On Computing Voronoi Diagrams of Convex Polyhedra by Means of Wavefront Propagation

Martin Held*
Dept. of Applied Mathematics and Statistics
State University of New York at Stony Brook
Stony Brook, NY 11794-3600

Abstract

We present a wavefront-propagation algorithm for computing Voronoi diagrams (under the Euclidean metric) of convex polyhedra in 3D. The algorithm needs only $O(n_V \log n_V)$ time in the worst case, where n_V denotes the size of the output Voronoi diagram. The algorithm is output-sensitive, i.e., exactly those Voronoi faces and Voronoi edges are generated which are part of $\mathcal{VD}(\mathcal{B})$. In particular, the number of computationally expensive floating-point operations is linearly related to the actual number of Voronoi faces and edges. The algorithm traverses the Voronoi diagram under construction from outwards to inwards, similar to the 2D approaches of Preparata and Persson. For this purpose it maintains Voronoi nodes which are not yet processed in a priority queue of size $O(n_V)$, and it is the cost of this bookkeeping which contributes the log-term in the complexity bound.

1 Introduction

1.1 Motivation

Consider a convex polyhedron $\mathcal{P}$ within the three-dimensional Euclidean space $\mathbb{E}$, and let $\mathcal{B}$ be its polygonal boundary. The Voronoi diagram $\mathcal{VD}(\mathcal{B})$ partitions $\mathcal{P}$ into mutually disjoint regions, the so-called Voronoi regions, where every region is associated with exactly one boundary facet. Clearly, if there were a fast algorithm for computing Voronoi diagrams of polyhedra in $3D$, at least some of the successful 2D applications of Voronoi diagrams could also be extended to 3D. However, in sharp contrast to 2D where several worst-case or average-case optimal algorithms are known, up to now little is known for the 3D setting. Most of the work on Voronoi diagrams in higher dimensions focussed on sets of discrete points. For extensive references we refer to [PS90, OBS92].

Milenkovic [Mil93] proposed a robust algorithm for computing Voronoi diagrams of general polyhedra. His algorithm has a running time of $O(n_P n_V \log^2 b)$, where n_P is the size of the input polyhedron, i.e., the number of the polyhedron's facets, edges, and vertices, and where n_V is the size of the output Voronoi diagram, and b is the number of desired bits of precision. Hoffmann published an algorithm for computing the skeleton of a CSG solid, cf. [Hof90]. Nackman and Srinivasan [NS91] prove that the bisector of a linearly separable set is a manifold (in any dimension).

A detailed description of an approach to computing discretized versions of the Voronoi diagram of a solid shape is presented by Okabe et al. in [OBS92]. After sampling the shape's surface the Voronoi diagram of the set of sample points is computed. By eliminating certain Voronoi surfaces an approximation of the solid's Voronoi diagram is obtained. See also the work by Armstrong [Arm91], Chiang [Chi92], and Goldack et al. [GYKD91].

We note that the size of a Voronoi diagram of a polyhedron may be at least an order of magnitude larger than the size of its defining polyhedron. In particular, even a convex polyhedron of size $O(n_P)$ may define a Voronoi diagram which has up to $O(n_P^2)$ many Voronoi faces, cf. Fig. 1. Thus, achieving output-sensitivity is an important issue.

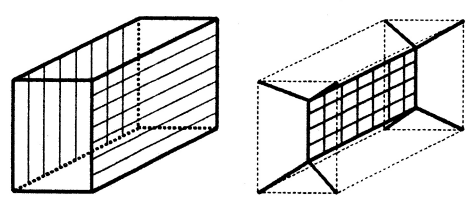

Fig. 1: A Convex Polyhedron with $O(n^2)$ Voronoi Faces

*Supported by grants from Boeing Computer Services, and by NSF Grant DMS-9312098. E-Mail: held@ams.sunysb.edu; Phone: (516) 632–8362; Fax: (516) 632–8490. On sabbatical leave from Universität Salzburg, A-5020 Salzburg, Austria.

