

# Voronoi Methods in GIS\*

(extended abstract)

Christopher M. Gold<sup>1</sup>, Peter R. Remmele<sup>2</sup>, and Thomas Roes<sup>2</sup>

<sup>1</sup> Centre de Recherche en Géomatique  
Pavillon Casault, Université Laval  
Ste-Foy, Québec, Canada G1K 7P4  
Email: christopher.gold@scg.ulaval.ca

<sup>2</sup> Department of Computer Science  
ETH Zentrum, CH-8092 Zürich, Switzerland  
Email: { remmele, roos}@inf.ethz.ch

## 1 Introduction and Historical Background

Traditional vector-based GIS organize basic objects of interest such as roads, rivers, towns, or houses in thematic multi-layered (polygonal) maps representing them as polygons, arcs, and nodes (see Corbett [6] for a summary). Much of the early work was based on the line-intersection model of space where the global detection of intersecting arcs or lines was the sole method of determining connectivity – see Dutton [10] and Chrisman et al. [5]. Relations between polygons, arcs, and nodes imposed a “topology” which linked the basic map objects.

At that time, GIS systems were designed to process polygonal thematic maps, in order to respond to simple queries of the type “Show me all the areas with Type 1 or Type 2 soil, not zoned as industrial, within 50m of a lake”. It turned out that such queries can be answered using only three basic operations. The first operation, “Reclassify and Merge”, reclassifies a polygonal map on the basis of each polygon’s attributes and combines adjacent polygons of the same resulting class. The second operation, “Corridor” or “Buffer Zone”, constructs a polygon set with boundaries at a specified distance around the source object set (lakes in our example). The third operation, “Polygon Overlay”, combines two polygon layers into a single layer such that each polygon gets the properties from each original layer (e.g. Type 1 soil and industrial in our example). All operations are global and depend therefore on a complete rebuild of the topology after each operation.

Many years of experience allowed us to identify the weaknesses of this technique. The detection of intersections itself is both an expensive operation and prone to errors due to digitizing limitations; with the consequence that if an

- 
- A full version of this paper will appear in the special issue of *Algorithmica* on Cartography and Geographic Information Systems [18].

intersection is not found the result is an incomplete knowledge of topology. As adjacency is defined via intersection, disconnected features such as islands are difficult to handle correctly. As a result, there is no readily-available approach to locally modify the topology and hence no true dynamic system permitting the addition and deletion of individual map objects. Dissatisfaction has started to grow concerning these limitations, because even minor modifications to the map require a global rebuild. Although a variety of techniques can be used to modify a small patch and then to sew it back into the main map, subsequent synchronization problems and localization problems (finding locally modifiable patches) arise.

As the emphasis on the user interface continues to grow, it is clear that the current non-interactive spatial model will be superseded by something closer to the human interaction with a paper map using a pen – or even our interaction with real-world geographic space. This requires that our actions take effect immediately, both in spatial queries and spatial modifications of our map. If we can detect the adjacent objects to our moving pen at any time during map construction – so must the GIS; this provides the possibility of testing line segment intersections and snapping points to line segments. If we can detect polygons or other structures that fail to close exactly – so must the GIS. If we can locate the polygon containing our pen, an isolated data point, or an island without special processing – so must the GIS. We thus need a spatial model possessing the tiling and adjacency properties of the raster model but the direct relationship to real-world objects of the vector model.

While this has puzzled the GIS community for some years, at least one potential answer is readily available from Computational Geometry: the Voronoi diagram [32], a universal data structure for representing proximity (see, e.g., the early works by Shamos and Hoey [29] and by Green and Sibson [21]). It supports a multitude of nearest neighbor queries (details can be found in the textbooks by Preparata and Shamos [27] and Okabe et al. [26]). As we will see, the Voronoi diagram for points and line segments with its dual, the Delaunay multigraph [7], provides the basic spatial adjacency properties between map objects. This not only resolves the basic difficulties with the line-intersection model, but allows us to maintain the topology of a GIS.

Nowadays, the Voronoi diagram and its variants are well known in many areas of science (see Aurenhammer [2] for a survey) and many algorithmic paradigms of Computational Geometry apply to compute them efficiently (see [3,11,22,29]). In Geomatics and GIS, Gold [14-16] discovered the Voronoi diagram as a fundamental tool for representing and maintaining topology in a map. Generalizations towards dynamic operations [9] and kinematic motions [4,8,23,31] have been investigated for about one decade.

