VORONOI DIAGRAM AND DELAUNAY TRIANGULATION SUPPORTING AUTOMATED GENERALIZATION

Wanning Peng, K. Sijmons, A. Brown
Department of Geoinformatics, ITC
P.O. Box 6, 7500 AA Enschede, The Netherlands

Abstract

This paper presents the interim results of some research on how to apply the Voronoi diagram and the Delaunay triangulation to support some aspects of generalization decision-making. Particular attention is given to pattern detection, spatial conflict detection, neighbour identification, object displacement, and object aggregation/merging. Concrete examples that illustrate the power of the method are also given. The work has been implemented using C++ in a Windows environment.

1 Introduction

Apart from the lack of generalization theory and rules which are difficult to formalize due to the partly subjective nature of traditional map generalization, technical difficulties associated with geographical structural analysis by computers in order to make a generalization decision, are equally important impediments in developing an operational approach to automated generalization.

In many cases, generalization decision-making requires not only the information about the target object, but also the information about the environment (such as neighbours) that provides **constraint** over the behaviour of the object. For instance, if a group of objects forms a certain kind of significant (regular) pattern, it should be considered as a whole when dealing with any of its members in a generalization process. Object displacement and aggregation/merging are also constrained by the surrounding objects: moving an object should not cause it to hit others; two objects should not be aggregated if there is another object between them. Decisions should not be made without taking constraint (which can be thematical or geometrical) into account.

One of the major reasons for requiring generalization is spatial conflict caused by scale reduction. How to detect a spatial conflict (including location, orientation, and degree of conflict) is a question that must be answered before starting the decision-making process. Detecting a spatial conflict and constraint are two important aspects of geographical understanding in the sense of automated generalization decision-making. They require adequate supporting data structures to avoid complicated algorithms and heavy computation. The (adjacent) topological data structure is the one that can serve the purpose in many cases. However, this structure is not appropriate for handling geometrically unconnected objects for which structures that support spatial analysis and queries involving spatial proximity are required.

The Voronoi diagram and Delaunay triangulation are important data structures in computational geometry, and have been used in many applications (e.g. DTM, closest-site, shortest-path, and robot motion, etc.) [2,8]. This paper presents a research on how to apply these data structures to support our problem solving, by particularly looking into pattern detection, spatial conflict detection, neighbour identification, object displacement, and object aggregation/merging. Since the Voronoi diagram and the Delaunay triangulation are geometrically dual, our discussion will concentrate on the Delaunay triangulation.

2 Basic properties of the Delaunay triangulation

The Delaunay triangulation is a triangulation of a set of points V with the empty circle property, that is that the circumcircle of any of its triangles does not contain any point of the given set [9]. The Delaunay triangulation is unique and locally equiangular [10], hence, it maximizes the minimum angle of its triangles compared to all other triangulations (see Figure 1).

Constrained Delaunay triangulation is an extension of the standard Delaunay triangulation by allowing (pre-described) non-intersecting line segments (except at their endpoints) to be forced in as part of the triangulation. Therefore, triangles that contain any of such pre-described edges may not be Delaunay triangles (see Figure 2).

Detailed discussions about the Voronoi diagram, the Delaunay triangulation, and constrained Delaunay triangulation can be found in [2,4,8,9,10].

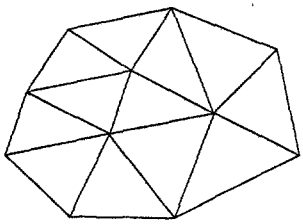

Figure 1: An example of standard Delaunay triangulation

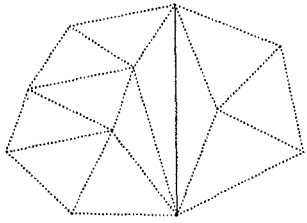

Figure 2: An example of constrained Delaunay triangulation

3 Applications in automated generalization

Although the Voronoi diagram and Delaunay triangulation are powerful tools in solving many computational problems and have attracted enormous research interest in many areas, they have received only a little attention from the world of automated generalization [3,7]. In fact many generalization problems can benefit from applying these structures.

