

Fully Dynamic and Kinematic Voronoi Diagrams in GIS

Christopher M. Gold*

Faculté de Foresterie et de Géomatique
Université Laval, Québec, Canada

Peter R. Remmele**

Thomas Roos**

Department of Computer Science
Federal Institute of Technology
Zurich, Switzerland

ABSTRACT This paper gives a survey of static, dynamic, and kinematic Voronoi diagrams as a basic tool for Geographic Information Systems (GIS). We show how the Voronoi diagram with its dual, the Delaunay multigraph, can be used to maintain the topology of the map objects of a GIS. The presented method 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 using expected linear storage.

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.

KEYWORDS Computational Geometry, Delaunay triangulation, dynamic and kinematic operations, GIS, Voronoi diagram.

*Address: Centre de recherche en géomatique, Pavillon Casault, Université Laval, Ste-Foy, Québec, Canada G1K 7P4. **Fax: +1 418 656 7411**. Email: christopher.gold@scg.ulaval.ca

Address: Dept. of Computer Science, ETH Zentrum, **CH-8092 Zürich, Switzerland. **Fax: +41 1 632 1172**. Email: {remmele, roos}@inf.ethz.ch

1 Introduction

1.1 Traditional spatial data structures

Traditional vector-based GIS topology was created for thematic (polygonal) maps, linking the basic objects of interest like roads, rivers, towns, or houses, which are represented by polygons, arcs, and nodes in various ways (see Corbett [6] for a summary). Much of the early work was done by the Harvard Laboratory for Spatial Analysis, and was based on a 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]. These systems were designed for the processing of polygonal thematic maps, in order to respond to a basic set of queries. It was found that 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" could be addressed by three basic operations. The first operation, "Reclassify and Merge" or "Dissolve", reclassified a polygonal map on the basis of each polygon's attributes and combined adjacent polygons of the same resulting class. The second operation, "Corridor" or "Buffer Zone", constructed a polygon set with boundaries at a specified distance around the target object set (lakes in our example). The third operation, "Polygon Overlay", combined two polygon coverages into a single coverage such that each polygon had properties from each original coverage (e.g. Type 1 and industrial in our example). All three of these are global operations on polygon sets, and depend for their operation on a complete rebuild of the topology (i.e., the polygon/arc/node relations) after each operation.

Sufficient experience has been gained using this technique in the last fifteen or so years to identify its weaknesses. Basically, if an intersection is not found, then there is no knowledge of neighbors. Detection of intersections is both an expensive operation and prone to errors due to digitizing limitations - and thus some intersections may not be detected. In addition, there is no definition of adjacency other than that of intersecting lines, and features such as islands are difficult to place correctly. As a result, there is no readily-available incremental approach to modifying the spatial linkages and hence no true dynamic system permitting the addition or deletion of individual map objects. This implies that changes of the topology occurring at specified times, can not easily be handled, as these must be added or removed one at a time by a global recalculation.

Dissatisfaction has started to grow concerning the limitations of the global, batch, polygon based spatial model being used. Even minor modifications to the map require a global rebuild, although a variety of techniques are used to modify a small patch and then to sew it back into the main map. Also the digitizing process becomes extremely error-prone, as arcs are

added incrementally, but the topology required to define the connectivity is only added as a batch operation afterwards. As connectivity is defined by the intersections of arcs, this is only functional for polygonal map tilings (and, to a much lesser extent, networks). Thus no spatial structure of this type can readily be defined for disconnected objects such as point sets or islands.

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.

1.2 A fully dynamic approach

All of the problems previously mentioned arise because our spatial relationships are based entirely on the vector model of intersecting line segments being our only definition of adjacency. Yet even with the polygon-arc-node model there is a more stable definition: polygons are adjacent if they have a common boundary. In the raster spatial model (also frequently used in spatial analysis) this is trivially true. We thus need a spatial model possessing the tiling and adjacency properties of the raster 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 well-known data structure in Computational Geometry (see, e.g., the early work by Shamos and Hoey [29]).

For a given set of generators, the Voronoi diagram partitions the space in such a way that each generator is assigned to the points closest to that generator with respect to a given distance function. For reasonable classes of generators (such as points, line segments and circular arcs) and reasonable distance metrics (such as the Euclidean one), the Voronoi diagram allows a compact representation of the topology, and can be represented by a topological data structure such as, e.g., the quad-edge data structure by Guibas and Stolfi [23]. It supports a multitude of proximity and 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. With that, it resolves the basic difficulties with the line-intersection spatial model.

