NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES

MSc THESIS

Voronoi diagram of orthogonal polyhedra in two and three dimensions

Christina P. Katsamaki

SUPERVISOR: Ioannis Z. Emiris, Professor N.K.U.A.

ATHENS

JUNE 2019

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Διάγραμμα Voronoi ορθογώνιων πολυέδρων σε δύο και τρεις διαστάσεις

Χριστίνα Π. Κατσαμάκη

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Ιωάννης Ζ. Εμίρης, Καθηγητής Ε.Κ.Π.Α.

ΑΘΗΝΑ

ΙΟΥΝΙΟΣ 2019

MSc THESIS

Voronoi diagram of orthogonal polyhedra in two and three dimensions

Christina P. Katsamaki
Registration Number: M1517

SUPERVISOR: Ioannis Z. Emiris, Professor N.K.U.A.

TWO-MEMBER EXAMINATION COMMITTEE:

Ioannis Z. Emiris, Professor N.K.U.A.

Dimitrios Gunopulos, Professor N.K.U.A.

Examination Date: June 28, 2019

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Διάγραμμα Voronoi ορθογώνιων πολυέδρων σε δύο και τρεις διαστάσεις

Χριστίνα Π. Κατσαμάκη
A.M: M1517

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Ιωάννης Ζ. Εμίρης, Καθηγητής Ε.Κ.Π.Α.

ΔΙΜΕΛΗΣ ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:

Ιωάννης Ζ. Εμίρης, Καθηγητής Ε.Κ.Π.Α.

Δημήτριος Γουνόπουλος, Καθηγητής Ε.Κ.Π.Α.

Ημερομηνία Εξέτασης: 28 Ιουνίου 2019

ABSTRACT

Voronoi diagrams are a fundamental geometric data structure for obtaining proximity relations. We consider axis-aligned orthogonal polyhedra in two and three-dimensional space. These are polyhedra whose faces meet at right angles and their edges are aligned with the axes of a coordinate system. We construct the exact Voronoi diagram inside an axis-aligned orthogonal polyhedron with holes defined by such polyhedra, under the max-norm. This is a particularly useful scenario in certain application domains, including raster graphics and VLSI design.

Our approach avoids creating full-dimensional elements on the Voronoi diagram and yields a skeletal representation of the input object, equivalent to the straight skeleton. We introduce a complete algorithm in 2D and 3D that follows the subdivision paradigm relying on a bounding-volume hierarchy; this is an original approach to the problem. The algorithm reads in a region bounding the input polyhedron and performs a recursive subdivision into cells (using quadtrees and octrees for 2D and 3D resp.). Then, a reconstruction technique is applied to produce an isomorphic representation of the Voronoi diagram. An hierarchical data structure of bounding volumes is used to accelerate the 2D algorithm for certain inputs and is necessary for the efficiency of the 3D algorithm.

The complexity is adaptive and comparable to that of previous methods. Under a mild assumption it is $O(n/\Delta+1/\Delta^2)$ in 2D and $O(n\alpha^2/\Delta^2+1/\Delta^3)$ in 3D, where n is the number of sites, namely edges or facets respectively, Δ is the maximum cell size for the subdivision to stop (and is < 1 under the appropriate scaling), and α bounds vertex cardinality per facet. We also provide a numerically stable, open-source implementation in Julia, illustrating the practical nature of our algorithm.

Part of the current thesis is given in the paper "Voronoi diagram of orthogonal polyhedra in two and three dimensions", co-authored with Prof. Ioannis Z. Emiris, that is about to appear in Proceedings of SEA^2 2019 (Special Event on Analysis of Experimental Algorithms).