3.1 (Regular) Linear group detection

Regular patterns may be detected using the Delaunay triangulation as the structure reflects the distribution of the point set. As shown in Figure 3, dense points correspond to small triangles, while sparse points result in big triangles. Evenly distributed points give rise to triangles of similar sizes and shapes. For a subset of points having an orientation sensitive distribution, the shapes of the resulting Delaunay triangles exhibit a corresponding directional sensitivity. These are the properties based on which we could detect a regular pattern from a set of points/objects. Note that Ahuja [1] has given a similar observation of these properties (in relation to the Voronoi diagram). The following discussion concentrates on the detection of regular linear groups of objects within a larger group. However, the same methodology can also be applied to other kinds of patterns after modification.

The particular example which will be considered concerns the detection of linear groups of islands (Figure 5a). The human eye will detect such an linear group when 1) the centroids of the islands lie

on a straight or curved line; 2) the islands are rather similar in size; and 3) the distances between neighbouring islands in the group are similar and are less than the distance to the nearest island outside the group, from any member of the group. The algorithm which has been developed is based on this understanding and the Delaunay triangulation of the islands.

Since the algorithm is related to triangles, we need to find some variables to describe the nature of a triangle. In addition to area and perimeter, we introduce two more variables to describe the orientation and size of a triangle, called **bearing** and **span**. Their definitions are given below:

- Let $V=\{v_1,v_2,v_3\}$ be the vectors from the vertices $P=\{p_1,p_2,p_3\}$ of a triangle to the mid-points of their opposite edges $E=\{e_1,e_2,e_3\}$, the **bearing** of the triangle is defined as the orientation of $v_i \in V$ of which length $|v_i|$ is the maximum; the **span** is defined as the length of the corresponding edge e_i. See Figure 4.

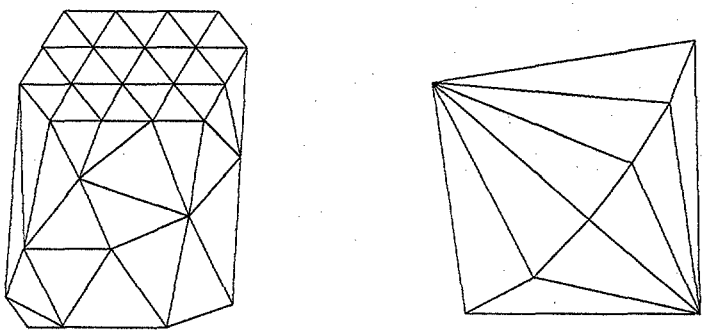

Figure 3: The structure reflects the distribution of the point set

Because our method is based on the similarity of related triangles, we still have to define criteria to determine the similarity and relation of a (sub)set of triangles:

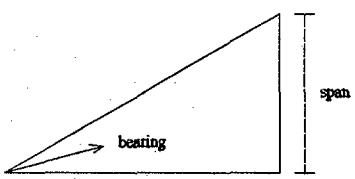

Figure 4: bearing and span of a triangle

- A (sub)set of adjacent triangles $T=\{t_1,t_2,...,t_n\}$ have radial orientations if their bearings start from the same point. This point is called the **radiant-point** of T.
- A (sub)set of adjacent triangles $T=\{t_1,t_2,...t_n\}$ have similar spans, if for each $t_i \in T$ the condition $|s_i-\bar{s}| \le f\bar{s}$ is true. Where s_i is the span of t_i, $\bar{s}$ (called local average) is the average of s_i, and $f=1-(\bar{s}/\bar{S})^k$ {if $\bar{s}<\bar{S}$}, or $f=1-(\bar{S}/\bar{s})^k$ {if $\bar{s}>\bar{S}$}. $\bar{S}$ (called global average) is the average of s_j of all the triangles in the network, k ($k \ge 1.0$) is a factor that controls the tolerance for the variation of s_i. This criterion is given based on the observation that the larger the difference between a local cluster (as a whole) and its global environment, the less critical the requirement concerning the differences among the members within

the local cluster. In other words, if a local cluster is very different from its environment, we may allow the cluster to have members which are slightly different from the rest of the group. The tolerance of the cluster difference depends on how the difference between the local cluster and the global environment.