This paper gives a survey of static, kinematic, and dynamic Voronoi diagrams as basic tools for Geographic Information Systems (GIS). We present a method that allows the insertion, deletion, and translation of points and line segments in a Voronoi diagram of  $n$  generators. All elementary operations are available in

$O(\log n)$  expected time and linear expected storage space. The Voronoi approach also greatly simplifies some of the basic traditional GIS queries and allows even new types of higher level queries. The concept of a persistent, locally-modifiable spatial data structure that is always complete provides an important new approach to spatial data handling that is not available with existing systems.

The paper is organized as follows. Section 2 provides the Computational Geometry basis by an introduction into static and kinematic Voronoi methods; we show how both can be combined towards a fully dynamic approach. In section 3, we look at the results from a practical point of view and show how the presented methods can be applied to obtain efficient algorithms for GIS.

## 2 Voronoi Diagrams

A brief summary of the elementary definitions and properties of Voronoi diagrams of points and line segments in the Euclidean plane is given in the following. Let  $d$  denote Euclidean distance and let

$$S := \{l_1, \dots, l_n\}$$

be a finite set of  $n \geq 3$  disjoint line segments (so-called generators) in the Euclidean plane. Line segments are allowed to degenerate to points. Later, we will also allow line segments to share endpoints, in which case we split off such endpoint from all incident line segments leaving behind open' line segments. Our results still apply, as we can leave an arbitrarily small gap between the line segments and the endpoint. We use the current assumption of disjoint line segments in the following, as it provides a more intuitive understanding.

In order to simplify the presentation, we assume the line segments of  $S$  to be in general **position**: we claim that there is no point in the Euclidean plane having the same distance to four different line segments and that there is no line in the plane tangent to three line segments, such that they all lie on the same side of this line.

### 2.1 Static Voronoi Diagrams

The static Voronoi diagram of  $S$  partitions the plane according to the nearest neighbor rule: each generator is associated with the region closest to it. The **bisector**

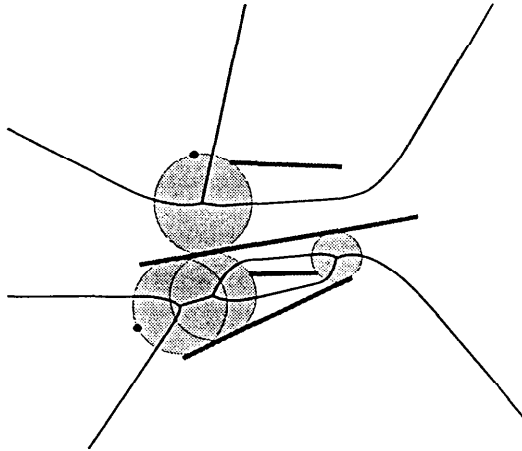
$$B_{ij} := \{X \in \mathbb{R}^2 \mid d(x, l_i) = d(x, l_j)\}$$

of two line segments  $l_i$  and  $l_j$  is a differentiable curve consisting of line and parabola segments. The **Voronoi** region or **Voronoi cell**

$$v_i := \{x \in \mathbb{R}^2 \mid d(x, l_i) \leq d(x, l_j) \text{ for all } j\}$$

---

<sup>1</sup> We call a line segment **open** if one or two endpoints have been split off.



**Fig. 1. Voronoi diagram and empty-circle property.**

of  $l_i$  is generalized-star-shaped [25] with respect to  $l_i$ . The “vertices” of the Voronoi regions, i.e. the non-differentiable points on the boundary  $\partial v_i$ , are called **Voronoi points** and the bisector parts on the boundary are called **Voronoi edges**. Finally, the **Voronoi diagram  $V(S)$**  is the collection of its Voronoi polygons. Figure 1 shows a Voronoi diagram of six line segments and points. Using Euler’s formula, it is easy to see that the number of Voronoi vertices and edges is bounded by  $O(n)$ ; the space complexity needed to store the Voronoi diagram is therefore linear in the number of line segments.

Three line segments  $l_i, l_j, l_k \in S$  form a Voronoi point in  $V(S)$  iff there exists a circle of contact touching these three generators and no other line segment intersects the interior of this circle<sup>2</sup> (cf. Figure 1). This fact is well-known under the term **empty-circle property** for Voronoi diagrams.

When talking about topology and neighborhood relations of geometric objects, the dual multigraph  $D(S)$  of the Voronoi diagram  $V(S)$  is often more intuitive. In  $D(S)$ , each generator of  $S$  corresponds to a node and each Voronoi edge of  $V(S)$  corresponds to an edge; this is done in such a way that the cyclical order of the dual edges around a dual node corresponds to the ordered sequence of Voronoi edges of the corresponding Voronoi region (see Figure 2). In general position, the dual multigraph  $D(S)$  can be extended to form a complete triangulation by adding a point at infinity (c.f. [28]); this extension is called the **topological structure** of the Voronoi diagram.

