/merge options, to add or delete points or line segments. Polygons, when defined, are specified as the set of Voronoi tiles of points and half-lines that are entirely enclosed by line boundaries.

3.3 Basic spatial queries

The basic query is the local “proximal” query, returning the nearest object any time the cursor is moved. The “neighbour” query returns the set of neighbouring objects to an existing object specified with the proximal query. Another operation inserts the query point into the map, creating its own Voronoi region. This is used primarily for various types of interpolation. A fourth local query is “scan”, where the topological network is searched radially outwards from some starting location, until some terminating condition (e.g. moving out of a search box, or perhaps reaching a linear
boundary). This last is used for polygon labelling, as a polygon is defined as a set of Voronoi zones bounded by line segments. The fifth local query is “trace”; where all Voronoi zones crossed by an imaginary “line segment” are reported.

3.4 Vector GIS emulation

These, along with the construction commands, are used in the emulation of traditional GIS analysis queries involving polygon boundary dissolve, buffer zone generation, and polygon overlay. The first of these is implemented using standard construction commands to delete line segments in a local manner, the second is performed by simple geometric construction within each Voronoi zone of the target object set, and the third by using the “trace” query to draw boundaries of one layer through the second (and then the second layer through the first). The resulting line segments, split and labelled with the appropriate polygon “colours”, are reconstructed in a third Voronoi layer. In the current implementation the “source” and “trace-through” layers may be viewed in two windows, while the resulting overlay is being incrementally constructed in the third.

4. SYSTEM ARCHITECTURE

The central Voronoi module is a closed box, accessible as a dynamic linked library (DLL) from the calling system. Messages are passed to it, in the same fashion as to a database engine, and responses are returned. Input messages, as with a database engine, are either updates or (non-invasive) queries. Responses consist of a message, a list of objects (objects found for queries, objects modified for updates), plus a list of coordinate pairs if necessary. Input commands must be addressed to an existing object in an existing layer, and the full information for the command is contained in one or two “cursors” (an object name and/or a coordinate pair). A macro name must be given in the event that a collision is detected during the update.

Within the Voronoi module are three components that may differ between versions. The memory management system preserves the triangles (“topology”) and map objects (points, half-lines and coordinate pairs) during map update. This may vary internally depending on the assumptions about system memory, paging, etc. The geometry module contains the small number of arithmetic operations necessary with spatial coordinates (determinant, intersection, etc.) Currently this operates in the Euclidean plane, but versions for the sphere are anticipated. and others are possible. The macro list contains the small number of feasible actions in the case of a collision: line intersection and continuation; create node and snap; cancel and undo the previous command, etc.

In the case of a simple query, a message from the calling system results in a simple response: the list of objects found within the spatial conditions (neighbours, within a box, etc.) For a simple map update the same is true, but where a collision is detected (usually when one line attempts to cross another) the Voronoi module must return with the original command only partially completed in order to report the condition and obtain the correct instruction (macro name). Thus there is a loop within the calling system to permit selection of the appropriate response. In addition, if actions, or graphic display, is desired after each topological event, this must also be permitted by a loop within the calling system.

4.1 Platform selection

This architecture is designed to provide the maximum of flexibility (imposing the fewest possible constraints on the potential applications using this spatial server) while preserving the integrity of the server itself. Ongoing experience indicates that new applications are found every few weeks, and great effort has been taken to impose the fewest possible presuppositions on the basic design. The same considerations applied in the selection of the system platform. We wished to work within the PC environment, as we envisaged decision support system applications equivalent to a “spatial
spreadsheet", rather than as a replacement for a large corporate GIS. We wished to preserve the integrity of the spatial server, while permitting ready access to its functionality. This suggested a dynamic linked library (DLL) format, accessible by an interpreter for the development of applications prototypes. This led to the selection of Microsoft’s Visual Basic as the application development environment, not least because the same considerations applied to access to existing commercial products - e.g. attribute databases and spreadsheets. These could be purchased as required, and the appropriate modules integrated with the application in the same fashion as the Voronoi DLL, permitting the customer (e.g. graduate student) to concentrate on the application rather than the graphics interface or low-level tools. Preliminary experiments suggest that this approach is appropriate, and simple applications are currently being developed.

