7

A Methodology for Automated Cartographic Data Input, Drawing and Editing Using Kinetic Delaunay/Voronoi Diagrams

Christopher M. Gold[1], Darka Mioc[2], François Anton[3], Ojaswa Sharma[3], and Maciej Dakowicz[1]

[1] Faculty of Advanced Technology, University of Glamorgan, Pontypridd, Wales, CF37 1DL, UK
{cmgold,mdakowic}@glam.ac.uk

[2] Department of Geodesy and Geomatics Engineering, University of New Brunswick, P.O. Box 4400, Fredericton, New Brunswick, Canada E3B 5A3
dmioc@unb.ca

[3] Informatics and Mathematical Modelling, Technical University of Denmark, Richard Petersens Plads, Building 321, 2800 Kgs. Lyngby, Denmark
{fa,os}@imm.dtu.dk

Summary. This chapter presents a methodology for automated cartographic data input, drawing and editing. This methodology is based on kinematic algorithms for point and line Delaunay triangulation and the Voronoi diagram. It allows one to automate some parts of the manual digitization process and the topological editing of maps that preserve map updates. The manual digitization process is replaced by computer assisted skeletonization using scanned paper maps. We are using the Delaunay triangulation and the Voronoi diagram in order to extract the skeletons that are guaranteed to be topologically correct. The features thus extracted as object centrelines can be stored as vector maps in a Geographic Information System after labelling and editing. This research work can also be used for updates from sources that are either paper copy maps or digital raster images. A prototype application that was developed as part of the research has been presented.

We also describe two reversible line-drawing methods for cartographic applications based on the kinetic (moving-point) Voronoi diagram. Our objectives were to optimize the user's ability to draw and edit the map, rather than to produce the most efficient batch-oriented algorithm for large data sets, and all our algorithms are based on local operations (except for basic point location). Because the deletion of individual points or line segments is a necessary part of the manual editing process, incremental insertion and deletion is used. The original concept used here is that, as a curve (line) is the locus of a moving point, then segments are drawn by maintaining the topology of a single moving point (abbreviated as MP hereafter, or the "pen") as it moves through the topological network (visualized as either the Voronoi diagram or Delaunay triangulation). This approach also has the interesting property that a "log file" of all operations may be preserved, allowing reversion to previous map states, or "dates", as required.

7.1 Introduction

In recent years, GISs have undergone major changes. Commercial GISs can now handle dynamic updates and topology, versioning and advanced data editing. However, there are still many functionalities missing in commercial GISs. One of them is automated raster to vector conversion. Automated feature extraction from images is the key to real-time map updates, automated digitization and many other tasks that involve robotic vision. This research work is concerned with centreline extraction from colour scanned maps. Various map features have been considered but our primary interest is in linear features, since they provide excellent identification of objects in any analysis.

However, in many cases in cartography the observable features are man-made, and known to be composed of straight edges. While these may be approximately extracted by various generalisation techniques, (e.g. early Douglas-Peucker methods), they are then incompatible with the point Delaunay triangulation (abbreviated DT hereafter) / Voronoi diagram (abbreviated VD hereafter) forming the underlying spatial structure.

In an interactive editing environment it is desirable to be able to adjust features individually within the current spatial model — in this case the VD. The problem has previously been addressed for the case of terrain modelling, where the terrain is represented by the DT and additional constraints such as river bottoms need to be added. Here the constrained DT is often used — this is constructed from the simple DT by inserting triangle edges of arbitrary length, flagged so that they may not be changed by subsequent triangle flips. This may also be used with the results of our feature extraction from skeletons, rather than for terrain.

One problem, however concerns the meaning of the triangle edges — some represent particular linear features and some merely represent Voronoi adjacency relationships between data elements. In particular, features usually require additional attribute information, unlike the remaining edges. The solution in this case should be a complete separation of the features (points, line segments) and the relationship information (topology: triangulation). For urban features in particular, the line segment VD holds strong theoretical attractions, as the dual triangulation merely represents the topological relations, while the map features are preserved separately with the associated attribute data. While functioning software exists (see [45]) it is not interactive, thus not allowing the manual editing we desire, and the algorithms are extremely complex.

This chapter is organised as follows. Section 7.2 introduces the Voronoi diagram of points and open oriented straight line segments and describes its core properties used in Section 7.3. Readers interested only in applications may skip this section and go directly to Section 7.3. Section 7.3 describes the extraction of features from images to produce "polylines" representing the desired entities — coastlines, rivers, etc. Section 7.4 addresses this issue by looking at the Moving Point VD as the basic algorithm, with modifications added to generate first the Constrained DT and secondly the Line Segment VD, with operations that permit the insertion and deletion of map features. This has not previously been achieved robustly — that is, handling degenerate cases with finite precision computer arithmetic.

7.2 Introduction to the Voronoi Diagram of Points and Open Oriented Straight Line Segments

Let us first introduce the definition of the Voronoi diagram for a set of sites (i.e. objects or subsets) in the Euclidean affine plane. Let us now consider a set of points and open oriented[1] segments $\mathcal{O}=\{O_1,...,O_s\}$ in the Euclidean plane. We recall that an open oriented segment is the open set formed from a (closed straight line) segment $[AB]$ by removing its extremities A and B and considering as positive orientation along the line (AB) the orientation from A to B. The distance from a point M to an object O_i is defined as: either the Euclidean distance between the two points if O_i is a point, or ∞ if the point M lies on the right of O_i, or $d\,(M, O_i) := \inf_{P\in O_i} d_e\,(M, P)$ where d_e denotes the Euclidean distance between two points, otherwise. The Voronoi diagram for a set of points and open oriented line segments is a generalized Voronoi diagram. Let us now introduce the definition of a generalized Voronoi diagram (see (45)), in order to be able to introduce the definition of the Voronoi diagram for a set of points and oriented line segments as a generalized Voronoi diagram. Let S be the metric space in which we place ourselves (typically $\mathbb{R}^2$).

Definition 1. *A mapping $\delta : S \times \mathcal{O} \to \{0,1\}$ defined by $(p, O_i) \mapsto \delta\,(p, O_i)$ such that:*

$$\delta\,(p, O_i) = \begin{cases} 1, & \textit{if } p \textit{ is assigned to } O_i \\ 0, & \textit{otherwise} \end{cases}$$

is called an assignment rule.

Under an assignment rule δ, we consider the set $V(O_i)$ of points assigned to O_i, and the set $e(O_i, O_j)$ of points assigned to both O_i and O_j with $i \neq j$.

Definition 2. *A Voronoi tessellation is a set $\mathcal{V}\,(\mathcal{O}, \delta, S)$ such that the assignment rule δ satisfies the following two conditions:*

- *every point in S is assigned to at least one element of $\mathcal{O}$ i.e., $\forall p \in S, \sum_{i=1}^{n} \delta\,(p, O_i) \geq 1$;*
- *the set $e\,(O_i, O_j)$ pertains to the boundaries of $V\,(O_i)$ and of $V\,(O_j)$, i.e., $\forall \varepsilon > 0, \forall p \in e\,(O_i, O_j)$:*
  - $N_\varepsilon\,(p) \cap [V\,(O_i) \setminus e\,(O_i, O_j)] \neq \emptyset$ *and*
  - $N_\varepsilon\,(p) \cap [S \setminus V\,(O_i)] \neq \emptyset$ *and*
  - $N_\varepsilon\,(p) \cap [V\,(O_j) \setminus e\,(O_i, O_j)] \neq \emptyset$ *and*
  - $N_\varepsilon\,(p) \cap [S \setminus V\,(O_j)] \neq \emptyset$,

  where $N_\varepsilon\,(p)$ is the open ball centred at p of radius the real number ε.

[1] As in simplex orientation or segment direction, that is not the same as the orientation of the underlying topological space.

Indeed, the first condition implies that the elements in $\mathcal{V}(\mathcal{O}, \delta, S)$ are collectively exhaustive i.e., $\bigcup_{i=1}^{n} V(O_i) = S$.

The definitions of $V(O_i)$ and of $e(O_i, O_j)$ together with the second condition imply that the elements in $\mathcal{V}(\mathcal{O}, \delta, S)$ are mutually exclusive except for boundaries i.e.,

$[V(O_i) \cap V(O_j)] \setminus e(O_i, O_j) = \emptyset$ for all $i \neq j$.

Definition 3. *The Voronoi region of O_i is: $V(O_i) = \{p \in S \mid \delta(p, O_i) = 1\}$.*

Definition 4. *The Voronoi edge of O_i and O_j with $i \neq j$ is:*

$$e(O_i, O_j) = \{p \in S \mid \delta(p, O_i) = \delta(p, O_j) = 1\}.$$

Definition 5. *The Voronoi tessellation is the set $\mathcal{V}(\mathcal{O}, \delta, S)$=$\{V(O_1), ..., V(O_n)\}$.*

We designate this tessellation the generalized Voronoi diagram generated by the generator set $\mathcal{O}$ with assignment rule δ in space S, and $V(O_i)$ the generalized Voronoi region associated with O_i. We call the assignment rule δ that generates a generalized Voronoi diagram, the Voronoi generation assignment rule, or, shortly, the $V-$assignment rule.

Definition 6. *The Voronoi diagram for a set of points and open oriented segments in the Euclidean plane is a generalized Voronoi diagram where the space is the Euclidean plane, the generator set is comprised of points and/or pairs of open oriented straight line segments in the Euclidean plane such that their extremities belong also to the generator set, and the generator assignement rule is as follows.*

If O_i is a point, then
$$\delta(p, O_i) = \begin{cases} 1, & \text{if } d(p, O_i) \leq d(p, O_j), \forall j \\ 0, & \text{otherwise} \end{cases}.$$
If O_i is an open oriented line segment, then
$$\delta(p, O_i) = \begin{cases} 1, & \text{if } d(p, O_i) \leq d(p, O_j), \forall j \text{ and} \\ & p \text{ is on the left of } O_i \text{ or on } O_i \\ 0, & \text{otherwise} \end{cases}.$$

The Voronoi cell $V(O_i)$ of O_i is the set of points that are closer (in the sense of the distance between a point and an object defined just above) to O_i than to other sites $O_j : j \neq i$ of $\mathcal{O}$. Then, let us introduce the definition of the Delaunay graph of a set of sites (or objects) in the Euclidean affine plane.

Definition 7. *The Delaunay triangulation of $\mathcal{O}$ is the geometric dual of the Voronoi diagram of $\mathcal{O}$: two sites of $\mathcal{O}$ are linked by an edge in the Delaunay triangulation if and only if their cells are incident in the Voronoi diagram of $\mathcal{O}$.*

7.2.1 Quad-Edge Based Voronoi Data Structure

Guibas and Stolfi (26) developed a convenient mathematical structure for representing the topological relationships among edges of a pair of dual subdivisions on a two-dimensional manifold[2]. A subdivision of a manifold M is (26) a partition S of M into three finite collections of disjoint parts, the vertices (denoted by $\mathcal{VS}$), the edges (denoted by $\mathcal{ES}$) and the faces (denoted by $\mathcal{FS}$) with the following properties:

- Every vertex is a point of M,
- Every edge is a line of M,
- Every face is a disk of M,
- The boundary of every face is a closed path of edges and vertices.

A directed edge of a subdivision P is an edge of P together with a direction along it (see page 80 in (26)). Since directions and orientations can be chosen independently, for every edge of a subdivision there are four directed, oriented edges (26). For any oriented directed edge e we can define unambiguously its vertex of *origin* $e.Org$, its *destination,* $e.Dest$, its *left face*, and its *right face*. The flipped version $e.Flip$ of an edge e is the same unoriented edge taken with *opposite orientation* and same direction. The edge $e.Rot$ is the edge of the dual subdivision that goes from the right face of e to the left face of e and oriented so that moving counterclockwise around the right face of e corresponds to moving counterclockwise around the origin of $e.Rot$. The *next edge with the same origin,* $e.Onext$ is defined as the one immediately following e (counterclockwise) in the ring of edges out of the origin of e (see Figure 7.1). Edge functions (see Figure 7.1) allow the traversal of the pair of dual subdivisions.

