

An Algorithmic Approach to a Marine GIS

Christopher Gold
Chair in Geomatics
Laval University
Quebec, Canada

INTRODUCTION

The concept of a “Marine GIS” is in some ways an oxymoron: a GIS (Geographical Information System) almost presupposes that “Geography” equals “Land”. The classical GIS structure is based on manual cartography, with registered transparent overlays of different themes, and the manual extraction of relevant combinations - an approach that is much less useful in a full marine environment (although still relevant in coastal zone management). Thus polygon overlay, polygon reclassification and buffer zone generation are the central operations. These are based on some form of underlying topological structure defining polygon connectivity. Other modules, such as network management or terrain modelling, are handled by other structures.

In a marine GIS, however, many of these concepts are open to question. Where are the polygons, the topographic models or the networks? On land most of the features are presumed to be stable, with only occasional updates. At sea, how many features are stable, and how many continuously moving? In other words, is a “Marine GIS” a GIS at all, within the usual definitions? Clearly, in the long term, it is not - but in that case we need to define more fully which operations we need to be able to perform, and then to examine the relevant structures.

Two properties immediately come to mind when thinking about the sea: that it is three-dimensional, and that it is continuously changing. The definition of a three-dimensional spatial marine model is of great interest, but probably beyond our current capabilities in this paper. We will therefore restrict ourselves to two dimensions, even though this is even less plausible than it is on land. Nevertheless, if our 2-D concepts may be expanded to 3-D then this will be a bonus.

Thus we will concentrate on 2-D models - either of the water surface or of the sea floor and the bathymetry. Nevertheless, even these are fundamentally different from a land-based GIS: many things are continuously changing their position. Any spatial structure must be able to handle routine positional update, or else we will not be able to implement any meaningful spatial model. By a spatial model we mean the selection of the appropriate properties of our real-world space that we intend to simulate in the computer. This has been discussed in detail in Gold (1996), and will be summarized briefly here.

One of the properties of real-world space is that it appears to be continuous. This is clear in the case of “field” data, such as temperature or (terrestrial) elevation. It is also true for “object” data, such as houses, as we are really dealing with an abrupt density change between the surrounding air and the solid house. Is a cloud an object or a field? Treated as a (less abrupt) density change it is best represented as a field. The same is true of many marine features - currents, plankton fields, polynyas, etc.

Nevertheless, in our (human) management of information, we perform a classification of these changes in field value, to provide identifying names. Within a computer system, our “space” must be broken up into appropriate pieces for storage, classification and management purposes - but it is important to remember that we *are* performing a classification and subdivision of space, with all its oversimplification and potential for error. Even when we are working with straightforward objects such as houses, there is an underlying spatial classification of “house” and “not house”, and notions of connectivity, proximity and adjacency.

On the basis of these thoughts, which become less esoteric and more meaningful when working with something as non-solid as the sea, it becomes clear that our computer simulation of space must be able to handle field-type data, must be able to embed solid objects within this space, and must be able to handle local movement (either of water or of solid objects such as ships) within our spatial data structure. As the overall objectives of our Marine GIS have still to be defined, we will content ourselves with suggesting that we must be able to simulate the water movement or object motion, and be able to monitor the system state at any moment in time. (By simulation we mean either the use of some mathematical algorithm, as for water flow, or a series of updates of ship position or other observations.) We need to be able to combine field and object information, to preserve such spatial information as connectivity, proximity and adjacency, and to prohibit the existence of more than one object in one location at the same time.

Since we are looking at a form of spatial partition, it is natural to look first at the grid or raster model. This partitions space into a tessellation on the basis of some superimposed coordinate system. It has the particular advantage that adjacent cells are readily detected, and many algorithms have been developed to handle questions of proximity and connectivity, among other questions. Its limitations are due to the fact that it *is* purely a partition of space, and is not directly related to any embedded objects or samples of field values. Where objects are concerned, it is difficult to manipulate. In addition, it is hard to generate for the globe with satisfactory results, it has a fixed spatial resolution that must be defined in advance, and it may have very high storage costs.

A second spatial partition is the “vector” polygon structure, as used in most commercial GISs. After the real-world space has been classified, this works fairly well. However, it was defined for static polygonal structures, not for discrete objects or data points, and is not easy to update locally. It is hard, for example, to imagine a polygon-based system to manage the navigation of a ship between various moving obstacles. It has the attraction of being a tessellation, as is the raster, can be represented as a graph structure inside the computer (a big advantage) but does not readily adapt to many unconnected island structures, or to

networks or other planar graphs. Its biggest handicap lies in its construction from digitized arcs, which is typically slow and error-prone.

Other spatial partitions are also used for specialized purposes. 2D-trees, quadtrees (of various types) and R-trees are spatial indexing schemes for the rapid storage and retrieval of spatially-located objects, and TINs (Triangulated Irregular Networks) are used for the modelling of terrain from irregularly distributed elevation values. These and other variants are discussed in Worboys (1995).

It will be noted that all of these structures are concerned only with the spatial component of our information, and not with its attributes - although this has been touched on briefly in our consideration of clouds and houses. Much work has been done on this aspect by many other researchers - our interest here is only that changes in values of some property or properties over space. This has been discussed further in Gold (1996), and here we will simply assume that some classification may be made between spatial locations, so as to distinguish them. For simplicity this will usually be referred to as the "colour" of the object.