There exist several algorithmic techniques for constructing Voronoi diagrams of points and line segments in  $O(n \log n)$  time and  $O(n)$  space (see [9,11]).

<sup>2</sup> Notice that if three line segments of  $S$  generate two circles of contact, these circles have different orientation with respect to the cyclical order of the generators on their boundary.

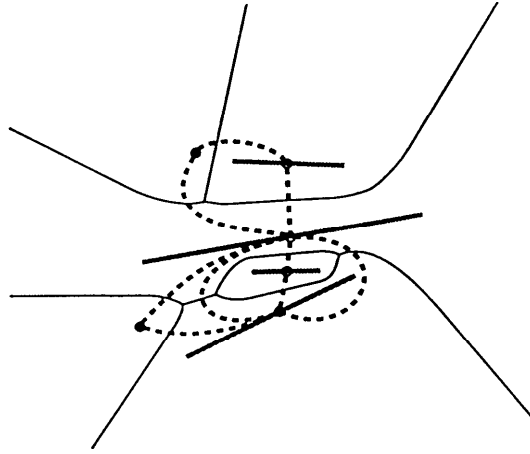


Fig. 2. The dual multigraph.

## 2.2 Kinematic Voronoi Diagrams

We now extend the notion of the Voronoi diagram to continuously moving generators (see [1] for a recent survey). For our purpose, it is sufficient to consider one point  $p$  moving along a straight line within a static scene  $S$  of line segments (compare [28]).

As point  $p$  moves, the Voronoi diagram  $V(S \cup p)$  changes continuously, but at certain critical instants in time, topological **events** occur that cause a change in the topology. At each instant of time, the topology of the Voronoi diagram is completely determined by the Voronoi points in  $V(S \cup p)$  and by the unbounded Voronoi edges. Therefore, the topological structure of  $V(S \cup p)$  can only change when a Voronoi point appears/disappears, or when a change on the boundary of the convex hull  $\partial CH(S \cup p)$  occurs.

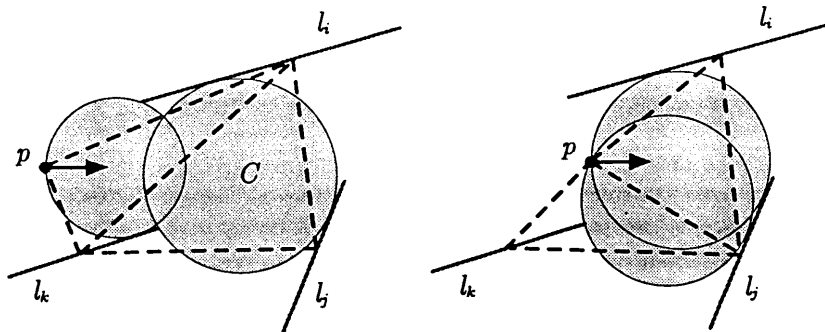
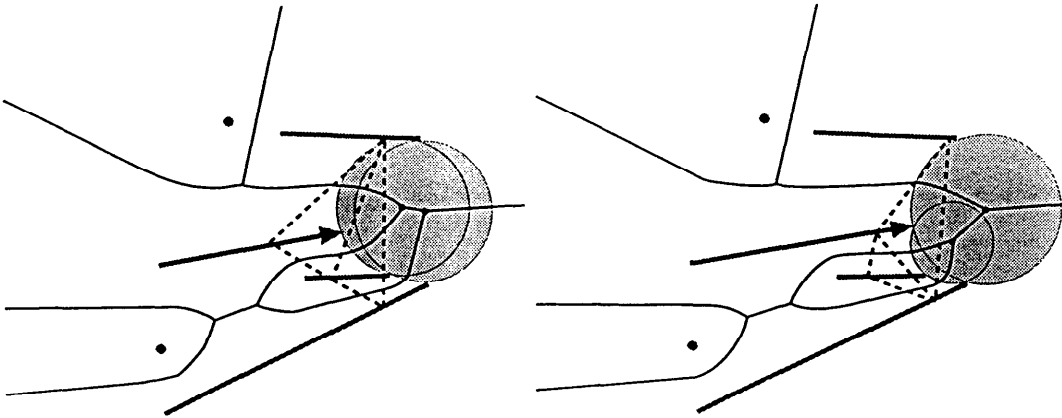


Fig. 3. The quadrilateral before and after the swap.