An edge algebra (26) is the mathematical structure used for representing simultaneously a pair of dual subdivisions (in our use of the Quad-Edge data structure, the Delaunay triangulation and the Voronoi diagram). It captures all the topological properties of a subdivision (26). The topology of the subdivision is completely determined by its edge algebra, and vice versa. This allows all the edge functions to be expressed using three basic primitives, $Flip$, Rot, and $Onext$ described above (26). The edge algebra (26) is an abstract algebra $(E, E*, Onext, Flip, Rot)$ where E and E^* are arbitrary finite sets (of edges), and $Onext$, Rot, and $Flip$ are functions on E and E^*. The main advantage of the Quad-Edge data structure is that all the construction and modification of planar graphs can be done using two basic topological operators, and the complex topological operations built from these two basic topological operators:

- $e := MakeEdge[]$ creates an edge e to a newly created data structure representing an empty manifold;
- $Splice[a, b]$ joins or separates the two edge rings $a.Org$ and $b.Org$, and independently, the two dual edge rings $a.Left$ and $b.Left$ (see Figure 7.2).

[2] A two-dimensional manifold is a topological space with the property that every point has an open neighbourhood which is a disk.

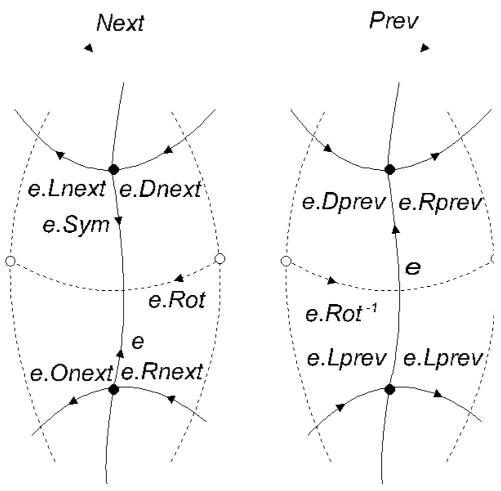


Fig. 7.1. The edge functions (adapted from Guibas and Stolfi (26))

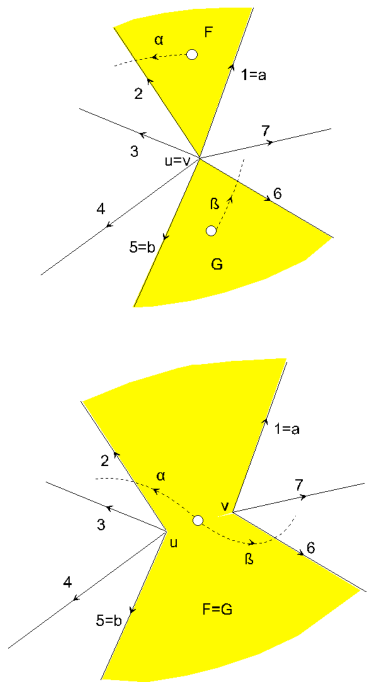


Fig. 7.2. The Splice topological operator

7.2.2 The Operations on the Dynamic Voronoi Data Structure

The complex operations (18) can be decomposed into sequences of atomic actions. Each atomic action in a complex operation executes the geometric algorithm for addition, deletion or change of objects and corresponding Voronoi cells.

The *atomic actions* are:

- the *Split* action, which inserts a new point into the structure by splitting the nearest point from the pointed location into two points;
- the *Merge* action, which deletes the selected point by merging it with its nearest neighbour;
- the *Switch* action, which is performed when a point moves and a "topological event" — defined in Section 7.4.2 — occurs (i.e. the moving point enters or exits a circle circumscribed to a Delaunay triangle);
- the *Move* (topological event) action, which moves the selected point from its current position to a new position or until the next topological event;
- the *Link* action, which adds a line segment between the points obtained after a Split action; the Link action must occur after a Split action, and adds a line segment between the point selected for splitting and the newly created point;
- the *Unlink* action, which removes the selected line segment; the Unlink action must occur before a Merge action, and removes the line segment between the selected point and its nearest object.

These actions compose the set of atomic actions of the dynamic spatial Voronoi data structure (37).

The *atomic actions* are the basis upon which *complex operations* have been built. All the complex operations (18; 40) of this dynamic Voronoi data structure are complex operations composed of atomic actions. The composition of atomic actions into complex operations is provided by syntactic rules.

The *complex operations* are composed of atomic operations, and the exact decomposition of complex operations into sequences of atomic actions is given in (38).

7.3 A Methodology for Raster to Vector Conversion of Colour Scanned Maps and Satellite Imagery

Manual digitization is very time consuming and tedious in nature. The semi-automated algorithms (35) for digitization involve tracing the object from a black and white map and then picking up the centreline of the object. Other algorithms ask for human input for decisions, such as to connect lines at junctions. Interactive systems for map digitization and query, that involve human interaction with the machine, have been designed (47). A fully automated approach is still an open problem, since the maps are produced for human consumption and make use of the heuristic reasoning and world knowledge of human beings (34).

7.3.1 Skeletonization

As discussed in (7), the skeletons produced by any skeletonization algorithm are expected to have homotopy, one pixel thickness, mediality, motion invariance, noise immunity and reconstructability. In practice, however, all of these properties cannot be satisfied simultaneously (7) and some of the properties are even contradictory. Therefore, the choice of properties is entirely dependent on the application. Popular methods of skeleton extraction are thinning using mathematical morphology (24, chap. 9) and skeletonization using distance transform (9). This research is concerned with skeleton extraction using the Voronoi diagram (1; 17; 43; 42).

Image Segmentation

Edge detection produces global edges in an image. This means that there is no object definition attached to the edges. Therefore it is required to somehow define the objects first and then obtain edges from them. This can be achieved by using *image segmentation.* The main goal of image segmentation is to divide an image into parts that have a strong correlation with objects or areas of the real world depicted in the image (52, chap. 5). Thus, image segmentation divides the whole image into homogeneous regions based on colour information. The regions can be loosely defined as representatives of objects present in the image.

Feature space analysis is used extensively in image understanding tasks. Comaniciu and Meer (11) provide a comparatively new and efficient segmentation algorithm, that is based on feature space analysis and relies on the *mean-shift algorithm* to robustly determine the cluster means. A *feature space* is a space of feature vectors. These features can be object descriptors or patterns in case of an image. A colour vector corresponding to a pixel from an image can be represented as a point in the feature space.

In order to understand the mean shift algorithm, consider n data points x_i, $i = 1, \ldots, n$ in the d-dimensional space R^d. A *flat kernel* that is the characteristic function of the λ-ball in R^d is defined as

$$K(x) = \begin{cases} 1 \text{ if } \|x\| \leq \lambda \\ 0 \text{ if } \|x\| > \lambda \end{cases} \tag{7.1}$$

The *mean shift* vector at a location x is defined as

$$M_\lambda(x) = \frac{\displaystyle\sum_{r \in R^d} xK(r-x)}{\displaystyle\sum_{r \in R^d} K(r-x)} - x \tag{7.2}$$

The mean shift vector, a vector of difference between the local mean and the center of the window $K(x)$, is proportional to the gradient of the probability density at x (10). Thus, the mean shift is the steepest ascent with a varying step size that is the magnitude of the gradient. Comaniciu and Meer (12) use the mean shift vector in seeking the mode of a density by shifting the kernel window by the magnitude of the mean shift vector repeatedly. The authors also prove that the mean shift vector converges to zero and eventually reaches the basin of attraction of that mode.

A simple, adaptive steepest ascent mode seeking algorithm is suggested in (11).

1. Choose the radius r of the search window (i.e, radius of the kernel).
2. Choose the initial location of the window.
3. Compute the mean shift vector and translate the search window by that amount.
4. Repeat till convergence.

The mean shift algorithm gives a general technique of clustering multi-dimensional data that can be applied in colour image segmentation. The fundamental use of the mean shift is in seeking the modes that gives the regions of high density in any data.

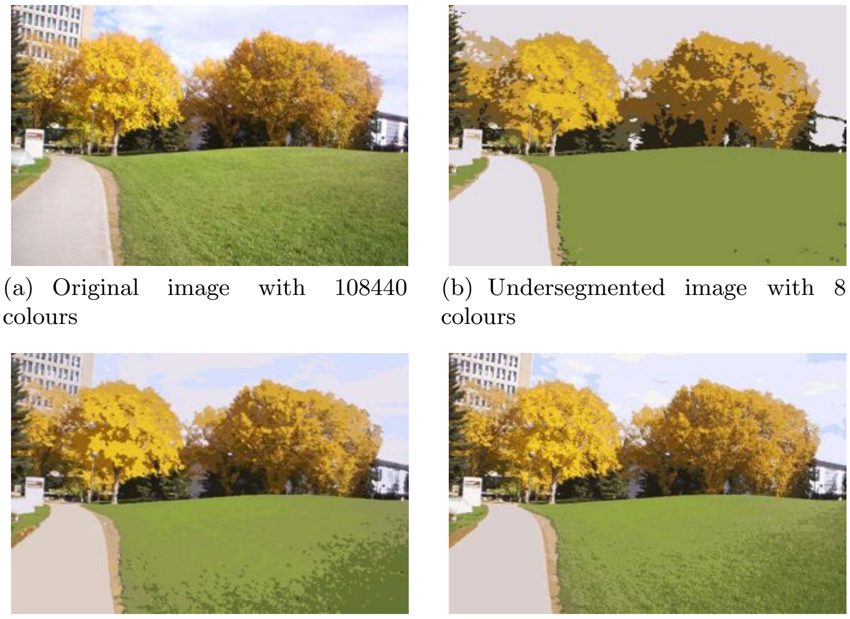
The method described in (11) provides an autonomous segmentation technique with only the type of segmentation to be specified by the user. This method emphasizes the importance of utilizing the image space along with the feature space to efficiently perform the task of segmentation. The segmentation has three characteristic input parameters:

- Radius of the search window, r,
- Smallest number of elements required for a significant colour, N_{min}, and
- Smallest number of connected pixels necessary for a significant image region, N_{con}.

The size of the search window determines the resolution of the segmentation, smaller values corresponding to higher resolutions. The authors use square root of the trace of global covariance matrix of the image, σ, as a measure of the visual activity in the image. The radius r is taken proportional to σ. For the implementation of the segmentation algorithm, the authors provide three segmentation resolution classes:

1. **Undersegmentation** refers to the coarsest resolution with a minimum number of colours and only dominant regions of the image. The three parameters for this class are:
   $(0.4\sigma, 400, 10)$.
2. **Oversegmentation** refers to intermediate resolution and represents objects with some level of detail. The three parameters for this class are: $(0.3\sigma, 100, 10)$.
3. **Quantization** refers to the finest resolution and produces images with all the important colours with no object connectivity requirement. The three parameters for this class are: $(0.2\sigma, 50, 0)$.

Figure 7.3 shows the results of this segmentation algorithm on a natural image. Note the variation in number of colours for each segmentation type.



(a) Original image with 108440 colours

(b) Undersegmented image with 8 colours

(c) Oversegmented image with 34 colours

(d) Quantized image with 49 colours

Fig. 7.3. Colour image segmentation by (11)

Later, Comaniciu and Meer provide an improvement (12) over this segmentation algorithm by merging the image domain and the feature (range) space into a joint spatial-range domain of dimension $d = p + 2$, where p is the dimension of the range domain. This gives an added advantage of considering both spaces together and gives good results in cases where non-uniform illumination produces false contours when the previous segmentation algorithm is used. Therefore, the new algorithm is particularly useful to segment natural images with man-made objects. An added computational overhead to process higher dimensional space is inevitable here. In this research, since we are dealing with scanned maps, the simple mean shift based segmentation algorithm provides satisfactory results.