SUBJECT AREA: Computational Geometry

KEYWORDS: max norm, axis-aligned, rectilinear, straight skeleton, subdivision method, numeric implementation

ΠΕΡΙΛΗΨΗ

Τα διαγράμματα Voronoi αποτελούν μία θεμελιώδη γεωμετρική δομή δεδομένων και εκφράζουν αποστάσεις σημείων στο χώρο από ένα σύνολο αντικειμένων. Θεωρούμε ορθογώνια πολύεδρα ευθυγραμμισμένα με τους άξονες. Πρόκειται για πολύεδρα των οποίων οι έδρες σχηματίζουν ορθές γωνίες, και οι ακμές είναι παράλληλες προς τους άξονες ενός καρτεσιανού συστήματος συντεταγμένων. Κατασκευάζουμε το διάγραμμα Voronoi στο εσωτερικό ενός ορθογώνιου πολυέδρου με τρύπες που ορίζονται από αντίστοιχα πολύεδρα, χρησιμοποιώντας την max-νόρμα. Πρόκειται για έναν συνδυασμό που βρίσκει πολλές εφαρμογές σε τομείς όπως τα raster graphics και ο σχεδιασμός κυκλωμάτων VLSI.

Παρουσιάζουμε έναν αλγόριθμο για την κατασκευή αυτών των διαγραμμάτων Voronoi σε δύο και τρεις διαστάσεις. Ακολουθούμε τη μέθοδο υποδιαίρεσης και βασιζόμαστε σε μία δομή δεδομένων από bounding-volumes: πρόκειται για μία μη τετριμμένη προσέγγιση του προβλήματος. Επιπλέον αναλύουμε την πολυπλοκότητα του αλγορίθμου, η οποία είναι γραμμική στο πλήθος των εδρών κάτω από μία υπόθεση ομοιόμορφα κατανεμημένης εισόδου.

Μέρος της παρούσας εργασίας πρόκειται να δημοσιευθεί στα πρακτικά του συνεδρίου SEA^2 2019 (Special Event on Analysis of Experimental Algorithms).

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Υπολογιστική Γεωμετρία

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: max νόρμα, μέθοδος υποδιαίρεσης, αριθμητική υλοποίηση

ACKNOWLEDGEMENTS

First, I would like to express my deep gratitude to my advisor, Prof. Ioannis Emiris, for his support, guidance and patience during the process of writing this thesis. I want to extend my thanks to all the people of ErGa Lab, for helping me in various ways and of course to my family and friends for encouraging me.

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Orthogonal shapes are ubiquitous in numerous applications including raster graphics and VLSI design. In this thesis, we address Voronoi diagrams of 2- and 3-dimensional orthogonal shapes. We focus on the L_∞ metric which is used in the relevant applications and has been studied much less than L_2.

A *Voronoi diagram* partitions space into regions based on distances to a given set $\mathcal{S}$ of geometric objects in $\mathbb{R}^d$. Every $s \in \mathcal{S}$ is a *Voronoi site* (or simply a *site*) and its *Voronoi region* under metric μ is

$$V_\mu(s) = \{x \in \mathbb{R}^d \mid \mu(s,x) < \mu(x,s'),\ s' \in \mathcal{S} \setminus s\},$$

comprising all points closer to s than to any other site. The *Voronoi diagram* is the set

$$\mathcal{V}_\mu(\mathcal{S}) = \mathbb{R}^d \setminus \bigcup_{s \in \mathcal{S}} V_\mu(s),$$

consisting of all points that attain their minimum distance to $\mathcal{S}$ by at least two Voronoi sites. For general input, the Voronoi diagram is a collection of faces of dimension $0, 1, \ldots, d-1$. A face of dimension k comprises points equidistant to at least $d+1-k$ sites. Faces of dimension 0 and 1 are called *Voronoi vertices* and *Voronoi edges* respectively. The union of Voronoi edges and vertices is the *1-skeleton*.

Equivalently, a Voronoi diagram is defined as the *minimization diagram* of the distance functions to the sites. The minimization diagram is a partitioning of space into regions, each region consisting of points where some function has lower value than any other function.