It is well-known that in the dual multigraph  $D(S \text{ Up})$ , both types of changes can be described as swaps of the diagonal edge of two neighboring triangles (see Figure 3). As  $\mathbf{p}$  alone is moving while all other generators in  $S$  are fixed at their current position, a topological event can only occur when  $\mathbf{p}$  runs into or out of a circle of contact or when  $\mathbf{p}$  enters or leaves the boundary of the convex hull  $\partial CH(S \text{ Up})$  (see [18] for more details). During the motion of  $\mathbf{p}$  a list of swaps can be maintained in a priority queue which uses  $O(\log I;)$  time per event in the worst case; here,  $k$  is the current degree of  $\mathbf{p}$  in the dual multigraph. The swap itself can be performed in  $\mathbf{O(1)}$  time when using the quad-edge data structure by Guibas and Stolfi [24] (see [28] for more details).

### 2.3 Dynamic Voronoi Diagrams

We now generalize our concept to allow **dynamic** operations such as insertions and deletions of points and line segments in a Voronoi diagram; we present a surprisingly simple and general framework to handle insertions and deletions in  $O(\log n)$  expected time. For inserting and deleting a point  $\mathbf{p}$  into the generator set  $S$ , we adopt the framework of Devillers et al. [9] using a slightly modified definition of the conflict region by Boissonnat et al. [3].

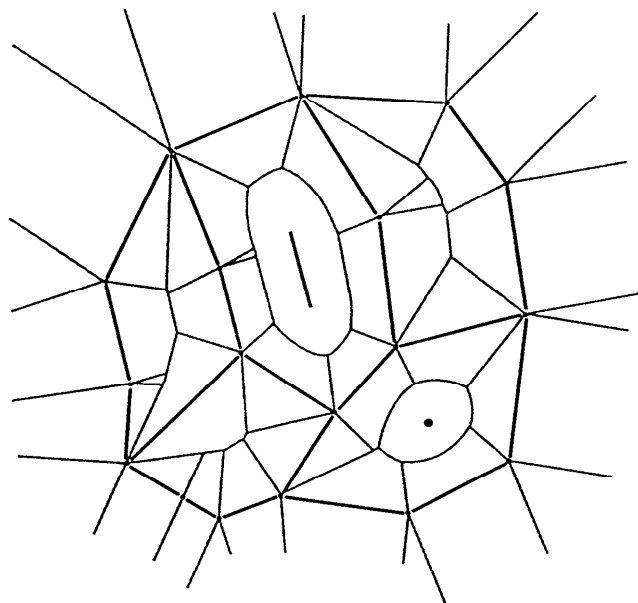


**Fig. 4.** A Voronoi diagram during an expansion.

For handling line segments, we add a tool for expanding and shrinking line segments using kinematic Voronoi methods. More precisely, we **insert a line** segment into a Voronoi diagram of line segments by first inserting one of its endpoints into the Voronoi diagram. Afterwards, we expand the generator by **moving** the point to the other end of the line segment (cf. Figure 4). In this way, a line segment is generated by a rubber band pinned down at one end and dragged to the other. In this setting, the topological events are even simpler

than in the general case since the moving point - the free endpoint of the line segment - only runs into (but never out of) circles of contact corresponding to Voronoi points (see [19,28]). We delete a line segment in a Voronoi diagram of line segments by reversing this operation: we first shrink the line segment to a single point and remove this point from the Voronoi diagram.

Using input randomization, one can prove that the expected time for expanding/shrinking a line segment is bounded by  $O(\log n)$  (see [18] for details).



**Fig. 5.** A Voronoi diagram of points and open line segments.

### 3 GIS - A new approach

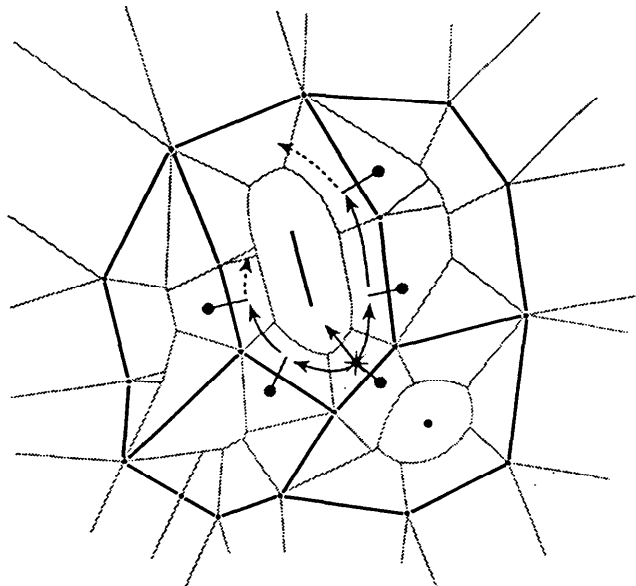
Traditional GIS normally are coordinate-based systems, which means that a uniform grid will span the geographic environment and this grid will define the environment model's resolution. Objects of interest such as mountain peaks, corner stones, or even towns (depending on the resolution) will be represented by a point on the model's grid. In order to incorporate extended objects such as rivers and estates, it is useful not only to employ single points but also line segments and polygons. Besides a variety of information attached to these basic objects, a GIS has to store some kind of neighborhood relationship to allow local updates and to answer queries on locality. Therefore, a data structure designed

as the base of a GIS must be able to handle at least these objects within its neighborhood relations.