At this stage we are ready to specify a spatial model and an associated spatial data structure that satisfies many of our proposed requirements. The spatial model - the Voronoï diagram - has been discovered many times in different applications over the last century or so (Descartes, Dirichlet, Voronoï, and Thiessen, among others), and more recently has had many versions and algorithms developed within the discipline of Computational Geometry (Aurenhammer 1991, Okabe *et al.*, 1992). Our task here is to describe it as an appropriate topological data structure for many geomatics applications, especially in dynamic contexts such as the marine environment.

THE SIMPLE VORONOÏ DIAGRAM - A STATIC DATA STRUCTURE

We will restrict ourselves initially to the simple nearest-point Voronoï diagram (VD) (sometimes known as the Thiessen diagram) in the Euclidean plane, although many extensions exist, and are described in the references above. (The VD may readily be embedded in the surface of the sphere, simplifying the boundary conditions in the plane.) The diagram may be defined as the partitioning of the plane such that, for any set of distinct data points, the cell associated with a particular data point contains all spatial locations closer to that point than to any other. (This definition may be extended beyond data points, to line segments and other, more complex objects such as houses or road segments if needed.) The definitions and algorithms which follow may be extended to two or more dimensions as required, permitting the extension of this approach to a fully three dimensional marine GIS. The simple Voronoï/Thiessen diagram may be found in various GIS packages, usually for determining the regions closest to certain resources, such as fire stations or shopping centres.

A simple VD is shown in **Figure 1**, along with the associated dual Delaunay triangulation (DT). This triangulation, often used in terrain modelling, has a triangle edge connecting each pair of objects whose Voronoï cells are adjacent. (The DT consists of triangles whose circumcircles are empty. If four or

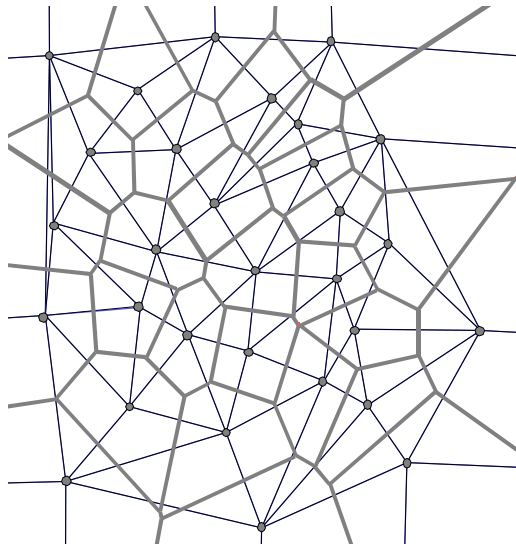


Figure 1
Voronoi
diagram and
Delaunay
triangulation

more points are cocircular, the interior triangle edges may be arbitrarily selected.) It has the advantage of the raster model in that it is a tessellation of the plane, and hence neighbouring points are well defined. It has the advantage of the vector polygon model in that only one object is associated with each cell. (We will see later how to use these point objects to generate more complex ones.) Per item, the VD requires more storage than the raster model, although fewer cells are needed to

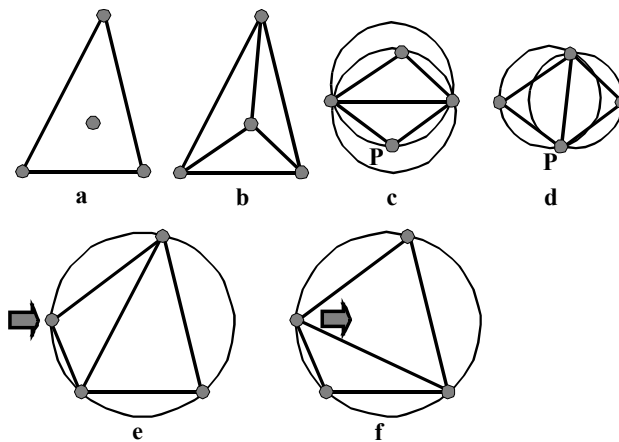


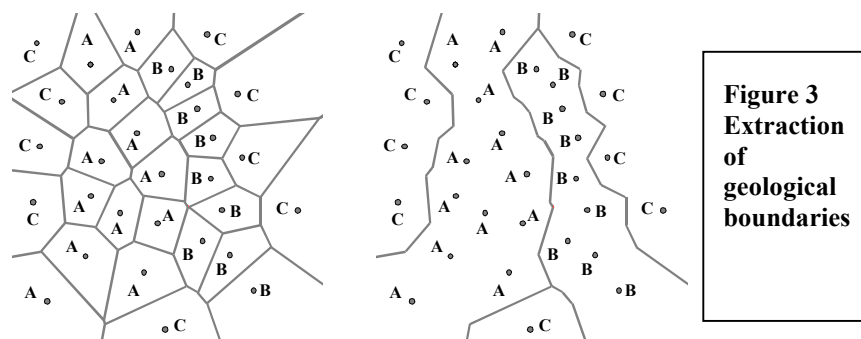
Figure 2
Point
insertion and
diagonal
switching

represent most spatial features. Unlike the vector polygon model, the VD is readily updated locally. (This means that if a point is added, only a small number of neighbouring cells need to be modified. Thus updating is very fast.) We will start by describing a simple implementation of this algorithm and an appropriate data structure. We will then show a variety of applications that become surprisingly simple with this approach.