Image segmentation provides us with definite object boundaries that are used to extract sampling points around an object. These sample points can then be used to compute the skeleton and the boundary of the object.

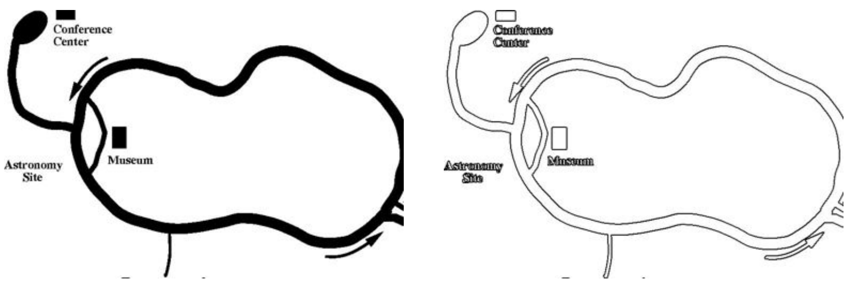
Crust Extraction

Amenta et al. (1) perform the extraction of object boundaries from a set of sufficiently well sampled data points. The vertices of the Voronoi diagram approximate the medial axis of a set of sample points from a smooth curve. Vertices of the Voronoi diagram of the sample points were inserted into the original set of sample points and a new Delaunay triangulation was computed (1). The circumcircles of this new triangulation approximate empty circles between the original boundary of the object and its skeleton. Thus, any Delaunay edge connecting a pair of the original sample points in the new triangulation is a part of the border (1).

Further research by Gold (17) leads to a one-step border (crust) extraction algorithm. In a Delaunay triangulation, each Delaunay edge that is not on the convex hull of the triangulation is adjacent to two triangles and the circumcircles of these triangles are the Voronoi vertices. A Voronoi edge connecting these two circumcenters is the dual edge to the Delaunay edge considered here. According to Gold (17), a Delaunay edge is a part of the border if it has a circle that does not contain any Voronoi vertex. It is sufficient to test only the vertices of the dual Voronoi edge. The test is the standard *InCircle* test. Considering two triangles (p, q, r) and (r, q, s) sharing an edge (q, r) in a Delaunay triangulation and let v be the a vector orthogonal to edge $(r - q)$ in clockwise order, then the test becomes:

$$(s-q)\cdot(s-r) * (p-q)\cdot(p-r) \geq -(s-r)\cdot v * (p-q)\cdot v \tag{7.3}$$

This test will be true for an edge in the border set. Furthermore, those Delaunay edges internal to the object that are not the part of the border set have their dual Voronoi edges as being part of the skeleton (shown in Figure 7.4).



(a) Map (b) Boundary

Fig. 7.4. Result of boundary (crust) extraction using algorithm by Gold

Skeleton Extraction

Research in (4) suggests a new algorithm for skeleton extraction. This is based on the concept of *Gabriel graph* (15).

A Gabriel graph G (highlighted in Figure 7.5) is a connected subset of the Delaunay graph $\mathscr{D}$ of points in set S, such that two points p_i and p_j in S are connected by an edge of the Gabriel graph if, and only if, the circle with diameter p_ip_j does not contain any other point of S in its interior. In other words, the edges in G are those edges from $\mathscr{D}$ whose dual Voronoi edges intersect with them.

Given the Delaunay triangulation $\mathscr{D}$ and the Voronoi diagram V of sample points S from the boundary of an object, the algorithm for centreline extration proceeds by selecting all the Gabriel edges in graph G. Each dual Voronoi edge

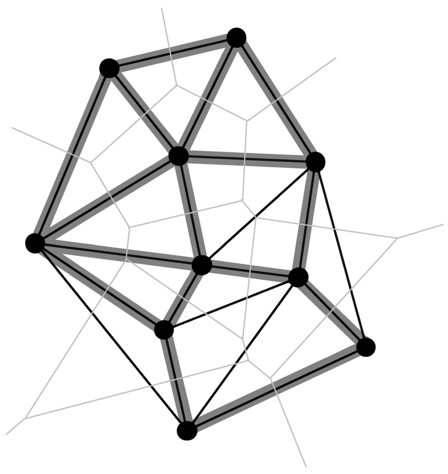
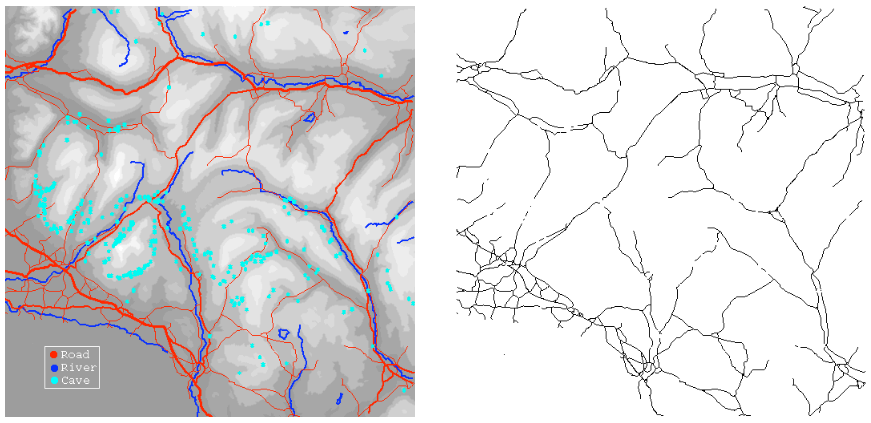


Fig. 7.5. Gabriel graph highlighted in a Delaunay triangulation



(a) Map (b) Skeletons of streams in red

Fig. 7.6. Result of skeleton extraction using the algorithm by Anton et al.

v of the Gabriel edge g from G is inserted in the skeleton K if the following condition is met:

$$\begin{array}{c} g.Org.Col \neq g.Dst.Col \\ \text{Or} \\ g.Org.Col \neq v.Org.Col \\ \text{Or} \\ g.Org.Col \neq v.Dst.Col \\ \text{And} \\ \|g.Org.Col - g.Dst.Col\| \geq \|v.Org.Col - v.Dst.Col\| \end{array} \tag{7.4}$$

Here, *Org.Col* and *Dst.Col* are colour values from the gray scale image corresponding to the location of the origin and the destination of an edge respectively. Figure 7.6(b) shows the result of skeleton extraction from streams present in a map (Figure 7.6(a)). However, a close observation reveals that the skeleton thus obtained has gaps. These gaps are prominent if the object under consideration has sharp turns in its geometry, which is further amplified if the object is thick. An ongoing research project tries to overcome this by locating skeleton edges by moving along the border set.

7.3.2 Automated Approach to Skeletonization of Scanned Map Features

Colour images provide more contextual details about the objects present in the image. Therefore, processing colour images rather than gray scale images can provide much more accurate information. The general approach adopted here is:

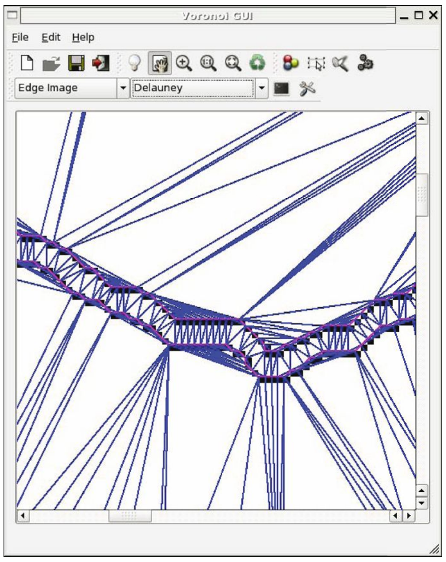
1. Segment a colour image into prominent objects,
2. Ask the user to select an object or process all the objects independently,
3. Collect sample points for each object to be processed, and
4. Extract the skeletons using a Delaunay/Voronoi diagram based algorithm.

These steps are explained next in detail.

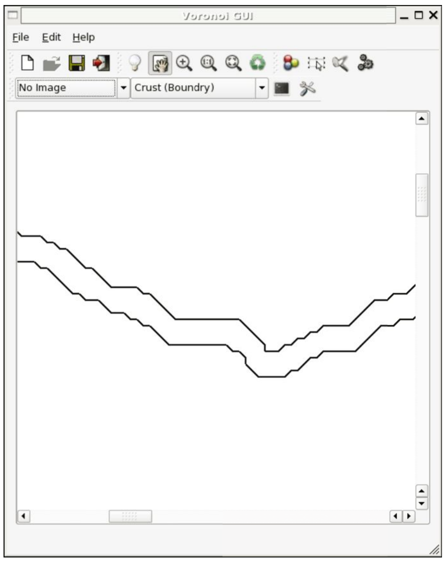
Once objects are defined as homogeneous regions by the segmenter, the next step is to select them and operate on them. This is implemented as an interactive object selection in the application VGUI. To achieve this, the user is allowed to select a region on the image. If an object is composed of more than one regions then multiple object selection can be made and regions combined to form a single object. A wrongly selected region can be removed from the selection. The user input is processed and the selected region is highlighted and selected for subsequent processing.

Once we have an object selected from an image, the next step is to sample its boundary in order to generate points used to construct the Delaunay triangulation. In order to automatically generate these sample points, edge pixels that are returned by the morphological edge detector are used. Using edge pixels also helps in generating a dense sampling which is required to give a better approximation of the skeleton (1). Morphological edge detection on the binary image containing the selected object is performed and the edge pixels are then sequentially inserted into the Delaunay triangulation. The triangulation is updated after every insertion (using the incremental algorithm).

The Delaunay triangulation of the sample points (see Figure 7.7(a)) is computed using the incremental algorithm given in (25), which is stored in the quad-edge data structure. This is followed by the computation of the Voronoi vertices for all faces of the triangulation. The boundary of the object is extracted using the criteria given in (17). The edges in the Delaunay triangulation are analyzed and flagged as being part of the boundary. Figure 7.7 shows the Delaunay triangulation and extracted boundary for a linear object. The sample points of the object are shown as black pixels on the DT in Figure 7.7.



(a) The Delaunay triangulation



(b) Extracted boundary

Fig. 7.7. Boundary extraction from colour image

Skeleton extraction from the Voronoi diagram

Amenta et al. (1) show that the "crust" or the boundary of a polygon can be extracted from an unstructured set of points provided the data points are well sampled. Gold et al. (22) further simplify their method and show that the boundary can be extracted in a single step (see section 7.3.1). Gold (17) discusses the "anti-crust" in the context of skeleton extraction citing a brief introduction of this term in (1). The idea behind getting the skeleton is that a Voronoi edge is a part of the skeleton, if its corresponding dual Delaunay edge is not a part of the border set (crust) and it lies completely within the selected object. Thus, selecting the Voronoi edges lying inside the selected object that are dual to the non-crust Delaunay edges should give us the skeleton (see Figure 7.8). The Voronoi edges thus selected form a tree structure called the "anti-crust" (17) that extend towards the boundary but do not cross it.

The anti-crust of an object, as described above, forms a tree like structure that contains the skeleton. Once all the Delaunay edges belonging to the border set or the crust are identified using the condition given in (17), it is easy to identify the Voronoi edges belonging to the anti-crust. In Figure 7.9, consider the Delaunay triangulation (dashed edges), the corresponding Voronoi diagram (dotted edges) and the crust edges (solid red edges).

