

Environmental mapping made simple

Christopher Gold

Geomatics Research Centre, Laval University,

Québec City, Québec, Canada G1K 7P4.

Christopher.Gold@scg.ulaval.ca

Abstract

In environmental applications such as forestry, the capture of the polygonal structuring of the data is expensive in terms of hardware, software and labour - due largely to difficulties in constructing a topologically complete map. This paper will show that, for many applications, topological structuring and data entry methods may be made quite simple,

The example used is a paper map of forest polygons. The map must usually have the relevant polygons digitized manually, but instead of drawing individual boundaries to be connected later the operator is instructed to draw around the interior of each polygon in turn, generating a set of 'fringe' points labelled with the relevant polygon's number. If the map has no other information on it then it may be scanned instead, and saved as an image. Simple image processing techniques may be used to generate fringe points as in the manual process.

Polygon boundaries are generated by first creating the Voronoi diagram of these fringe points. These triangle edges are examined in a single-pass 'visibility order' to extract the Voronoi cell edges only between fringe points with differing labels. These edges are connected with no topological errors. They may be exported to a GIS as arcs, or preserved internally for polygon labelling and other processing.

Both Voronoi cells and extracted arcs are saved internally using the Quad-Arc, a modification of the Quad-Edge data structure, which is guaranteed to manage connected graphs on the manifold using just two operators. In an object-oriented language the data structure management is only a few lines of code, giving a simple and elegant approach to topology management.

Introduction

This work is based on three observations. The first is that traditional digitizing of polygon maps is complicated both at the user level, and at the level of the computer algorithm. The algorithm is complex because we are trying to connect sets of arcs (strings of x-y points) which exist independently of each other, and which interact in unpredictable ways.

The second observation is that the point Voronoi diagram may be constructed in a simple, robust fashion using something like the well-known incremental algorithm. Its simplicity is due to the completeness of the data structure into which we insert each new point. This allows us to find the enclosing dual Delaunay triangle, add the point and check the neighbouring edges while working in a consistent data structure: all the previous points and edges are embedded in the working surface.

A third observation is that we can extract the relevant edges in the Voronoi diagram to form the desired set of arcs in our polygon map. These would normally require a different data structure from that needed to preserve the Voronoi diagram, but a minor modification of an existing data structure - the Quad-Edge data structure of Guibas and Stolfi [1] - allows us to work with a single data structure, and within any embedding manifold. This greatly simplifies the algorithm development, and allows us to consider most operations as connecting individual objects to the data structure in the embedding manifold, or disconnecting objects from it. It also permits extremely concise coding in an object-oriented environment, although at the expense of somewhat higher storage costs.

Manually digitized, and scanned, forest maps

A typical Quebec forest map may be quite complex, with a large number of different features, potentially obscuring the polygons representing the forest stands. In the case of traditional manual digitizing, the operator is capable of distinguishing between the different features on the map, and of drawing the individual boundaries between pairs of polygons. However, the operation is awkward, as it is the unlabelled boundaries, rather than the labelled polygons, that must be input - and the standard problems of ensuring that each new arc intersects with the other arcs only adds to the number of digitizing errors made.

Gold et al. [2] addressed this problem by having the operator digitize each of the features of interest (the polygons) in some convenient order - in effect “double digitizing” each boundary. A red dot was placed at the centre of the cursor, and the operator was instructed to “roll” it around the interior boundary of each polygon in turn. Interior “fringe” points were generated a short distance apart from each other, each with the same label as the polygon. At the end of the session the points were processed. The first step was to generate the point Voronoi diagram, using a simple incremental algorithm. The resulting dual triangulation was scanned in visibility order (Gold and Cormack, [3]), using a single stack, and the vertices of each triangle checked to see if they were all the same, or different - giving 15 different combinations. This permitted the extraction of the Voronoi boundaries of interest - those between points with differing labels. These could be extracted as complete arcs between nodes during one pass through the triangulation, using one extra stack. These arcs could then be extracted to a traditional GIS for reconstruction of the topological connectedness. This “rapid digitizing” method, a combination of an operational procedure and an appropriate algorithm, was capable of guaranteeing that the topology was error-free, because the original set of Voronoi cells was topologically complete. Of course, input errors were always possible, but the resulting topological structure was always a complete polygon set. The method was perhaps four times faster than traditional arc digitizing, both because it was more intuitive (focusing on the objects of interest, the polygons) and because fewer errors were made. An additional, unforeseen, benefit was that operator training time was drastically reduced.

