

# Dynamic Spatial Object Condensation Based on the Voronoi Diagram

Weiping Yang and Christopher Gold  
Chaire industrielle en géomatique  
Centre de recherche en géomatique. Université Laval  
Québec, Qc. Canada G 1K 7P4  
E-mail: weiping@gmt.ulaval.ca

**Abstract:** A spatial data handling system deals with large spatial data sets which, in order to permit analysis, must be organized using certain geometric structures. This leads to specifications that identify diverse spatial data types or objects in the system. How to handle large, complex spatial objects in an efficient way is a demanding research topic in the fields of computational geometry and spatial database theory.

This paper reports a dynamic algorithm aimed to partition the Voronoi diagram with point and line details. The spatial object, i.e. the partitioned topological subspace, can reside in the same spatial page, with topological linkage to the original topological subspace removed: or it can be saved into secondary memory. The abilities of traversing the whole topological space and modifying partitioned subspaces are preserved in a seamless and hierarchical fashion. The spatial page holding the topological space with sparse spatial indices can be compressed. The condensed spatial objects are readily workable subspaces and can be loaded independently or attached to other subspaces partitioned with the Voronoi diagrams. The partitioning algorithm has a time and space cost proportional to the complexity of partitioning polygons. The time in compressing spatial page is linear to the number of objects compressed.

**Keywords:** *Voronoi diagram, Delaunay triangulation, computational geometry and topology, algorithms, spatial data modelling, paging and memory compressing hierarchical data structure, Geographic Information System.*

## 1 Introduction

A spatial data handling system deals with large spatial data sets which, in order to permit queries and analysis, must be organized using certain geometric structures. Well known geometric data structures include variations of trees, grids, and linear indexing schemes. These data structures are based on subdivisions of space into smaller ones such that overheads in retrieving geometric objects can be reduced. Depending on criteria on how space is split and concerns about geometric data types, the resulting data structures vary in shape and size, ability to represent geometric objects of differing complexity, and efficiency in performing queries.

GIS applications impose even more demanding requirements on the geometric data structure. Due to the nature and quantity of spatial data generally dealt with by a GIS, the geometric data structures should support dynamic modification of spatial objects, spatial queries involving complex data types such as polygons and polylines, secondary storage of spatial objects corresponding to disk pages, and an integration with other spatial data models (e.g. topological and thematic models).

The Voronoi diagram is a spatial data model which integrates both- geometric embedding of objects into the real space and topological incidence relationships between the elementary components of the geometric structure. In the Voronoi diagram, space is decomposed and each subspace is associated with an object that causes the subdivision. The incidence relationships between objects are observed from the tessellation and are conveniently represented by the dual structure, the Delaunay triangulation. The intuitive object embedding, the mathematical properties, and the dual representations of the Voronoi diagram make it a fundamental structure solving many graph and spatial proximity problems [1], [7].

---

Both authors gratefully acknowledge the support by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Association de l'Industrie Forestière du Québec (AIFQ).

The  $R^2$  Euclidean Voronoi diagram of spatial objects (points and lines) allocates space to each underlying object. Data structures have to be designed to maintain the result of this decomposition. This implies that, besides given geometric objects, the local adjacency relationships between objects which is principle to guide the partition must be preserved. In common practice the computer representation of the Voronoi diagram is based on the dual structure, the Delaunay triangulation. One of the obvious reasons for this is due to the simplicity of the Delaunay triangulation.

Among three approaches (divide-and-conquer, sweepline, and incremental) for the construction of the Voronoi diagrams, the incremental method is the one which is most simple to understand and implement. In the incremental approach, the knowledge of objects to be constructed is not assumed. The Voronoi diagram is modified dynamically when a new object is inserted or deleting an existing object from the object set. A randomized incremental algorithm [2] reports an optimal time  $O(n \log n)$  construction for  $n$  randomly input objects. Another incremental algorithm is based on the technique of kinematically adjusting the nearest neighbour Voronoi diagram [3], [8]. A recent experiment constructing  $n$  disjoint line segments with this technique reports an optimal  $O((n+t) \log n)$  time and  $O((n^2 \log n))$  time in worst-case [4], where  $t$  is the number of topological events occurred while expanding a line in between two points. The interesting characteristic about this approach is the kinematic knowledge about the neighbours of a changing object, which can find many applications in dynamic digitizing,- motion planning, and robot navigation.