Navigation from a Delaunay edge to its dual Voronoi edge can be achieved by using the $Rot()$ operator in the quad-edge data structure. A Voronoi edge $e.Rot()$ of the dual Delaunay edge e is marked as an edge belonging to the anti-crust if the following conditions are satisfied:

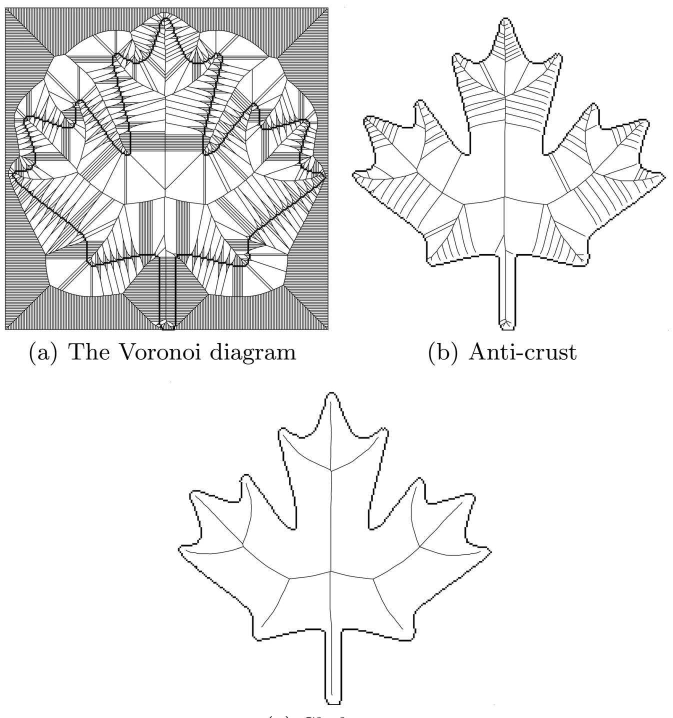
1. $e \notin Crust$
2. $e.Rot().Org \in I$
3. $e.Rot().Dst \in I$

Where $e.Rot().Org$ is the origin coordinate of edge $e.Rot()$, $e.Rot().Dest$ is the destination coordinate of edge $e.Rot()$ and I is the selected object. This marks all the Voronoi edges belonging to the anti-crust that fall inside the selected object. Negating conditions (2) and (3) so that the coordinates do not fall inside the object will give us the exterior skeleton or the *exoskeleton*. Once the anti-crust is identified, an appropriate pruning method can be applied to get rid of the unwanted edges.

Skeleton Pruning

The "hairs" around the result of the skeletonization are due to the presence of three adjacent sample points whose circumcircle does not contain any other sample point - either near the end of a main skeleton branch or at locations on the boundary where there is minor perturbation because of raster sampling (17). A skeleton retraction scheme suggested in (23) gets rid of the hairs and also results in smoothing of the boundary of the object. Ogniewicz (41) presents an elaborate skeleton pruning scheme based on various residual functions. Thus, a hierarchic skeleton is created which is good for multiscale representation. The problem of identifying skeleton edges now reduces to reasonably prune the anti-crust.

Gold and Thibault (23) present a retraction scheme for the leaf nodes in the anti-crust. The skeleton is simplified by retracting the leaf nodes of the skeleton


(a) The Voronoi diagram (b) Anti-crust

(c) Skeleton

Fig. 7.8. Skeleton as seen as the anti-crust

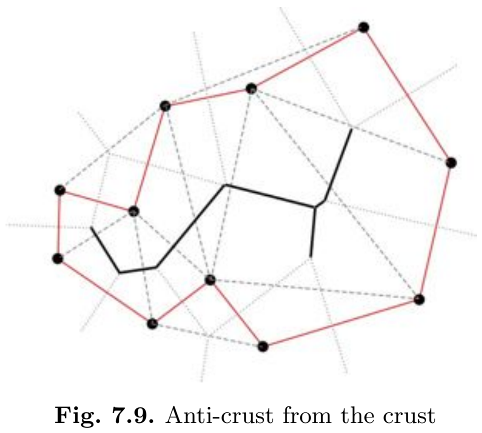


Fig. 7.9. Anti-crust from the crust

to their parent nodes. They (23) recommend performing the retraction operation repeatedly until no further changes take place. An observation reveals that an unwanted branch in a skeleton may be composed of more than one edge (see Figure 7.10). Therefore, single retraction may not be sufficient to provide an acceptable skeleton.

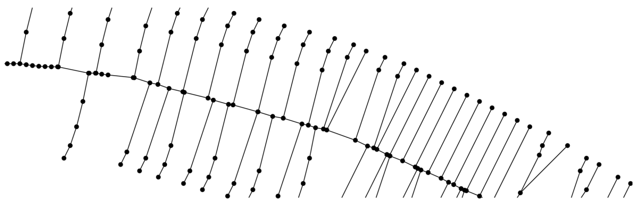


Fig. 7.10. Hair around the skeleton composed of multiple edges

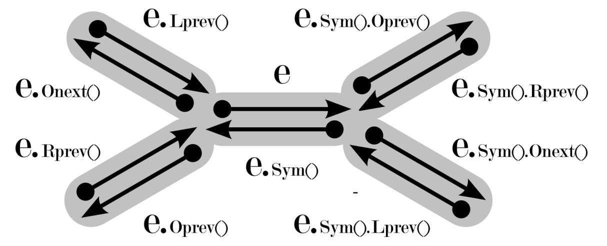
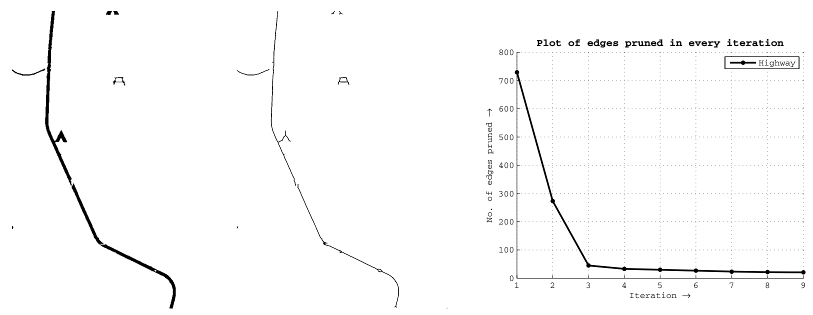


Fig. 7.11. Accessing neighboring edges in a quad-edge



(a) National Highway (b) Skeleton after pruning (c) Pruning plot of object

Fig. 7.12. Skeleton obtained after pruning leaf edges: Test image 1

A similar simplification can be achieved by pruning the leaf edges instead of retracting the leaf nodes. Leaf edge pruning produces satisfactory results and requires only two or three levels of pruning. Before pruning the leaf edges, these must be identified in the anti-crust using the operations provided by the quad-edge data structure. Figure 7.11 shows operators to access neighboring edges of an edge e.

An edge e from a tree of edges $T \subset V$, where V is the Voronoi diagram, is marked as a leaf edge if the following condition is satisfied:

$$\begin{array}{c} e.Oprev() \notin T \text{ And } e.Onext() \notin T \\ \text{Or} \\ e.Sym().Oprev() \notin T \text{ And } e.Sym().Onext() \notin T \end{array} \tag{7.5}$$

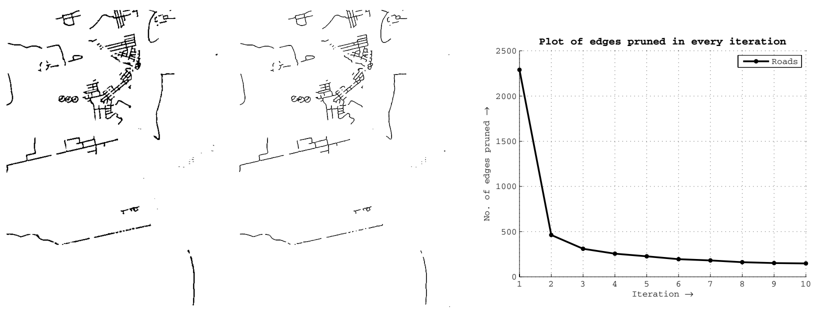
This condition essentially selects all the Voronoi edges belonging to the anti-crust that have at least one end point free (i.e., connected to an edge not belonging to the anti-crust). This condition is used to locate leaf edges followed by their removal from the skeleton. Experiments show that removing leaf edges two to three times simplifies the skeleton to a major extent for linear features.

Presented next are a few examples showing removal of extraneous hair from the skeleton by pruning the leaf edges. Figure 7.12(a) shows a national highway selected from a scanned map. This serves as an example of a thick linear feature. The result of leaf edge pruning is shown in Figure 7.12(b). A plot of the edges pruned in every iteration (see Figure 7.12(c)) underlines the fact that first two iterations are enough to produce an acceptable skeleton and that further pruning results in more contraction of the skeleton (indicated by the horizontal plot after a steep drop after third iteration).

Another example shown in Figure 7.13 of roads selected from a scanned map serves as a case of thin linear features. Again the plot shown in Figure 7.13(c) confirms the adequacy of two iterations of pruning. It should be noted, however, that the case of thin linear features is even simpler than the previous case of thick linear objects due to the presence of majority of single edge hairs. Since the plot shows a steep slope followed by a curve that is almost horizontal, one can expect most of the extraneous edges to be removed after the first iteration itself.

7.3.3 Results with Maps and Satellite Images

Image segmentation allows partitioning the image into homogeneous regions based on their colour that has been achieved here by clustering using the mean shift algorithm. The quality of the extracted objects, in terms of geometry and



(a) Roads (b) Skeleton after second pruning (c) Pruning plot of object

Fig. 7.13. Skeleton obtained after pruning leaf edges: Test image 2

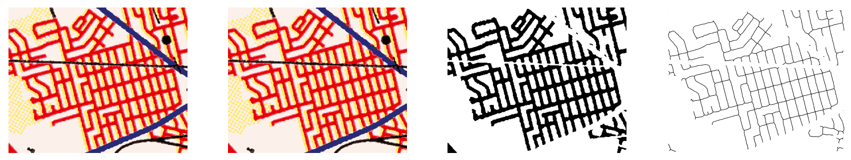

(a) Map with 65415 colours (b) Over-segmented image with 7 colours (c) Roads (d) Skeleton of roads

Fig. 7.14. Extraction of dense urban roads

connectivity, depends on the quality of the scanned image. A noisy image may result in disconnected regions of an object that may produce a broken skeleton. For sharp images with homogeneous regions, quantization produces good results while for noisy images over-segmentation or under-segmentation gives satisfactory results. A low pass (blurring) filter applied to an image, before processing it, often results in a well segmented image since it subdues the high frequency textural details and the noise. The developed application has the flexibility to select and combine the regions. Thus, different regions corresponding to an object with varying hue of a colour can be combined together as a single selection.

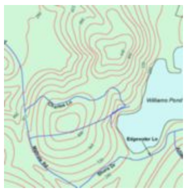
The skeletons and boundaries of map objects shown in this section are extracted using the algorithms implemented in the application. The skeletons are obtained by first computing the anti-crust and then pruning them by removing the leaf edges. The boundary (crust) is generated using the single step algorithm (17). A few examples of segmentation on scanned maps are shown next.

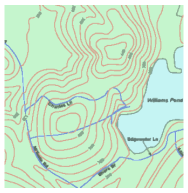
Figure 7.14 shows a typical case of a dense urban road network. The extracted skeletons are of excellent quality and maintain good connectivity except at the places where other features were drawn over the roads in the original map.

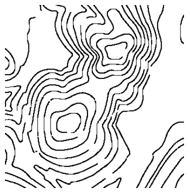
The next example shows a contour map obtained on-line from:
http://www.maptext.com/maps/Contour2.htm.