The easiest way to start is with a large triangle on the plane, big enough to hold all the data. (Alternatively, one may use a tetrahedron or octahedron on the sphere.) The simplest data structure element is a triangle, with pointers to each of the three vertices and to each of the adjacent triangles (Gold *et al.*, 1977). A “walk” through the triangulation from any starting triangle easily gives the triangle containing the point to be added (**Figure 2a**). This triangle is split in three (**Figure 2b**), and all the exterior edges are tested to see if the three new triangles have empty circumcircles (**Figure 2c**). If they do not, the edge forming the diagonal between them is switched, connecting the other two vertices (**Figure 2d**), and the edges of the new triangles are also tested. On the average there are only a few switches (Gold *et al.*, 1977). This is sufficient to create the triangulation, and dual Voronoï diagram, as shown in **Figure 1**, with reasonable efficiency. One modification is to replace the triangle data structure with the Quad-Edge structure (Guibas and Stolfi, 1985) or as modified by Gold (1998) as the “Quad-Arc” structure, for more complex cases to be described later. Here the topological element is no longer the triangle but the edge, and it may be used to store any connected graph on the orientable manifold. Only a few lines of code are needed for the implementation, as shown in Gold (1998).

EXTRACTING BOUNDARIES FROM DATA

While well known and easy to implement, the simple VD is sufficient for a variety of applications when combined with the “visibility ordering” of Gold and Maydell (1978) or Gold and Cormack (1987), which gives a method for traversing a triangulation so that each triangle is visited once only. Based on this we may give each vertex of the input data a colour, and then extract the edges of the VD between vertices of different colours, discarding the rest. This is described in Gold *et al.* (1996). **Figure 3a** gives the example of a possible geological map, where each data point identifies the rock type of an outcrop. Discarding the unwanted



edges gives a first approximation to the geological boundaries (**Figure 3b**). Gold *et al.* (1996) used this approach for forest map input by digitizing around the interior of each polygon in turn, and labelling each point with the polygon number. This results in a rapid approximation to the polygon map, with significant data input savings. Arcs were then exported to a conventional GIS. Gold (1997) extended this

to scanned maps, where “fringe” points were generated around the black pixels on the scanned image (**Figure 4a**). These were then coloured using a “flood-fill” algorithm for each polygon interior, and then processed as before. **Figure 4b** shows the VD/DT of the points from **Figure 4a**, and **Figure 4c** shows the extracted polygon boundaries. Gold (1998) extended this again by using the Quad-Arc structure mentioned previously, permitting the preservation of the topological structure within small PC based programs. This structure is capable of maintaining any connected planar graph, unlike the previous triangle structure. Thus for a static VD (or for a snapshot of a dynamic VD at some specified time) contours can be generated separating differing classes of data, as identified by the vertex colour. This is applicable to the Marine GIS as well as to the terrestrial applications given.

A concept closely associated with the VD is the Medial Axis Transform (MAT), or “skeleton”. This is a subset of the VD, and is usually applied to polygons - each point on the MAT is equidistant to at least two edges of the polygon. The resulting graph may be used to classify the polygon shape (see Blum, 1967, 1973 and Blum and Nagel, 1978) - for example in character recognition. In **Figure 4** the discrete topology was really the skeleton of the linework, identified as separating vertices of differing colour.

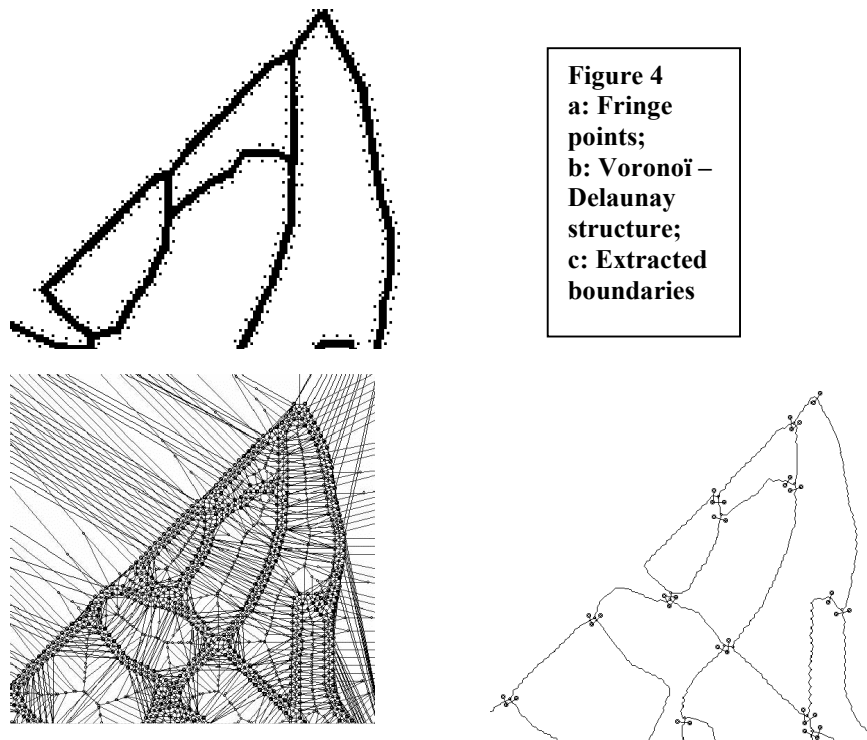


Figure 4
a: Fringe points;
b: Voronoi – Delaunay structure;
c: Extracted boundaries