Two problems can be observed in applying the dynamic Voronoi diagram as a fundamental data model for a dynamic GIS, within the framework of incremental construction. First, compared with other geometric and topological data models embedded in most current GISs, the Voronoi diagram apparently consumes more memory space. This observation is obvious because, in addition to create indices to geometric objects and their coordinates, the structure must maintain the topology for each single triangle. For a complex polygonal map composed of hundreds of thousands of short line segments, the size of the Delaunay triangulation could be extremely large. Second, since the Voronoi diagram is generated dynamically and incrementally, the input of objects does not necessarily follow a particular order. This may result in an instance of the data structure with poor spatial indexing. That is, objects and triangulation that are close in space may be stored far apart in memory, possibly scattered in different data blocks. This makes the ordering of the space according to some pattern impossible. It follows that no matter how big a map is, all the object and topological data must be loaded in memory in order to ensure the presence of the neighbourhood around any area of interest.

Large memory use with unnecessarily detailed objects lowers the overall performance of the system and hinders operators from concentrating on more important properties of a data set. Managing spatial data in a non-splittable fashion fails to meet the requirements for modern data structures which must be dynamic, hierarchical, with levels of details, and divisible into disk pages [6].

If some part of the Voronoi diagram of detailed objects can be taken away from the organizing structure and represented by some compound objects, the number of objects present in memory would be much smaller. Again, if the separated object space preserves partitions of the space, it can be loaded into memory any time as an independent workspace. The whole space can then be managed seamlessly as if there is no partition. Furthermore, if scattered objects and local topological relationships in a subspace can be compressed, they can be more efficiently managed and retrieved. This paper concentrates on developing these possible solutions by splitting the Voronoi diagram around boundaries of simple polygons.

## 2. Fundamentals

We first make clear some terms, notions, and data structures for objects and their space embedding which will be referred to in this paper..

### 2.1 The Objects and Their Data Structures

The set  $\mathcal{S}$  of  $n$  objects is composed of points (O-cells) and line segments (1-cells). Each line segment is treated as composed of two *half-he* segments and each half-line is associated with one end point. The data structures for the geometric part of the object set is composed of two files: an object file and a coordinate file. The object file consists of  $n$  tuples of two fields. For a point object, the first field points to the tuple in the coordinate file containing its coordinates, and the second field is null. For a half-line segment, the first field points to an end point in anti-clockwise sense, and the second field points to the other half-line (Figure 1)

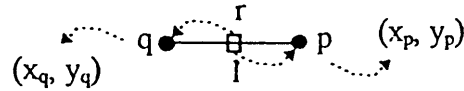


Figure 1. Object Data Structures

The separation of object identifications and their coordinates allows us to treat objects in a topological space and their coordinates in a geometric space. Some mapping functions can be designed to transform results from both spaces.' Typical transforming functions include scaling, rotating, and translating (Figure 2).

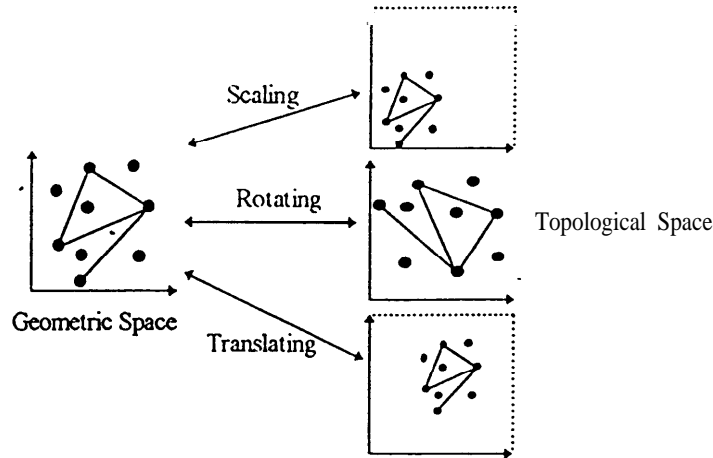


Figure 2. Mapping between Topological and Geometric Spaces

### 2.2 Voronoi Diagram: The Object Space Structure

Given a set of  $n$  ( $n > 1$ ) objects  $\mathcal{S} = \{\text{points, line segments}\}$  in the  $R^2$  Euclidean space, the Voronoi diagram,  $V(\mathcal{S})$ , is constructed in an incremental approach which partitions the plane into *Voronoi regions*,  $v$ , such that for two distinct objects  $s_i, s_j \in \mathcal{S}$ , the dominance of  $s_i$  over  $s_j$  is defined as the subset of the plane being at least as close to  $s_i$  as to  $s_j$ :

