A Topology-Oriented Algorithm for the Voronoi Diagram of Polygons

Toshiyuki Imai
Department of Mathematical Engineering and Information Physics
University of Tokyo

1 Abstract

An algorithm for constructing Voronoi diagrams whose generators are points, line segments, segment chains and polygons is shown. In this algorithm, topological structure of the diagrams is treated as prior information to numerical values of reals. If the computation is precise enough for conventional algorithms to construct the correct Voronoi diagrams, this algorithm also works well. Moreover, even when the precision is not enough, this algorithm never fails and ends up with a diagram as an output with several topological properties which Voronoi diagrams must have. In this sense, this algorithm is robust against numerical error. This algorithm is designed in the practical assumption that numerical error exists; hence it has not any exceptional rules for degeneracy but still works for degenerate inputs.

2 Voronoi Diagrams

The algorithm in this paper belongs to an incremental type. At first, we introduce some definitions and notations. In this paper, polygon is considered as a simple closed chain of line segments.

Let g_i be a point, a line segment, a segment chain or a polygon for each i $(1 \leq i \leq N)$. Let us assume that each g_i is simple and any g_i, g_j $(i \neq j)$ do not intersect.

We also assume that the set $\{g_1, \cdots, g_n\}$ is a plane graph in which the degree of any vertex is at most 2. We call the set $G = \{g_1, \cdots, g_N\}$ a generator set. We define the distance d between a point v and a generator g_i as $d(v, g_i) = \inf\{\|u - v\| \mid u \in g_i\}$, where $\| \bullet \|$ is the Euclidean norm. For g_i, g_j $(i \neq j)$, we define region $R(g_i, g_j)$ as follows:

$$R(g_i, g_j) = \{v \in \mathbf{R}^2 \mid d(v, g_i) < d(v, g_j)\}. \tag{1}$$

Let $R(g_i) = \bigcap_{j \neq i} R(g_i, g_j)$. Those regions $R(g_1), \cdots, R(g_N)$ define a partition $V(G)$ of the plane $\mathbf{R}^2$ and we call the partition the (generalized) **Voronoi diagram** for the set G. We call the region $R(g_i)$ **Voronoi region** of the generator g_i. The **Voronoi edge** is defined as a curve segment which is the common boundary of 2 Voronoi regions and the **Voronoi vertex** is defined as an endpoint of a Voronoi edge. Voronoi vertices and Voronoi edges have a structure like a plane graph. In this paper, we use the terms **region, edge** and **vertex** instead of the Voronoi region, the Voronoi edge and the Voronoi vertex respectively. We also use the term the **Voronoi diagram of polygons** for simplicity even if its generators are points, line segments, segment chains and polygons

We consider the next problem.

- For a set of generators in the unit square $[0, 1]^2$, construct the Voronoi diagram of the generators in the same square.

This setting of the problem does not lose generality, because the generators can be transformed in this square by scale transformation and parallel displacement.

In order to describe the algorithm, we introduce another partition of the plane called the **separated Voronoi diagram** of polygons. For each generator $g_i \in G$, we make a set G_0 of points by gathering

1. if g_i is a point, g_i itself,
2. if g_i is a line segment, 2 endpoints,
3. if g_i is a segment chain, all endpoints of all segments,
4. if g_i is a polygon, all corner points of g_i.

We denote $G_0 = \{p_1, \cdots, p_m\}$. By removing the points in G_0 from each generator g_i, nothing is left if g_i is a point, and some segments without their endpoints are left otherwise. We call the segment without its endpoints the **open segment**. Suppose $\{e_1, \cdots, e_n\}$ is a set of all open segments. For each k $(0 < k \leq n)$, we denote $G_k = G_0 \cup \{e_1, \cdots, e_k\}$.