The procedure of detecting a regular linear group within a larger group is formulated as follows:

- calculate the centroid and area of each polygon.
- triangulate the centroids (Figure 5b).
- find a subset of adjacent triangles $T_s=\{t_{s1},t_{s2},...,t_{sn}\}$ having radial orientations (Figure 5b, triangles with thin solid lines and connected by arcs).
- from T_s, select a subset of adjacent triangles $T_t=\{t_{t1},t_{t2},...,t_{tm}\}$ having similar spans (Figure 5b, triangles with thick solid lines).
- from T_t, select all the vertices except the radiant-point and form in sequence a list of points $P=\{p_1,p_2,...,p_q\}$ which are the centroids of the corresponding islands.
- from P, select a sublist of "adjacent" points $P_s=\{p_{s1},p_{s2},...,p_{st}\}$ under the condition that the areas of all the corresponding objects are similar. The similarity of areas is defined in the same way as for spans.
- P_s forms an linear group (islands connected by solid lines in Figure 5c), the characteristics of which can be described by the following variables, i.e., the average distance $\bar{D}$ between adjacent points of P_s and its standard deviation S_D, the average angle $\bar{L}$ included between adjacent edges connecting two adjacent points of P_s, and its standard deviation S_L, the average area $\bar{T}$ and its standard deviation S_T. Based on these variables, extend both sides of the existing pattern by including new islands that fall into the pattern but were not detected by the above procedure: including neighbours which meet the following three conditions: 1) $|Di-\bar{D}|<KS_D$; 2) $|Li-\bar{L}|<KS_L$; 3) $|Ti-\bar{T}|<KS_T$. Where K equals 2 for 95% confidence (or probability), or equals 3 for 99% confidence (islands connected by dashed lines in Figure 5c).

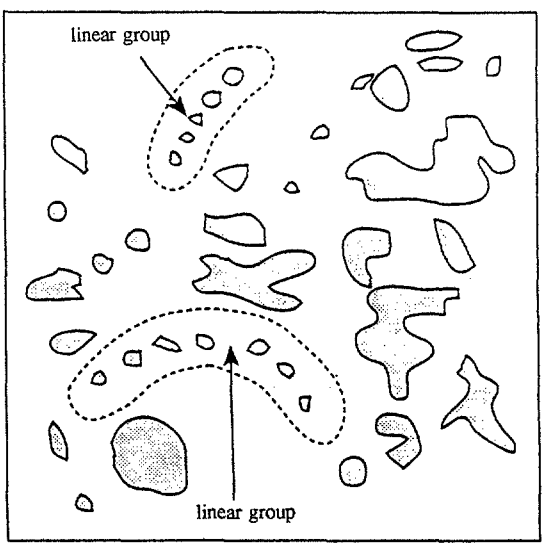

Figure 5a: A group of islands (source: [6])

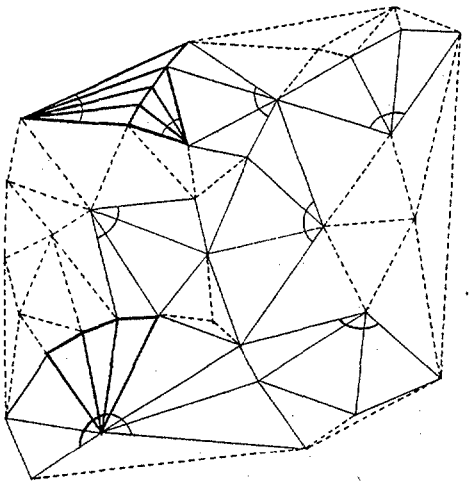

Figure 5b: Delaunay triangulation of the centroids

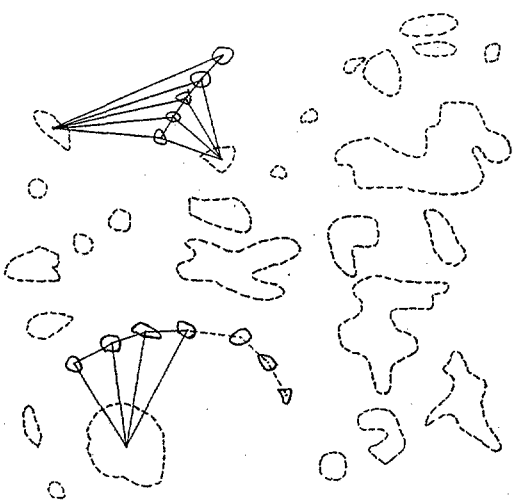

Figure 5c: (Regular) Linear groups detected

3.2 Spatial conflict detection

Detecting a spatial conflict may involve complicated algorithms and heavy computation without adequate supporting data structures. Moreover, apart from knowing which objects are in conflict, we also need to know the status of the conflict (such as location, orientation, and degree) in order to solve the problem properly. With constrained Delaunay triangulation, these problems can be solved easily.