$$D(s_i, s_j) = \{x \in R^2 \mid \delta(x, s_i) \leq \delta(x, s_j)\},$$

for  $\delta$  denoting the Euclidean distance function.  $D(s_i, \bullet)$  is a closed half plane bounded by the bisector of  $s_i$  and  $s_j$ , denoted by  $B(s_i, s_j)$ :

$$B(s_i, s_j) = \partial D(s_i, s_j) = \{ x \in R^2 \mid \delta(x, s_i) = \delta(x, s_j) \}.$$

The Voronoi region  $v(s_i)$  is the portion of the plane lying in all of the dominances of  $s_i$  over the remaining objects in  $S$ :

$$v(s_i) = \bigcap_{s_j \in S - \{s_i\}} D(s_i, s_j).$$

If two bisectors  $B(p, q)$  and  $B(p, r)$  ( $p, q$ , and  $r \in S$ ) intersect, i.e.

$$B(p, q) \cap B(p, r) \neq \emptyset,$$

the intersection is called a *Voronoi point* and the bisector delimited by two consecutive Voronoi points is called a *Voronoi edge*. Two objects are *neighbours* if they share a common Voronoi edge (possibly extended to infinity). The Voronoi point has an equal distance to at least three objects and is therefore the circumcentre defined by these objects.

A  $v(s_i)$  is unbounded iff there exists a hyperplane separating  $s_i$  and its complements  $S - \{s_i\}$ . It turns out that unbounded Voronoi regions occur iff an object  $s_i$  lies on the boundary of the convex hull defined by  $S$ .

If  $S$  is a point set, the region associated with a  $p \in S$  will be the intersection of all half-planes containing  $p$  and delimited by the bisectors between  $p$  and the other members in  $S$ . It follows that the Voronoi regions are (possibly unbounded) convex polygons. This nice property does not hold for a general Voronoi region if  $S$  contains both points and lines. The bisector between a point and a line is no longer a straight line but a locus of points bisecting the point and the line. While the definition of the dominance  $D(p, l)$  still holds, the bound  $B(p, q)$  may become a parabola and the Voronoi region  $v(p)$  is not, in general, a convex polygon. Figure 3 shows the Voronoi diagram (gray) of points and lines (dark).

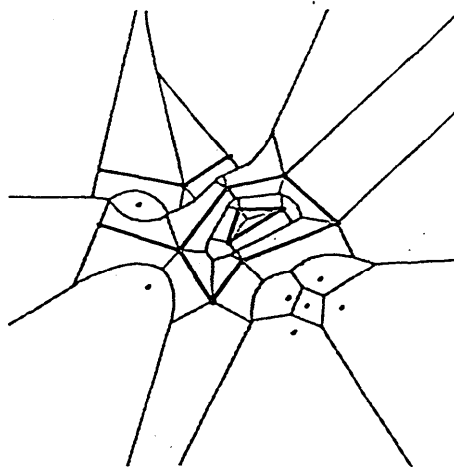


Figure 3. The Voronoi Diagram of Points and Lines

### 2.3 Delaunay Triangulation: The Dual Topological Space

The Voronoi edge separating  $p$  and  $q$  indicates that  $p$  and  $q$  are neighbours. A straight line linking  $p$  and  $q$  can be used to represent this relationship. It follows that for  $e$  Voronoi edges we can have the same number of straight lines. Each Voronoi point corresponds one triangle formed by three straight lines linking three objects on the circumcircle<sup>2</sup>. The union of these triangles is called the dual structure of the Voronoi diagram, the Delaunay Triangulation  $D(S)$ . Each straight line linking two objects is called a **Delaunay edge**. For  $S = (\text{points, line segments})$ , the corresponding Delaunay edges between a line and a point can be graphically implemented by linking the point and the middle of the line segment. Figure 4 illustrates the Delaunay triangulation (dotted) for the objects (solid) in Figure 3.