Like the Voronoi diagram of polygons, we define a region $R(p, q)$ for $p, q \in G_n$, as follows: If p, q are an open segment and its endpoint, (assume $\{p, q\} = \{p_i, e_k\}$ and p_i, p_j are endpoints of e_k),

$$R(p_i, e_k) = \{x \in \mathbf{R}^2 \mid (x - p_i) \cdot (p_j - p_i) < 0\},\ R(e_k, p_i) = \{x \in \mathbf{R}^2 \mid (x - p_i) \cdot (p_j - p_i) > 0\}. \tag{2}$$

Otherwise, we use the definition by the formula 1. Let $R(q) = \bigcap_{p \neq q} R(q,p)$. We call the partition of the plane defined by $R(q)$ $(q \in G_k)$ the **separated Voronoi diagram** for the generator set G_k. We use the notation $V(G_k)$ for the separated Voronoi diagram. The separated Voronoi diagram and the Voronoi diagram of polygons use the same notation $V(\bullet)$ but both can be distinguished by the generator sets G and G_k. We also use terms **region, edge, vertex** and so on in common.

We can get easily the Voronoi diagram $V(G)$ from the separated Voronoi diagram $V(G_n)$ by removing edges whose both sides are regions of the same generator in G (Figure 1). That's why the algorithm constructs the separated Voronoi diagram $V(G_n)$ instead of the Voronoi diagram $V(G)$ of polygons. Especially, in the separated Voronoi diagram $V(G_k)$, we call edges between regions of an open segment and its endpoints a **separator**.

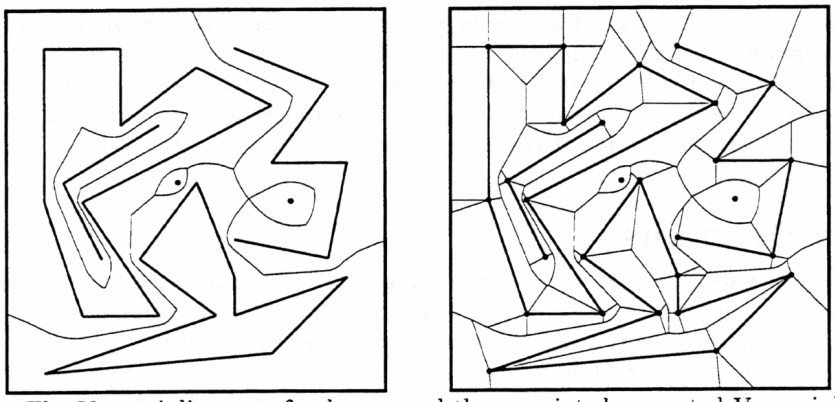

Figure 1: The Voronoi diagram of polygons and the associated separated Voronoi diagram.

The Voronoi diagram has some unbounded edges. This is the difference between the Voronoi diagrams and plane graphs. The algorithm adds 3 extra points to the generators. We use 3 points of a triangle which includes the square $[-1,2]^2$ to make the number of the unbounded edges just 3 without any influence on the Voronoi diagram in the unit square $[0,1]^2$.

3 Topology-Oriented Algorithm

The algorithm takes the topology-oriented method. Geometric algorithms are considered as a series of processing data of figures by arithmetic of reals (floating point numbers). In general, arithmetics of reals have numerical errors. The topology-oriented method is an approach such that the topological properties which the figures must have are considered prior to the results of arithmetic. Conventional approaches to geometric algorithms assume the complete accuracy of the computation. Just this assumption assures the data free from topologically inconsistent structures automatically, but this assumption fails when the algorithm is implemented. As a result of the assumption, there occurs a problem that the program by the algorithm leads to inconsistent judgements, endless loops, failures, or data destructions. The topology-oriented method avoids the problem by keeping the topological properties of the data.

We first treat the algorithm for the Voronoi diagram of line segments and then we extend it to the algorithm for the Voronoi diagram of polygons.