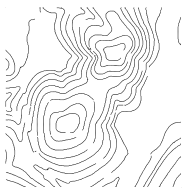
Fig. 7.15(b) shows a segmented image of the map. The map was treated with a low pass filter to even out noise in the image. The contour object in the segmented image is well separated from other objects (like roads and the background). Figure 7.15(d) shows the extracted centrelines of the contour curves, while Figure 7.15(f) shows the extracted road centrelines. The extracted contour centrelines, despite being broken at a few places, are valuable to be imported into a GIS, to attach labels and optionally to generate a digital elevation model.

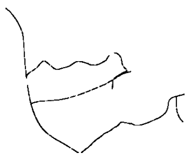
In order to test the applicability of the designed system to satellite images as well, a few experiments were conducted. Since the system is designed to make use of homogeneity in colours of objects, natural objects are better suited for our analysis. A few cases of coastline extraction have been considered here. A coastline is defined as the boundary between land and water. Coastline mapping is important for coastal activity monitoring, resource mapping, navigation, etc. A lot of work on coastline extraction from SAR (Synthetic Aperture Radar)


(a) Map with 8894 colours


(b) Segmented image with 9 colours


(c) Contours


(d) Skeleton of contours


(e) Roads

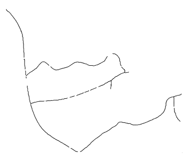

(f) Skeleton of roads

Fig. 7.15. Extraction of contours and roads from a map

and multi-spectral imagery has been done. Bo et al. (8) provide a technique for coastline extraction from remotely sensed images using texture analysis. Liu and Jezek (36) showed delineation of the complete coastline of Antarctica using SAR imagery. Bagli and Soille (6) suggest a morphological segmentation based automated approach for coastline extraction. Di et al. (14) use the image segmentation algorithm in (12) to segment an image and detect the shoreline. The final shoreline is obtained by local refinement. In the following examples, the coastline is extracted as the crust of the selected object. The accuracy of the coastline depends on the spatial resolution of the imagery.

Figure 7.16 shows the complex coastline of Guinea Bissau. Segmentation results in four colours that define the water body out of which three define the coastline. Multiple selection allows the combination of these three regions together to form the complete coastline as shown in Figure 7.16(c).

Next, the coastline of the Hebrides is shown in Figure 7.17. As can be seen, the coastline is corrugated. The extracted coastline is shown in Figure 7.17(c).

Applicability of the developed methodology can be easily extended to natural colour satellite imagery to extract homogeneous features. Coastline delineation, snow cover mapping, cloud detection, and dense forest mapping are a few areas where satisfactory results can be obtained. In the next section we will introduce kinetic Voronoi diagrams and Delaunay triangulations and we will present their applicability in GIS.

In this section, we have described an approach for extracting features from images using the point VD and skeletonisation techniques. Where the resulting features are linear (usually man made) an incremental generalisation process is

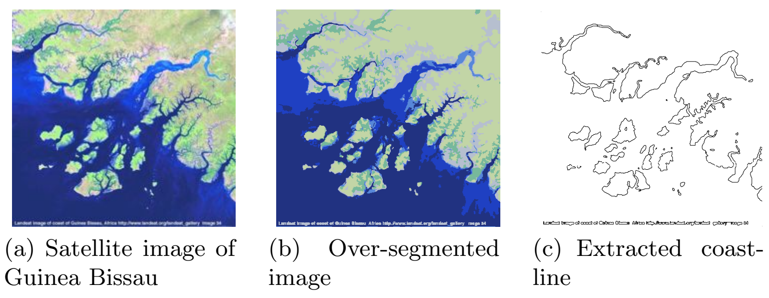

(a) Satellite image of Guinea Bissau (b) Over-segmented image (c) Extracted coastline

Fig. 7.16. Feature polygon extraction from the satellite image of Guinea Bissau

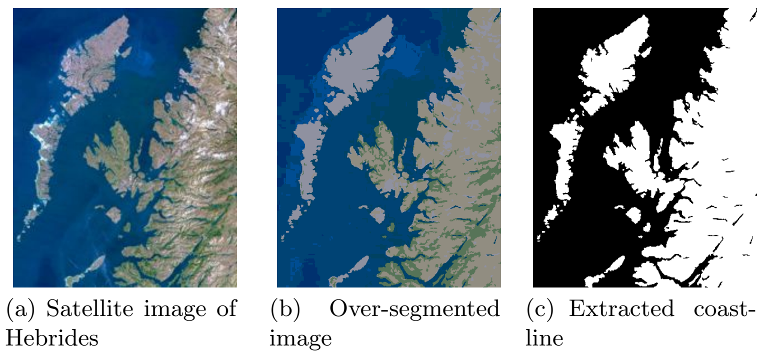

(a) Satellite image of Hebrides (b) Over-segmented image (c) Extracted coastline

Fig. 7.17. Feature polygon extraction from the satellite image of Hebrides west coast

needed to produce the simpler vector representation of the features, preferably within the VD/DT space used for the skeletonisation itself. This is addressed by the incremental DT or, preferably, the Line Segment VD described in the next section. The result is a toolkit permitting feature extraction from imagery and subsequent derivation of linear features within a single spatial model and software environment.

7.4 Kinetic Voronoi/Delaunay Drawing Tools

Kinetic data structures can be used for simulation of a complex system of multiple moving objects (27), that can not be handled by commercially available GISs. Although GISs have evolved a lot recently, they are still a few "real world" features that they can not handle.

7.4.1 Map Drawing and Editing

Constructing digital maps with line data has various difficulties, especially with the connectivity ("topology") between various polylines. This is implicit in the spatial model used — one-dimensional features floating loose in two-dimensional space. A more hopeful approach is based on tessellating the whole map space, so that all cells are bounded by other cells. Connectivity is thus automatic. Tessellations may be regular or irregular. A regular grid ("raster") has simple spatial relationships but, because it is based on the coordinate system and not on the features being represented, has difficulties in managing complete features or polylines. Irregular tessellations may be triangulations or other cell structures, e.g. the Voronoi diagram (VD). Triangulations connecting data points may follow any chosen properties, but the Delaunay triangulation (DT), the dual of the VD, has a geometric (coordinate-based) specification that gives a unique solution, except in degenerate cases, and most importantly it can be updated locally while perturbing only the immediately neighbouring triangles.

However, the triangle edges may not exactly follow the desired polyline. Voronoi cells are usually constructed for sets of data points, giving the proximal region to each point, but they are not readily aggregated into polylines. A topological model is required for two purposes: to perform analysis on the final map (e.g. network flow) or to construct the map in the first place (e.g. snapping one polyline to another). Saving individual features (polygons, polylines) to a database and reconstructing the connectivity as required may suffice for some types of analysis (but not network flow), but this still leaves the construction problem. Node construction from the intersections of individual polylines is a classical GIS problem, caused because the intersections of individual one-dimensional entities, and their subsequent merging to form nodes (in a polygon map for example) do not always give a well-defined ordering of edges around the nodes — a requirement for an elementary topology on a 2-manifold. Here a tessellation is an attractive option. In addition, traditional topology-building algorithms in GIS are batch-oriented: all polylines are inserted at once, and any local changes require a complete rebuild (although in some cases a "patch" may be recalculated and reinserted into the larger map). In practice, many construction and editing operations are incremental, mimicking the manual use of a pen (and an eraser) to make local changes. This suggests the use of a kinetic algorithm to simulate pen movement.

7.4.2 An Integrated Approach

We propose here an integrated approach to tessellated map construction, which uses locally-updated tiles as a consistent and kinetic definition of adjacency. The first component is the moving-point VD/DT, which manages the topology of the moving pen. The second component is the Constrained DT, which permits the specification of certain triangle edges, even if they fail the Delaunay conditions, thus allowing the construction and connection of line segments. The third component is the Line-Segment VD, where the VD defines proximal regions around line segments as well as points. This integrated approach uses the Moving-Point VD (or DT) to "draw" the desired line segment in both modes, thus applying a common underlying algorithm to both processes. This approach to preserving topology assists in the construction process, and also in the analysis of the final map, for example for map generalization or network flow. Its main drawback, as with all large-scale graph structures, is the lack of an efficient mapping to a database system — although some object-oriented databases may assist in this. This algorithmic approach has the following stages:

The Moving-Point VD or DT

This requires the dynamic incremental insertion and deletion of data points. In addition, individual points may be moved from their previous location to some subsequent location — the "trajectory". In order to maintain the VD/DT geometric properties there must be a predictive tool to specify at what location the neighbouring VD/DT edges must be modified — a "topological event" (abbreviated hereafter TE).

The Kinetic Constrained DT

The Moving Point (MP) in the moving point VD is split from a previous "old" point (making two new adjacent triangles with "zero" area) and moved towards its "new" destination. The initial zero length triangle edge between the old and new points is flagged as constrained (abbreviated hereafter CE), and any TE generated by the moving point is ignored if it involves switching any CE.

The Kinetic Line-Segment VD

Instead of flagging the CE in the initial position of the Constrained DT, a pair of "open oriented line segments" (also called half-lines hereafter) is generated. These are two new generators in the VD — one for each side of the line, in addition to the two end points. Each of these is the potential generator of a Voronoi proximal region. As the MP moves, TEs are identified as before, and the topology updated, thus giving an expanding region associated with each open oriented line segment.

In this model the topological events are the same as before, but the circumcircle (CC) calculation must be expanded in order to work with distances from line segments as well as points. In earlier work (16; 55) a direct calculation of Voronoi boundary intersections was used to find the circumcentre. This failed on occasion as arithmetic precision limitations could place the centre on the wrong side of a line segment, thus destroying the node-ordering necessary for topology maintenance. A new iterative algorithm was developed (2; 3) that converged on the correct solution from an initial condition while preserving the necessary order of the generator locations around the circumcircle. All the operations used have their inverses, as MP movement may expand or contract the trailing line (38). Preserving the topological relationships during construction means that potential collisions may be detected in advance, and the appropriate join operations implemented. This is simplified as the lines and their proximal regions are embedded in the two-dimensional space, guaranteeing that, for example, one VD line segment may detect an imminent collision and form the appropriate junction that preserves the correct node and region ordering around the junction point.

7.4.3 The Kinetic Point VD and Its Dual DT

"Kinetic" data structures maintain their topological structure while the entities move; dynamic ones merely permit local insertion and deletion of these entities (points, segments, etc.) Insertion algorithms are well known, but published point deletion algorithms are relatively recent.

The dynamic VD and DT

The simple VD can be constructed in many ways (5; 44), but the incremental algorithm has often been found to be both stable and simple (26). In simple terms, each new point is inserted into the existing DT by first finding the enclosing triangle, using the CCW test of (26), splitting it into three triangles using the new point, and then testing each edge recursively to see if it conforms to the Delaunay criterion: that neither of the adjacent triangles CCs have an interior point. If they do, the common diagonal is switched and the new edges are added to the stack of edges to be tested (26). This CC test (INCIRCLE) (26) can be shown to be equivalent to calculating the VD vertices and testing if the VD edges cross. Point deletion can be performed by approximately following the inverse process: switch DT edges if the result gives an exterior triangle whose CC is empty except for the point being deleted. When only three triangles remain the central point is deleted. There are two similar approaches: (13; 39). Thus, the VD is updated at the same time as the DT. Both insertion and deletion may be considered as partitioning the DT into two parts: the valid DT exterior area and the valid DT interior area. Boundary edges are then switched until the two parts merge.