Here, we study Voronoi diagrams in the interior of an axis-aligned *orthogonal polyhedron*; its faces meet at right angles, and the edges are aligned with the axes of a coordinate system. It may have arbitrarily high genus with holes defined by axis-aligned orthogonal polyhedra, not necessarily convex. All facets are simply connected (without holes) for

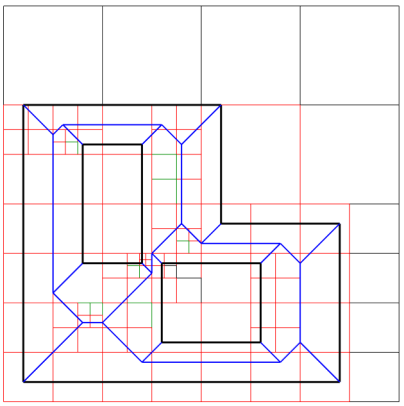


Figure 1: Voronoi diagram (in blue) of a rectilinear polygon with 2 holes.

simplicity. The two dimensional analog of such a polyhedron is also known as *rectilinear* polygon. The sites are all facets on the boundary of all polyhedra.

The L_∞ or Chebyshev distance is a metric defined on a vector space where the distance between two vectors is the greatest of their differences along any coordinate dimension. For two points $x, y \in \mathbb{R}^d$, with standard coordinates x_i and y_i respectively, their L_∞ distance is $\mu_\infty(x, y) = \max_i\{|x_i - y_i|\}$. The L_∞ distance of x to a set $S \subset \mathbb{R}^d$ is $\mu_\infty(x, S) = \inf\{\mu_\infty(x, y) \mid y \in S\}$.

The **L_∞ Voronoi diagram** uses the L_∞ metric to measure distances. In Figure 1, the Voronoi diagram[1] of a rectilinear polygon with 2 holes is shown in blue. Our algorithm follows the *Subdivision Paradigm* and handles 2D and 3D sites. It reads in a region bounding all input sites and performs a recursive subdivision into cells (using quadtrees or octrees). Then, a reconstruction technique is applied to produce an isomorphic representation of the Voronoi diagram.

1.1 Previous Work

If V is the number of polyhedral vertices, the combinatorial complexity of our Voronoi diagrams equals $O(V)$ in 2D [18] and $O(V^2)$ in 3D [4]. In 3D, it is estimated experimentally to be, in general, $O(V)$ [16].

Related work in 2D concerns L_∞ Voronoi diagrams of segments. In [18], they introduce an $O(n \log n)$ sweep-line algorithm, where n is the number of segments; they offer a robust implementation for segments with $O(1)$ number of orientations. Another algorithm implemented in library CGAL [6] is incremental. The L_∞ Voronoi diagram of orthogonal polyhedra (with holes) is addressed in [16] in view of generalizing the sweep-line paradigm to 3D: in 2D it runs in $O(n \log n)$ as in [18], and in 3D the sweep-plane version runs in $O(kV)$, where $k = O(V^2)$ is the number of events.

When the diagram is restricted in the interior of a polygon or polyhedron, it serves as a skeletal representation. A skeleton reduces the dimension of the input capturing its boundary's geometric and topological properties. In particular, *straight skeletons* are very related to the L_∞ Voronoi diagram of rectilinear polygons [2]. An algorithm for the straight skeleton of a simple polygon (not necessarily rectilinear) has complexity $O(V^{\frac{17}{11}+\varepsilon})$ for fixed $\varepsilon > 0$ [9]. For x-monotone rectilinear polygons, a linear time algorithm was recently introduced [7]. In 3D, an analogous equivalence of the straight skeleton of orthogonal polyhedra and the L_∞ Voronoi diagram exists [4] and a complete analysis of 3D straight skeletons is provided in [3]. Specifically for 3D orthogonal polyhedra, in [4] they offer two algorithms that construct the skeleton in $O(\min\{V^2 \log V, k \log^{O(1)} V\})$, where $k = O(V^2)$ is the number of skeleton features. Both algorithms are rather theoretical and follow a wavefront propagation process. Recently, the straight skeleton of a 3D polyhedral terrain was addressed [13].

[1] computed by our software and visualized with `Axl` viewer.