In this paper, we take numerical errors for granted. We also take the following assumptions:

- Any combinatorial computation (computation of integers) is correct,
- the same computation of reals always has the same numerical errors.

In general, those assumptions hold on conventional computer systems. We also take assumptions on the input segments and the computer as follows:

- The result of computation of the length of a line segment is always positive.
- The intersection test reports no intersecting pair of the line segments is found.

The algorithm we present in this paper is in the incremental type. It is as follows:

Algorithm 1 (Construction of a separated Voronoi diagram)

1.1 Construct the Voronoi diagram $V(G_0)$ of points.
1.2 For each $k = 1, \cdots, n$, do **1.2.1**.
1.2.1 Add the open line segment e_k to the generator set, to make $V(G_{k-1})$ into $V(G_k)$.

After the execution of the algorithm 1, we obtain the separated Voronoi diagram $V(G_n)$. As we mentioned in the previous section, we can easily get the Voronoi diagram of line segments from this $V(G_n)$ by removing the edges of separators. We use Sugihara-Iri's algorithm [6] to construct the Voronoi diagram $V(G_0)$ of points at **1.1** in the algorithm 1, Sugihara-Iri's algorithm also takes a topology-oriented method. Therefore, we just show the detail of **1.2.1** in the algorithm 1. This part of the algorithm adds an open segment to the generator set and modifies the Voronoi diagram.

Let the open segment e_k be added to the generator set G_{k-1}. We take the data structure in which all of open segments have arbitrary orientation. The endpoints of an open segment consist of the **initial point** and the **terminal point**. Let p_i and p_j be the initial and terminal points of the open segment e_k respectively. The difference between $V(G_{k-1})$ and $V(G_k)$ has the following properties (Figure 2):

- Newly born vertices and edges make a closed path which surrounds the vertices and edges to be removed.
- The vertices and edges to be removed constitute a tree structure.
- The tree structure has the unique path between the regions of the endpoints p_i and p_j.
- The path goes between the two groups of regions; one is right to the open segment e_k ant the other is left.
- Any separator is not completely removed.

The algorithm we present takes such properties. The following algorithm 2 adds the open segment e_k and

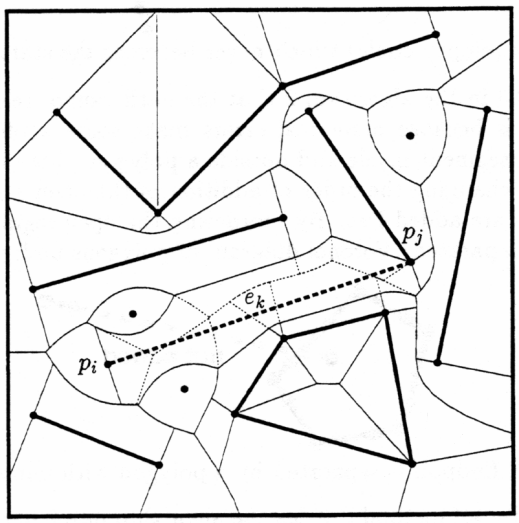

Figure 2: Adding an open segment.

modifies the Voronoi diagram from $V(G_{k-1})$ to $V(G_k)$:

Algorithm 2 (Addition of an open segment e_k.)
2.1 Find the start vertex on the boundary of $R(p_i)$.
2.2 Set the search point to the start vertex.
2.3 Until the search point get to the region $R(p_j)$, do **2.3.1** and find the path.
2.3.1 Find the unsearched edge which lies between the right and left regions to e_k and which has the search point as an end vertex, and then set the other end vertex to the new search point.
2.4 Search the edges to be removed from the vertices on the path, and determine the tree structure to be removed. **2.5** add new vertices and edges which surround the tree and remove the tree to make the new region, i.e., the region of e_k.

