Some Tools for Modeling and Analysis of Surfaces

Carsten Dorgerloh[*]
University of Bonn

Jens Lüssem[†]
University of Bonn

Morakot Pilouk[‡]
ESRI-Redlands

Jürgen Wirtgen[§]
University of Bonn

Abstract

We present some algorithms which construct a triangular graph given a set of points, where each face of the graph complies with the Delaunay criteria. Then we develop an linear time algorithm to construct the contour of such a triangular graph, where each face is coloured either black or white. Our techniques avoid expensive trigonometric computations. We intoduce our algorithms for the 2-dimensional case and show how to extend them to the d-dimensional case in a straightforward manner.

[*]Dept. of Computer Science, University of Bonn, 53117 Bonn, Germany. Email: carsten@cs.uni-bonn.de

[†]Dept. of Computer Science, University of Bonn, 53117 Bonn, Germany. Email: jens@cs.uni-bonn.de

[‡]ESRI, 380 New York Street, Redlands, CA 92373, USA, email: mpilouk@esri.com

[§]Dept. of Computer Science, University of Bonn, 53117 Bonn, Germany. Email: wirtgen@cs.uni-bonn.de

1 Introduction

Consider the following scenario: An oil company drills n boreholes $p_1, \ldots, p_n$. With each p_i an value $val(p_i)$ will be associated, which represents the expected amount of oil supposed to be in the neighborhood of p_i. They want to determine which connected regions in the plane promise to give a lot of oil.
In order to solve problems of this type, we perform three phases.

Triangulation phase: Based on the n points, we construct a planar graph structure T consisting of non-intersecting triangles.

Rating phase: For each triangle $t = (p_1, p_2, p_3)$ (with p_1, p_2, p_3 being the vertices of t) we evaluate $f(p_1, p_2, p_3)$, where f is an appropriate rating, e.g. the mean of the $val(p_i)$. Depending on some threshold we color the triangle black or white.

Contour phase: In this phase we compute the set of cycles C defined by edges of T, which are on a common boundary of a black and a white triangle. Furthermore, C is constrained to contain only those cycles which are not enclosed by another cycle.

With growing n, this construction tends to be similar to the real world, since more points give us more information. Therefore our modeling of the reality will become more and more close grained.
The paper is organized as follows. Section 2 describes the triangulation phase. To be more precise, we discuss an triangulation algorithm which complies with the Delaunay criteria. Section 3 contains the elementary steps of the algorithm for the contour problem and shows how to implement them efficiently. From the proof of correctness of the elementary steps the correctness of the main algorithm of the sequence is immediately clear. Finally, we describe that the algorithm can easily be extended to the d-dimensional case.

2 Constructing the Delaunay Triangulation

Triangulation has been applied in many disciplines especially for modeling and analysis of surface, e.g. terrain modeling in GIS, civil engineering, landscape architecture. From a triangulated structure representing a surface, we can compute slope, aspect, visibility, isolines, light reflectance from the surface, volume above or below the surface with respect to a given datum. For our algorithms, we use the Delaunay triangulation, having some nice properties [Au 91].

Definition 1 *Let $\mathcal{P} = \{p_1, ..., p_n\} \subseteq \mathbb{R}^d$ be a set of d-dimensional points. For all $p \in \mathcal{P}$ we define the Voronoi cell $V(p)$ to be the subset of $\mathbb{R}^d$ which is closer to p than to any other point in $\mathcal{P}$. Formally*

$$V(p) = \{x \in \mathbb{R}^d : \|x - p\| < \|x - q\| \, \forall q \in \mathcal{P} \setminus \{p\}\}$$

Let $H(p, q)$ the halfspace defined through the bisecting hyperplane of p and q containing p. So we have another definition of $V(p)$:

$$V(p) = \bigcap_{q \in \mathcal{P} \setminus \{p\}} H(p, q)$$

The cells $V(p)$ partition the $\mathbb{R}^d$ and form by this way the Voronoi diagram. The dual graph will be called the Delaunay triangulation. Here we have some elementary properties of the Delaunay triangulation.

Property 1

1. *The delaunay triangulation is a partition of the $\mathbb{R}^d$ in simplices, whose vertices are the points of $\mathcal{P}$.*
2. *The Delaunay triangulation of a Voronoi diagram is unique:*
   The cells which circumscribe the simplices do not contain any point of $\mathcal{P}$ in their interior.