For polyhedral objects it is common to consider as Voronoi sites the vertices, edges and higher-dimensional faces (when $d \geq 3$) that form their boundary [21, 10, 14]. However, the combination of the underlying metric and the structure of the input polyhedron may cause the appearance of full-dimensional faces in the Voronoi diagram. Under this event, the Voronoi diagram cannot serve as a skeletal representation of the input shape and the design of an effective algorithm for its computation is significantly more difficult. For example, the L_2 Voronoi diagram of a polygon in the plane, under the standard definition, contains two dimensional regions of points equidistant to a reflex vertex and its adjacent edges. A usual approach towards eliminating the appearance of these two-dimensional regions is to consider as Voronoi sites the *open* edges and the polygon vertices and define by convention 1-dimensional bisectors between them. A similar concept is this of defining the 'cones of influence" [12] or "zones" [22] of each site.

Full-dimensional faces on the Voronoi diagram in our setting are very frequent [16]; under L_∞, when two points have same coordinate value, their bisector is full dimensional. Conventions have been adopted, to ensure bisectors between sites are not full-dimensional [18, 16, 6]. We address this issue in the next chapter.

The literature on subdivision algorithms for Voronoi diagrams is vast, e.g. [5, 8, 22] and references therein. Our work is closely related to [22, 5]. These algorithms are quite efficient, since they adapt to the input, and rather simple to implement. None exists for our problem.

1.2 Our contribution

We express the problem by means of the minimization diagram of a set of algebraic functions with restricted domain, that express the L_∞ distance of points to the boundary. The resulting Voronoi diagram, for general input, is $(d-1)-$dimensional. We focus on 2D and 3D orthogonal polyhedra with holes, where the resulting Voronoi diagram is equivalent to the straight skeleton. We introduce an efficient and complete algorithm for both dimensions, following the subdivision paradigm which is, to the best of our knowledge, the first subdivision algorithm for this problem. We compute the exact Voronoi diagram (since L_∞ bisectors are linear). The output data structure can also be used for nearest-site searching.

The overall complexity is output-sensitive, which is a major advantage. Under Hypothesis 1, which captures the expected geometry of the input as opposed to worst-case behaviour, the complexity is $O(n/\Delta + 1/\Delta^2)$ in 2D, where n is the number of sites (facets) and Δ the separation bound (maximum edge length of cells that guarantees termination). This bound is to be juxtaposed to the worst-case bound of $O(n \log n)$ of previous methods. In 3D it is $O(n\,\alpha^2/\Delta^2 + 1/\Delta^3)$ where α bounds the vertex cardinality per facet (usually constant). Under a further assumption (Remark 4.1) this bound becomes $O(V/\Delta^2 + 1/\Delta^3)$ whereas existing worst-case bounds are quasi-quadratic or cubic in V. Δ is measured under appropriate scaling such the bounding box has edge length 1. Scaling does not affect arithmetic complexity, but may be adapted to reduce the denominators' size in rational

calculations. The algorithm's relative simplicity has allowed us to develop a numerically stable software in Julia[2], a user-friendly platform for efficient numeric computation; it consists of about 5000 lines of code and is the first open-source code in 3D. We use the algebraic geometric modeler `Axl` for visualization. All experiments conducted for the purpose of this thesis were run on a 64-bit machine with an Intel(R) Core(TM) i7-8550U CPU @1.80GHz and 8.00 GB of RAM.

The rest of this thesis is organized as follows. The next chapter provides structural properties of these Voronoi diagrams. In Chapter 3 we introduce our 2D algorithm: the 2D and 3D versions share some basic ideas which are discussed in detail in this chapter. In particular, we describe a hierarchical data structure of bounding volumes, used to accelerate the 2D algorithm for certain inputs and is necessary for the efficiency of the 3D algorithm. Then we provide the complexity analysis of the 2D algorithm. In Chapter 4 we extend our algorithm and analysis to 3D. In Chapter 5 we give some experimental results and in Chapter 6 we conclude with some remarks and we discuss on possible future research directions.

[2] `https://gitlab.inria.fr/ckatsama/L_infinity_Voronoi/`

2. BASIC DEFINITIONS AND PROPERTIES

We introduce useful concepts in general dimension. Let $\mathcal{P}$ be an orthogonal polyhedron of full dimension in d dimensions, whose boundary consists of n *simply connected* (without holes) facets; these are edges or flats in 2D and 3D, respectively. Note that $\mathcal{P}$ includes the shape's interior and boundary. Now the set of Voronoi sites $\mathcal{S}$ consists of the *closed facets* that form the boundary of $\mathcal{P}$, including all facets of the interior polyhedra. There are as many such polyhedra as the genus.