Nowadays, the Voronoi diagram and its variants are well known in many areas of science (see Aurenhammer [2] for a fundamental survey). Nearly all algorithmic paradigms known in Computational Geometry apply for computing planar Voronoi diagrams of n points in $O(n \log n)$ time and $O(n)$ space. Among them are the classic divide & conquer algorithm by Shamos and Hoey [29], the sweepline method by Fortune [11], and randomized incremental techniques by Boissonnat et al. [3] and Guibas et al. [21].

Kinematic Voronoi diagrams, where generators are allowed to move along given continuous trajectories, have been investigated since about one decade. Starting with the early work by Tokuyama [31] (for two groups of points) and Guibas, Mitchell, and Roos [22] (for independent motions), kinematic Voronoi diagrams found more and more interest in the Computational Geometry research community (see [4, 8]). Independently, in Geomatics and GIS, Green and Sibson [20] and Gold [13, 14, 16] discovered the Voronoi diagram as a fundamental tool for representing and maintaining topology in a map.

The paper is organized as follows. Section 2 provides the Computational Geometry basis by giving a brief introduction into static and kinematic Voronoi methods. Section 3 discusses extensions towards fully dynamic and efficient maintenance of Voronoi diagrams with respect to insertions and deletions of generators. In Section 4, we look at the results from a practical point of view and show where the presented methods can be applied to obtain practical and very efficient algorithms in GIS. Finally, a discussion and an outlook can be found in the Conclusions.

2 The Voronoi diagram

This section briefly summarizes the elementary definitions and properties of Voronoi diagrams of points and line segments in the Euclidean plane. As usual, we let d denote Euclidean distance (extended to compact objects). We are given a finite set

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

of $n > 3$ disjoint line segments (so-called generators) in the Euclidean plane. We allow line segments to degenerate to points. Later on, for polygonal maps in GIS, we will have to allow line segments that share endpoints. In this case, if an endpoint of a line segment is incident to more than one

line segment, we split off the endpoint from all incident line segments; this leaves open¹ line segments behind. For reasons of continuity (we can leave an arbitrarily small gap between the line segments and the endpoint), all our theoretical investigations apply. However at the moment, the current assumption of disjoint line segments provides a more intuitive understanding,

In order to simplify the present at ion, we assume the line segments of S to be in general position. For this, 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 Diagram

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.

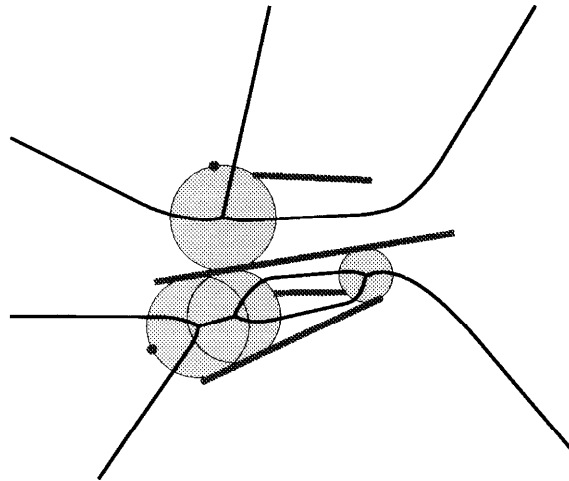


FIGURE 1. Voronoi diagram and empty-circle property.

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**

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

¹ We call a line segment open, if one or two endpoints have been split off.

of l_i is generalized-star-shaped [24] with respect to l_i . The “vertices” of the Voronoi regions, i.e. the non-differentiable points on the boundary ∂v_i , are called *Voronoi points* and the bisector parts on the boundary are called *Voronoi edges*. Finally the *Voronoi diagram* of S is defined by

$$V(S) := \bigcup_{i=1}^n \partial v_i$$

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 $O(n)$; 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 (cf. Figure 1). This fact is well-known under the term *empty-circle property* for Voronoi diagrams. 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; this allows to distinguish them and the corresponding Voronoi points,