This property gives us the simple algorithm $SimpleDelaunay(\mathcal{P})$ with a worst-case complexity of $O(n^{\lceil n/2 \rceil + 1})$ [Bo 81].

```
SimpleDelaunay(P)
Input:  A set $\mathcal{P} = \{p_1, ..., p_n\} \subseteq \mathbb{R}^d$ of points.
Output: The Delaunay triangulation of $\mathcal{P}$.

1. Let $s_1 := (p_1, ..., p_{d+1})$ be the first simplex.
2. Define $S = \{s_1\}$ to be the set of actual simplices. Set $card\_simp := 1$, the number
   of actual simplices.
3. For all $i = d+2, ..., n$ do
   (a) $R(p_i) := \emptyset$. In the following $R(p_i)$ will be the region defined by the union of the
       simplices in $S$, whose circumscribing ball contain $p_i$.
   (b) For all $j = 1, ..., card\_simp$ do
       • If $circumball(p_i, s_j)$ then
         – $R(p_i) := R(p_i) \cup s_j$,
         – $S := S \setminus \{s_j\}$,
   (c) Let $F(p_i)$ the set of facets defined by $R(p_i)$.
   (d) Construct new simplices by connecting $p_i$ to the elements of $F(p_i)$.
   (e) Update the number $card_simp$
```

To complete this algorithm we have to describe the subroutine $circumball(p, s)$ which returns *true* iff p is contained in the circumscribing ball of the simplex s. The easiest way to do this, is to compute this ball and test whether p is inside or not.
In the 2-dimensional case, we know from the elemental geometry, that the center of the circumscribing circle of a triangle is defined by the intersecting point of the orthogonals on the centers of the sides. It suffices to calculate two of these and solving the linear system.
It is easy to see that we can generalize the algorithm to d dimensions with the following recursive procedure $ConstructCircumball_d(s)$:

```
ConstructCircumball_d(s)
Input:  A simplex $s = (p_1, ..., p_{d+1})$.
Output: The center of the circumscribing ball of $s$.

1. If $d = 1$, return the center of $s$. ($s$ is a line segment)
   Else, take two arbitrary facets $f_1$ and $f_2$ of $s$ and calculate
   • $c_1 = ConstructCircumball_{d-1}(f_1)$,
```

- $c_2 = ConstructCircumball_{d-1}(f_2)$.

2. Construct the orthogonals o_1 and o_2 on f_1 and f_2 thru c_1 and c_2.
3. Calculate the intersection of o_1 and o_2, which gives us the center c.
4. Return c.

Now it is easy to implement *circumball*(p, s). We calculate the center c of the ball which circumscribes the simplex s and take some arbitrary vertex p_i $(i = 1, ..., d+1)$ of s. Now we have only to check whether

$$||c - p_i|| \geq ||c - p||,$$

or not.

3 The Contour Algorithm

Before we explain the elementary steps of the contour algorithm we need the following definitions.
Let $G = (V, E)$ be a planar graph. Consider a fixed plane embedding of G. The unbounded region is called the *exterior face*. Other faces are called *interior faces*. The vertices and the edges on the exterior face are called *exterior vertices* and *exterior edges*, respectively. For each vertex v, $N(v)$ denotes the set of neighbors of v. A planar graph $T = (V_T, E_T)$ has a *triangular embedding*, if every face of T, except the exterior face, is a triangle. The triangles of a *colored triangular embedding* are colored black and white, respectively. The color of the external face is always assumed to be white. Let S be a set that contains all edges of T which are on a common boundary of a black and a white triangle. In fact, S is a collection of edge-disjoint simple cycles (the points of the cycles being evident by context) of T, which are separating white and black regions. The problem of finding simple cycles is one of the most basic and natural algorithmic graph problems (see [Le 90]) and was considered by many researchers e.g. [Mo 85], [AYZ 95], [DW 97a], [DW 97b]. However, in the present paper we search for a special set of cycles - namely the external contour.
An *(external) contour* C (introduced in [DL 95]) of a colored triangular embedding T consists of those cycles of S, which are not enclosed by another cycle of S. The task of computing the external contour of T can now be formalized to produce the set C.
The following well known lemma (see e.g. [Ha 69] [Ev 79]) is helpful, because for planar graphs it implies that an $\mathcal{O}(|V| + |E|)$ algorithm is really an $\mathcal{O}(|V|)$ algorithm.