5. COMPLEX OPERATIONS

The local nature of the basic construction commands and queries gives great flexibility in designing higher level queries. As the topology is always complete, robot-navigation methods may be used to steer the cursor away from unwanted collisions, or to follow along a pre-existing boundary judged to be sufficiently close to the trajectory specified (thus eliminating unwanted sliver polygons). Robot-navigation problems themselves may be addressed, with or without operator interaction, and with or without ongoing changes in the map data. An outline of a marine GIS using these properties is suggested in Gold and Condal (1994).

Based on the low level messages described previously, using both the Voronoi and database modules, additional layers of application functions may be developed between these modules and the Visual Basic interface. A few simple examples should suffice.

5.1 Buffer zone generation

The previously-mentioned buffer-zone generation is performed by inserting the target object set into a Voronoi layer, and then performing a query to extract a list of the names of all the map objects. For each map object a neighbour query is made, returning a list of all its neighbouring objects (together with a list of the Voronoi cell nodes). From this description of the cell outline, the intersection of the corridor boundary (at the specified distance from the generating object) may be calculated (if it is present at all). For a line object this will be a linear segment, for a point object it will be a circular arc (broken into small line segments). These segments will be inserted into another Voronoi layer, and the operation reported for the next object in the list. Figure 4 shows a simple buffer zone, and that the point-in-polygon problem may be resolved by searching for the nearest object and identifying its tile attribute.

5.2 Interpolation

For a field-type application, rather than the previous object-based application, the Voronoi layer may be generated as before, composed of points and line segments for which attribute (e.g. elevation) information is available. If an elevation grid is desired, each grid node will be taken in turn and inserted into the Voronoi layer using an “add” command. A list of the neighbours of this cell will be found by then sending a “neighbour” query for this inserted point. For each neighbour in the list another neighbour query returns the neighbours and voronoi vertices around this, as in the previous case, and from this its cell area may be calculated. The query point is now removed with a “delete” command, and the original set of neighbours processed again to re-calculate their cell areas. The difference in the areas is the weighting to be assigned to each neighbour’s elevation value in order to produce the elevation estimate at the grid node (see Gold 1989 and Gold and Roos 1994 for details). In practice a more efficient algorithm may be developed and implemented as a basic Voronoi query, but the method described here is directly achievable with the basic Voronoi command set.
5.3 Polygon overlay

Similar procedures may be developed for polygon overlay, where each line segment from one polygon layer is used in a "trace" command sent to the second polygon layer, which returns with a list of Voronoi cells intersected. This list is used to determine any line intersections and the polygons traversed by the original line segment, which is broken up accordingly and inserted, with the combined set of polygon attributes attached, into the final polygon layer under construction. This process is repeated with each line segment in layer one being traced through layer two, and then layer two segments traced through layer one. The result in the final layer is the combined polygon set. This works for simple overlay cases, but where near-coincidence and related problems occur additional macros are required to allow navigation along existing boundaries within a Voronoi layer. Work is ongoing on this application.

Further complex operations could be described for terrain runoff modelling applications, robot navigation on the basis of both obstacle objects and field-type terrain information, for point cluster analysis and image object recognition, and others that will be reported in detail as they are completed.

6. CONCLUSIONS

The spatial system as currently operational has great flexibility, and may be used as a library of spatial operations, or a "spatial server", to be used by many application shells. Interpolation problems suppose a "field" view of space, with data objects embedded in it. These data objects may be the usual points, but also line segments with specific surface continuity constraints attached - e.g., slope-continuous across contour line input. only surface-continuous across ridge and valley lines, and surface-discontinuous across geological fault lines. Surfaces always honour all data objects and values (Gold and Roos. 199-I). The marine application mentioned above includes Voronoi neighbour

Figure 4. Simple buffer zone about a polygon boundary.
detection to navigate a boat (a moving query point embedded in the Voronoi map) between obstacles, as well as the same embedded query point to interpolate depth directly from bathymetric observations. Features that are displaced over time may also be handled. Applications under development currently include digital elevation modelling, surface runoff modelling, interactive spatial decision support systems, flow simulation, network analysis, and forest map maintenance.

ACKNOWLEDGEMENTS

This research was made possible by the foundation of an Industrial Research Chair In Geomatics applied to Forestry at Laval University, jointly funded by the Natural Sciences and Engineering Research Council of Canada and the Association de l’Industrie Forestiere du Quebec.

REFERENCES