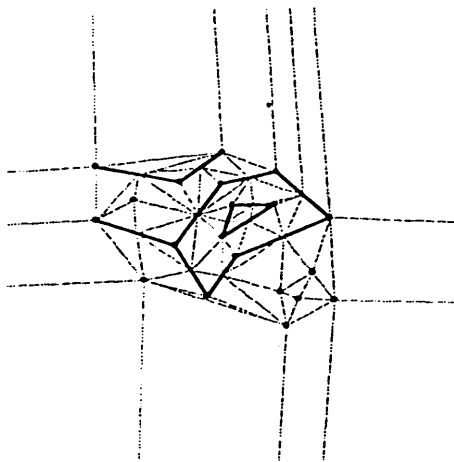


Figure 4. The Delaunay Triangulation of Points and Lines

Unlike its counterpart for a point set, the Delaunay triangulation for points and lines is not necessarily optimal in angles. It simply becomes a graph representing adjacency relationships between objects. Nevertheless, the circle tangent to three vertex objects is object-free, i.e. none of the given objects is contained in its interior.

The Delaunay triangulation is the actual structure maintained in the computer for the partitioning of space. In our implementation, the data structure for the Delaunay triangulation is composed of tuples, each with six fields:

$$DT(v1, v2, v3, t1, t2, t3)$$

where  $v1, v2, v3$  are pointers to three vertex objects, in anti-clockwise order, and  $t1, t2, t3$  are pointers to three adjacent triangles, opposite the equivalent vertices in the triangle (Figure 5).

Since the Delaunay triangulation also forms a complete decomposition of the object space and the adjacency relationships among its components (objects and triangles) are defined with respect to the object space and in data structures, a closed set of a few atomic operations based on the data structure of

---

<sup>2</sup>By saying so we assume that  $S$  is in general position, i.e. there are no cocircular cases and not all objects in  $S$  are colinear. These degenerate cases can be handled by some additional rules.

the Delaunay triangulation can be developed in the system similar to that used by the quad-edge data structure [5]. We call the object space structured by the Delaunay triangulation the **topological space**

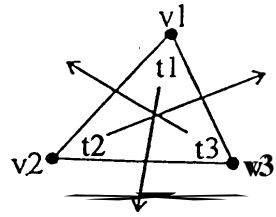


Figure 5. Data Structure for the Delaunay Triangulation

### 2.4 Topological Events

In the Voronoi diagram, objects are inter-related based on their neighbourhood relationships defined with bisectors. At any stage, the Voronoi diagram forms a complete partition (no overlapped Voronoi regions and no neglected space) if the circumcircle defined by any three adjacent objects is object-free. In a kinematic incremental construction system, a new object can be created and moved to some given location. As the object moves, the shape of its Voronoi regions changes. If the distance of the movement is small, the change of the shape will not alter the set of neighbours of this object. If the displacement is large enough, the neighbourhood relationships must be modified at some critical time, generating a topological **event** [3]. A moving object may trigger two types of topological events (Figure 6). When the Voronoi tile of the moving point is about to separate from the tile of a neighbouring object, it is moving out of the circumcircle defined by the neighbouring object and their two common neighbours (Figure 6a). At this stage, the triangle edge linking the moving object and the object to be separated should be switched. When the moving object is entering the circumcircle defined by two of its neighbouring objects and another non-neighbouring object, it is **moving in** and is about to include the object as its new neighbour (Figure 6b). At this stage, the triangle between two neighbours should be switched to link the moving point and the new neighbour.