1.2 Survey of the Algorithm Proposed

In the sequel we will outline a fast algorithm for computing the Voronoi diagram $\mathcal{VD}(\mathcal{B})$ of $\mathcal{B}$ for a convex polyhedron bounded by $\mathcal{B}$. Our algorithm needs only $O(n_V \log n_V)$ time in the worst case. The algorithm is output-sensitive, i.e., exactly those Voronoi faces and Voronoi edges are generated which are part of $\mathcal{VD}(\mathcal{B})$. In particular, the number of computationally expensive floating-point operations – such as intersecting bisectors – is linearly related to the actual number of Voronoi faces and edges.

The algorithm can be expected to be reasonably fast even when n_P gets large. A 2D version of the algorithm proposed in this paper – for general polygons – was implemented by the author and appealing results were obtained, cf. Held [Hel93]. In fact, the running time of the 2D algorithm usually also behaves well for general polygons, although the algorithm is no longer output-sensitive for general polygons.

The algorithm traverses the Voronoi diagram under construction from outwards to inwards. Basically, it is a wavefront-propagation approach, akin to a 3D prarie fire, where the event points are given by the nodes of the Voronoi diagram, and where the wavefront status captures the topological and metric data of the Voronoi diagram intersected by the wavefront. The wavefront starts at the boundary $\mathcal{B}$ and uniformly propagates inwards. For this purpose the algorithm maintains Voronoi nodes which are not yet processed in a priority queue of size $O(n_V)$, and it is the cost of this bookkeeping which contributes the log-term in the complexity bound.

Wavefront-propagation approaches to the computation of Voronoi diagrams of 2D shapes have first been proposed by Preparata [Pre77] and Persson [Per78]. Preparata's algorithm computes the Voronoi diagram of a convex n-gon in $O(n \log n)$ time, and the algorithm presented here is an extension of Preparata and Persson's ideas to 3D. Recently, Gürsoy and Patrikalakis [GP92] employed an algorithm based on Preparata's approach for finite-element meshing in 2D.

This paper is structured as follows: The next section introduces the notation used in this paper and establishes some basic facts which will be used in the analysis of the algorithm. Section 3 contains the algorithm. A (sketch of a) formal analysis of the algorithm is carried out in Section 4.

2 Preparing for the Algorithm

Throughout the paper we assume that the polyhedron's surface $\mathcal{B}$ is given by a boundary representation, as widely used in the fields of solid modeling. Furthermore, we assume that no neighboring facets are coplanar. The 2D entities of $\mathcal{B}$ are referred to as '(boundary) facets' whereas the 2D entities of $\mathcal{VD}(\mathcal{B})$ are called '(Voronoi) faces'.

2.1 Basic Definitions

For points $p, q \in \mathbb{E}$, we denote their Euclidean distance by $d(p,q)$. As usual, for two sets $P, Q \subset \mathbb{E}$, we denote by $d(P,Q)$ the infimum over all distances between pairs of points belonging to P and Q, i.e., $d(P,Q) := \inf\{d(p,q) : p \in P, q \in Q\}$. The following definitions are fairly standard.

Definition 2.1 (Voronoi Region) For a boundary facet $f \in \mathcal{B}$, its *Voronoi region* $\mathcal{VR}(f,\mathcal{B})$ with respect to $\mathcal{B}$ (within $\mathcal{P}$) is defined as

$$\mathcal{VR}(f,\mathcal{B}) := \{p \in \mathcal{P} : d(p,f) \leq d(p,\mathcal{B})\}.$$

Definition 2.2 (Voronoi Polyhedron) For a boundary facet $f \in \mathcal{B}$, its *Voronoi polyhedron* $\mathcal{VP}(f,\mathcal{B})$ with respect to $\mathcal{B}$ (within $\mathcal{P}$) is defined as

$$\mathcal{VP}(f,\mathcal{B}) := \cup_{f' \neq f \in \mathcal{B}} \ \mathcal{VR}(f,\mathcal{B}) \cap \mathcal{VR}(f',\mathcal{B}).$$

Definition 2.3 (Voronoi Diagram) The *Voronoi diagram* $\mathcal{VD}(\mathcal{B})$ (within $\mathcal{P}$) is defined as

$$\mathcal{VD}(\mathcal{B}) := \cup_{f \in \mathcal{B}} \mathcal{VP}(f,\mathcal{B}).$$