One problem, however, is that it is not always easy to colour the black boundary pixels so that edges may be distinguished. If there are no closed polygons – in a river network for example - all input points to the VD will have the same

colour, and no skeleton will be produced. Amenta *et al.* (1998) showed that if a curve or boundary is sufficiently well sampled then the boundary may be extracted from the unordered points as the set of DT edges whose circumcircles do not contain part of the skeleton. (Sufficiently well sampled was shown to be a function of the distance from the boundary to the MAT. It is usually easily achievable except at sharp corners.) Gold (1999) and Gold and Snoeyink (1999) showed that the extraction process may be reduced to a simple local test on each Quad-Edge, which is assigned to be either part of the crust (DT edge) or the skeleton (VD edge). Of interest to us is that the discrete skeleton, as an approximation to the MAT, is also extracted. A useful terrestrial example is the extraction of approximate watershed boundaries from a scanned hydrological network - the MAT gives the watershed between watercourses (**Figure 5**).

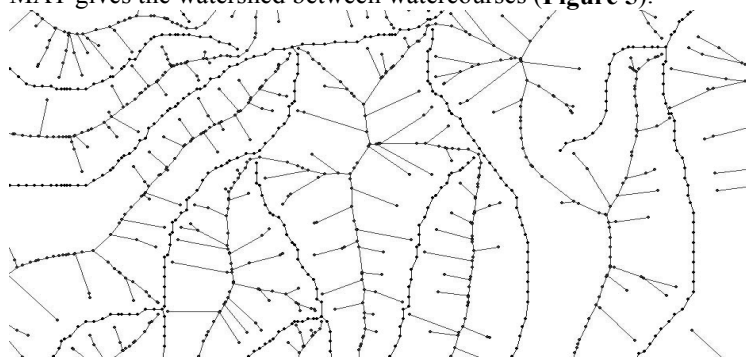


Figure 5 Watershed estimation from hydrology network

While irregular in location and shape, the simple VD has many of the advantages of a raster that were described above. This becomes valuable once we consider flow simulation, an application of interest in a Marine GIS. In a raster model the flow domain (perhaps an aquifer or watershed in a terrestrial setting) is approximated by a relatively coarse grid. Each cell is given initial conditions (land elevation and water level), and the slope calculated between each pair of adjacent cells (approximated by their centroids). For each time step, flow is estimated between cell pairs, the water levels are updated, and the cycle repeated - either to a steady state, or for the duration of the simulation. The VD may also be used in the same manner - the governing equations take a little more derivation, but this is facilitated by the cell boundaries being perpendicular bisectors between the generating points (see Narasihman and Witherspoon 1976). This is particularly efficient because the VD is more easily adapted to the form of the flow domain and boundary conditions (constant head or impermeable boundaries), requiring fewer cells than the grid.

The same advantage holds true for simple mapping, as in the geological map previously mentioned. A simple terrain model may be generated where the data point attribute is elevation rather than rock type. It is appropriate for the above flow modelling, and forms a useful technique whenever the closest data value is the safest elevation estimate, rather than some more complex function. It has

proved very useful for regional volume estimates in coal resource evaluation, even though the surface generated (a “prism map”) has abrupt elevation changes.

A more common approach to terrain (or sea floor) modelling is to use the DT of the same data, rather than the VD, as this gives surface continuity, even

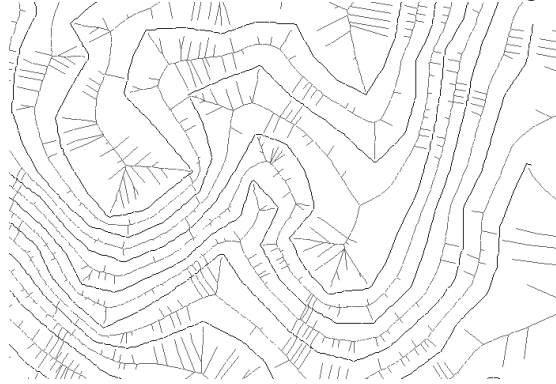


Figure 6a Skeleton extracted from contours

though slopes and higher order derivatives are discontinuous. Attempts to generate “smooth” surfaces across triangle boundaries have not been entirely successful. However, the adaptability of being able to use data of any distribution is extremely valuable, especially with dense data such as along ships’ tracks, with airborne observations and even with contours digitized from existing charts. In this last case the crust/skeleton algorithm is of value in the generation of a correct triangulation (see **Figure 6a**), as it enables the detection of gullies associated with indentations in individual contours. (**Figure 6b** shows the same map generalized by retracting