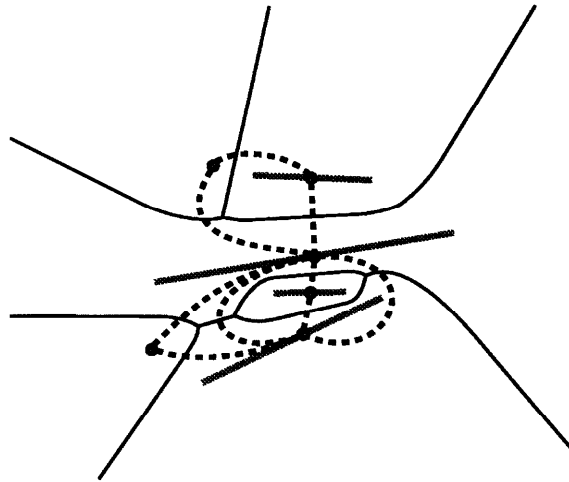


FIGURE 2. The dual multigraph.

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. Thereby each generator of S corresponds to a node and each Voronoi edge of $V(S)$ to an edge in the dual multigraph. 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 (cf. [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. Among them are the sweepline algorithm by Fortune [11] and a randomized incremental technique by Devillers et al. [9]. A common property of these algorithms is that they are either static or semi-dynamic, i.e. they only allow insertions of new generators.

2.2 Kinematic Voronoi Diagrams

In this section, we 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 (cf. [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. Notice that 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.

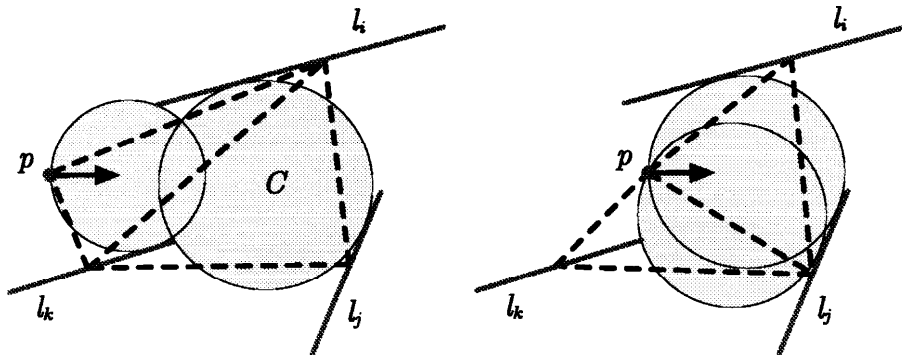


FIGURE 3. The quadrilateral before and after the swap.

For ease of description, we only consider the disappearance of a Voronoi point, i.e. the case when the moving generator p enters a circle of contact C of three line segments $l_i, l_j, l_k \in S$; the reverse transition of p leaving the circle, as well as topological changes on the boundary of the convex hull

$\partial CH(S \cup p)$, can be handled analogously. Shortly before p enters the circle of contact corresponding to $l_i, l_j, l_k \in S$, there exists another circle of contact, of the three generators p, l_i, l_k ; both circles of contact correspond to Voronoi points in $V(S \cup p)$. At the instant when p enters C , both Voronoi points disappear while two new Voronoi points corresponding to the generators p, l_i, l_j and p, l_j, l_k are generated (see Figure 3).

In the dual multigraph $D(S \cup p)$, the topological event can be described as a *swap* of the diagonal edge of two neighboring triangles (see again Figure 3). As p alone is moving while all other generators in S are fixed at their current position, a swap can only occur when p runs into or out of a circle of contact or when p enters or leaves the boundary of the convex hull $\partial CH(S \cup p)$. For computing the potential swaps, we calculate for each quadrilateral, i.e. for each pair of adjacent dual triangles of the topological structure involving p , the (at most two) instants when p crosses the corresponding circle of contact of the remaining generators (three line segments) of the quadrilateral, or when p crosses the corresponding edge on the boundary of the convex hull $\partial CH(S)$. During the motion of p a list of swaps can be maintained in a priority queue which uses $O(\log k)$ time per event in the worst case; here, k is the current degree of p in the dual multigraph. The swap itself can be performed in $O(1)$ time when using the quad-edge data structure by Guibas and Stolfi [23] (see also [28] for more details).

3 Dynamization

Operations such as insertions and deletions of points and line segments in a Voronoi diagram are the topic of this section; we present a surprisingly simple and general framework to handle insertions and deletions in $O(\log n)$ expected time. For this purpose, we adopt a technique by Boissonnat et al. [3] for inserting and deleting points. 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. 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. The point location step necessary for inserting a point at the right place in the Voronoi diagram is the crucial step of any insertion algorithm with respect to time complexity (see, e.g., [21]).