Conventionally, we call the faces of $\mathcal{VD}(\mathcal{B})$ *Voronoi faces.* The collection of all edges of the Voronoi regions corresponds to the so-called (internal) *Voronoi skeleton*, $\mathcal{SK}(\mathcal{B})$. In analogy to the 2D setting we call these edges *bisectors.* Points where bisectors meet are called *(Voronoi) nodes.*

Definition 2.4 (Boundary Clearance) For a point $p \in \mathcal{P}$, we call $d(p,\mathcal{B})$ its (boundary) *clearance.*

2.2 Basic Facts

We note that for every boundary facet $f \in \mathcal{B}$, the Voronoi region $\mathcal{VR}(f,\mathcal{B})$ is a convex polyhedron bounded by f and by $\mathcal{VP}(f,\mathcal{B})$. In the sequel, we will make extensive use of the offset set of $\mathcal{P}$.

Definition 2.5 (Offset Set) For $t \geq 0$ we denote by $\mathcal{P}_t$ the (interior) *offset set* of $\mathcal{P}$ with minimum clearance t with respect to $\mathcal{B}$ (within $\mathcal{P}$), i.e.,

$$\mathcal{P}_t := \{p \in \mathcal{P} : d(p,\mathcal{B}) \geq t\}.$$

Let $t_{max} := \max\{t \geq 0 : \mathcal{P}_t \neq \emptyset\}$. Then, for $0 \leq t \leq t_{max}$ we denote the boundary of $\mathcal{P}_t$ by $\mathcal{B}_t$.

Lemma 2.1 (Offsetting $\mathcal{P}$) For every $0 \leq t \leq t_{max}$, $\mathcal{VD}(\mathcal{B}) \cap \mathcal{P}_t = \mathcal{VD}(\mathcal{B}_t)$.

Corollary 2.1 For every $0 \leq t \leq t_{max}$, the vertices of $\mathcal{B}_t$ are given by those points on $\mathcal{SK}(\mathcal{B})$ which have boundary clearance t.

Lemma 2.2 (Restriction Lemma) Let $t \geq 0$ be less than the minimum clearance of any Voronoi node of $\mathcal{VD}(\mathcal{B})$. Then the Voronoi diagram $\mathcal{VD}(\mathcal{B})$ restricted to $\mathcal{P} \setminus \mathcal{P}_t$ is given by the restriction of all Voronoi faces and bisectors originating at the boundary $\mathcal{B}$ to $\mathcal{P} \setminus \mathcal{P}_t$.

Definition 2.6 (Incoming/Outgoing) Let ν be a Voronoi node and b one of the bisectors incident on ν. Then b is classified as *incoming* if the clearance of ν is larger than the clearance of the second endpoint of b, and *outgoing* otherwise.

3 The Algorithm

3.1 Initialization

With every Voronoi face under construction we associate its boundary – i.e., the two chains of bisectors – generated so far. We call these two (chains of) bisectors associated with a face its 'loose ends'. In the beginning, the boundary Voronoi faces only have the boundary bisectors originating at the vertices of $\mathcal{B}$ associated with them, with two boundary bisectors being associated with one Voronoi face. Eventually, the loose ends of a face will intersect, thus finishing the construction of this face.

We start with describing the initialization of the set of candidate nodes. The following algorithm determines a suitable initial set CN of candidate nodes.

1. $CN := \emptyset$.
2. For every boundary vertex ν compute the boundary bisectors originating at ν, and associate them with their defining Voronoi faces.
3. For every boundary vertex ν and bisector b originating at ν:
   (a) Intersect b with the other loose ends of the Voronoi faces it is associated with.
   (b) If b has intersections with these loose ends then select and store[1] the intersection ν' with minimum clearance, link the intersecting bisectors, and associate ν' with them.
4. For every boundary bisector b, if b has an intersection ν associated with it then terminate it at ν.
5. Arrange CN as a priority queue such that the front end is an intersection with smallest boundary clearance.

Of course, all except the last step of this algorithm can be carried out by traversing $\mathcal{B}$ in a manner guaranteeing that every boundary facet is visited only a constant number of times.