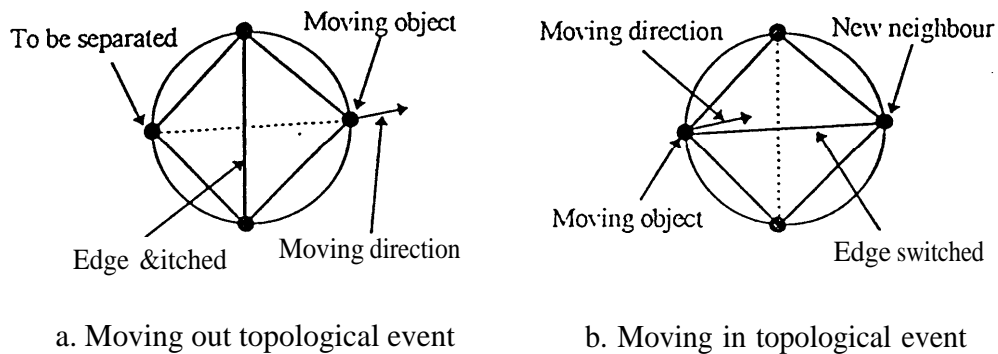


Figure 6. Topological Events of a Moving Object

### 3 Properties of the Line Voronoi Diagram

The treatment of a line segment,  $d$ , as two half-lines,  $(l, r)$ , implies that, in between each full line segment, there exists a bisector separating its two half-lines into two half planes. This observation explains some interesting properties demonstrated in the line Voronoi diagram.

By definition,  $l$  and  $r$  are two objects with zero distance in between. There is one and **only** one bisector  $B(l, r)$  passing through the zero sized space between them. This brings us the following axiom:

Axiom 3.1: Two half-lines ( $l, r$ ) of a line segment  $d$  are immediate neighbours.

If two objects lie on both sides of a line having their projections onto the range of the line, no matter how close they are, they can never be adjacent to each other. This is true because they have each half-line as an immediate neighbour. From Axiom 3.1, the bisector between the two half-lines ensures that no other bisectors can cross it and there are three bisectors in between them (Figure 7). From this argument, we have the following property stated as Corollary 3.1:

Corollary 3.1: If  $p, q \in \mathcal{S}$  lie in two half-planes divided by a line segment<sup>3</sup>  $d = \{l, r\}$ , it is impossible that their Voronoi regions  $v(p)$  and  $v(q)$  be adjacent to each other.

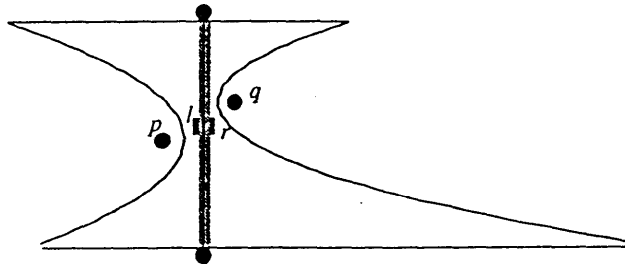
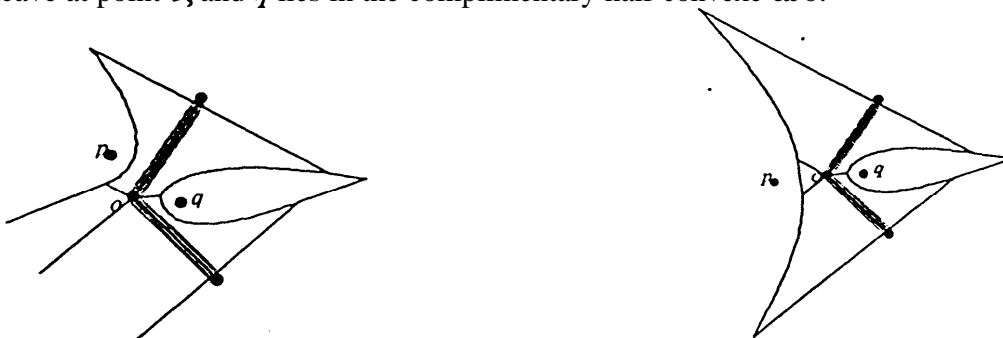


Figure 7. Voronoi Regions  $v(p)$  and  $v(q)$  are not adjacent

Corollary 3.1 is also true if two objects are divided by two incident line segments. We assume that two incident line segments are not colinear. As is shown in Figure 8, objects  $p$  lies in the half-plane concave at point  $o$ , and  $q$  lies in the complimentary half convexe at  $o$ .



a: The Projection of Object  $p$  is on a Line Segment

b: The Projection of Object  $p$  is not on a Line Segment

Figure 8. Two objects lying in two half-spaces are not adjacent

<sup>3</sup> For simplicity, we assume that a line segment has a finite length. The half-planes divided by the line segment are actually bounded by two perpendicular hyperplanes at both ends. This assumption can be removed when we later talk about polygons closed by line segments.