At **2.3** in the algorithm 2, if the path gets the wrong edge such as an unbounded edge, a separator, a edge which makes a closed path or which returns to the start region $R(p_i)$, we backtrack the path to the vertex where the

judgement is the most unsure. At **2.4** in the algorithm 2, if the tree gets the wrong edge above, we make a vertex on the edge to keep the tree structure to be removed. For the generator set of points and line segments as an input, the algorithm 2 guarantees the output with the following properties under any numerical error:

- Any endpoint or open segment has a simply connected region.
- Regions of any open segment and its endpoints are adjacent on the separator.

We adapt the algorithm 2 (which is for the set of points and line segments) to the one for polygons. Suppose the open segment e_k is to be added and e_k is adjacent to already added open segment e_l at the initial point p_i of e_k. Then, at **2.1** in the algorithm 2, such property holds that one of vertices on the separator between e_k and p_i never becomes the start point of the path (Figure 3). The corresponding property holds at the terminal point p_j of the open segment e_k if the open segment has an adjacent open segment already added at the endpoint p_j. The algorithm for the set of polygons keep those properties. In the case of the Voronoi diagram of polygons,

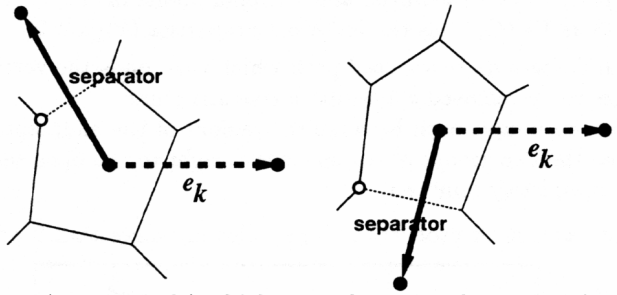

Figure 3: Vertex (empty circle) which never becomes the start point of the path.

such problem may occurs at **2.3** in the algorithm 2 that the path cannot reach the destination without taking a separator (Figure 4). This is because numerical errors make some wrong judgements which separate the endpoints of an unadded open segment inside and outside a polygon whose boundary has already been added. We can avoid this problem by changing the order of addition of the open segments in such way that the open segments which close polygons are added last. By reordering the open segments to be added, the algorithm 2 guarantees the output of the (separated) Voronoi diagram of polygons under any numerical errors.

Figure 4: Endpoints separated by a polygon with numerical errors.

In general, the topology-oriented method divides the algorithm into 2 parts. One is the combinatorical part and the other is numerical part. This means we can first make a combinatorical part of the algorithm, which is robust against numerical errors and which ends with an output which keeps some topological properties and then, we can improve the way of numerical computations in the algorithm. In the algorithm for the Voronoi diagram of polygons, it is also important to improve the way of numerical computations; however we skip the details because of space limitation.

At **2.3** in the algorithm 2, to find the next edge on the path to be removed, we compute for each neighbor generator, a value of how left the generator is to the open segment e_k at the search point v of the path. We call this value **left-value**. The sign of the left-value shows whether the generator is left to e_k or not. If the generator is an open segment e_l, we use the orthogonal projection p of v to e_l for the computation of the left-value, and if the generator is a point, we use this point as p. We can compute the left-value of the generator at v as the signed distance from the point p to the segment e_k (Figure 5).

Especially, in the case of computing the left-value of the open segment e_l at v, it takes a lot of computations with errors to compute the coordinates of v and p. For the Voronoi diagram of polygons, if the open segments e_k and e_l are adjacent, the sign of the left-value of e_l is independent from the search point v of the path. In

such case, to diminish the influence of errors, we first test whether e_l is left to e_k or not, without computing the left-value, directly from the coordinates of the endpoints of e_k and e_l and then, we set a constant as the left-value of e_l.