3.2 Main Loop

Let us now turn to the main part of our algorithm. The following algorithm computes the Voronoi diagram of $\mathcal{B}$ in a step-by-step manner, thereby proceeding from outwards to inwards:

1. Fetch and delete the front end ν of CN until ν can be accepted or $d(\nu, \mathcal{B}) > t_{max}$. Let $t = d(\nu, \mathcal{B})$. If[2] $t > t_{max}$ then goto Step 5.
2. Update B_t, compute the boundary bisectors of $\mathcal{B}_t$ at ν, and associate them with their Voronoi faces.
3. For every boundary bisector b of $\mathcal{B}_t$ at ν:
   (a) Intersect b with the other loose ends of the Voronoi faces it is associated with.
   (b) If b has intersections with these loose ends then select and store[3] the intersection ν' with minimum clearance, link the intersecting bisectors, and associate ν' with them.
4. Goto Step 1.
5. Report 'finished' and stop.

We note that an intersection ν can be accepted in Step 1 as a Voronoi node if it still is contained in the bisectors it is associated with, i.e., if it still is part of $\mathcal{VD}(\mathcal{B})$ as constructed so far. Intersecting a bisector with a loose end merely means to compute the intersection between this bisector and the last bisector of the loose end; as it is proved in Section 4, no intersection which has been accepted will be discarded lateron. Thus, there is no need for any sophisticated scanning procedure such as the Lee-Drysdale scanning scheme employed in the general 2D setting.

[1] In case that two bisectors are intersected by b at ν' which are already linked (and thus already have an intersection associated and stored in CN), do not store ν' in CN.

[2] This can be checked by updating $\mathcal{B}_t$ as the wavefront moves inwards.

[3] In case that two bisectors are intersected by b at ν' which are already linked (and thus already have an intersection associated and stored in CN), do not store ν' in CN.

Computing the boundary bisectors of $\mathcal{B}_t$ at a Voronoi node ν in Step 2 of the algorithm is no demanding task as long as the number of incoming bisectors of ν is small. For instance, if the vertex ν of $\mathcal{P}_t$ has three incident boundary edges then the one and only boundary bisector of $\mathcal{B}_t$ at ν is determined as follows, cf. Fig. 2, where the dot denotes ν and the numbers refer to the facets of $\mathcal{B}$:

1. Let $b(f_1, f_2, f_3)$, $b(f_1, f_2, f_4)$, and $b(f_1, f_3, f_4)$, be the bisectors intersecting at ν.
2. Compute the bisector $b(f_2, f_3, f_4)$.

In this case, three bisectors intersect at ν such that a Voronoi region is closed.

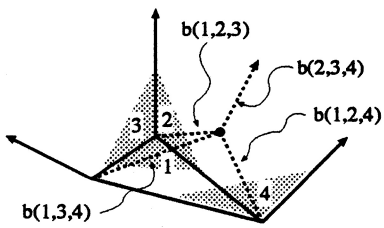

Fig. 2: Three incoming bisectors.

If ν has two incoming bisectors, the following simple subalgorithm computes the new boundary bisectors, cf. Fig. 3:

1. Let $b(f_1, f_2, f_3)$ and $b(f_2, f_3, f_4)$ be the bisectors intersecting at ν.
2. Compute the bisectors $b(f_1, f_2, f_4)$ and $b(f_1, f_3, f_4)$.

In this case, ν regarded as a vertex of $\mathcal{P}_t$ has four incident boundary edges.

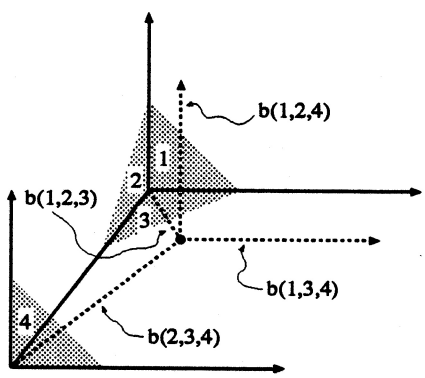

Fig. 3: Two incoming bisectors.