The Voronoi diagram and its dual, the Delaunay multigraph, exactly fulfill these properties; the Voronoi diagram allows us to locate the nearest generator to a given query point whereas the dual gives us fast access to the neighbors of a generator stored in the GIS. Manipulations on the data such as insertions, deletions, or translations can easily be handled as we have seen before. In order to design the complex objects of a polygonal map, we have to allow line segments that share endpoints (as described in Section 2). Figure 5 shows a Voronoi diagram of a polygonal map.

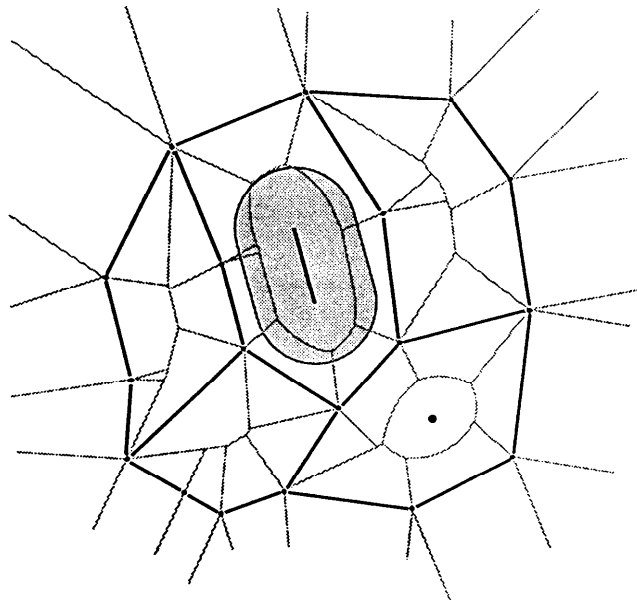
### 3.1 Traditional GIS operations

Within the Voronoi diagram, a map polygon creates Voronoi cells in its interior. **Labeling** a **polygon** consists of selecting any of these cells (by using the proximal query) and performing a flood-fill to collect all adjacent cells (see Figure 6). Attributes are then associated with the polygon label. When reclassification gives the same class on each side of a line segment, this segment is removed using the line deletion process described above.



**Fig. 6.** A flood-fill of a polygon.

Buffer-zone generation is performed by inserting the target object set, e.g. the boundaries of a set of lakes, into a Voronoi layer (see Figure 7). For each Voronoi cell the intersection of the corridor boundary (at the specified distance from the generating object) may be calculated - if it is present at all. This is a complex task within a traditional GIS, but is easily available with the Voronoi diagram [14], being an expression of the “wavefront” or “prairie fire” analogy.



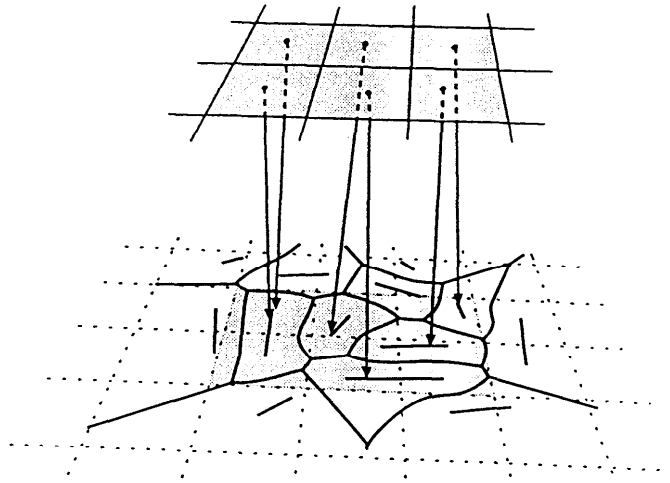
**Fig. 7. Buffer zone generation.**

**Polygon overlay** in the Voronoi context may be performed by drawing the second polygon set, line by line, onto the Voronoi diagram of the first polygon set. The attributes associated with each line segment are preserved. A form of flood-fill is then used to ensure that labeling is correct in the resulting map, even in the case of islands.