The moving-point VD and DT

When a point MP moves as part of a DT/VD it may either travel a short distance without requiring a topology update, or else triangle edges must be switched to maintain the Delaunay criterion. These TEs (16; 49; 28) occur when MP moves into or out of a CC. Real CCs are those formed from triangles immediately exterior to the star or set of triangles connected to MP. "Imaginary" CCs are formed by triangles that would be created if MP was moved out of its CC, and are formed by triples of adjacent points around MPs star. Thus, if MP moved into a constellation of points in a DT it would first enter the CC of a triangle, causing a triangle edge switch and adding the furthest point of the triangle to the star of MP. The original real CC is now preserved as an imaginary one. As MP continued to move, at some later time it would move out of this imaginary CC, the original triangle would be recreated and the CC would become real again. As MP moves in any direction, it may enter or leave many CCs, and

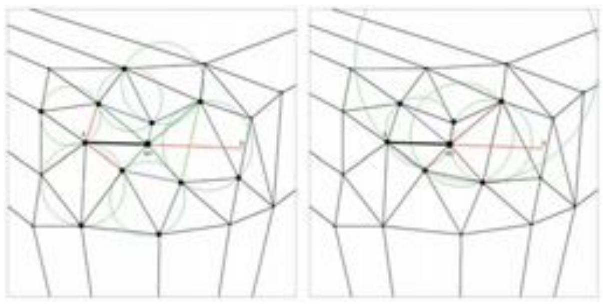


Fig. 7.18. Left: "Real" circumcircles to MP; right: "Imaginary" circumcircles to MP

the triangle edges must be updated accordingly. Topological operators are used to extract the real CCs surrounding MPs star, and the imaginary CCs formed by consecutive triples around MPs star. Real CCs are dropped if they are re-formed from the imaginary ones. (Initially, before a point is first moved, all surrounding real CCs are found and tested against the proposed trajectory.) To maintain the DT during MPs movement it is therefore necessary only to find the intersection of its trajectory with the first CC (either real or imaginary), move MP to the intersection point, switch the affected edge, and then repeat the process (see Figure 7.18). To avoid problems due to degenerate cases, e.g. when several CCs are superimposed due to a regular square grid of data points for example, the "first" intersection must be clearly defined. In practice it is critical not to "forget" an intersection because it is behind MPs current position, usually due to computer arithmetic limitations. This is achieved by always using the earliest intersection, even if it is behind MP, subject to a test that the intersection point is associated with an arc of the triangle edge to be switched.

This looping process: find the next intersection with a CC; move MP; switch the DT edge — is continued until the MP's destination is reached. If there is already a node at the destination then the MP is merged with that node (removing that point and two triangles adjacent to the edge connecting those nodes). It is, however, possible that MP collides with an existing point, destroying the DT structure. Thus, at each iteration of the loop the distance from MP to each new neighbouring point is tested, and if it is below some tolerance the collision is detected — the MP is rolled back to its origin, then moved again towards the collision point CP. Eventually the MP and CP are merged and the process is continued from the CP by splitting the MP again and moving it towards the original destination.

The Kinetic Constrained DT

Where MP collides exactly with a neighbouring point, the points are merged by removing MP along with the two triangles adjacent to the edge between these points. The reverse process may be used to create a new MP from a previous

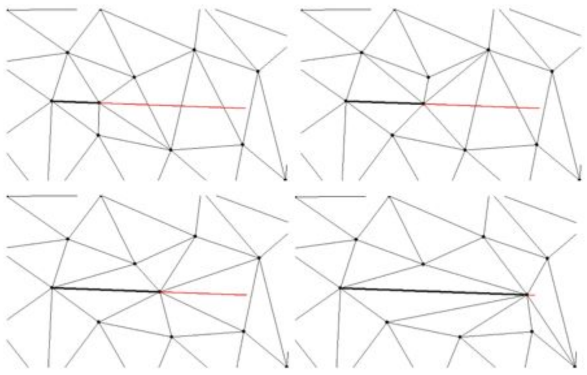


Fig. 7.19. Four stages in the construction of a constrained edge

point (split). This creates a zero-length edge between them, which expands as MP moves away. In the normal course of events this edge will be switched once MP moves outside the imaginary CC formed by the previous point and its two adjacent points in the star. However, for many applications it would be desirable if the edge was preserved, and MP used to “draw” a triangle edge between two locations. In this case the trailing edge of MP is flagged “do not switch”, and all tests to switch it are ignored. This generates the Constrained DT, where specific triangle edges are fixed (constrained, CE) and do not follow the DT/VD condition (46; 50) and (48). See Figure 7.19.

The interesting thing about this approach, as opposes to those in the literature, is that it is incremental, allowing the addition of further points or constraints as required. A further property is that it is reversible — MP may be moved back along its (constrained) trajectory, “rolling it up” as it goes, until it reaches its starting point, with which it may be merged. Thus, each operation has its inverse, giving a kinetic data structure which allows the construction and incremental or interactive modification of the desired map. In addition, the construction commands may be preserved as a “log file” for later reconstruction or modification. If timestamps are associated with each command the map may be rolled forwards or rolled backwards to any desired time (38). Because of this reversibility, intersections of pairs of CEs may be managed. If MP finds a new CE (CE1) as part of its star, and attempts to switch it, then a potential collision exists. Then, if necessary, MP is rolled back to its origin, an intersection point IP between the trajectory for CE and CE1 is computed, CE1 is rolled back to IP and a new portion of CE1 is created by splitting a new node from IP and reconstructing CE1 (leaving IP as a new point on CE1). Then, MP is moved again towards IP. They eventually merge and then a new portion of CE is drawn from the intersection point to the original destination of MP. This may be repeated for as many intersections as necessary (See Figure 7.20).

Figure 7.21 shows the construction of the Constrained DT for a UK urban data set. This is of particular interest because of the work described in (31; 32; 54) that

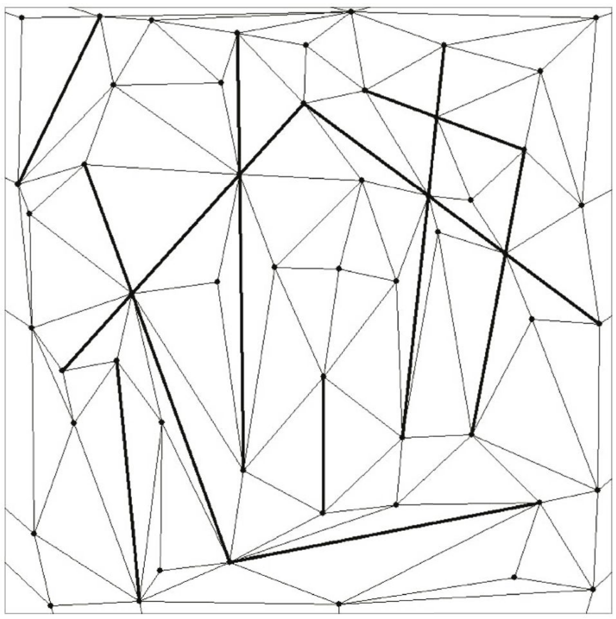


Fig. 7.20. Constrained DT with intersections

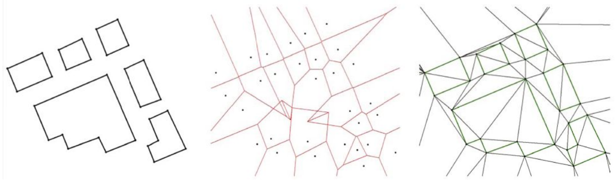


Fig. 7.21. a) Building boundaries; b) the Voronoi Diagram (note the two overlapping Voronoi boundaries where the constrained edge differs from the unconstrained edge); c) constrained Delaunay Triangulation

constructed the CDT of roads and building outlines and then used the adjacency information to modify and move the buildings as part of the process of map generalization. Figure 7.21c shows that for this example only two constrained edges are absolutely necessary (their Voronoi edges overlap) — these prevent the switching to give a valid VD/DT — while the others would be unchanged. Figure 7.22 shows the constrained DT for several buildings and roads.

7.4.4 The Kinetic Line-Segment VD

The primary problem with the Constrained DT is the confusion of entities. For a simple point DT/VD the primary objects are data points, which are the generators of the proximal Voronoi cells. The DT merely describes the dual

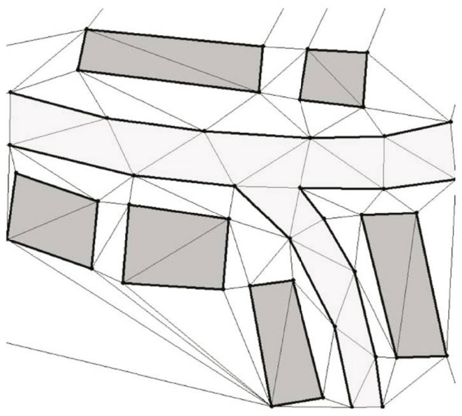


Fig. 7.22. The Constrained DT for several buildings and roads

relationships of the Voronoi edges: Delaunay edges are merely pointers expressing which pairs of data points are separated by Voronoi boundaries. For a simple TIN model it is convenient to imagine that these are geometrically defined as "straight", as the triangle is a 2D simplex and hence forms a basis for linear interpolation within, but their real function is to support the set of equidistant boundaries that form the VD of a set of generators. (These generators may, if required, be any set of non-overlapping objects: the dual DT remains a triangulation.) However, with the Constrained DT there is a confusion between triangle edges that express duals of VD edges and those that have been manually added as objects — in the sense that a building outline is formed of point and line-segment objects. Thus, the VD of a Constrained DT is broken at each constrained edge, and the VD edges that are correct on one side of a constrained edge are invalid when they penetrate to the other side. It is more correct to define the mapped objects separately (perhaps composed of points and line segments) and then to construct the DT/VD expressing the spatial relationships between them.

Unfortunately, the construction of the VD of points and line segments has proved to be a difficult task, primarily due to the limited precision of computer arithmetic. This causes no great difficulty for well-separated individual line segments (e.g (20)), but map objects constructed from connected points and line segments need to have tight guarantees that, for example, the circumcircle for a line segment, its end-point, and a line segment connected to that point, falls on the correct side of the polyline. (Geometrically it falls precisely on the common end-point, but topologically it must be associated with the correct side.) This has proved difficult to achieve, and workers have spent a great deal of time attempting to construct robust algorithms (e.g (29; 30; 53)).

In addition, we have wanted to allow incremental, rather than batch, construction, so we followed the approach of (20) which was based on the concept of the moving point VD described above. Instead of preserving a trailing triangle

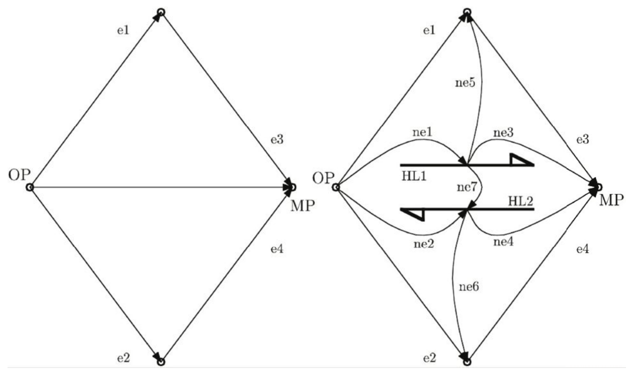


Fig. 7.23. Half-lines (open oriented line segments) between two data points

edge, as described above for the Constrained DT, the "old point" OP and the "moving point" MP are connected with additional map objects: open oriented line segments connecting OP and MP that stretch as MP moves away. Here an "open oriented line segment" or half-line is similar to the "half-edge" structure used in CAD consisting of one side of the desired line segment, in anticlockwise orientation viewed from the face. As both end points and both open oriented line segments are map objects they are therefore generators of the VD, and thus they are vertices of the dual DT, as shown in Figure 7.23. Half-lines HL1 and HL2 are linked with DT edge ne7, and they are linked by DT edges to endpoints OP and MP. In (33) an incremental algorithm was produced for exact arithmetic which allowed intersections by splitting line segments in advance, using exact arithmetic and implicit coordinates for intersections, as they have the original end points as support. We can not do this, as we allow arbitrary line segment deletion, but we allow MP to move from its origin to its destination and manage any collisions as it detects them. Thus, calculation of the circumcircles of these triangles is more complex than for point data sets. In (55) this was calculated from the intersections of the curves forming the Voronoi boundaries, but this suffered from the arithmetic precision problems mentioned above.