Under the L_∞, the Voronoi diagram $\mathcal{V}_\infty(\mathcal{S})$ often contains full-dimensional regions (e.g. Figure 2a). Aiming at a $(d-1)$-dimensional Voronoi diagram, we will define an appropriate set of distance functions and restrict their minimization diagram to $\mathcal{P}$.

Let $V_\infty(s)$ denote the Voronoi region of site s under the L_∞ metric. Lemma 2.1 gives a property of standard L_∞ Voronoi diagram preserved by Definition 2.1.

(a) **(b)**

Figure 2: Voronoi diagrams (in red): (a) standard, under L_∞, (b) under Def. 2.1.

Lemma 2.1. *Let $s \in \mathcal{S}$. For every point $p \in V_\infty(s)$ it holds that $\mu_\infty(p,s) = \mu_\infty(p, \textit{aff}(s))$, where aff$(s)$ is the affine hull of s.*

Proof. Assume without loss of generality that $s \subset \{x \in \mathbb{R}^d \mid x_j = c\}$, $j \in [d]$ and $c \in \mathbb{R}$. If $\mu_\infty(p,s) \neq \mu_\infty(p, \text{aff}(s))$, then $\mu_\infty(p,s) = \inf\{\max_{i\in[d]\setminus j}\{|p_i - q_i|\} \mid \forall q \in s\}$ and there is $q \in \partial s$ such that $\mu_\infty(p,s) = \mu_\infty(p,q)$. To see this, suppose on the contrary that q is in the interior of s. Then, we can find another point $q' \in s$ ε-close to q such that $|p_i - q'_i| = |p_i - q_i| - \varepsilon \Rightarrow \mu_\infty(p,q') = \mu_\infty(p,q) - \varepsilon$, for any $\varepsilon > 0$. This leads to a contradiction. Therefore, there is a site $s' \neq s$ with $q \in s'$. Since $p \in V_\infty(s)$, then $\mu_\infty(p,s) < \mu_\infty(p,s') \leq \mu_\infty(p,q)$; contradiction. □

For $s \in \mathcal{S}$ let $\mathcal{H}(s)$ be the *closed* halfspace of $\mathbb{R}^d$ induced by aff(s) such that for every $p \in s$ there exists a point $q \in \mathcal{H}(s) : q \in int(\mathcal{P})$ and $\mu_\infty(p,q) < \epsilon$, $\forall \epsilon > 0$. We define the **(unoriented) zone** of s as $\mathcal{Z}(s) := \{p \in \mathbb{R}^d \mid \mu_\infty(p,s) = \mu_\infty(p, \text{aff}(s))\}$. The **oriented zone** of s is $\mathcal{Z}^+(s) := \mathcal{H}(s) \cap \mathcal{Z}(s)$ (Figure 3).

We associate to s the distance function

$$D_s(\cdot) : \mathbb{R}^d \to \mathbb{R} : p \mapsto \begin{cases} \mu_\infty(p,s), & \text{if } p \in \mathcal{Z}^+(s), \\ \infty, & \text{otherwise.} \end{cases}$$

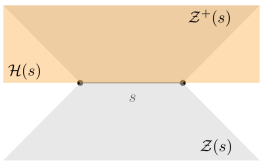


Figure 3: $\mathcal{H}(s)$**,** $\mathcal{Z}(s)$**,** $\mathcal{Z}^+(s)$ **for 2D site** s**.**

The minimization diagram of $\mathcal{D} = \{D_s \mid s \in \mathcal{S}\}$ restricted to $\mathcal{P}$ yields a Voronoi partitioning. The *Voronoi region* of s with respect to $D_s(\cdot)$ is

$$V_{\mathcal{D}}(s) = \{p \in \mathcal{P} \mid D_s(p) < \infty \text{ and } \forall s' \in \mathcal{S} \setminus s \;\; D_s(p) < D_{s'}(p)\}.$$

Definition 2.1. *The Voronoi diagram of* $\mathcal{P}$ *with respect to* $\mathcal{D}$ *is the set* $\mathcal{V}_{\mathcal{D}}(\mathcal{P}) = \mathcal{P} \setminus \bigcup_{s \in \mathcal{S}} V_{\mathcal{D}}(s)$.