There are two possible cases for object  $p$ : 1) it can be projected on to a line segment; 2) it cannot be projected on to a line segment. In both cases,  $v(p)$  and  $v(q)$  cannot be adjacent, because object  $q$  can only have two half-lines forming a convex angle as its immediate neighbours. Object  $o$  can be adjacent to object  $p$  but will never be adjacent to object  $q$ .

Similarly, it can be deduced that

Corollary 3.2: If  $P = \{L\} \subset S$  is a closed polygon, then  $P$  partitions  $S$  into two topological subspaces,  $S_1$  and  $S_2$ . Any member object in  $S_1$  will not be adjacent to any object in  $S_2$  and vice versa. We denote  $P$  as the *partitioning polygon*.

With the above corollaries, we are now ready to prove the following theorems which form the basis of our algorithm.

Theorem 3.1: Rearranging the topological space inside a partitioning polygon  $P$  will not affect the topological space outside  $P$ , and vice versa.

Proof The rearrangement of the topological space inside  $P$  can be triggered by adding, deleting, or moving an object inside  $P$ . The sequence of topological events involves switching out or switching in neighbouring objects, possibly including line segments on  $P$ . However, Corollary 3.2 ensures that no objects outside  $P$  will be included as neighbouring objects of any object inside  $P$ . In other words, at any time, no triangle edges will be extended directly to link objects at both sides of  $P$ . Therefore, bisectors around outside  $P$  will not change and not does the whole outside topological space.0

Theorem 3.2: Two topological subspaces divided by a line segment are inter-related by two triangles in the interior of the line.'

Proof As is shown in Figure 8, each line segment  $d$  has two half-lines ( $l, r$ ) and is associated with two ending points,  $p$  and  $q$ . By Axiom 3.1,  $l$  and  $r$  are immediate neighbours, a Delaunay edge links  $l$  and  $r$ . Thus forms two triangles  $t_1$  and  $t_2$  inside the line. Referring to the data structure (Figure 5) for the Delaunay triangulation,  $t_1$  and  $t_2$  contains pointers to triangles, facing half-lines, in both top and bottom topological subspaces.

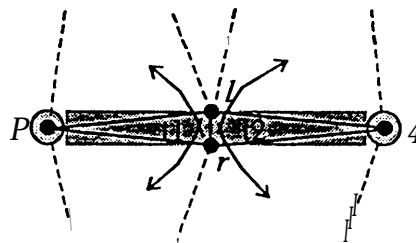


Figure 8. Critical Triangles Relating Two Topological Subspaces

The two triangles inside  $d$  are called *critical triangles* and are denoted by  $D(d)$ .

Theorem 3.2 shows that no matter how topological space changes in one side of the line, by modifying only pointers to changed adjacent triangles in either  $t_1$  or  $t_2$ , the whole topological space maintains integrity. There is no need to adjust the topological relationships on the other side of the line. This confirms Theorem 3.1. Theorem 3.2 can be readily applied to two topological subspaces divided by a chain of line segments and closed polygons. The proof of it is straight forward and is omitted from this paper.

#### 4 The Algorithm

Now we are ready to develop algorithms for condensing the Voronoi diagram of objects inside a simple polygon  $P$ . There are two steps. The first one partitions the topological subspaces delimited by the polygon, i.e., cut off adjacent relationships between the two subspaces such that details of the inside subspace can be removed from the original space and saved in one disk page. The existence of space bound at either subspace implies that  $P$  has to be repeated in both subspaces. The second step compresses the memory used for the Voronoi diagram inside the polygon.