3.1 Point location

For our purpose, we adopt a technique by Boissonnat et al. [3] that uses a conflict graph. Similar methods that avoid building a conflict graph can be found in the textbook by Mulmuley [25].

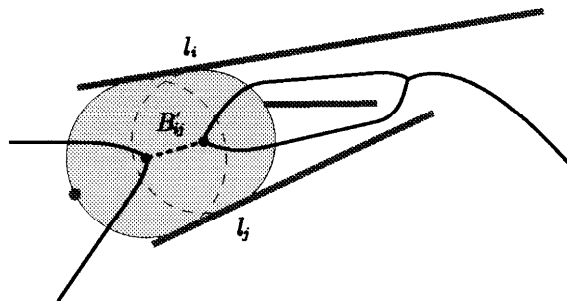


FIGURE 4. Two adjacent dual triangles defining a conflict region.

To locate a query point p within a Voronoi diagram $V(S)$, we construct a *point location structure* as follows. Let B'_{ij} be the part of the bisector B_{ij} that is defined by two adjacent dual triangles with l_i, l_j as the nodes of the common dual edge. The conflict region of these two adjacent dual triangles is the union of all circles of contact of l_i and l_j with center on B'_{ij} (see Figure 4). A line segment is in conflict with a conflict region iff both intersect.

Starting with four initial line segments, we incrementally build a history of conflict regions by inserting the line segments one by one using the technique below. Whenever a new line segment is inserted, new Voronoi edges - and, with them, new conflict regions - are created (cf. Figure 5). These overlapping conflict regions become sons of the destroyed conflict region; this leads to a directed acyclic graph, the so-called *history* of conflict regions. In this way, the leaves of the search structure represents the current conflict regions. We answer a point location query by following a path of conflict regions given by the point location structure; notice that it takes constant time to decide if a point lies within a given conflict region. Standard input randomization arguments show that the expected query time for a point p in the point location structure is $O(\log n)$; the expected size of the point location structure is $O(n)$ (see [3, 9] for details).

3.2 Insertion and deletion of a point

For inserting a point p into the generator set S , we adopt the framework of Devillers et al. [9] using our definition of the conflict region. We first

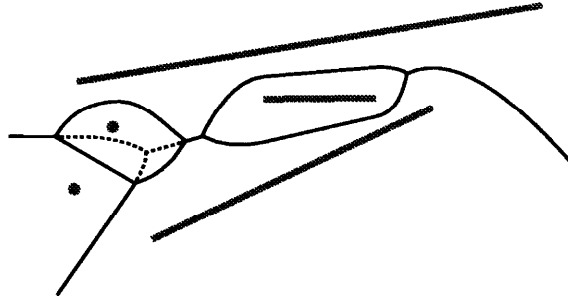


FIGURE 5. Update after an insertion of a point.

locate the point p in the Voronoi diagram $V(S)$ of line segments by searching the point location structure described above. This gives us a conflict region that is destroyed by the insertion of point p and, with that, an initial line segment l_i for which the dual edge (l_i, p) exists in the dual multigraph $D(S \cup p)$. Afterwards, breadth-first-search (similar to [23]) returns all conflict regions destroyed by p . The proof of Devillers et al. [9] carries over and proves that there are only an expected number of $O(\log n)$ swaps per insertion; the same time **suffices** to update the point location structure. So, the insertion of a point can be handled in $O(\log n)$ expected time.

In the case of the deletion of a point p from the generator set, S , the main problem is to update the history. The point p has been inserted some steps ago meaning that all following insertions in conflict with conflict regions created by the insertion of p have to be invalidated. In other words, the history has to be updated as if the insertion of p had never taken place and all other generators had been inserted in the same order. The deletion of the point p will influence all conflict regions including it; therefore we have to remove all these contiguous conflict regions. This leaves a “star-shaped hole” in the dual multigraph that has to be refilled in exact the way p had never been inserted. The key idea is a clever way to find all points that have been inserted after p and which are contained in the conflict regions created by p . These points will be reinserted in order to construct the successive part of the history without p (see [9] for details). With that, the deletion of a point p can be performed in $O(\log n)$ expected time if p is chosen with equal probability from the generators.