Let us consider the example shown in Figure 6a. Given a minimum accepted distance D_{min}, we assume that we want to know if object C is in conflict with others, and if yes, what is the status. To answer the question, we follow the following procedure:

- construct a constrained triangulation of all the corner points, using polygon edges as constraints (Figure 6b).
- find Delaunay neighbours by checking edges connecting object C and other objects (solid connecting lines in Figure 6c).
- for each Delaunay neighbour, find all the triangles $T=\{t_1,t_2,...,t_n\}$ which connect object C and the currently considered neighbour (e.g., triangles T_1 and T_2 for neighbour A, Figure 6d).
- assuming that v_i, v_j, and v_k are three vertices of a triangle, where v_i and v_j are corner points of object C (or the neighbour), v_k is a corner point of the neighbour (or object C), let d_i be the distance between v_k and v_i, d_j be the distance between v_k and v_j, d_h be the distance between v_k and the line l_{ij} connecting v_i and v_j, for each triangle $t_i \in T$, calculate the minimum distance d_{min} between v_k and l_{ij}: let $d_{min}=\min(d_i,d_j)$ if any of the two vertex angles (associated with v_i and v_j respectively) is an obtuse angle, otherwise, if none of them is an obtuse angle, let $d_{min}=d_h$. Note: $\min(d_i,d_j)$ means that select d_i if $d_i<d_j$, otherwise select d_j.
- among all the d_{min}s, select the one d_0 which has the minimum value.
- If $d_0<D_{min}$ then object C is in conflict with the neighbour. The orientation is identified by the direction associated with d_0. This direction is also the most efficient direction for the conflicted objects to move away from each other in order to solve the conflict. In other words, *if moving along this direction, the displacement is minimized*, that is, $D_{min}-d_0$ (Figure 6e).

3.3 Neighbour identification

In subsection 3.2, we have seen how neighbours can be found out by using the constrained Delaunay triangulation. For two objects, if any part of them are connected by a triangle edge, then they are neighbours. Because this neighbour relation is different from the adjacent relation, we call a neighbour identified by the Delaunay triangulation **Delaunay neighbour**. Delaunay neighbours have several important characteristics [1]:

- The Delaunay neighbours are symmetric by definition.
- The Delaunay neighbours of an object may change if the object changes its spatial status.
- The Delaunay neighbours of an object are not necessarily its *nearest* neighbours. The Delaunay neighbours of an object must "surround" it. Hence, distant points may be accepted as neighbours on the sparsely populated side, whereas relatively close objects may not be accepted as neighbours on the dense side if they occur "behind" other closer objects.

Many automated generalization problems can benefit from the Delaunay neighbours. For example, for a given object, the analysis for spatial conflicts, object aggregation/merging, and object displacement will be limited to only its Delaunay neighbours.