##### 4.1 Partitioning Topological Subspaces

Given  $\mathcal{S}$ ,  $V(\mathcal{S})$ ,  $D(\mathcal{S})$ , and  $P = \{L\} \subset \mathcal{S}$ , where  $L$  are  $m$  line segments forming  $P$ , the objective is to cut  $D(\mathcal{S})$  into  $D(\mathcal{S}_1)$  and  $D(\mathcal{S}_2)$ , for  $\mathcal{S}_1 \subset \mathcal{S}$ ,  $\mathcal{S}_2 \subset \mathcal{S}$ , such that  $\mathcal{S}_1 \cap \mathcal{S}_2 = P$ , and  $D(\mathcal{S}_1) \cap D(\mathcal{S}_2) = D(P)$ , the critical triangles inside each line segment in  $P$ . We assume the original  $D(\mathcal{S})$  was not constructed with critical triangles for each line object in  $\mathcal{S}$ . This is advantageous in practice because critical triangles consume memory. They can be inserted when needed. Partitioning subspaces involves inserting critical triangles, modifying pointers in adjacent and critical triangles along the polygon boundary such that both subspaces do not point to each other. After this, the two subspaces delimited by  $P$  can be treated separately. Properties demonstrated in Section 3 ensure inter-influence-free operations within each topological subspace, provided that any line segment in  $P$  is not broken.

The technique of cutting the space along  $P$  can be described using an analogy named “guard the jail”:

Imagine a jail enclosed by a boundary  $P$ . A double-doored-wall (critical triangles) is built on each side of  $P$  (Figure 9). The inner doors (critical triangle edges extended from the inner half-lines) direct members of the jail to go out (arrows heading out). The outer doors (critical triangle edges extended from the outer half-lines) direct outside visitors in (arrows heading in).

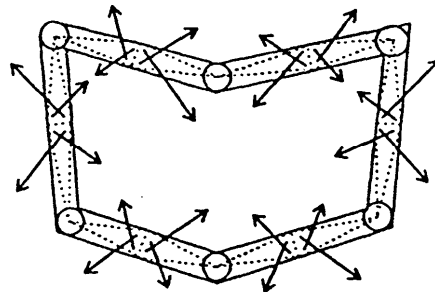


Figure 9. Critical Triangles linking Interior and Exterior Subspaces

To prevent members in the jail from getting out, the guards on!y have to devise paths out from inner doors to go around each incident corner and head to the adjacent inner door (Figure' 10). This guarantees that the entire subspace in the jail is visible, but not the subspace outside the wall.

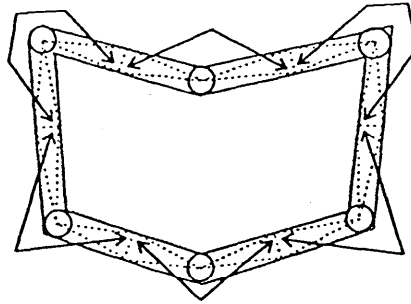


Figure 10. Taking off Linkage from In-Out

Similarly, to prevent visitors from getting into the jail, each of the paths led by outer doors is modified to head to adjacent outer doors incident to the common corners (Figure 11).

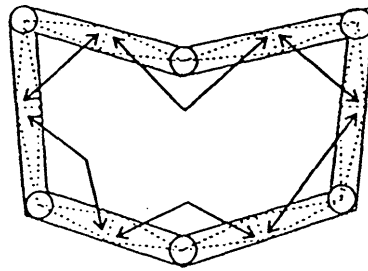


Figure 11. Taking off Linkage from Out-In

Obviously, the additional storage for critical triangles' costs  $O(m)$ . The time spent for the algorithm includes collecting adjacent triangle information around each line segment in  $P$ , inserting critical triangles in each  $L \in P$ , and modifying pointers in adjacent and critical triangles. The time spent on inserting critical triangles and modifying pointers in triangles is constant for each line and is proportional to  $m$ , the number of lines in  $P$ . Collecting adjacent triangles has an optimal time  $O(m * k)$  where  $k$  is the average number of adjacent triangles for all lines in  $P$ .

The above algorithm does not use any simplified version of geometric structures to mend the void space inside  $P$ . While the ability to traverse the whole space at the original level is not affected, the attempt to add new objects in the void space is prohibited. Editing the dropped space must be done at lower levels where the details reside.

If the void space is supported by some skeleton structures of  $P$  based on the Voronoi data model, e.g. by constructing a medial axis of  $P$ , the interior of the new topological space is then editable. This allows us to create structured layers of different thematic contents, aligned vertically and residing in the same memory. This "layered" memory management possesses interesting features which can be beneficial to many applications.