### **3.2 Separating the topology from the map data**

There are three main data types stored in a Voronoi system: coordinates, 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 open line segments (and have pointers, as required, to the coordinate records and the matching line segment). Equivalently, the quad-edge data structure of Guibas and Stolfi [24] may be used. Unlike the DCEL structure (see,

e.g., [27]), no pointers are associated with the map objects themselves; instead, all topology is contained within the Delaunay triangle records. This is an extremely desirable property in a GIS, as the **same** object may then be inserted in several layers simultaneously. This eliminates object duplication in the attribute database, as well as avoiding having different coordinate representations of the same object in various layers - eliminating **sliver** polygons when overlaid.



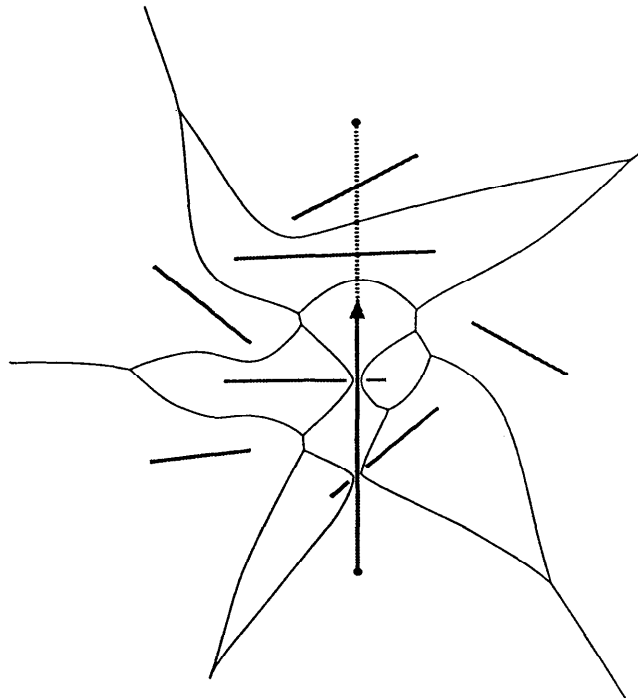
**Fig. 8. Point location by a grid.**

As a consequence of this design decision, there remains the basic question of how to access the portion of the triangulation associated with a particular map object, or with any specified  $(x, y)$  location. In practice, a simple walk through the triangulation from some previously accessed triangle (see, e.g., [24]) has been found quite sufficient to date, despite its poor theoretical efficiency. The next object to be processed is nearly always close to the previous object. Where necessary, a grid-based point location process may be employed (see [26] and Figure 8), at the cost of a more elaborate structure. However, it should be noted that even a simple grid, containing a pointer to some object falling within the grid cell, followed by a simple walk, will significantly improve the performance.

### **3.3 The intersection problem**

We showed how a line could be generated as the locus of a moving point and how to create and delete points and line segments. This works satisfactorily when the end points are known in advance. For real-world digitizing, connections are formed when the line being drawn crosses a previously-defined line. In this case, intersections must be detected in advance of the moving point. Gold [13] showed

that a line segment can only be intersected by an expanding line segment if both have been Delaunay neighbors shortly before. So, if a line segment becomes a new Delaunay neighbor of an expanding line segment, an intersection test is made and, if necessary, the intersection point calculated, and both the intersected line segment and the expanding line segment are split off at the intersection point as described above. This allows us to split and merge Voronoi diagrams along a common line (as shown in Figure 9).



**Fig. 9.** Separating and merging a Voronoi diagram.

This gives a fully dynamic algorithm for the creation and deletion of points and line segments, for point movement and for intersection and the snapping together of lines during digitizing.

### 3.4 Update management

As the topological structure is a triangulation, both global and local searching techniques are directly achievable as we have seen before (compare [30]). Because the update process is incremental, the addition and deletion of objects is immediate, and queries may **be** asked at once. This is of great value in identifying objects to be joined, for example, as it is only necessary to point within the

Voronoi cell of an object in order to select it. This proximal query is basic to all interaction and is equivalent to normal human gestures for specifying objects. The objective, as with much recent work on graphic interfaces, is to conform as far as possible to natural human gestures. Also, a fully dynamic map construction opens various ways of managing the history of a map; this appears to be an interesting approach to spatio-temporal GIS (compare Frank et al. [12]).

Finally, spatial operations not normally associated with the topology may be implemented using a single spatial data structure. These include robot navigation, interpolation, fluid flow simulation, dynamic network analysis, etc. (see [17] for some examples). The structure gives a fully local means of preserving topology, allowing fully interactive map update in response to the user's graphical actions (see [13]). It also permits the mixing of many types of data - fully connected polygons, connected hydrography, discrete points and line segments - within the same overlay.