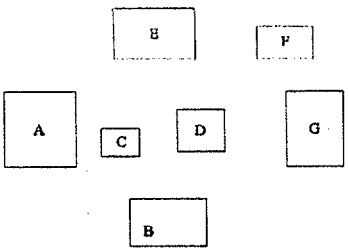

Figure 6a: A group of unconnected objects

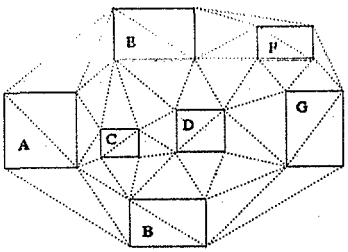

Figure 6b: Constrained Delaunay triangulation

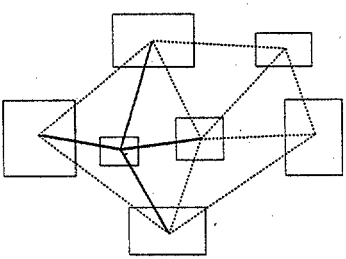

Figure 6c: Delaunay neighbours

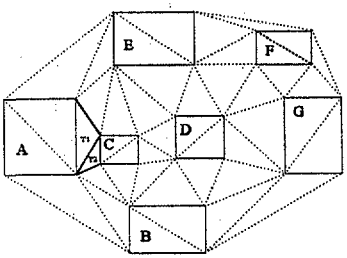

Figure 6d: Triangles connecting A and C

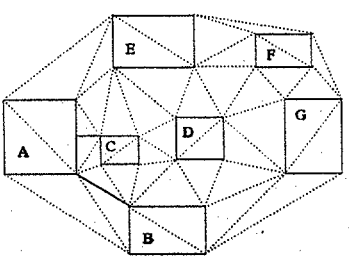

Figure 6e: Status of spatial conflicts

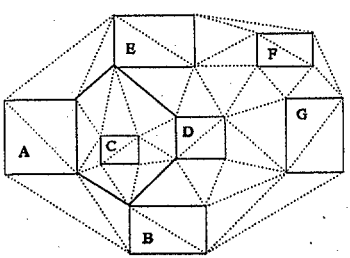

Figure 6f: Safe-region

3.4 Object displacement

Object displacement is one of the major problems in the course of automation of manual generalization. One of the main reasons is that when an object has to displace because of spatial conflict(s), there are no adequate measures and sufficient information to guide the movement of the object so that it will not "hit" or "cross" other objects. Another reason is that, in some cases, the displacement of an object relies on the displacement of another (neighbour) object, the resolution of which is even more complicated and difficult. It is unwise and perhaps infeasible to try to solve these problems without using an adequate data structure. In the rest of this subsection, we will discuss how these problems can be solved by using the Delaunay triangulation. First let us consider the problem of "safe movement", that is, how can we control an object so that its movement will not "hit" or "cross" other objects.

In Figure 6, if studied carefully, we may find, that each object is surrounded by a group of triangles T_s which do not enclose any other part of the given set of objects. The polygon formed by the external edges of the triangles of T_s is called the **safe-region** of the enclosed object o_i and is denoted by p_i. p_i has an important property, that is, it encloses only object o_i, and as long as o_i is inside the region, it is safe. In other words, o_i can move freely without "hitting" or "crossing" any other objects as long as it keeps itself inside p_i (that is why we called p_i safe-region). Apparently, the bigger the p_i, the more the space that o_i can move around. Of course, p_i is not the maximum safe-region of o_i, however, it is an efficient and useful measure to guide the displacement of o_i in the sense of solving spatial conflict, because the constraints by the surrounding objects are embedded in it. Because of the difference between p_i and the maximum safe-region, sometimes p_i may reject o_i's request for moving to a place which has no danger at all. The important fact is that p_i will never allow o_i to move to a dangerous place, however. In many cases, p_i reject o_i's request because it is not necessary to move in that direction. A solution to the problem is to check with all the neighbours to see whether o_i (in the new position) will be in conflict with them or not, after p_i has rejected its request. Because we want to keep a certain distance between two objects, we must apply a buffer on top of o_i when it moves. The width of the buffer should equal the requested distance.

The problem that "the displacement of an object relies on the displacement of another" can be solved by using the safe-region and neighbour relation:

- in subsection 3.2, we were able to detect a spatial conflict and know the optimum direction for displacement. Suppose object C (see Figure 6) is in conflict with object A, and C must move away from A.
- from subsection 3.2 we know the best direction α is to the right and horizontal.
- from the above discussion we can find the safe-region p_C of object C.
- apply a buffer on the top of C. The new and temporal object is denoted C′.
- let $C''=C'+\Delta C$, $\Delta C=f(d,\alpha)$, where $d=D_{min}-d_0$ (see subsection 3.2).
- if any vertex in C″ is outside of p_C, or any vertex in p_C falls into C″, then C can not move. Assuming that this is the case. Note that in practice, if the movement along α fails, we still can try by slightly changing the direction.
- by comparing α with the azimuths from C to its neighbours, we know object D is probably the one which blocked the movement of C′, therefore, it must move away.
- the request for the displacement of $\Delta C=f(d,\alpha)$ is now propagated onto object D.
- carry out a similar procedure and try to move object D. If D can not move, then pass the request to another object in the same way as done for C, or simply stop the process. How far this propagation should go depends on the practice. In principle, it can go on until it reaches the outspace. In practice, however, we may stop the propagation if the second or third object fails to displace, and think about other solutions.