Lemma 2 *If $G = (V, E)$ is any planar graph with $|V| \geq 3$. Then G has at most $3|V| - 6$ edges.*

The algorithm is built of the following steps.

3.1 Construction of the Dual Graph

Given a colored triangular embedding T, its *colored dual* $G = T^*$ is constructed as follows: simply trace the boundary of each face, place a vertex in G for each face of T (excluding the exterior face) and assign the corresponding color to it. If two faces of T have an edge e_T in common, join the corresponding vertices in G by an edge e.

```
$extend(G)$
    $V := V \cup \{v_{ext}\}$
    for each $u \in V - \{v_{ext}\}$ do
        if $degree(u) < 3$ then
            $E := E \cup \{u, v_{ext}\}$
        end if
    end for
```

Figure 1: Extend G by v_{ext}

It is quite straightforward to solve the problem in $\mathcal{O}(|V|)$ time sequentially, if we trace the boundaries by following cyclic linked lists.
In the following we denote by *construct_dual*(T) the procedure that executes the step as described above.

3.2 The Extension of the Dual

Now, we extend G by introducing a new vertex v_{ext}. v_{ext} corresponds to the external face of T and is assigned the color white. We connect this vertex to each node $u \in V - \{v_{ext}\}$ with $degree(u) < 3$ (see Figure 1). That are exactly those vertices in G corresponding to faces in T, which have a common boundary with the external face of T. Hence, adding v_{ext} cannot destroy the planarity of G.

3.3 The Hollow Out Step

What is the purpose of introducing the special vertex v_{ext}? The reason is, that we are now able to hollow out the white "regions" of T starting at the external face of T. This is done by changing G, while T remains unchanged. Furthermore, black vertices corresponding to faces in T which contribute edges to the contour of T are marked. The procedure which executes this step is given in Figure 2.
Before we prove the correctness of *hollow_out*(G), we need the following definition. We say a vertex $u \in V$ is *enclosed by black*, if there is no path from v_{ext} to u in G such that all vertices on that path, except maybe u, are coloured white.

Lemma 3 *hollow_out*(G) *applied to the dual* $G = (V, E)$ *(extended by* v_{ext}*) of any triangular embedding of* $T = (V_T, E_T)$ *requires* $\mathcal{O}(|V|)$ *time.*

PROOF: S initially contains all neighbors of v_{ext}. In the course of the algorithm the set S is modified as follows. In each execution of the while-loop one vertex $u \in S$ is picked. If the color of u is white, then all neighbors of u, except v_{ext}, are inserted into S. Furthermore, G is changed: each $v \in N(u) - \{v_{ext}\}$ is connected to v_{ext}, u and all incident edges are deleted from G. On the other hand, if the color of u is black, then u is marked. Finally, u is deleted from S. It follows by an inductive argument, that every white vertex which is not enclosed by black is deleted from G. Moreover, every black vertex which is not enclosed by black is marked.

```
$hollow\_out(G)$
    $S := N(v_{ext})$
    while $S \neq \emptyset$ do
        $u :=$ first element of $S$
        if $colour(u) = white$ then
            for each $v \in N(u) - \{v_{ext}\}$ do
                if $v_{ext} \notin N(v)$ then
                    $E := E \cup \{v, v_{ext}\}$
                $E := E - \{v, u\}$
                $S := S \cup \{v\}$
            end for
            $E := E - \{u, v_{ext}\}$
            $V := V - \{u\}$
        else
            $mark(u)$
        end if
        $S := S - \{u\}$
    end while
```

Figure 2: Formulation of the procedure *hollow_out*

Lemma 2 guarantees that the while loop is executed at most $\mathcal{O}(|V|)$ times. Each of the $\mathcal{O}(|V|)$ runs of the while loop needs constant time because each vertex $u \in V - \{v_{ext}\}$ has at most three neighbors. ∎

3.4 Finding the Black Components

In this step we first remove the vertex v_{ext} and all edges which are incident to that node from G (see Figure 3). The remaining graph is made up of what we call the *black components* of G. The computation of the black components can be done using standard algorithms for connected components which are based on depth-first-search or breadth-first search (see e.g. [AHU 83]). This step runs in $\mathcal{O}(max(|V|, |E|))$ time and is denoted by $compute\_black\_components(G)$.