This means one gets the Voronoi diagram of Figure 2b. Clearly $\mathcal{P} \subset \bigcup_{s \in \mathcal{S}} \mathcal{Z}^+(s)$. Denoting by $\overline{X}$ the closure of a set X, then $V_\infty(s) \subseteq V_{\mathcal{D}}(s) \subseteq \overline{V_\infty(s)} \subseteq \mathcal{Z}^+(s)$ (Figure 2a). The bisector of $s, s' \in \mathcal{S}$ with respect to $\mathcal{D}$ is $\text{bis}_{\mathcal{D}}(s, s') = \{x \in \mathbb{R}^d \mid D_s(x) = D_{s'}(x) < \infty\}$. Then $\text{bis}_{\mathcal{D}}(s, s') \subset \text{affbis}(s, s')$, where $\text{affbis}(s, s')$ denotes the L_∞ (affine) bisector of $\text{aff}(s), \text{aff}(s')$. In 2D (resp. 3D) if sites have not the same affine hull, bisectors under $\mathcal{D}$ lie on lines (resp. planes) parallel to one coordinate axis (resp. plane) or to the bisector of two perpendicular coordinate axes (resp. planes). Although the latter consists of two lines (resp. planes), $\text{bis}_{\mathcal{D}}(s, s')$ lies only on one, and it can be uniquely determined by the orientation of the zones.

Degeneracy of full-dimensional bisectors, between sites with the same affine hull, can therefore be avoided by infinitesimal perturbation of the corresponding sites. This is equivalent to assigning priorities to the sites; the full dimensional region of the former diagram is 'to the limit' assigned to the site with the highest priority (Figure 4b). Such a perturbation always exists, both for 2D [16, Lem. 13] and 3D [16, Lem. 31].

(a) **(b)**

Figure 4: (a) 2D Voronoi diagram for polygon with colinear edges. (b) 1D Voronoi diagram after infinitesimal perturbation of edges, where $\epsilon \to 0^+$**.**

Set X is *weakly star shaped* with respect to $Y \subseteq X$ if $\forall x \in X$, $\exists y \in Y$ such that the segment (x, y) belongs to X.

Lemma 2.2. *For every* $s \in S$, $\overline{V_{\mathcal{D}}(s)}$ *is weakly star shaped with respect to* s.

Proof. Let $p \in \overline{V_{\mathcal{D}}(s)}$ and $\rho = D_s(p) = \mu_\infty(p, q)$, $q \in s$. The open ball $B_\infty(p, \rho)$ centered at p with radius ρ is empty of sites. For $t \in (0, 1)$, let $w = tp + (1 - t)q$ on the line segment (p, q). Then, since for every $i \in [d]$ it is $|w_i - q_i| = t|p_i - q_i|$, it holds that $w \in \mathcal{Z}^+(s)$ and $D_s(w) = t\rho$. If $w \notin \overline{V_{\mathcal{D}}(s)}$ there is a site s' such that $D_{s'}(w) < t\rho$. But $B_\infty(w, t\rho) \subseteq B_\infty(p, \rho)$ and s' intersects $B_\infty(p, \rho)$, leading to a contradiction. □

Therefore, since every s is simply connected, from Lemma 2.2 $\overline{V_{\mathcal{D}}(s)}$ is *simply connected* and $V_{\mathcal{D}}(s)$ is also simply connected.

Let the *degree* of a Voronoi vertex be the number of sites to which it is equidistant. If the degree is $> d+1$, the vertex is *degenerate*. Lem. 2.3 is nontrivial: in metrics like L_2 degree is arbitrarily large. For $d = 2, 3$ this bound is tight [16].