### **3.5 Navigation and interpolation**

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 using a "cursor" or "observer" embedded within the Voronoi diagram may be used to steer 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 has been suggested by Gold and Condal [17].

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. 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 (see [20] for details).

### **3.6 Map partitioning and parallel processing**

When maps become very large - maybe even global - additional issues become critical, e.g., partitioning the map into portions that are manageable in memory becomes a significant issue. Traditional GIS, with map sheet boundaries, do not have a good answer. Some suggestions for partitioning maps along line segment boundaries or constrained triangulation edges are given in [33]. This has the added attraction that such a partitioning may be used to control access to particular portions of the map, either so that several operators may simultaneously be working on map update within assigned regions, or else to allow parallel

processing to build the whole map more quickly without danger of conflict. Splitting and merging Voronoi diagrams as described above are fundamental in this process.

## Conclusions

**The Voronoi diagram embodies a form of “automated” topology, which - used as a basic data structure in GIS - may be the tool for the appropriate problems in the domain of Geomatics, and also the basic framework for algorithmic correctness and efficiency.**

The concept of an always-complete, persistent, locally-modifiable spatial data structure provides an important new approach to spatial data handling. We have implemented the concept to a level that validates the proposed system design. However, analysis of future GIS needs leads to significant implementation issues where the field of Computational Geometry could contribute to a systematic resolution of an important field of spatial analysis. Clarification and simplification of problems concerning spatial or topological relationships could be the gain of a collaboration of Geomatics with Computational Geometry.

## Acknowledgements

The funding of the first author for this research was made possible by the foundation of an Industrial Research Chair in Geomatics at Laval University, jointly funded by the Natural Sciences and Engineering Research Council of Canada and the Association de l'Industrie Forestière du Québec. The second and third author gratefully acknowledge the support by the Swiss National Science Foundation (SNF) under grant 21-39328.93 and 20-45407.95.

The authors would like to thank Jack Snoeyink and the anonymous referees for their helpful comments.

## References

1. G. Albers, L.J. Guibas, J.S.B. Mitchell, and T. Roos, *Voronoi diagrams of moving points*, Tech. Rep. 235, ETH Zurich, 1995, to appear in *Intl. J. Comp. Geom. & Appl.*
2. F. Aurenhammer, *Voronoi diagrams: A survey of a fundamental geometric data structure*, *ACM Comput. Surv.*, Vol. 23, pp 345-405, 1991.
3. J.-D. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec, *Applications of random sampling to on-line algorithms in Computational Geometry*, *Discrete & Computational Geometry*, Vol. 8, pp 51-71, 1992.
4. P. Chew, *Near-quadratic bounds for the  $L_1$  Voronoi diagram of moving points*, *Proc. 5<sup>th</sup> Canad. Conf. Comp. Geom. CCCG'93*, pp 364-369, 1993.