3.5 Computation of the External Contour

We describe the procedure *contour_of_component* (G_i, T, G, G_{init}), which computes the external contour of a component G_i of G (see the Appendix).
The correctness of the procedure will be immediately clear from the following lemma. Again, we need several definitions. Each $u \in V$ corresponds to a face in T. Let us denote by $\lambda(u)$ the set of vertices and by $\gamma(u)$ the set of edges of the corresponding triangle in T. By G_{init} we denote the graph produced by $construct\_dual(T)$.

Lemma 4 *The edge list returned by* $contour\_of\_component(G_i, T, G, G_{init})$ *is the external contour of* G_i. *The procedure runs in time* $\mathcal{O}(|V|)$.

```
$remove\_v\_ext(G)$
    for all $u \in N(v_{ext})$
        $E := E - \{u, v_{ext}\}$
    end for
    $V := V - \{v_{ext}\}$
```

Figure 3: Removal of v_{ext}

```
$external\_contour()$
    $construct\_dual(T)$
    $extend(G)$
    $hollow\_out(G)$
    $remove\_v\_ext(G)$
    $compute\_black\_components(G)$
    for each black component $G_i$ of $G$
        $contour\_of\_component(G_i, T, G, G_{init})$
```

Figure 4: The main procedure

PROOF: Since only the faces corresponding to marked vertices contribute edges to the external contour $E_{contour}(i)$ of G_i, it suffices to investigate only such vertices. Let u be a marked vertex with $degree_{G_{init}}(u) = 3$ and $v \in N_{G_{init}}(u)$. Let $e_c \in E_T$ be the common edge of the faces corresponding to the vertices u and v, respectively. Furthermore, assume that $colour(v) = white$. We claim that $e_c \in E_{contour}(i)$. Suppose, to the contrary, that $e_c \notin E_{contour}(i)$. As an immediate consequence, the other edges of $\gamma(u)$ are not in $E_{contour}(i)$. But this implies that u is not marked, which is a contradiction. The other cases consider exterior edges and can be proven similarly. Since the computation of a common edge of two faces takes $\mathcal{O}(1)$ time, $contour\_of\_component(G_i, T, G, G_{init})$ runs in time $\mathcal{O}(|V|)$. ■

3.6 The Main Procedure

Now we present our main procedure, (see Figure 4), putting the developed things together. We summarize the analysis of the previous section in the following theorem.

Theorem 5 *The external contour of a triangular graph can be computed in time $\mathcal{O}(|V|)$.*

An algorithm with this complexity is given explicitely.

3.7 Extension to the d-Dimensional Case

The above algorithm generalizes to the d-dimensional ($d > 2$) case where we consider d-simplexes.
First, we illustrate the changes neccessary by describing the 3-dimensional case. Here, we have to consider tetrahedrons instead of triangles. The dual graph is now constructed by identifing each tetrahedron by a vertex. Two vertices are joined by an edge, if their corresponding tetrahedrons have a face in common. Thus, each vertex in the dual graph has at most four neighbours. The adaption of the other steps is straightforward and it can easily be shown that the algorithm runs in time linear in the number of tetrahedrons. This method can be extended to the d-dimensional ($d > 3$) case as well: Two d-simplexes are adjacent if they have a $(d-1)$-simplex in common. A vertex in the dual graph has at most $d+1$ neighbours. Again, it can be shown that the algorithm runs in time linear in the number of d-simplexes.

4 Further Extensions and Related Problems

The algorithms presented here only handle a set of points. Many surface modelling problems, however, may also involve preservation of the geometry of linear features. These features may represent lines or boundaries of polygonal objects. For the 3-dimensional case, the geometry of the surface itself (faces) may need to be preserved within the tetrahedral network. The preservation of geometry of those features allows to extract and reconstruct the features directly from the irregular network. Algorithms that perform fast constrained network formation close to linear time still need further studies.

Acknowledgements

We thank Armin Cremers and Marek Karpinski for helpful comments.

References

[AHU 83] Aho, A., Hopcroft, J., Ullman, J., *Data Structures and Algorithms*, Addison-Wesley Publishing Company, 1983.