The 'degenerate' case of $k \geq 4$ boundary edges of $\mathcal{P}_t$ being incident upon ν needs a more careful treatment. Observe that the boundary facets corresponding to these edges form a convex cone centered at ν. Fig. 4 shows such a cone and its Voronoi diagram. We apply a divide-and-conquer scheme to this cone which is akin to Lee's divide-and-conquer algorithm [Lee82] for constructing the Voronoi diagram of a 2D polygon. Conceptually, the facets of the cone correspond to the polygon's edges and the Voronoi faces correspond to the polygon's bisectors. Thus, in time $O(k \log k)$ we can construct all boundary bisectors at a degree-k vertex ν.

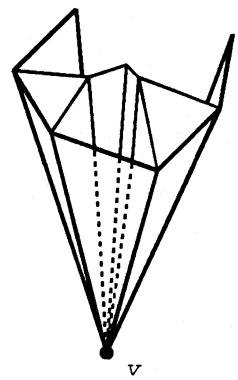

Fig. 4: Cone of boundary facets and its Voronoi diagram.

4 Formal Analysis

We start with a sketch of the correctness proof of the algorithm.

Lemma 4.1 (Initialization Lemma) Let $V_1(\mathcal{B})$ be the set of nodes of $\mathcal{VD}(\mathcal{B})$ which have smallest boundary clearance among all Voronoi nodes. Let $V_2(\mathcal{B})$ be the set of those nodes stored in CN which have smallest boundary clearance among all nodes initially stored in CN. Then $V_1(\mathcal{B}) = V_2(\mathcal{B})$ holds.

Proof:
We first prove that all Voronoi nodes of smallest clearance are contained in CN, i.e., that $V_1(\mathcal{B}) \subset V_2(\mathcal{B})$. Then we show that $V_2(\mathcal{B}) \subset V_1(\mathcal{B})$.
Let $\nu \in V_1(\mathcal{B})$, and suppose that it is not contained in CN. Let $t := d(\nu, \mathcal{B})$. Assume that there is at least one boundary bisector b incident upon ν, and let $b_1, \ldots, b_k$ be the bisectors incident upon ν which are contained in the Voronoi faces that share b, cf. Fig. 5 for $k = 3$; e_1, e_2, e_3 denote boundary edges. Since ν is not contained in CN, none of $b_1, \ldots, b_k$ can be a boundary bisector. Thus, $b_1, \ldots, b_k$ are line segments extending between ν and some Voronoi nodes $\nu_1, \ldots, \nu_k$, where none of $\nu_1, \ldots, \nu_k$, lies on $\mathcal{B}$. Since the boundary clearance of ν is minimal among all Voronoi nodes, none of $\nu_1, \ldots, \nu_k$ can have a smaller

clearance than ν, and since no two neighboring facets are coplanar, at least one of these nodes has a strictly larger clearance than ν. Thus, $b_1, \ldots, b_k$ form a convex cone centered at ν. Now consider the intersection of this cone with a plane which cuts off ν. In Fig. 5, the convex polygon defined by this intersection is given by the shaded triangle. We note that the clearance distances of the polygon's vertices are greater than (or equal to) t, with at least one clearance being strictly larger. Due to the convexity of $\mathcal{P}$, the boundary clearance of any point belonging to this polygon's interior also is strictly larger than t. By virtue of ν being a Voronoi node, there has to exist a point p on the boundary of $\mathcal{B}$ which is contained in this cone such that $d(\nu, p) = t$. Thus, the line segment linking ν and p has to intersect the interior of the convex polygon in a point p'. This yields $d(p, p') > t$ and $d(\nu, p) = d(\nu, p') + d(p', p) = t$, i.e., a contradiction. The case that no boundary bisector is incident upon ν can be proved similarly. We conclude that ν is contained in CN. If there were a node contained in CN with boundary clearance smaller than ν than this node would be a Voronoi node, too, and $\nu \notin V_1(\mathcal{B})$. Thus, $\nu \in V_2(\mathcal{B})$.
Now assume that $\nu \in V_2(\mathcal{B})$ is no Voronoi node. Let $t := d(\nu, \mathcal{B})$, and let b_1 and b_2 be the boundary bisectors intersecting at ν. Since ν is no Voronoi node, ν lies completely in the interior of some Voronoi region, and b_1 and b_2 are intersected by some other bisectors. These intersections have a strictly smaller boundary clearance than ν because the boundary clearance increases strictly monotonously as one moves along b_1, b_2 away from the boundary (towards ν). Thus, there do exist Voronoi nodes with boundary clearance less than t. However, according to the first part of the proof, all Voronoi nodes with minimal clearance are contained in CN. Thus, CN contains nodes whose boundary clearence is less than t, which violates the definition of $V_2(\mathcal{B})$. □