Lemma 2.3. *(a) The maximum degree of a Voronoi vertex is less than or equal to* $2^d d$. *(b) When* $d = 2$*, a Voronoi vertex cannot have degree 7.*

Proof. (a) Consider the vertex placed at the origin; 2^d orthants are formed around the vertex. To obtain the maximum number of Voronoi regions that share this vertex in each orthant, we count the maximum number of Voronoi edges in the interior of an orthant that have this Voronoi vertex as endpoint; at most one such edge can exist in each orthant. Since these Voronoi edges are equidistant to d sites, result follows.

(b) Let $v^* = (x^*, y^*)$ be a Voronoi vertex of degree 7. Since 7 Voronoi edges meet at v^*, due to symmetry, we examine the two cases of Figure 5. When the configuration of Voronoi regions around the vertex is like in Figure 5a, then s_1 is a horizontal site and s_2, s_7 are vertical. Then, aff(s_2), aff$(s_7) \subset \{(x, y) \in \mathbb{R}^2 \mid x > x^*\}$. Since $v^* \in \mathcal{Z}^+(s_2) \cap \mathcal{Z}^+(s_7)$ and is equidistant to both s_2 and s_7, the affine hulls of s_2, s_7 coincide. Then, whichever is the orientation of s_1, the affine bisectors of s_1, s_2 and s_1, s_7 cannot intersect like in Figure 5a. When the configuration of Voronoi regions around the vertex is like in Figure 5b, since b_3 is vertical, s_1 is vertical. But since b_1 is horizontal, s_1 must be horizontal; a contradiction. □

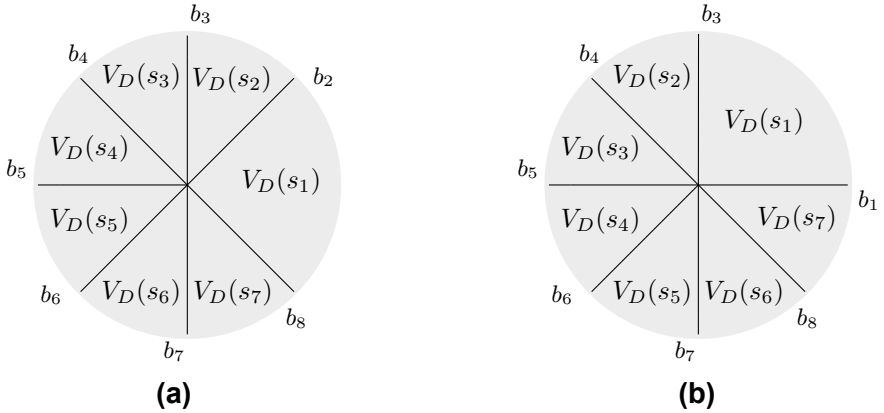


Figure 5: The two cases in proof of Lemma 2.3.

3. SUBDIVISION ALGORITHM IN TWO DIMENSIONS

Given manifold rectilinear polygon $\mathcal{P}$, i.e. every vertex being shared by exactly two edges, the input consists of $\mathcal{S}$ and a box $\mathcal{C}_0$ bounding $\mathcal{P}$. Non-manifold vertices can be trivially converted to manifold with an infinitesimal perturbation. Subdivision algorithms include two phases: the *subdivision* and the *reconstruction* phase. At the first phase $\mathcal{C}_0$ is recursively subdivided to 4 identical cells until some *termination criteria* are satisfied and the diagram's topology can be determined in $O(1)$ time inside each cell. The diagram is reconstructed in the second phase.

3.1 Subdivision Phase

We consider subdivision cells as closed. Given cell $\mathcal{C}$, let $\phi(\mathcal{C})$ be the set of sites whose closed Voronoi region intersects $\mathcal{C}$: $\phi(\mathcal{C}) = \{s \in \mathcal{S} \mid \overline{V_{\mathcal{D}}(s)} \cap \mathcal{C} \neq \emptyset\}$. For point $p \in \mathcal{P}$ we define its **label set** $\lambda(p) = \{s \in \mathcal{S} \mid p \in \overline{V_{\mathcal{D}}(s)}\}$. When $p \in \mathcal{P}^c$, where $\mathcal{P}^c$ is the complement of $\mathcal{P}$, then $\lambda(p) = \emptyset$. Using the definition of label sets, one can alternatively write $\phi(\mathcal{C})$ as $\phi(\mathcal{C}) = \bigcup_{p \in \mathcal{C}} \lambda(p)$.