In Gold [4] this approach was extended to scanned maps. If a clean polygonal map was available (Figure 1), then fringe points could be generated by image analysis techniques (Figure 2). First the black pixels might be thickened by some desired amount; then all pixels with both “black” and “white” neighbours were identified, forming a fringe around the black region. These fringe points were then thinned out. Finally, they were labelled by performing a flood-fill of each region of connected white space - a polygon interior - to give a machine-assigned identifier to each polygon. These fringe points were imported into the “rapid digitizing” program described above. The resulting maps were topologically complete. Figure 2 shows the extracted arcs in between the rows of fringe points. In the scanned map program currently in use by the forest industry (see Gold,

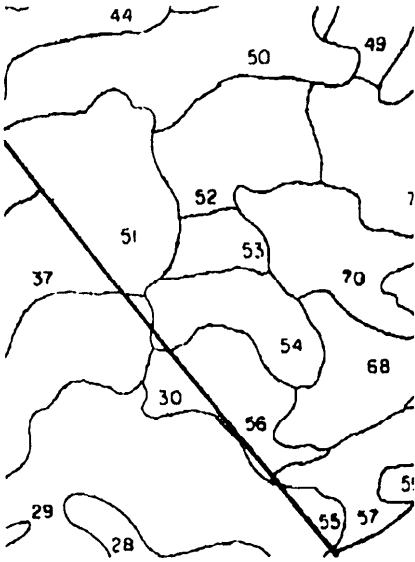


Figure 1 Scanned map image.

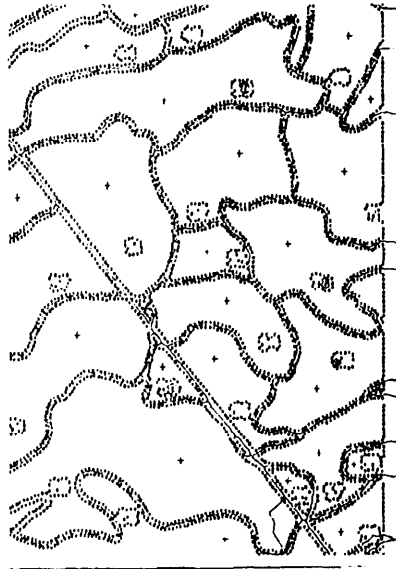


Figure 2 Fringe points and extracted boundaries.

[4]) these are then written to a file, to be imported by a GIS or similar program, Figure 3 shows part of a scanned forest map and Figure 4 shows the map processed and exported to the GIS. It is hard to see the differences. In each of the above cases the Voronoi diagram could be stored as the dual Delaunay triangulation, having triangles as objects, with pointers to adjacent triangles and vertices. The extraction of the polygon edges was performed by scanning through the triangulation and extracting the Voronoi edges and arcs of interest. These were exported as individual arcs, without any topological linkages between them. This was undesirable if we wished to work with the polygonal map within our own software programs. Is it possible to use a common data structure for both the triangulation and the arcs, and simply throw away the unwanted triangle edges?

The Quad-Edge data structure

There are various ways of storing and manipulating planar graphs in the computer. They are all designed to allow the program to move from one entity (Region, Edge, Node) in the graph to some adjacent entity. Worboys [5] gives a good summary.