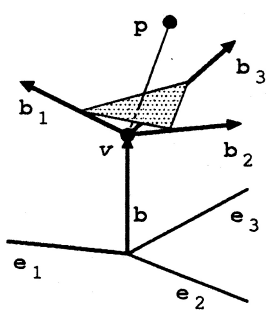

Fig. 5: CN contains all Voronoi nodes of minimum clearance.

The following lemma already captures the main correctness aspects of the algorithm proposed. To help intuition, imagine shrinking $\mathcal{P}$ successively. Let $t_1 < \ldots < t_m$ denote the (sorted) boundary clearances of the Voronoi nodes, with identical clearances listed only once. We say that 'pass i has been finished' if the algorithm accepts a Voronoi node with clearance t_{i+1} for the first time. By definition, the zeroth pass is the execution of the initialization phase, and $t_0 := 0$.

Lemma 4.2 (Loop Invariant) The following loop invariant holds after the i-th pass (for $0 \leq i \leq m$), where $V_1(\mathcal{B}_{t_i}), V_2(\mathcal{B}_{t_i})$ are defined as in the previous lemma:

- $V_1(\mathcal{B}_{t_i}) = V_2(\mathcal{B}_{t_i})$;
- $\mathcal{VD}(\mathcal{B}) \cap (\mathcal{P} \setminus \mathcal{P}_{t_{i+1}})$ has been computed.

Proof:
For $i = 0$, the loop invariant is fulfilled due to the Initialization Lemma 4.1 and the Restriction Lemma 2.2. Assume inductively that it is fulfilled after the $i-1$-th path, for some $1 \leq i \leq m$. W.l.o.g., $|V_1(\mathcal{B}_{t_i})| = 1$, and let ν be the node accepted at the beginning of the $i+1$-th pass. Thus, $t_{i+1} = d(\nu, \mathcal{B})$. We observe that all boundary bisectors of $\mathcal{B}_{t_i}$ which are not incident upon ν also are boundary bisectors of $\mathcal{B}_{t_{i+1}}$. Thus, intersections among them are already stored in CN. The only 'new' boundary bisectors of $\mathcal{B}_{t_{i+1}}$ are the bisectors incident upon ν which are constructed in Step 2 of the algorithm's main loop. Since the intersections of these new boundary bisectors with the already existing boundary bisectors of $\mathcal{B}_{t_{i+1}}$ are computed by the algorithm, we conclude that all intersections among boundary bisectors of $\mathcal{B}_{t_{i+1}}$ are stored in CN. Hence, CN contains all nodes which would have been stored in CN by applying the initialization phase directly to $\mathcal{B}_{t_{i+1}}$. By applying the Initialization Lemma 4.1 and the Restriction Lemma 2.2 to $\mathcal{B}_{t_{i+1}}$ we obtain that the loop invariant is fulfilled after the $i+1$-th pass. □

Corollary 4.1 The algorithm correctly computes the Voronoi diagram $\mathcal{VD}(\mathcal{B})$ of a convex polyhedron with boundary $\mathcal{B}$. In particular, the vertices of $\mathcal{VD}(\mathcal{B})$ are accepted exactly in the order of their boundary clearance.

Lemma 4.3 (Complexity Lemma) The algorithm spends $O(n_V \log n_V)$ time on computing $\mathcal{VD}(\mathcal{B})$, where n_V denotes the output size of $\mathcal{VD}(\mathcal{B})$, i.e., the number of faces, edges, and vertices of $\mathcal{VD}(\mathcal{B})$.