Circumcircle

In our new work we use the approach of (3) where the simple point circumcircle (CC) calculation (INCIRCLE) (26) was given an initial estimate based on the configuration of the points/line-segments used. Initially, points and the mid-points of valid portions of the line segments were used for the INCIRCLE test. The centre was then projected onto each line segment, and a new CC calculated based on INCIRCLE. When the new circumcentre was projected onto each line, and the projection point was outside the line segment, then a point half way between the old projection point and the appropriate endpoint of the line was used. This was iterated to a suitable level of precision, and the method was guaranteed to preserve the order of the generating points around the CC, thus keeping the initial Voronoi edge order around the circumcentre (and thus the correct DT order as well), (3). Figure 7.24 shows the rapid convergence of the method for two initial configurations.

The key improvement over the work of (2) was the trimming of the potential space for the initial estimate: the centre had to be on the correct side of the line segments, given the original anticlockwise order of the vertices obtained from the data structure; line segments had to be trimmed to be on the correct side of other line segments; data points had to be on the correct sides of line segments and projected within the trimmed segment, etc. (Figure 7.25). While all "real" CCs, as defined previously, had to have valid solutions, as they were part of a valid VD, "imaginary" CCs might not have valid solutions, and needed to be rejected in order to avoid false topological changes. After extensive testing, our current algorithm appears robust under any conditions of the three generators, and is able to detect invalid combinations successfully.

Line segment construction

As with the Constrained DT, the MP is used to draw the line segment using the open oriented line segments described above. As MP moves it acquires and loses Voronoi neighbours, as with the simple moving point, but when it loses them they are transferred to the trailing line segment: since this is the locus of MP, it retains all the neighbourhood relationships previously held by MP (see Figure 7.26).

For a simple line segment with two end points there are four Voronoi regions: one for each end point and one for each open oriented line segment. This permits the querying of each side of a line, e.g. to find if a point is inside or outside a polygon. As shown in (16) the partitioning of the map space into proximal regions also makes buffer-zone generation an elementary operation on each region. Figure 7.4.4 shows the line segment VD for the same urban dataset as shown previously. When the DT is also displayed, note that each line segment is also a DT vertex. This clearly distinguishes the DT function of expressing the adjacency relationships, and not being part of the map object. Each of the map objects may be edited by the insertion or deletion of line segments (open oriented line segment pairs) and free vertices. The method is dynamic, in the sense of being locally updatable, and

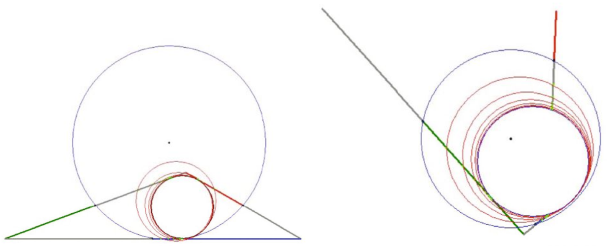


Fig. 7.24. Iterative circumcircle calculations

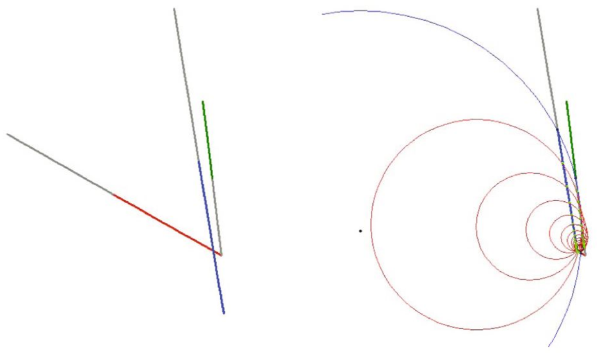


Fig. 7.25. Trimming segments a) Initial line segments; b) The resulting circle after trimming the lines and iterative circumcircle calculations

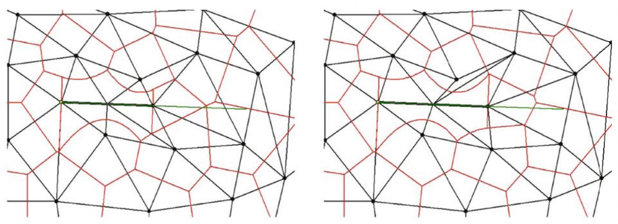


Fig. 7.26. Two stages in drawing a line segment VD

kinetic, in the sense that MP may move within the map space. However, line segments may only expand or shrink, and not sweep sideways, as collision detection and topology maintenance are based on MP alone. As with the Constrained DT, the Line-segment VD may have intersecting segments. The line segments are map objects having two separate sides (Figure 7.23), allowing attributes (such as polygon colour) to be assigned to each side.

7.4.5 Robustness

Space does not permit the description of all the details of the suite of algorithms described in this chapter. The key question in practice is the robustness of the method for all types of data input, given the problems of arithmetic precision. The underlying method described here consists of two parts: a geometric test and a topological update. Any arithmetic operation not resulting in a topological change causes no robustness problems — for example calculating the CC (Voronoi node) for display purposes, or the projection of a point onto the interior of a line segment. Only geometric tests used to trigger topological changes can cause robustness problems, and there are only two — calculation of CCs and a sidedness test ("walk" in (21), "CCW" in (26)). CCW and INCIRCLE

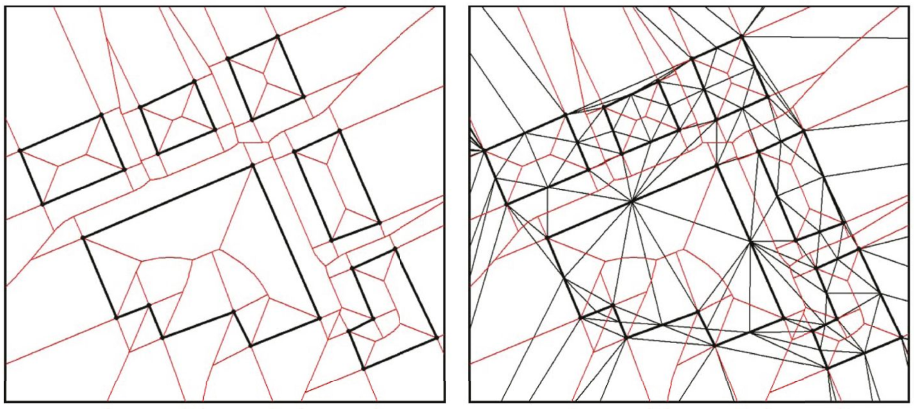


Fig. 7.27. a) Line segment VD; b) VD plus DT for the simple buildings of Figure 7.21

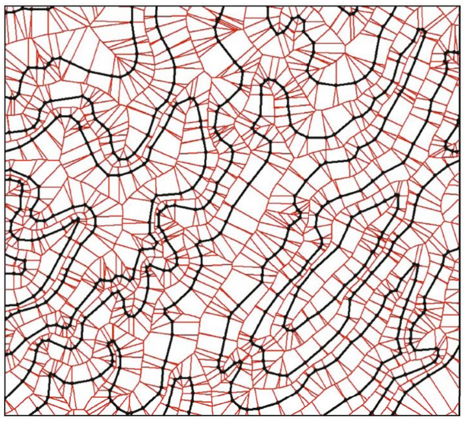


Fig. 7.28. Line-segment VD for contours

(for three points) are geometric predicates that have been studied extensively, and arbitrary-precision solutions are readily available (51). The iterative circle calculation for line-segments described above uses INCIRCLE, and although it is iterative (and therefore approximate) this causes no problem where the centre projects onto the interior of the line-segment. In practice, a tolerance value is needed (to allow for arithmetic imprecision) only in the specific cases described below.

1. For moving points, when a point is selected for movement or splitting, the CCs for "exterior" triangles must be put on a list if their projections onto the trajectory are in front of MP. A tolerance is used here and causes no difficulties for these approximately tangent cases.

2. When finding the next topological event for MP, the intersection of the trajectory with the CC is imprecise. A tolerance is used to check if the intersection of a "real" or "imaginary" CC is too close to the trajectory start or end, and are snapped to their coordinates if necessary (to avoid the oscillation of the MP at those locations). The tolerance is also used to test whether the intersection point falls on one of the vertices of the circle.
3. There is a tolerance check for collisions with objects in MPs path (points are treated as disks with a specified radius).

Another important issue while using floating point arithmetic is to guarantee the same results of determinant and circumcircle calculations for the same objects (for example a different order of three points in determinant calculation gives three slightly different results due to the computer's arithmetic precision). This is

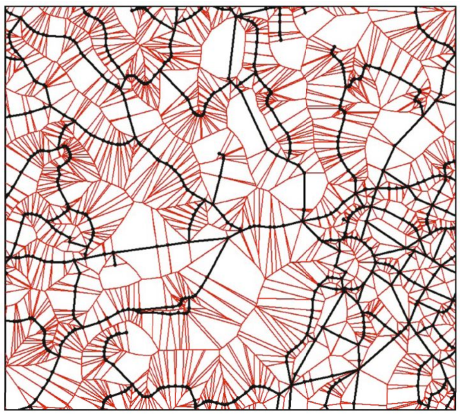


Fig. 7.29. a) Line-segment VD

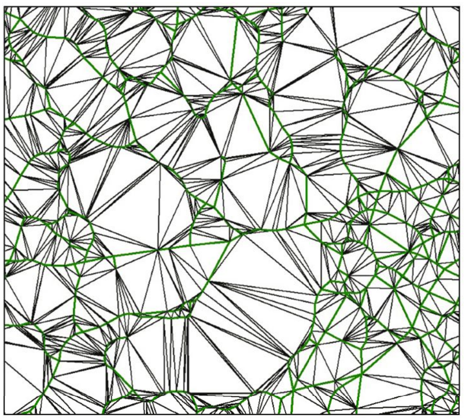


Fig. 7.30. b) Constrained DT for a road network

achieved by consistent ordering of objects (points and lines) before doing actual calculations.

7.4.6 Applications

Our previous examples have been urban applications, showing building and street boundaries, for potential application in map generalization (31; 32). We will briefly show two others. Figure 7.28 shows the Line-segment VD for a portion of a contour map. Both the points and line-segments forming the contours are map objects, as would have been the intention of the compilers. In addition, the medial axis, or skeleton, between or within the contours is clearly seen (see (22), for further discussion of the skeleton). This map is directly editable if required.

Figures 7.29 and 7.30 show the Line-segment VD and the Constrained DT for a road network. Again, the relationships between map objects are clearer in the VD, and both maps are directly editable.

7.5 Conclusions

In this chapter, we have presented an effective methodology for automated digitization of features from scanned maps. The methodology enables extraction of centreline as well as boundary of an object in a single step. The centrelines are required while digitizing linear features like roads, contours, streams, etc. On the other hand, boundaries are needed to obtain polygons from features like fields, islands, etc.

Based on the methodology, an interactive software application has been developed. It incorporates object extraction from input image using colour image segmentation. Segmentation based on clustering using mean shift algorithm in feature space has been adopted here. Mean shift algorithm is a popular and robust method of clustering and provides good results in segmentation. The application allows selection of multiple objects for extraction of the skeleton or the boundary, as well as automatic extraction for all the features in the scanned map. Skeletonization and boundary extraction is based on the Voronoi diagram and the Delaunay triangulation. This not only gives accurate skeletons and boundaries, but also preserves the topology of the extracted feature.