Figure 3 Input forest map.

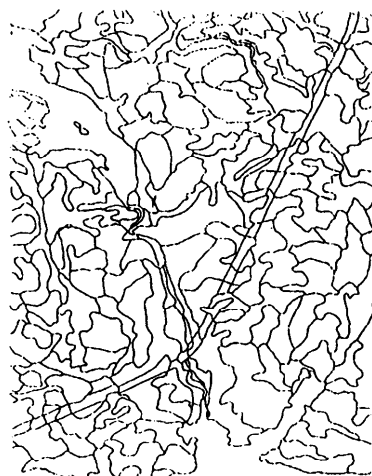


Figure 4 Output Forest map.

The Quad-Edge data structure (Guibas and Stolfi, [1]) allows navigation through the map from edge to edge only. Its advantages are firstly that there is no distinction between the primal and the dual representation (it is symmetric with respect to edges and faces/polygons), and secondly that all operations are performed as pointer operations only, thus giving an algebraic representation to its operations. Its disadvantage is that storage costs are high by comparison with other methods. It is, however, very easy to implement, especially in an object-oriented environment. The top part of Figure 5 shows the basic structure, with four branches for each edge of the graph being stored, one for each of the adjacent vertices or faces. There are two pointers on each branch: one to the vertex or face object, and one to the next anticlockwise Quad-Edge around that object. Thus all vertices or faces have complete topological loops around them.

Only two commands are used to modify a graph: “Make-Edge”, illustrated in the lower portion of Figure 5, to create a new edge on a manifold, and “Splice”, shown in Figure 6, to connect/disconnect Quad-Edges together. In the simplest case, Splice connects two separate “Next” loops, joining the two nodes together, and at the same time splitting the “Next” loop around the common face. (The process could equally well have split the “Next” loop around a node, and merged the loops around a face: Splice is its own inverse. See Guibas and Stolfi [I] for more details.)

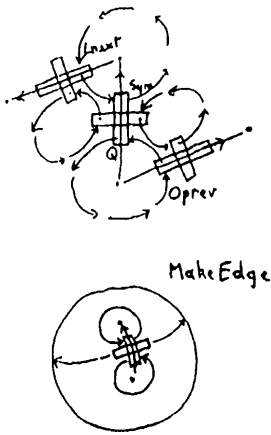


Figure 5 The Quad-Edge structure.

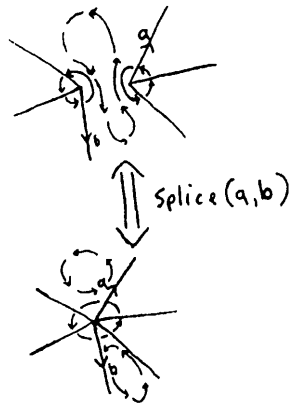


Figure 6 Splice.

The Quad-Arc structure

The scanned map process exported individual arcs to an appropriate GIS - but in this process we lose the topological structuring, which was always present in the Voronoi structure used to generate the arcs. The original Voronoi structure contains far too many individual points to be used as an archival mechanism, or for internal processing of polygonal maps. It would be desirable as an alternative to remove the large number of irrelevant Voronoi edges without destroying the desired polygonal structure. This would be difficult with an underlying triangulation, but straightforward with the Quad-Edge structure. The method is a modification of the approach used in Gold et al. [2], where the triangles (or edges) are traversed in visibility order, and the decision as to whether the edge is relevant or irrelevant is based on whether the triangle vertex labels are the same or different. In this case, instead of collecting arc (polygon boundary) segments for export, the unwanted segments are removed using the Splice operation, and the 'good' arcs are preserved.