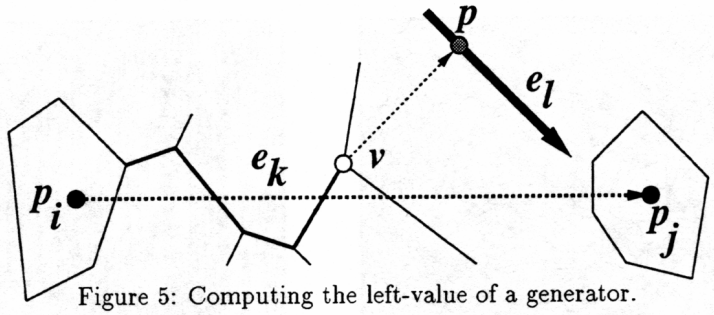

Figure 5: Computing the left-value of a generator.

As the format of the input for the set of points and line segments, just an integer and a series of reals are enough in which the integer is the number of segments and the series is the coordinates of the endpoints of segments and the isolated points. This format is not enough for the input of the Voronoi diagram of polygons. As the input, information about incidence of open segments and endpoints is required. The same kind of thing happens on the output of the algorithm. These mean that the input and output for the set of polygons become larger in their size than those for the set of points and line segments. We do not mention the details of the input or the output anymore for such difference make little influence on the algorithm.

Allowing the numerical error means that we cannot find any degeneracy in the input. Hence, the algorithm we present does not have any exceptional rules for the degenerate input. The algorithm treats a degenerate input in the same way as it treats a non-degenerate input.

4 Complexities

We discuss the complexity of the algorithm. Let n be the number of isolated points and open segments in the input. The Voronoi diagram and the separated Voronoi diagram of polygons have $O(n)$ vertices and $O(n)$ edges.

We discuss the time complexity of the algorithm first. The optimal one is $O(n \log n)$[1]. To construct the Voronoi diagram of all endpoints of open segments and isolated points, it takes $O(n^2)$ time by Sugihara and Iri's algorithm. When an open segment is added to the generators, it takes $O(n)$ time to find the initial point of the path, takes $O(n^2)$ ($O(n)$ if no backtrack happens) time to find the path and takes $O(n)$ time for determining the tree structure. It takes $O(n)$ to add and remove edges to create a region. Hence, the total time complexity is $O(n^3)$ in the worst case. It is reduced to $O(n^2)$ if the numerical errors are so small that no backtrack happens at the path search. If there is no numerical error, our algorithm has the $O(n^2)$ time complexity, which is not optimal but the same as the conventional incremental algorithm for the Voronoi diagram.

Next, let us discuss the space complexity of the algorithm. It is easy to see the space complexity is $O(n)$ and optimal because the number of edges and vertices is $O(n)$.

We show the practical time complexity of the algorithm. We first implemented the algorithm for the Voronoi diagram of line segments as a program SEGVOR and then we adapted SEGVOR to a program for the Voronoi diagram of polygons named PLVOR. We show the performance of PLVOR. Inputs consist of just polygons obtained by the following algorithm.

Algorithm 3 (Construction of an input set of polygons.)
3.1 Genearate the coordinates of endpoints $p_1, \cdots, p_n$ by random reals uniformly distributed in an interval $[0, 1]$.
3.2 Set the open segments $e_1, \cdots, e_n$ in the following way: $e_i = (p_i, p_{i+1})$ $(i = 1, \cdots, n-1); e_n = (p_n, p_1)$.
3.3 While an intersecting pair (e_i, e_j) of segments exists, do **3.**3.1.
3.3.1 If the exchange of the terminal points of e_i and e_j makes no pair of edges with the same endpoints,
exchange the terminal points of e_i, e_j,
otherwise,
exchange the initial point of e_i and the terminal point of e_j.