We have also attempted to show that a tessellated spatial model has definite advantages for cartographic applications, and facilitates a kinetic structure for map updating and simulation. Firstly, the moving-point DT/VD model approximates human thinking, and manages collision detection, snapping and intersection at the data input stage by maintaining a topology based on a complete tessellation. Secondly, the Constrained DT allows the simulation of edges, and not just points, with only minor changes to the moving-point model, but at the cost of confusing map objects and topological entities. Thirdly, the Line-segment VD is a better-specified model of the spatial relationships for compound map objects built from points and line segments than is the Constrained DT. However, until now it has been more difficult to develop. We believe that this method is now viable for 2D cartography, and in many cases it should replace the Constrained DT. However, whichever method is used, the concept of using the moving point as a pen, permitting interactive navigation within the map under construction, together with the ability to delete and add line segments as desired in the construction and updating process, appears to be a very useful approach.

In conclusion, we firstly describe an approach for extracting features from images using the point VD and skeletonisation techniques. Where the resulting features are linear (usually man made) an incremental generalisation process is needed to produce the simpler vector representation of the features, preferably within the VD/DT space used for the skeletonisation itself. This is addressed by the incremental DT or, preferably, the Line Segment VD described in the second part of the chapter. The result is a toolkit permitting feature extraction from imagery and subsequent derivation of linear features within a single spatial model and software environment.

Acknowledgements

This research work has received the financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC) and New Brunswick Innovation Funds (NBIF) for a project titled "Data structures and algorithms for the integration of raster and vector GIS".

We would like to acknowledge the financial support of the EU Marie-Curie Chair in GIS at the University of Glamorgan.

References

[1] Amenta, N., Bern, M., Eppstein, D.: The crust and the β-skeleton: Combinatorial curve reconstruction. Graphical models and image processing: GMIP 60(2), 125–135 (1998)

[2] Anton, F., Gold, C.: An iterative algorithm for the determination of Voronoi vertices in polygonal and non-polygonal domains. In: Proceedings of the Canadian Conference on Computational Geometry, Kingston, Canada, pp. 257–262 (1997)

[3] Anton, F., Snoeyink, J., Gold, C.: An iterative algorithm for the determination of Voronoi vertices in polygonal and non-polygonal domains on the plane and the sphere. In: 14th European Workshop on Computational Geometry (1998)

[4] Anton, F., Mioc, D., Fournier, A.: 2D image reconstruction using natural neighbour interpolation. The Visual Computer 17(3), 134–146 (2001)

[5] Aurenhammer, F.: Voronoi diagramsa survey of a fundamental geometric data structure. ACM Computing Surveys (CSUR) 23(3), 345–405 (1991)

[6] Bagli, S., Soille, P.: Morphological automatic extraction of coastline from pan-european landsat tm images. In: Proceedings of the Fifth International Symposium on GIS and Computer Cartography for Coastal Zone Management, vol. 3, pp. 58–59 (2003)

[7] Bernard, T.M., Manzanera, A.: Improved low complexity fully parallel thinning algorithm. In: ICIAP 1999: Proceedings of the 10th International Conference on Image Analysis and Processing, p. 215. IEEE Computer Society, Washington (1999)

[8] Bo, G., Delleplane, S., Laurentiis, R.D.: Coastline extraction in remotely sensed images by means of texture features analysis. In: Geoscience and Remote Sensing Symposium, IGARSS 2001, Sydney, NSW, Australia, vol. 3, pp. 1493–1495 (2001)
[9] Borgefors, G.: Distance transformations in arbitrary dimensions. Computer Vision, Graphics, and Image Processing 27(3), 321–345 (1984)
[10] Cheng, Y.: Mean shift, mode seeking, and clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence 17(8), 790–799 (1995)
[11] Comaniciu, D., Meer, P.: Robust analysis of feature spaces: color image segmentation. In: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR 1997), pp. 750–755. IEEE Computer Society, Washington (1997)
[12] Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis Machine Intelligence 24(5), 603–619 (2002)
[13] Devillers, O.: On deletion in Delaunay triangulations. In: Proceedings of the fifteenth annual symposium on Computational geometry, pp. 181–188 (1999)
[14] Di, K., Wang, J., Ma, R., Li, R.: Automatic shoreline extraction from high-resolution ikonos satellite imagery. In: Proceeding of ASPRS 2003 Annual Conference, vol. 3., Anchorage, Alaska (2003)
[15] Gabriel, K.R., Sokal, R.R.: A new statistical approach to geographic variation analysis. Systematic Zoology 18(3), 259–278 (1969)
[16] Gold, C.: Spatial Data Structures: the Extension from One to Two Dimensions. LF Pau (ad.), Mapping and Spatial Modelling for Navigation, NATO ASI Series F 65, 11–39 (1990)
[17] Gold, C.M.: Crust and anti-crust: A one-step boundary and skeleton extraction algorithm. In: Symposium on Computational Geometry, pp. 189–196. ACM Press, New York (1999)
[18] Gold, C.M.: An object-based dynamic spatial data model, and its applications in the development of a user-friendly digitizing system. In: Proceedings of the Fifth International Symposium on Spatial Data Handling, Charleston, pp. 495–504 (1992)
[19] Gold, C.M.: Three approaches to automated topology, and how computational geometry helps. In: Proceedings of the Sixth International Seminar on Spatial Data Handling, Edinburgh, Scotland, pp. 145–158 (1994)
[20] Gold, C., Remmele, P., Roos, T.: Voronoi diagrams of line segments made easy. Proc. 7th Canad. Conf. Comput. Geom, pp. 223–228 (1995)
[21] Gold, C., Charters, T., Ramsden, J.: Automated contour mapping using triangular element data structures and an interpolant over each irregular triangular domain. In: Proceedings of the 4th annual conference on Computer graphics and interactive techniques, pp. 170–175 (1977)
[22] Gold, C.M., Snoeyink, J.: A one-step crust and skeleton extraction algorithm. Algorithmica 30(2), 144–163 (2001)
[23] Gold, C.M., Thibault, D.: Map generalization by skeleton retraction. In: Proceedings of the 20th International Cartographic Conference (ICC), Beijing, China, pp. 2072–2081 (August 2001)
[24] Gonzalez, R.C., Woods, R.E.: Digital Image Procesisng, 2nd edn. Prentice Hall, Englewood Cliffs (2002)
[25] Green, P., Sibson, R.: Computing dirichlet tessellations in the plane. The Computer Journal 21(2), 168–173 (1977)

[26] Guibas, L., Stolfi, J.: Primitives for the manipulation of general subdivisions and the computation of voronoi diagrams. ACM Transactions on Graphics 4(2), 74–123 (1985)
[27] Guibas, L.: Kinetic data structures: A state of the art report (1998)
[28] Guibas, L., Mitchell, J., Roos, T.: Voronoi diagrams of moving points in the plane. 570, 113–125 (1992)
[29] Held, M.: VRONI: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments. Computational Geometry: Theory and Applications 18(2), 95–123 (2001)
[30] Imai, T.: A Topology Oriented Algorithm for the Voronoi Diagram of Polygons. In: Proceedings of the 8th Canadian Conference on Computational Geometry table of contents, pp. 107–112 (1996)
[31] Jones, C., Bundy, G., Ware, J.: Map generalization with a triangulated data structure. CARTOGR GEOGRAPH INF SYST. 22(4), 317–331 (1995)
[32] Jones, C., Ware, J.: Proximity Search with a Triangulated Spatial Model. The Computer Journal 41(2), 71 (1998)
[33] Karavelas, M.: A robust and efficient implementation for the segment Voronoi diagram. In: International Symposium on Voronoi Diagrams in Science and Engineering (VD 2004), pp. 51–62 (2004)
[34] Kasturi, R., Fernandez, R., Amlani, M.L., chun Feng, W.: Map data processing in geographic information systems. Computer 22(12), 10–21 (1989)
[35] Lee, K.H., Cho, S.B., Choy, Y.C.: A knowledge-based automated vectorizing system for geographic information system. In: ICPR 1998: Proceedings of the 14th International Conference on Pattern Recognition, vol. 2, p. 1546. IEEE Computer Society, Washington (1998)
[36] Liu, H., Jezek, K.C.: A complete high-resolution coastline of antarctica extracted from orthorectified radarsat sar imagery. Photogrammetric Engineering and Remote Sensing 70(5), 605–616 (2004)
[37] Mioc, D., Anton, F., Gold, C., Moulin, B.: Spatio-temporal change representation and map updates in a dynamic Voronoi data structure. In: Proceedings of the Eight International Symposium on Spatial Data Handling, Vancouver, Canada, pp. 441–452 (1998)
[38] Mioc, D., Anton, F., Gold, C., Moulin, B.: Time Travel. Visualization in a Dynamic Voronoi Data Structure. Cartography and Geographic Information Science 26(2) (1999)
[39] Mostafavi, M., Gold, C., Dakowicz, M.: Dynamic Voronoi/Delaunay Methods and Applications. Computers and Geosciences 29(4), 523–530 (2003)
[40] Mioc, D., Anton, F., Gold, C.M., Moulin, B.: Map updates in a dynamic Voronoi data structure. In: ISVD, pp. 264–269 (2006)
[41] Ogniewicz, R.L.: Skeleton-space: A multiscale shape description combining region and boundary information. In: Proceedings of Computer Vision and Pattern Recognition 1994, pp. 746–751 (1994)
[42] Ogniewicz, R.L., Kübler, O.: Hierarchic Voronoi skeletons. Pattern Recognition 28(3), 343–359 (1995)
[43] Ogniewicz, R.: Automatic medial axis pruning by mapping characteristics of boundaries evolving under the euclidean geometric heat flow onto Voronoi skeletons. Technical Report 95-4, Harvard Robotics Laboratory (1995)
[44] Okabe, A., Boots, B., Sugihara, K.: Spatial Tessellations: Concepts and Applications of Voronoi Diagrams. Wiley & Sons, Chichester (1992)

[45] Okabe, A., Boots, B., Sugihara, K., Chiu, S.N.: Spatial tessellations: concepts and applications of Voronoi diagrams, 2nd edn. John Wiley & Sons Ltd, Chichester (2000)

[46] Paul Chew, L.: Constrained delaunay triangulations. Algorithmica 4(1), 97–108 (1989)

[47] Quek, F.K.H., Petro, M.C.: Human-machine perceptual cooperation. In: CHI 1993: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 123–130. ACM Press, New York (1993)

[48] Rognant, L., Chassery, J.M., Goze, S., Planès, J.G.: The delaunay constrained triangulation: The delaunay stable algorithms. In: IV, pp. 147–152 (1999)

[49] Roos, T.: Voronoi diagrams over dynamic scenes. Discrete Appl. Math. 43(3), 243–259 (1993)

[50] Shewchuk, J.R.: Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In: Lin, M.C., Manocha, D. (eds.) FCRC-WS 1996 and WACG 1996. LNCS, vol. 1148, pp. 203–222. Springer, Heidelberg (1996)

[51] Shewchuk, J.R.: Adaptive precision floating-point arithmetic and fast robust geometric predicates. In: Discrete and Computational Geometry, vol. 18, pp. 305–363 (1997)

[52] Sonka, M., Hlavac, V., Boyle, R.: Image Processing, Analysis, and Machine Vision. PWS publishing (1999)

[53] Sugihara, K., Iri, M., Inagaki, H., Imai, T.: Topology-Oriented Implementation–An Approach to Robust Geometric Algorithms. Algorithmica 27(1), 5–20 (2000)

[54] Ware, J., Jones, C.: Conflict Reduction in Map Generalization Using Iterative Improvement. GeoInformatica 2(4), 383–407 (1998)

[55] Yang, W., Gold, C.: Dynamic spatial object condensation based on the Voronoi diagram. In: Proceedings, Fourth International Symposium of LIESMARS, vol. 95, pp. 134–145 (1995)