The polygon boundaries that remain consist of strings of Quad-Edges, connected at the ends. The original Quad-Edge structure was designed for general graphs, with pointers to face elements and node elements. In a mapping context, we wish to preserve the relationships between arcs

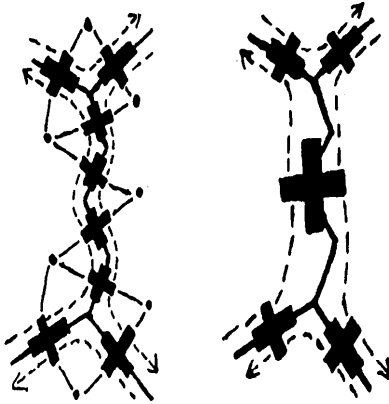


Figure 7 Quad-Edge to Quad-Arc conversion.

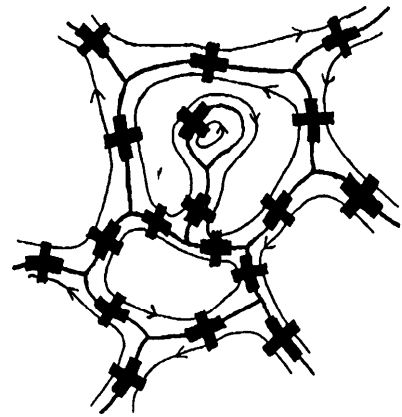


Figure 8 The Quad-Arc topological structure.

composed of multiple points. Since arcs usually include their nodal end points, it is feasible to replace the Quad-Edge node pointers with a pointer to the arc (and leave the face pointer pointing to the polygon definition). We have called this a Quad-Arc, to emphasize its similarity to the arcs used in a traditional GIS to define the boundaries between polygon pairs. As there are two node pointers in each Quad-Edge, it is possible to have each pointing to a directed arc, as might be the case in a divided highway. However, this is not usually the case, and as the arc is inevitably stored in a directed order, the appropriate node pointer points to the arc, while the opposed pointer is given a null value. In use, when the null pointer is detected the opposed pointer is checked to find the arc, which will be in reverse order. If the Quad-Arc is to be used for a simple triangulation, both of these pointers will refer to node objects (as before) rather than to arc objects. Two null pointers indicate a “dummy” Quad-Edge, used to retain the connectedness of the graph.

The strings of Quad-Edges representing polygon boundaries may therefore be traversed again, using a depth-first search or equivalent (see Sedgewick, [6]), to generate Quad-Arcs. This is illustrated in Figure 7. Figure 8 shows a small polygon map resulting from this process. In both of these figures the topological loops are shown around the perimeter of the polygons, but the loops around the nodes of the polygon map are not shown, for the sake of simplicity, nor are the pointers to the polygons or nodes. For more details see Guibas and Stolfi [1], or Mioc et al. [7].

Once the structure is completed, the resulting topological structure may be used to perform polygon shading, etc., as with any mapping program. Due to the overall simplicity of the algorithm, the method may be used in a wide variety of applications. Current work is on a simple mapping package where hand drawn maps may be scanned into a PC with an economical scanning unit, and the resulting topologically complete map is available in a very few minutes. This may then be exported to a commercial mapping program or GIS, or preserved internally for simple mapping projects.

Advantages of the Quad-Arc structure

The resulting data structure is more compact than the original Quad-Edges and cleanly separates the topology from the attributes, in that the pointer structure is entirely within the topological elements: four pointers between the branches of each Quad-Arc in the object-oriented version, four pointers to adjacent Quad-Arcs, and four pointers to attribute information. These are traditionally the two faces and the two nodes associated with the Quad-Edge, but in the Quad-Arc the face pointers point directly to polygon attribute information, perhaps in a database, and the two node pointers may be used to reference the associated arc (chain of x-y coordinates) or other arc information as desired. Thus only one topology file is required, along with a geometry (arc) file and a polygon attribute storage mechanism. Islands may be integrated with dummy Quad-Arcs. The topology permits any connected graph (not only polygons), thus allowing the maintenance of network graphs.