- displace C by ΔC if D can move.

This is a simple example, further research on more sophisticated procedures is necessary.

3.5 Object aggregation

One of the major difficulties in object aggregation or merging is to decide which two (or more) objects should be aggregated. Generally speaking, only adjacent or the most proximate objects should be aggregated. Two objects should not be aggregated when there are others between them, that is, an object can only be aggregated with its neighbours. Based on this understanding, we can transform this problem into an operation of searching neighbours for an object, and then decide with which neighbour the object should be merged according to other criteria such as distance, area, and theme. This operation is relatively simple for geometrically connected objects because the powerful (adjacent) topological data structure (e.g., the formal data structure FDS [5]) can apply. For geometrically unconnected objects (such as islands, separated buildings), again, we can apply the Delaunay triangulation (see subsection 3.3). Note that, if dealing with area or line objects, we should use constrained Delaunay triangulation instead of the standard Delaunay triangulation, in order to prevent a triangle edge from crossing any objects.

4 Conclusion

In this article, we have shown the potential of the Delaunay triangulation (both unconstrained and constrained) in solving many of our problems, as a complement to (adjacent) topological data structure. All the examples presented in this article have been implemented using C++ in a Windows environment. The Voronoi diagram also can be used to solve these problems. However, Voronoi polygons can vary in number of edges. The associated Delaunay triangulation is simpler to handle (always three edges and three vertices), therefore, it seems more attractive.

The issue of geographical analysis is one of the major problems to be resolved for the automation of manual generalization: generalization rules must be translated into equivalent geometrical descriptions and operations, generalization decisions can be made only after having a good understanding of the environment and background, generalization operations have to deal with the geometrical descriptions of relevant objects. This issue must not be neglected or avoided because of the difficulty. However, we also should not try to solve these problems without having an adequate supporting data structure.

5 References

[1] Ahuja, N., 1982. Dot Pattern Processing Using Voronoi Neighbourhoods. IEEE, Vol. PAMI-4, No. 3, pp. 336-343.

[2] Aurenhammer, F., 1991. Voronoi Diagram - A Survey of a Fundamental Geometric Data Structure. ACM Computing Surveys, Vol. 23, No. 3, pp.345-405.

[3] Chithambaram, R., Beard, K., and Barrera, R., 1991. Skeletonizing Polygons for Map Generalization. Technical Papers ACSM-ASPRS Convention, Vol. 2, Cartography and GIS/LIS, pp. 44-55.

[4] Floriano, L., and Puppo, E., 1988. Constrained Delaunay Triangulation for Multiresolution Surface Description. Proceedings of the 9th International Conference on Pattern Recognition, pp. 566-569.

[5] Molenaar, M., 1991. Terrain Objects, Data Structures and Query Spaces. In: Geo-Informatik (Schilcher M., ed.), Siemens-Nixdorf Informationssystemme A.G.,Munchen, pp. 53-70.

[6] Muller, J. C., 1992. Area-patch generalization: a competitive. The Cartographic Journal, Vol. 29, pp. 137-144.

[7] Muller, J. C., Weibel, R., Lagrance, J. P., and Salge, F., 1993. Generalization: State of the Art and Issues. GISDATE Task Force on Generalization 1993, European Science Foundation, 21p.
[8] Okabe, A., Boots, B., and Sugihara, K., 1994. Nearest neighbourhood operations with generalized Voronoi diagrams: a review. INT. J. Geographical Information Systems, Vol. 8, No. 1, pp.43-71.
[9] Preparata, F., P., and Shamos, M. I., 1985. Computational Geometry: An Introduction. Springer, New York.
[10] Sibson, R., 1977. Locally equiangular triangulations. Comput. J., Vol. 21, No. 3, pp. 243-245.