3.3 Insertion and deletion of a line segment

We insert a line segment l into an existing Voronoi diagram $V(S)$ of n generators by first inserting one of its endpoints p into $V(S)$. Afterwards, we use the fact that a line segment can be viewed as a rubber band pinned down at one end and dragged to the other. In this way, we expand the

inserted point \mathbf{p} to the line segment l using kinematic Voronoi diagram techniques. Obviously, the expansion of line segment l (cf. Figure 6) can be considered as the motion of the free endpoint of l in the Voronoi diagram $V(S)$ of line segments (compare Section 2). In our setting, the topological events are even simpler since the moving point - the free endpoint of l - only runs into (but never out of) circles of contact corresponding to Voronoi points in $V(S)$ (see [18, 28]).

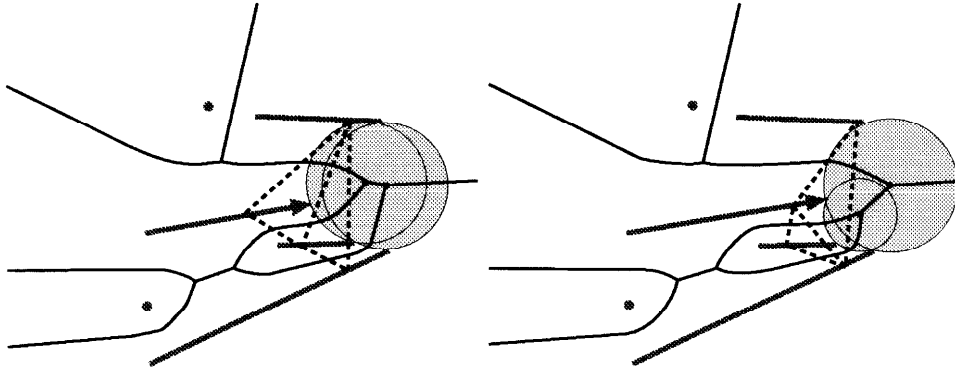


FIGURE 6. A Voronoi diagram during an expansion.

The overall time complexity of inserting a line segment is therefore determined by the cost of inserting a single endpoint of the line segment - which we know is $O(\log n)$ in the expected case - plus the time needed to process the topological events for expanding the line segment. Let d_l denote the degree of l in the dual multigraph $D(S)$. Notice that during the expansion of l , the degree of l in the dual multigraph $D(S)$ increases by one for each topological event until it reaches the value d_l . As only one line segment is moving at a time, the time for processing each topological event is $O(\log d_l)$. Using input randomization, the expected time for expanding the last line segment is

$$\frac{1}{n} \sum_{l \in S} O(d_l \log d_l) \in O(\log n)$$

as $O(\sum_{l \in S} d_l) = O(n)$. To delete a line segment l , we simply reverse this process. We shrink the line segment l to a single point \mathbf{p} again using kinematic Voronoi diagram techniques. Afterwards, we remove \mathbf{p} from the Voronoi diagram as seen before. So, if we select the line segment to be deleted with equal probability among all line segments, the expected time complexity of a deletion is the same as for an insertion which is $O(\log n)$.

Theorem 1 *The Voronoi diagram of n line segments can be computed with $O(n)$ expected space and $O(\log n)$ expected update time for inserting and deleting line segments.*

4 GIS - A new approach

Traditional GIS normally are coordinate-based systems, which means that a uniform grid will represent the geographic environment and this grid will define the environment model's resolution. The managed objects consists of points on the grid. For example, points of interest such as mountain peaks, corner stones, or even towns, depending on the resolution, will be represented by a point in the model. Looking at rivers and estates it is useful not only to employ single points but also line segments and polygons. At a basic level, a GIS has to manage any kind of information attached to points, line segments, and polygons. It also has to represent some kind of neighborhood relationship in order 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 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. Further implications of the given method to the area of GIS are given herein.