Queries in this structure are based on the edge-algebra developed by Guibas and Stolfi [1]. Finding the boundaries of a polygon for example requires successive calls to *Oprev* or *Lnext* (see Figure 5), until one returns to the starting arc. To find adjacent polygons at the same time involves looking at the dual branches of each Quad-Arc at each step. Since travelling from any Quad-Arc to any other involves only a sequence of pointer movements around the arms of the Quad-Arc, or to the next Quad-Arc around a node or face, these may be encoded as a binary string for any path within the map.

A boundary between two polygons may be deleted by calling *Splice* twice, to disconnect each end. If a boundary is to be introduced between

two nodes, then Splice is again called twice. If a polygon is to be cut in half by a new boundary, then first the intersections with the two old boundaries have to be found. Each of these is deleted, as described above, and the associated arc cut in half, to give two new arcs with the intersection point in common. These are then reconnected with the Splice function, and the new arc added in the same fashion. It should be 'noted that all these operations are local in extent, permitting real-time map modification,

An interesting possibility with the Quad-Arc data structure is that, because the Quad-Edge edge algebra is valid on any (orientable) manifold, it is possible to move beyond the plane. The structure on the sphere has been described in Gold [8]. Even further, manifolds include surfaces with 'handles', thus allow the modelling of overpasses and underpasses, etc. The Voronoi diagram and the Delaunay triangulation are preserved directly in the same structure, without the necessity of performing any additional searches around nodes or polygons. The same is true of a polygonal map and its dual, or indeed for any connected graph. The work of Mioc et al. [7] shows the value of the Quad-Edge or Quad-Arc data structure for managing the temporal component of a map: how to handle changes over time. They have shown that with the edge algebra it is possible to define reversible operations in a spatio-temporal data structure that permit moving forwards or backwards in time, with consequent map updating.

Conclusions

As a consequence of this approach we can read a scanned polygonal map, extract the fringe points, generate the Voronoi diagram and remove the unnecessary elements in order to generate a topologically complete map. The structure permits straightforward topology maintenance and spatial queries after simplification. It holds strong promise of being able to handle spatio-temporal queries in the future, and extends readily to the sphere or any manifold. This may be done entirely within a simple and concise program, and the resulting map used for display or modification immediately. Topologically structured maps have ceased to be beyond the reach of small-scale developers.

Acknowledgements

Many thanks are due to Weiping Yang, Luc Dubois, Darka Mioc and François Anton for assistance with the programming. This research was jointly funded by the Natural Sciences and Engineering Research Council of Canada, and the Association de l'industrie forestière du Québec.

References

- [1] Guibas, L. and J. Stolfi, Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams, *Transactions on Graphics*, 4, pp. 74- 123, 1985.
- [2] Gold, C.M., J. Nantel and W. Yang, Outside-in: an alternative approach to forest map digitizing, *International Journal of Geographical Information Systems*, 10, no. 3, pp. 29 1-3 10, 1996.
- [3] Gold, C.M. and S. Cormack, Spatially ordered networks and topographic reconstructions, *International Journal of Geographical Information Systems*, 1, pp. 137-148, 1987.
- [4] Gold, CM., Simple topology generation from scanned maps, *Proceedings, Auto-Carto 13, ACM/ASPRS*, Seattle, pp, 337-346, April 1997.
- [5] Worboys, M.F., *GIS: A Computing Perspective*, Taylor and Francis, London, 376 pp., 1995.
- [6] Sedgewick, R., *Algorithms*, Addison-Wesley, Reading, Mass., 551p., 1983.
- [7] Mioc, D., F. Anton, C.M. Gold and B. Moulin, Spatio-temporal management for dynamic maps, *Proceedings, Eighth International Symposium on Spatial Data Handling*, Vancouver, July 1998.
- [8] Gold, CM., The global GIS, *Proceedings, International Workshop on Dynamic and Multi-Dimensional GIS*, Hong Kong, pp. 1-4, August 1997.