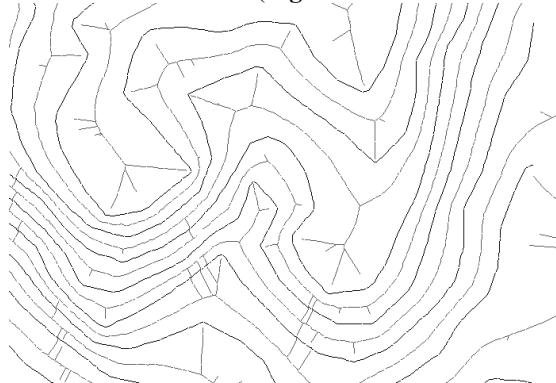


Figure 6b Simplified skeleton

the minor “hair” branches on the skeleton – this smooths minor perturbations in the “crust”.) A big advantage of the DT, as opposed to any other possible triangulation, is the guarantee that the same triangulation will be produced independent of data

input order (other than for cocircular points). It is also locally updateable. An improved interpolation approach using “area-stealing” will be described below.

Apart from polygons and terrain models, the third data structure used in conventional GIS is to manage networks (roads, rivers) for various kinds of analysis, especially network flow. As the above structures are planar graphs, this application may be implemented with the DT, flagging the relevant edges accordingly. It is even more appropriate to use the Quad-Arc structure extracted from the original VD, as full connectivity is preserved, and bidirectional flow may be handled by the ordering structure imposed by the basic Quad-Edge topology. Thus we have shown that, even with the simple incremental VD algorithm, most topological structures used in a traditional GIS may be maintained using the VD/Quad-Arc approach.

POINT DELETION - A DYNAMIC DATA STRUCTURE

In computer science terms, a data structure is “dynamic” if items may be added and deleted individually without having to rebuild a major portion of the whole thing, thus permitting rapid updating of the structure. We must now define a delete operation to add to our simple incremental procedure in order to have a dynamic VD/DT data structure. While mentioned briefly in the literature as “the inverse of the incremental insertion algorithm”, we have not found any detailed description of the algorithm, which we have recently implemented. Briefly, it goes as follows: locate a triangle having the point to be deleted, by performing the “walk” operation as mentioned above. For each pair of the triangles surrounding the point to be deleted, examine the three remaining vertices: could they be a valid Delaunay triangle in the absence of the central point? If so, switch their common edge, so that the central of the three vertices is no longer a neighbour of the point to be deleted. (If the three vertices form a valid Delaunay triangle then there must be a valid circumcircle - the vertices forming the triangle are in anticlockwise order - and this circumcircle must be empty - none of the other neighbours of the point to be deleted fall within this circle.) This process is repeated with the reduced set of neighbours until only three remain. These three triangles are then reduced to one by removing the central point. As each of the triangles “cut off” is Delaunay, the resulting triangulation is also Delaunay. This is equivalent to removing the point to be deleted as the first step, and then retriangulating the remaining polygon. However, the process just described is simpler, and consistent with the point addition algorithm. In this case **Figure 2d** reverts to **Figure 2c** for each vertex switched out. **Figure 2b** reverts to **Figure 2a** when the central point is removed.

Given the ability to add and delete points at will, there are various extensions that can be made to the previous applications - most obviously the ability to modify the data set interactively. (This is particularly useful in the rapid digitizing application, where initial errors need to be corrected once they are detected.) Single points or complex objects can be “moved” by deleting their composing points, and re-inserting them at the new locations. However, the biggest

advantage comes when we wish to perform interpolation on our data set, and the flat triangular plates of the terrain model are inadequate.

Interpolation is the ability to extract estimates of the value of a curve or surface at intermediate locations between the data points. The simple “proximal query” or prism map described above is a form of interpolation, as is the triangulated terrain model. Where some form of “smooth” surface is desired, another approach must be taken. A good summary is given in Lam (1983). The most common approach is to take some form of weighted average of a set of neighbouring points to the location where the estimate is desired - the biggest problem is how to select these neighbours. If a “counting circle” is used, the results depend directly on its radius. The trick is to select a set of neighbours so that the contribution of a particular data point decreases to zero before it is eliminated from the set of neighbours - if not, then there is a discontinuity in the surface as a result of that point’s sudden removal from the set of neighbours. Details are found in Gold (1989), where an alternative is proposed as described below. (A mathematical definition of a similar solution is found in Sibson (1981)).

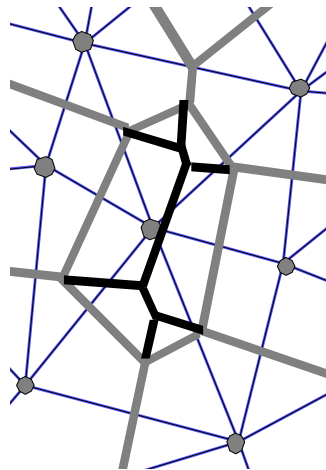


Figure 7 Areas stolen by point insertion

The VD gives an intuitive definition of the neighbours of any data point - so to find the neighbours of some arbitrary query, its location should be inserted into the VD, and its immediate neighbours determined. This new point has “stolen” some of the area of each of these neighbours - that is why they are neighbours, having a boundary in common: see **Figure 7**. If these stolen areas (normalized so they sum to unity) are used as weights for the elevation values of the data points, then we have a surface with certain properties. The surface is continuous, and it is “smooth” everywhere except at the data points. Its derivatives may be calculated (Anton *et al.*, 1998). A simple function may be used to make it smooth at the data points, and also to conform to any slope data that may exist. Most importantly, it conforms to every data point, no matter what the distribution. Given our previous discussion, the algorithm is simple: insert the query point; find the neighbours and

calculate the area of their Voronoï cells; remove the query point; recalculate the Voronoï area of the neighbours; calculate the weighted average based on the differences in the areas.