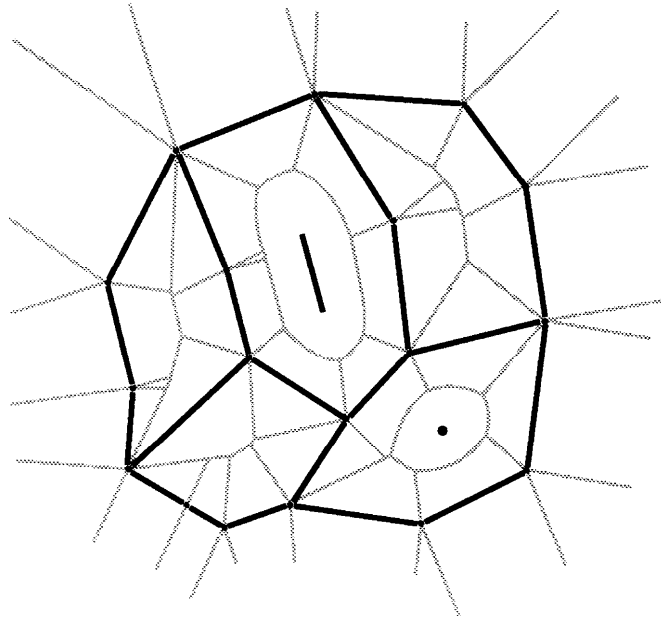


FIGURE 7. A Voronoi diagram of points and open line segments.

In order to design the complex objects of a polygonal map, we have to allow line segments that share endpoints. To this end, we split off the endpoint from all incident line segments; this leaves open line segments behind. All basic definitions and properties of Voronoi diagrams can be transferred. Figure 7 shows a Voronoi diagram of a polygonal map.

4.1 *Traditional GIS operations*

Within the Voronoi diagram, a map polygon creates Voronoi cells in its interior (see Figure 7). **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. 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.

Buffer-zone generation is performed by inserting the target object set (for example the boundaries of a set of lakes) into a Voronoi layer. 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 a basic property of the Voronoi diagram, being an expression of the “wavefront” or “prairie fire” analogy.

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 unconnected islands.

4.2 *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 [23] 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 the consequent sliver polygons when overlaid.

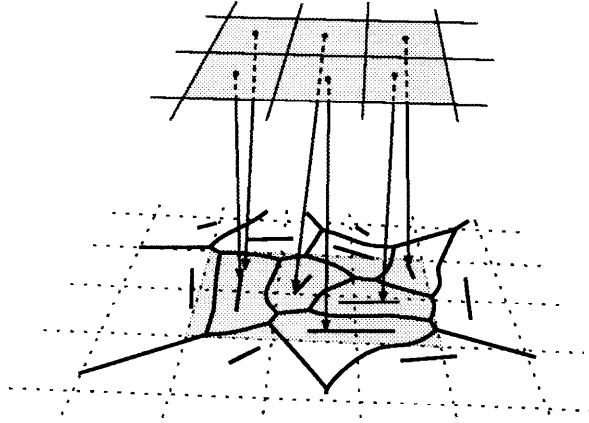


FIGURE 8. Point, location by a grid.

As a consequence of this design decision, there remains a basic question concerning how to access the portion of the triangulation associated with a particular map object, or with any specified (X,Y) location. The reason is that in practice, a simple walk through the triangulation from some previously accessed triangle (see, e.g., [23]) has been found quite sufficient to date, despite its poor theoretical efficiency. In practice, 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 triangle falling within the grid cell, followed by a simple walk, will significantly improve the performance.

4.3 *The intersection problem*

We showed how a line could be generated as the locus of a moving point as well as 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 [15] showed that a line segment can only be intersected by an expanding line segment if it has been a Delaunay neighbor 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 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. In addition, this allows to split and merge Voronoi diagrams along a common line (as shown in Figure 9).

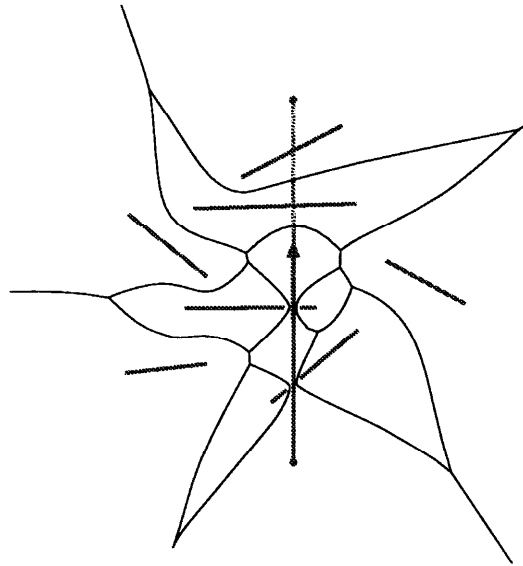


FIGURE 9. Separating and merging a Voronoi diagram.