5. N.R. Chrisman, J.A. Dougenik, and D. White, *Lessons for the design of polygon overlay processing from the Odyssey Whirlpool algorithm*, Proc. 5<sup>th</sup> Intl. Symp. on Spatial Data Handling SDH'92, Charleston, pp 401410, 1992.
6. J.P. Corbett, *A general topology for spatial reference*, SORSA Report, 1985.
7. B.N. Delaunay, *Sur la sphere* vide, Bull. Acad. Science USSR VII: Class. Science Math., pp 793-800, 1934.
8. O. Devillers and M. Golin, *Dog bites postman: Point location in the moving Voronoi diagram and related problems*, Proc. 1<sup>st</sup> Annual European Symp. on Algorithms **ESA'93**, LNCS 726, pp 133-144, 1993.
9. O. Devillers, S. Meiser, and M. Teillaud, *Fully dynamic Deiaunay triangulation in logarithmic time per operation*, Comp. Geom. Theory & Appl., Vol. 2, pp 55-80, 1992.
10. G. Dutton (Ed.), *First International Advanced Symposium on Topological Data Structures for GIS*, Harvard University, Cambridge, MA, Vol. 8, 1978.
11. S. Fortune, *A sweepline algorithm for Voronoi diagrams*, Algorithmica, Vol. 2, pp 153-174, 1987.
12. A.U. Frank, I. Campari, and U. Formentini (Eds.), *Theories and Methods of spatio-Temporal Reasoning in Geographic Space*, LNCS 639, 1992.
13. C.M. Gold, *An object-based dynamic spatial model, and its application in the development of a user-friendly digitizing system*, Proceedings, 5<sup>th</sup> Intl. Symp. on Spatial Data Handling **SDH'92**, Charleston, pp 495-504, 1992.
14. C.M. Gold, *Problems with handling spatial data - the Voronoi approach*, CISM Journal, Vol. 45, No. 1, pp 65-80, 1991.
15. C.M. Gold, *Spatial data structures - the extension from one to two dimensions*, In: L.F. Pau (Ed.), Mapping and Spatial Modelling for Navigation, NATO ASI Series F, No. 65, Springer-Verlag, Berlin, pp 11-39, 1990.
16. C.M. Gold, *The interactive map*, In: M. Molenaar and S. de Hoop (Eds.), Advanced Geographic Data Modelling and Query Languages for 2D and 3D Applications, Netherlands Geodetic Commission, Publications on Geodesy, No. 40, pp 121-128, 1994.
17. C.M. Gold and A.R. Condal, *A spatial data structure integrating GIS and simulation in a marine environment*, Marine Geodesy, Vol. 18, pp 213-228, 1995.
18. C.M. Gold, P.R. Remmele, and T. Roos *Fully dynamic and kinematic Voronoi diagrams in GIS*, Special Issue on Cartography and Geographic Information Systems, Algorithmica, to appear.
19. C.M. Gold, P.R. Remmele, and T. Roos, *Voronoi diagrams of line segments made easy*, Proc. 7<sup>th</sup> Canadian Conference on Computational Geometry CCCG '95, Laval University, Quebec City, pp 223-228, 1995.
20. C.M. Gold and T. Roos, *Surface Modeiing with Guaranteed Consistency - An Object-Based Approach*, Proc. Int. Workshop on Advanced Research in GIS **IGIS'94**, LNCS 884, pp 70-87, 1994.
21. P.J. Green and R. Sibson, *Computing Dirichiet tessellations in the plane*, The Computer Journal, Vol. 21, pp 168-173, 1978.
22. L.J. Guibas, D.E. Knuth, and M. Sharir, *Randomized incremental construction of Deiaunay and Voronoi diagrams*, Proc. 17<sup>th</sup> Intl. Colloquium on Automata, Languages and Programming **ICALP'90**, LNCS 443, Springer, pp 414 - 431, 1990.
23. L.J. Guibas, J.S.B. Mitchell, and T. Roos, *Voronoi diagrams of moving points in the plane*, Proc. 17<sup>th</sup> Intl. Workshop on Graph Theoretic Concepts in Computer Science **WG'91**, Fischbachau, Germany, LNCS 570, pp 113-125, 1991.

24. L.J. Guibas and J. Stolfi, *Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams*, ACM Transactions on Graphics, Vol. 4, pp 74-123, 1985.
25. D.T. Lee and R.L. Drysdale, *Generalization of Voronoi diagrams in the plane*, SIAM J. Comput., Vol. 10, No. 1, pp 73-87, 1981
26. A. Okabe, B. Boots, and K. Sugihara, *Spatial tessellations - concepts and applications of Voronoi diagrams*, John Wiley and Sons, Chichester, 1992.
27. F.P. Preparata and M.I. Shamos, *Computational Geometry: An introduction*, Springer, New York, 1985.
28. T. Roos, *Dynamic Voronoi diagrams*, PhD thesis, University of Würzburg, Germany, 1991.
29. M.I. Shamos and D. Hoey, *Closest point problems*, Proc. 16<sup>th</sup> Annual IEEE Symp. on Foundations of Computer Science FOCS'75, pp 151-162, 1975.
30. M. Teillaud, *Towards dynamic randomized algorithms in computational geometry*, LNCS 758, Springer-Verlag, Berlin, 1993.
31. T. Tokuyama, *Deformation of merged Voronoi diagrams with translations*, TRL Research Report TR87-0049, IBM Tokyo Research Laboratory, 1988.
32. G.F. Voronoi, *Nouvelles applications des paramtres wntinus à la théorie des formes quadratiques*. Premier Mémoire: *Sur quelques propriétés des formes quadratiques positives parfaites*, J. Reine Angew. Mathematik, Vol. 133, pp 97-178, 1907. Deuxieme Memoire: *Recherches sur les paralléloèdres primitifs*, J. Reine Angew. Mathematik, Vol. 134, pp 198 - 287, 1908 and Vol. 136, pp 67-181, 1909.
33. W. Yang and C.M. Gold, *Managing spatial objects with the VMO-Bee*, Proc. 7<sup>th</sup> Intl. Symp. on Spatial Data Handling SDH'96, Vol. 2, Delft, The Netherlands, pp 11B-15 to 11B-30, August 1996.