Thus we may use the “object” topological model defined by the VD to derive “field” properties. The VD combines the properties of both object and field views of space, and hence of both “raster” and “vector” data structures. In a Marine GIS based on the VD, bathymetric models may readily be generated, globally or locally. Of particular interest is the case of a boat navigating through a channel. Depths and slopes may be recalculated in real time, and the course adjusted accordingly. Recent depth soundings may be added to or removed from the data set, and used as required.

POINT MOVEMENT - A KINETIC DATA STRUCTURE

If a point represents our boat in the interpolation example above, it is natural to think of *moving* the boat, rather than merely deleting and re-inserting it. This has been done in the work of Roos (1990), Guibas *et al.* (1991) and Gold (1991) for two dimensions. If one point in the DT is perturbed slightly, then its Voronoï boundaries change slightly also. However, if the displacement is large enough that the circumcircle of a neighbouring triangle is entered (**Figure 2e**) then the diagonal is switched (**Figure 2f**). On leaving the circumcircle of three adjacent neighbours of the moving point the reverse operation takes place. If we can detect when the boat moves into or out of these circumcircles, then we can maintain the VD/DT topological structure. This clearly allows one of the basics of navigation - collision detection - as the examination of the boat’s immediate neighbours at any point in time will give those most likely to be hit if the current course is continued. These neighbouring points could be the outlines of islands, wrecks or other obstacles. We could also perform channel navigation with our moving boat, considering the neighbouring points to be depth soundings, using the interpolation techniques described above.

However, another application comes to mind. We have described water flow simulation by the finite difference method, where the cells (originally grids, but then the VD) expressed fixed portions of space through which the water flowed. This is the Eulerian model - but the Lagrangian model is also attractive, where “particles” (centres of mass) are free to move, along with their associated cells. “Free-Lagrange” methods have been developed using the VD, but without the dynamic data structure component (Fritts *et al.*, 1985). The dynamic VD is obviously attractive for this, and tidal modelling is being attempted using this approach. Global forces act on each particle (data point), which represents a fixed mass of water. Local forces act through the immediate Voronoï neighbours, and the result is a velocity vector associated with each point. Normally with the Free-Lagrange method a fixed time step is used, as with finite-difference or finite-element methods. This can cause problems, especially where turbulent flow is possible, as one fast-moving particle may collide with another, or else overshoot it, destroying the mesh structure. This problem may be eliminated by removing the

fixed time step, and replacing it with a priority-queue of the topological events described above. That is, each moving point has its next topological event, when a triangle switch must take place, and the time of this is dependent on the particle's velocity. By storing this list of events, and processing them in temporal sequence (updating the list after each event), the particle flow may be simulated without the usual mesh maintenance problems. Marine applications of this include simulation of tides, currents and pollution plumes. The method is directly extendable to three dimensions. Indeed, all the point insertion, deletion and movement algorithms previously described are also feasible in 3D, as is the crust and skeleton extraction process.

THE LINE SEGMENT VORONOI DIAGRAM

The above kinetic model may be extended towards more traditional GIS operations by the addition of one simple observation: a curve is the locus of a moving point. Thus, if we think of the moving point described above as the cartographer's "pen", then we may use it to draw our traditional map composed of line segments or polygons. Our two-dimensional algorithm for moving points needs to be modified slightly. Our triangles may now have either points or line segments at their vertices, instead of just points. As a result, a more elaborate circumcircle calculation is needed: instead of a circle touching the three points forming the triangle vertices, we must calculate a circle that is tangent to any line-segment vertices as well. The work of Anton and Gold (1997) shows how this may be done in an iterative fashion - indeed, it may be used with any objects at the vertices of the triangle. In the case of line segments, it is also necessary to know the *side* of the line segment that is connected to the triangle: this may be done with the half-line data structure of Yang and Gold (1996) or equivalent. When the "pen" is placed at the start of a line a point is inserted at that location, and as the "pen" is moved a point is split off from this point and moved towards the destination. This is just point creation and movement, as described above. However, the connection between the parent point and the child (moving) point is itself an object - the desired line segment, which is distinct from its end points. As the moving point progresses, the adjacency relationships between the moving point and its current neighbours are transferred to the trailing line - the line segment has Voronoi boundaries with all the old objects passed by the moving point. At its destination the moving point may merely stop, or may be merged with an object (point or line segment) at that location, allowing the connection of line segments to form polygons or other map objects. Indeed, the point may cross various intermediate line segments, generating intersections as it goes. Line deletion is performed by performing the reverse operations to insertion. **Figure 8** shows a simple map constructed in this fashion.