#### 4.2 Compressing the Space Bounded by $P$

When taking off the topological subspace  $D(S_1)$  from its original space  $D(S)$  and saving it as another spatial page, the memory populated with  $D(S_1)$  can be sparse and should be compressed to achieve both storage and spatial search efficiency.

The procedure of compressing memory is based on the flood-fill algorithm utilizing a priority queue. This is equivalent to traversing  $D(S_1)$  in preorder and has the effect of spirally expanding traversed space from any starting triangle. In the procedure below, we assume that both data structures for the Delaunay triangulation and the objects have a flag field of pointer type.

Procedure Compress-D(&)

Begin

Allocate memory A4 for the spatial page holding  $D(S_1)$ ;

Start from a given triangle  $oldt$ , get a new pointer  $newt$  from  $M$  and flag  $oldt$  with  $newt$ ;

Push the triangle into the priority queue;

While the queue is not empty, do the following:

Pop one triangle  $oldt$  from the queue;

Obtain  $newname$  of  $oldt$  from the flag field.

Loop for each of three adjacent triangles  $adjt$  in  $oldt$ , anti-clockwise

If  $adjt$  is not visited, get a  $newt$  from  $N$  for  $adjt$ , flag  $adjt$  with  $newt$ , push  $adjt$  in the queue;

Fill  $newname$  with  $newt$  and  $newt$  with  $newname$ ;

Examine the vertex  $oldv$  in  $adjt$ , opposing  $oldt$

If  $oldv$  is not visited, get a new pointer  $newv$  from  $N$  and flag  $oldv$  with  $newv$ ;

Fill  $newt$  with  $newv$ ;

Examine the vertex  $oldv$  in  $oldt$ , opposing  $adjt$

If  $oldv$  is not visited, get a new pointer  $newv$  from  $N$  and flag  $oldv$  with  $newv$ ;

Fill  $newname$  with  $newv$ ;

End Loop;

End While;

Save A4 into a disk page.

Release memory  $M$ .

End.

The procedure takes optimal  $O(nt)$  time, where  $nt = |D(S_1)|$ , the cardinality of the triangulation inside  $P$ . If  $nl = |S_1|$ ,  $nt$  is generally 2 ~ 4 times bigger than  $nl$ . Therefore the time complexity is linearly proportional to the number of objects compressed.

## 5 Conclusions and Future Work

We have devised algorithms for condensing topological spaces with detailed partitions by the dynamic point and line Voronoi diagram, based on the exploration of interesting properties demonstrated by the line Voronoi diagram. The algorithms are simple. The cost in time is proportional to the complexity of the polygon boundary in the splitting algorithm and is linear to the number of objects in memory compression. Additional storage for separating subspaces is caused by repeated storage of the cutting boundary and its critical triangles. The condensation creates a vertical dimension for structuring spatial data in the Voronoi diagram. It reduces the heavy load in memory for a large data and allows us to examine details in a dynamic and hierarchical fashion.

The condensation algorithm has been implemented on a PC running MS Windows with 8 MB memory, and tested with some polygonal subspaces of a small number of objects. Further work

includes more tests with variant data sets and collecting statistics for the running time. Extending the algorithm from simple polygons to general ones is also under consideration.

### References

- [1] F. Aurenhammer. Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure, *ACM Computing Surveys*, 23(3), 1991, 345-405.
- [2] 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, 1992, pp. 51-71.
- [3] C. M. Gold. Spatial Data Structures: the Extension from One to Two Dimensions, in: L.F. Parr (ed.), *Mapping and Spatial Modelling for Navigation*, NATO ASI Series, F, No. 65, Springer, Berlin, 1990, pp. 11-39.
- [4] C. M. Gold, P. R. Remmele, and T. Roos. Voronoi Diagrams of Line Segments Made Easy, *Proceedings of the 7th Canadian Conference on Computational Geometry*, August 1995, Quebec, Canada, pp. 223-28.
- [5] L. Guibas and J. Stolfi. Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams. *ACM Transaction on Graphics*, **4(2)**, 1985, pp. 74-123.
- [6] P. van Oosterom. *Reactive Data Structures for Geographic Information Systems*. Oxford University Press. 1993.
- [7] F. Preparata and M. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, 1985.
- [8] T. Roos. *Dynamic Voronoi Diagrams*. Ph.D Thesis, Universitat Würzburg, Switzerland, 1991.