[AYZ 95] Alon, N., Yuster, R., Zwick, U., *Color-coding*, Proc. 42^{nd} Journal of the ACM (1995), pp. 844–856.

[Au 91] Aurenhammer, F., *Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure*, ACM Computing Surveys **23**(3) (1991), pp. 345–405.

[Bo 81] Bowyer, A., *Computing Dirichlet tessellations*, Computer J. **24** (1981), pp. 162–166.

[DL 95] Dorgerloh, C., Lüssem, J., *A simple linear-time algorithm to find the contour in a coloured triangular graph*, Research Report 85146-CS, Institut für Informatik der Universität Bonn, 1995.

[DW 97a] Dorgerloh, C., Wirtgen, J., *Faster Finding of Simple Cycles in Planar Graphs on a randomized EREW-PRAM*, Proc. 2^{nd} Workshop on Randomized Parallel Computing (1997), held in conjunction with IPPS'97.

[DW 97b] Dorgerloh, C., Wirtgen, J., *Once again: Finding simple cycles*, Research Report 85165-CS, Institut für Informatik der Universität Bonn, 1997.

[Ev 79] Even, S., *Graph Algorithms*, Computer Science Press, 1979.

[Ha 69] Harary, F., *Graph Theory*, Addison-Wesley Publishing Company, 1969.

[Le 90] Leeuwen, J. v., *Graph Algorithms. Handbook of Theoretical Computer Science, Volume A, Algorithms and Comlexity*, chapter 10, pp. 525–631, Elsevier and The MIT Press, 1990.

[Mo 85] Monien, B., *How to find long paths efficiently*, Annals of Discrete Mathematics **25** (1985), pp. 239–254.

Appendix

```
$contour\_of\_component(G_i, T, G, G_{init})$
  $M := \{u \in V_i | u$ is marked$\}$
  $E_{contour}(i) := V_{contour}(i) := \emptyset$
  for each $u \in M$ do
    $S := V_{aux} := \emptyset$
    if $degree_{G\_init}(u) = 3$ then
      for each $v \in N_{G_{init}}(u)$ do
        if $colour(v) = white$ then
          for each $u_T \in \lambda(u)$, $v_T \in \lambda(v)$ do
            if $u_T = v_T$ then $V_{contour}(i) := V_{contour}(i) \cup \{u_T\}$
            if $|V_{contour}(i)| = 2$ then $E_{contour}(i) := E_{contour}(i)$
              $\cup\{Pop(V_{contour}(i)), Pop(V_{contour}(i))\}$
          end for
        end if
      end for
    else if $degree_{G\_init}(u) = 2$ then
      for each $v \in N_{G_{init}}(u)$ do
        for each $u_T \in \lambda(u)$, $v_T \in \lambda(v)$ do
          if $u_T = v_T$ then
            if $u_T \in V_{aux}$ then $inner\_point := u_T$
            $V_{aux} := V_{aux} \cup \{u_T\}$
            if $colour(v) = white$ then $V_{contour}(i) := V_{contour}(i) \cup \{u_T\}$
            if $|V_{contour}(i)| = 2$ then $E_{contour}(i) := E_{contour}(i)$
              $\cup\{Pop(V_{contour}(i)), Pop(V_{contour}(i))\}$
          end if
        end for
      end for
      $V_{aux} := V_{aux} - \{inner\_point\}$
      $E_{contour}(i) := E_{contour}(i) \cup \{Pop(V_{aux}), Pop(V_{aux})\}$
    else if $degree_{G\_init}(u) = 1$ then
      for each $u_T \in \lambda(u)$ do
        $S := S \cup \{u_T\}$
        for each $v_T \in \lambda(u) - S$ do
          $E_{contour}(i) := E_{contour}(i) \cup \{u_T, v_T\}$
        end for
      end for
      $v :=$ neighbour of $u$ in $G_{init}$
      if $colour(v) = black$ then
        for each $u_T \in \lambda(u)$, $v_T \in \lambda(v)$ do
          if $u_T = v_T$ then $V_{aux} := V_{aux} \cup \{u_T\}$
          if $|V_{aux}| = 2$ then $E_{contour}(i) := E_{contour}(i) - \{Pop(V_{aux}), Pop(V_{aux})\}$
        end for
      end if
    end if
  end for
```