Gold and Condal (1995) described this as a "boat" approach to map construction - the pen is embedded in the map data structure, and interacts directly with its neighbouring objects, permitting "intelligent" operation, such as automatic polygon closure, snapping to other objects, and the detection and processing of

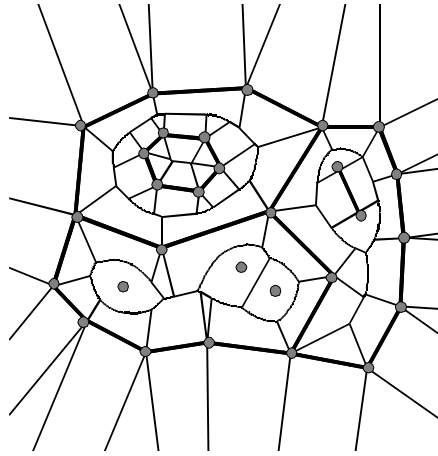


Figure 8 Simple line-segment Voronoi diagram

intermediate intersections. Traditional map generation systems use the “plane” approach, otherwise known as spaghetti digitizing, where there is no interaction between the line drawing process and the underlying map (if any). The interactive approach holds many potential advantages for intelligent operations in addition to the ones described above.

In particular, the boat (or robot) model allows for a large variety of navigation or simulation applications. The “boat” may represent a real boat, and the detection of adjacent obstacles (because their Voronoi cells touch) can allow for evasive action in a simulation. The addition of some “long-distance intelligence”, or goal-seeking, would permit the development of automated navigation systems. It should be remembered that it is not only the boat that may be moved, but, corresponding to the marine reality, obstacles, markers, etc. may appear, disappear or move over time within the dynamic Voronoi structure. This brings us closer to the goal of a freely-modifiable Marine “GIS” - although few of the operations correspond to those of a terrestrial GIS. (Gold 1991 has discussed the use of the Voronoi method for the standard GIS operations of buffer zone, polygon-merge and overlay. Buffer zones are particularly easy, as they consist only of lines or circular arcs drawn at the appropriate distance within the Voronoi cells of line segments or points respectively. Polygon merging may be performed by the local deletion of the common boundary line segments. Polygon overlay can be performed by drawing the first map onto the second, snapping to the second map if the pen is closer than some specified tolerance.)

But perhaps the most interesting observation from Gold and Condal (1995) is that we use exactly the same techniques for two-dimensional navigation (an object view of the world) as we do for interpolation of bathymetry (a field view). As shown in **Figure 9**, boat “A” is able to navigate towards the harbour mouth while evading the island, by examining its Voronoi neighbours. At the same time boat “B” is able to navigate to the deepest channel by using the “area-stealing” interpolation method on its adjacent depth soundings, which are

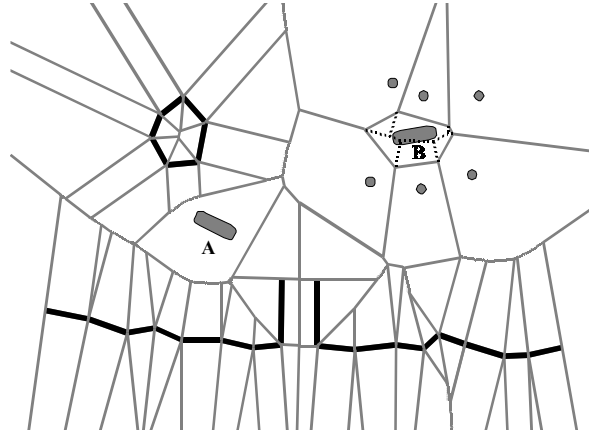


Figure 9 Marine navigation using the dynamic Voronoi diagram

preserved in a different layer than the surface obstacles. Finally, when boats A and B approach too closely to each other, this will be detected by them becoming Voronoi neighbours. This gives us a straightforward model for integrating the main components of a Marine GIS.

CONCLUSIONS AND ACKNOWLEDGEMENTS

We have discussed the algorithmic basics of a potential Marine GIS, based on the VD spatial model, and associated data structures. While our examples have been restricted to two dimensions, most of the methods are extendable to three. Because the fundamental operations of such a GIS are still not well defined, flexibility in the underlying operations is of paramount importance at this time, so as not to restrict our abilities by a too hasty specification of the needs of such a system - a more exploratory approach is required. But this philosophy is just as applicable to a terrestrial GIS. As stated in the Introduction, current systems are based on concepts of manual cartography, whereas today we are not so restricted. The algorithms described here have been implemented, are in use in two dimensions in our research, and may readily be reproduced, as they have in many cases been chosen for their simplicity. What perhaps is important is the concept - a simple set of operations may be sufficiently flexible to allow for a wide variety of applications. A greater openness to new spatial models and to the expansion of the set of basic operations, especially in the light of the increasing need for dynamic systems and simulation, can only be a good thing. The exploration of the needs of this new “floating world”, the examination of potential spatial models and the definition of an appropriate Marine GIS, can only be a benefit to all aspects of GIS and spatial analysis. The author would like to thank Mr. Weiping Yang, Mr. Mir Mostafavi and Mr. David Thibault for assistance with the figures.