4.4 *Update management*

Because 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, addition or 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 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, an incremental 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 [15]). It also permits the mixing of many types of data - fully connected polygons, connected hydrography, discrete points and line segments - within the same overlay.

4.5 *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 [19] for details).

4.6 *Map partitioning and parallel processing*

When maps become very large - maybe global - additional issues become critical. Then, the partitioning of the map into portions that are manageable in memory becomes a significant issue. Traditional GIS, with map sheet boundaries, does 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 (compare Figure 9).

Conclusions

Traditional GIS topology uses line intersection methods to create adjacency relationships, in a batch mode, after human digitizing has stopped. Modern Computational Geometry, as well as being concerned with line intersection detection and polygon building, is involved in a variety of other spatial operations, of which various forms of Voronoi diagrams or its dual are a significant, part.

Straightforward development of Voronoi methods has led from the simple incremental point Voronoi diagram algorithm, well known for many years, to the more recent moving point diagram, and finally to the fully dynamic diagram for points plus line segments. The Voronoi diagram – as a basic data structure in GIS – generates a form of “automated” topology, which may be used for the appropriate problem in the domain of Geomatics, and the basic framework for algorithm correctness and efficiency has been established.

Our approach has been implemented to a level that validates the proposed system design. However, even a successful analysis of future GIS needs leads to significant implementation issues, and the field of Computational Geometry could contribute greatly to a systematic resolution of an important field of spatial analysis. Nevertheless, the concept of a seamless, persistent, locally-modifiable spatial data structure that is always complete provides an important new approach to spatial data handling. It is clear that Geomatics has much to gain by collaborating with Computational Geometry, to clarify and simplify problems of spatial or topological relationships.

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.

5 References

- [1] G. Albers, L.J. Guibas, J.S.B. Mitchell, and T. Roos, **Voronoi diagrams of moving points**, Tech. Rep. 235, ETH Zürich, 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. 5th 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. 5th Intl. Symp. on Spatial Data Handling **SDH'92**, Charleston, pp 401-410, 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. 1st Annual European Symp. on Algorithms **ESA '93**, LNCS 726, pp 133-144, 1993.
- [9] O. Devillers, S. Meiser, and M. Teillaud, **Fully dynamic Delaunay 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, ***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.
- [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, ***An object-based dynamic spatial model, and its application in the development of a user-friendly digitizing system***, Proceedings, 5th Intl. Symp. on Spatial Data Handling ***SDH'92***, Charleston, pp 495-504, 1992.
- [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, ***Voronoi diagrams of line segments made easy***, Proc. 7th Canadian Conference on Computational Geometry CCCG'95, Laval University, Quebec City, pp 223-228, 1995.
- [19] C.M. Gold and T. Roos, ***Surface Modelling with Guaranteed Consistency - An Object-Based Approach***, Proc. Int. Workshop on Advanced Research in GIS IGIS'94, LNCS 884, pp 70-87, 1994.
- [20] P.J. Green and R. Sibson, ***Computing Dirichlet tessellations in the plane***, The Computer Journal, Vol. 21, pp 168-173, 1978.
- [21] L.J. Guibas, D.E. Knuth, and M. Sharir, ***Randomized incremental construction of Delaunay and Voronoi diagrams***, Proc. 17th Intl. Colloquium on Automata, Languages and Programming ***ICALP'90***, LNCS 443, Springer, pp 414 - 431, 1990.
- [22] L.J. Guibas, J.S.B. Mitchell, and T. Roos, ***Voronoi diagrams of moving points in the plane***, Proc. 17th Intl. Workshop on Graph Theoretic Concepts in Computer Science WG'91, Fischbachau, Germany, LNCS 570, pp 113-125, 1991.
- [23] 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.
- [24] 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

- [25] K. Mulmuley, ***Computational Geometry: An introduction through randomized algorithms***, Prentice Hall, 1994.
- [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. 16th Annual IEEE Symp. on Foundations of Computer Science FOCS'75, pp 151-162, 1975.
- [30] M. Teillaud, ***To 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 paramètres continus 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. Deuxième Mémoire: ***Recherches sur les parallélogrammes 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-Tree*** Proc. 7th Intl. Symp. on Spatial Data Handling ***SDH'96***, Delft, The Netherlands, August 1996.