Proof:
First note that the initialization phase can be carried out in $O(n_P)$ time[4], since computing and intersecting the boundary facets and bisectors takes constant time per item. The priority queue initially is of size $O(n_P)$ and can be generated in $O(n_P)$ time.
The bound on the overall complexity is easily derived by observing that the algorithm does not discard any Voronoi node after accepting it. Thus, the total number of candidate nodes which have to be inserted in resp. deleted from CN is $O(n_V)$. Similarly, the maximum number of intersections computed is $O(n_V)$. Inserting a node in resp. deleting a node from CN takes $\log n_V$ time at most. Computing the new boundary bisectors takes $O(k \log k)$ for a newly accepted degree-k Voronoi node ν. Since no accepted Voronoi nodes are ever discarded, this cost occurs only once per Voronoi node. We conclude that the algorithm's total worst-case complexity is $O(n_V \log n_V)$. □

We summarize our results obtained in the following theorem.

Theorem 4.1 The algorithm computes the Voronoi diagram $\mathcal{VD}(\mathcal{B})$ of a convex polyhedron with boundary $\mathcal{B}$ in time $O(n_V \log n_V)$, where n_V denotes the output size of $\mathcal{VD}(\mathcal{B})$. In particular, exactly those Voronoi faces and Voronoi edges are generated which are part of $\mathcal{VD}(\mathcal{B})$.

5 Summary

This paper presents an output-sensitive wavefront-propagation algorithm for computing the Voronoi diagram of a convex polyhedron. The algorithm's worst-case complexity is $O(n_V \log n_V)$, where n_V denotes the size of the output Voronoi diagram. We expect that the algorithm can be generalized to higher dimensions and can be adapted to handling more general objects than convex polyhedra. An implementation of the algorithm is planned for the near future.

References

[Arm91] C. Armstrong. Abstraction and meshing of 3d stress analysis models. Progress Report, SERC Directorate, UK, 1991.

[Chi92] C.-S. Chiang. *The Euclidean Distance Transform.* PhD thesis, CS Dept., Purdue Univ., West Lafayette, IN 47907, USA, Aug 1992.

[GP92] H.N. Gürsoy and N.M. Patrikalakis. An Automatic Coarse and Fine Surface Mesh Generation Scheme Based on Medial Axis Transform: Part I Algorithm. *Engineering with Computers*, 8:121–137, 1992.

[GYKD91] J.A. Goldack, X. Yu, A. Knight, and L. Dong. Constructing Discrete Medial Axis of 3D Objects. *Internat. J. Comput. Geom. Appl.*, 1(3):327–339, Sep 1991.

[Hel93] M. Held. A Fast Incremental Algorithm for Computing the Voronoi Diagram of a Planar Shape. In *Communicating with Virtual Worlds (CGI'93)*, pages 318–329, Lausanne, Switzerland, June 1993. Springer-Verlag.

[Hof90] C.M. Hoffmann. How to Construct the Skeleton of CSG Objects. Technical Report CSD-TR-1014, CS Dept., Purdue Univ., West Lafayette, IN 47907, USA, Sep 1990.

[Lee82] D.T. Lee. Medial Axis Transformation of a Planar Shape. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-4(4):363–369, 1982.

[Mil93] V. Milenkovic. Robust Construction of the Voronoi Diagram of a Polyhedron. In *Proc. 5th Canad. Conf. Comput. Geom.*, pages 473–478, Waterloo, Canada, 1993.

[NS91] L. R. Nackman and V. Srinivasan. Bisectors of Linearly Separable Sets. *Discrete Comput. Geom.*, 6:263–275, 1991.

[OBS92] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tesselations: Concepts and Applications of Voronoi Diagrams.* John Wiley & Sons, 1992. ISBN 0-471-93439-5.

[Per78] H. Persson. NC Machining of Arbitrarily Shaped Pockets. *Comput. Aided Design*, 10(3):169–174, May 1978.

[Pre77] F.P. Preparata. The Medial Axis of a Simple Polygon. In *Proc. 6th Int. Sympos. Math. Found. Comput. Sci.*, volume 53 of *Lecture Notes in Computer Science*, pages 443–450. Springer-Verlag, 1977.

[PS90] F.P. Preparata and M.I. Shamos. *Computational Geometry - An Introduction.* Springer-Verlag, third edition, Oct 1990. ISBN 3-540-96131-3.

[4] Recall that n_P denotes the input size of $\mathcal{B}$, i.e., the number of its facets, edges, and vertices.