BIBLIOGRAPHY

- Amenta, N., Bern, M. and Eppstein, D., 1998, The crust and the beta-skeleton: combinatorial curve reconstruction. *Graphical Models and Image Processing*, **60**, pp. 125-135.
- Anton, F. and Gold, C.M., 1997, An iterative algorithm for the determination of Voronoï vertices in polygonal and non-polygonal domains. In *Proceedings of the Ninth Canadian Conference on Computational Geometry*, Kingston, Ontario, pp. 257-262.
- Anton, F., Gold, C.M. and Mioc, D., 1998, Local coordinates and interpolation in a Voronoï diagram for a set of points and line segments. In *Proceedings of the Second Voronoï Conference on Analytic Number Theory and Space Tilings*, Kiev, Ukraine, pp. 9-12.
- Aurenhammer, F., 1991, Voronoï diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys*, **23**, pp. 345-405.
- Blum, H., 1967, A transformation for extracting new descriptors of shape. In *Models for the Perception of Speech and Visual Form*, edited by Whalen-Dunn, W., (Cambridge, Mass.: MIT Press), pp. 153-171.
- Blum, H., 1973, Biological shape and visual science (Part 1). *Journal of Theoretical Biology*, **38**, pp. 205-287.
- Blum, H. and Nagel, R.N., 1978, Shape description using weighted symmetric axis features. *Pattern Recognition*, **10**, pp. 167-180.
- Fritts, M.J., Crowley, W.P. and Trease, H.E., 1985, *The Free-Lagrange Method. Lecture Notes in Physics*, **238**, (Berlin: Springer-Verlag).
- Gold, C.M., 1989, Chapter 3 - Surface interpolation, spatial adjacency and G.I.S. In *Three Dimensional Applications in Geographic Information Systems*, edited by Raper, J., (London: Taylor and Francis).
- Gold, C.M., 1991, Problems with handling spatial data - the Voronoï approach. *CISM Journal*, **45**, pp. 65-80.
- Gold, C.M., 1996, An event-driven approach to spatio-temporal mapping. *Geomatica*, **50**, pp. 415-424.
- Gold, C.M., 1997, Simple topology generation from scanned maps. In *Proceedings of Auto-Carto 13*, Seattle, Washington, (ACM/ASPRS), pp. 337-346.

- Gold, C.M., 1998, The Quad-Arc data structure. In *Proceedings of the Eighth International Symposium on Spatial Data Handling*, Vancouver, BC, pp. 713-724.
- Gold, C.M., 1999, Crust and anti-crust: a one-step boundary and skeleton extraction algorithm. In *Proceedings of the ACM Conference on Computational Geometry*, Miami, Florida, (ACM).
- Gold, C.M., Charters, T.D. and Ramsden, J., 1977, Automated contour mapping using triangular element data structures and an interpolant over each triangular domain. In *Computer Graphics 11, Proceedings of Sigraph '77*, pp. 170-175.
- Gold, C.M. and Condal, A.R., 1995, A spatial data structure integrating GIS and simulation in a marine environment. *Marine Geodesy*, **18**, pp. 213-228.
- Gold, C.M. and Cormack, S., 1987, Spatially ordered networks and topographic reconstructions. *International Journal of Geographical Information Systems*, **1**, pp. 137-148.
- Gold, C.M. and Maydell, U.M., 1978, Triangulation and spatial ordering in computer cartography. In *Proceedings of the Canadian Cartographic Association Third Annual Meeting*, Vancouver, BC, pp. 69-81.
- Gold, C.M., Nantel, J. and Yang, W., 1996, Outside-in: an alternative approach to forest map digitizing. *International Journal of Geographical Information Systems*, **10**, pp. 291-310.
- Gold, C.M. and Snoeyink, J., 1999, A one-step crust and skeleton extraction algorithm. *Algorithmica*, (submitted)
- Guibas, L., Mitchell, J.S.B. and Roos, T., 1991, Voronoï diagrams of moving points in the plane. In *Proceedings, 17th. International Workshop on Graph Theoretic Concepts in Computer Science: Lecture Notes in Computer Science*, **570**, (Berlin: Springer-Verlag), pp. 113-125.
- Guibas, L. and Stolfi, J., 1985, Primitives for the manipulation of general subdivisions and the computation of Voronoï diagrams. *Transactions on Graphics*, **4**, pp. 74-123.
- Lam, N. S-N., 1983, Spatial interpolation methods: a review. *American Cartographer*, **10**, pp. 129-149.
- Narasihman, T.N. and Witherspoon, P.A., 1976, An integrated finite-difference method for analyzing fluid flow in porous media. *Water Resources Research*, **12**, pp. 57-64.

Okabe, A., Boots, B. and Sugihara, K., 1992, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. (Chichester: John Wiley and Sons).

Roos, T., 1990, Voronoi diagrams over dynamic scenes. *Proceedings of the Second Canadian Conference on Computational Geometry*, Ottawa, pp. 209-213.

Sibson, R., 1981, A brief description of natural neighbour interpolation. In), *Interpreting Multivariate Data*, edited by Barnett, V. (New York: John Wiley and Sons), pp. 21-36.

Worboys, M.F., 1995, *GIS: A Computing Perspective*. (London: Taylor and Francis).

Yang, W. and Gold, C.M., 1996, Managing spatial objects with the VMO-Tree. In *Proceedings of the Seventh International Symposium on Spatial Data Handling*, Delft, The Netherlands, pp. 711-726, 11B15-11B30.