At **3.**3 in the algorithm 3, we take 2 ways to search an intersecting pair (e_i, e_j) of open segments. The first

way is the search in the ascending lexicographical order of the indices (i, j) of open segments. We call the way A. The second way is based on the sweep line method by [2]. We call the way B. We show examples of 1024 segments in fig. 6. We can easily see that they are not alike. For each way and each number of segments, we

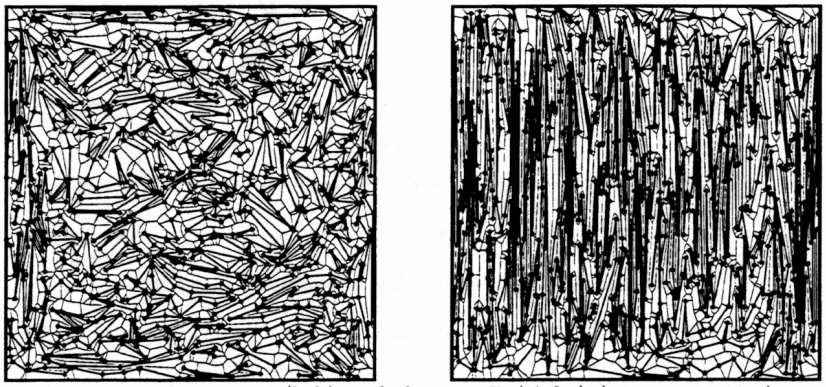

Figure 6: The way A (left) and the way B (right) (1024 segments).

generated 10 inputs and made the Voronoi diagrams. Because the execution times on SPARCstation ELC are almost the same for each way and each number of segments, we show the average in the table 1. The execution times are a little different between the way A and the way B, but the complexities are almost the same and nearer to $O(n)$ than to $O(n^2)$.

Table 1: The average of the execution time [s].

num. of segments	256	512	1024	2048	4096
the way A	2.94	5.90	11.87	24.50	51.78
the way B	3.11	6.31	12.73	26.64	57.19

5 Concluding Remarks

We present an algorithm for constructing the Voronoi diagram of points, line segments, segment chains and polygons and implement it as a program named PLVOR. With any numerical errors, the the algorithm ends with an output which holds some properties which the Voronoi diagram must have. If the numerical error is so small that the conventional algorithms works, the time complexity is $O(n^2)$ for the set of n open segments and isolated points, which is the same as that of the conventional incremental algorithm (not optimal). Under any (bad) accuracy, the execution is terminated in $O(n^3)$ time. For the experimental input constructed in 2 ways, the practical time complexity is nearer to the linear order than to the square order. As a subject for the future, we can point the extension of the algorithm for inputs in which three or more segments are adjacent at an endpoint. In such input, the combinatorical data itself can be inconsistent even if they are locally consistent. This means that the extended algorithm must prove that the input is combinatorically consistent or will require a consistent input. Moreover, by the format of the current input, the algorithm cannot determine in which polygon or chain each open segment is. This means that the input and the output will be more complicated for the extended algorithm.

References

[1] Aurenhammer, F.: Voronoi Diagrams – A Survey of a Fundamental Geometric Data Structure, *ACM Comput. Surv.*, vol. 23, pp. 345-405(1991).

[2] Bentley, J. L. and Ottmann, T., Algorithms for Reporting and Counting Geometric Intersections, *IEEE Trans. Comput.*, C-28, pp. 643–647.(1979)

[3] Imai, T. and Sugihara, K.: A Failure Free Algorithm for Constructing Voronoi Diagrams of Line Segments, *IPSJ Trans.*, vol. 35, No. 10, pp. 1966–1977(1989). (in Japanese).

[4] Imai, T.: A Combinatorial-Structure Oriented Algorithm for Voronoi Diagrams of Polygons, *IPSJ SIG Notes*, 95-AL-45-3 (1995), (in Japanese).

[5] Okabe, A., Boots, B. and Sugihara, K.: *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, John Wiley & Sons(1992).

[6] Sugihara, K. and Iri, M.: Construction of the Voronoi Diagram for "One Million" Generators in Single-precision Arithmetic. *Proc. IEEE*, vol. 80, pp. 1471-1484(1992).