Intuitively, storing $\phi(\mathcal{C})$ for every cell of the subdivision would ensure that upon termination and when the cardinality of this set for every leaf cell is a small constant, we could determine the topology of the Voronoi diagram inside each of them in constant time. The computation of $\phi(\mathcal{C})$ is hereditary, since $\phi(\mathcal{C}) \subseteq \phi(\mathcal{C}')$, if $\mathcal{C}'$ is the parent of $\mathcal{C}$. But it is rather costly; given $\phi(\mathcal{C}')$ with $|\phi(\mathcal{C}')| = \kappa$, it takes $O(\kappa^2)$ to compute $\phi(\mathcal{C})$, since the relative position of $\mathcal{C}$ to the bisector of every pair of sites in $\phi(\mathcal{C}')$ must be specified. Alternatively, following the work of [22, 5], instead of implementing the exact predicate $\phi(\cdot)$, we compute an approximate one. We denote by p_c, r_c the center and the L_∞-radius of $\mathcal{C}$ and define the **active set** of $\mathcal{C}$ as:

$$\widetilde{\phi}(\mathcal{C}) := \{s \in \mathcal{S} \mid \mathcal{Z}^+(s) \cap \mathcal{C} \neq \emptyset, \text{ and } \mu_\infty(p_c, s) \leq 2r_c + \delta_c\},$$

where $\delta_c = \min_s D_s(p_c)$, if $p_c \in \mathcal{P}$, and 0 otherwise. We now explain how $\widetilde{\phi}$ approximates ϕ by adapting [5, Lem.2], where $\widetilde{\phi}$ appears as a *soft version* of ϕ.

Lemma 3.1. *(a) For every $\mathcal{C}$, $\phi(\mathcal{C}) \subseteq \widetilde{\phi}(\mathcal{C})$.*
(b) For a sequence of cells $(\mathcal{C})_i$ monotonically convergent to point $p \in \mathcal{P}$, $\widetilde{\phi}(\mathcal{C}_i) = \phi(p)$ for i large enough.

Proof. (a) If $\phi(\mathcal{C}) = \emptyset$ the assertion is trivial. Let $s \in \phi(\mathcal{C})$ and $p \in \mathcal{C} \cap \overline{V_D(s)}$. It holds that $D_s(p) \leq D_{s'}(p) \Rightarrow \mu_\infty(p, s) \leq \mu_\infty(p, s')$ for every $s' \in \mathcal{S}$. We distinguish two cases according to the position of p_c relatively to $\mathcal{P}$. If $p_c \in \mathcal{P}$ and $p_c \in \overline{V_D(s^*)}$, where $s^* \in \mathcal{S}$, then:

$$\begin{aligned}\mu_\infty(p_c, s) &\leq \mu_\infty(p_c, p) + \mu_\infty(p, s) \leq \mu_\infty(p_c, p) + \mu_\infty(p, s^*) \leq \\ &\leq 2\mu_\infty(p_c, p) + \mu_\infty(p_c, s^*) \leq 2r_c + \mu_\infty(p_c, s^*).\end{aligned}$$

Otherwise, if $p_c \notin \mathcal{P}$, since $\mathcal{C} \cap \mathcal{P} \neq \emptyset$, there is a site s' intersecting $\mathcal{C}$ such that $\mu_\infty(p, s') \leq r_c$. Therefore,

$$\mu_\infty(p_c, s) \leq \mu_\infty(p_c, p) + \mu_\infty(p, s) \leq \mu_\infty(p_c, p) + \mu_\infty(p, s') \leq 2r_c.$$

(b) There exists $i_0 \in \mathbb{N}$ such that for $i \geq i_0$ $\mathcal{C}_i \cap \mathcal{P} \neq \emptyset$. Therefore, for $i \gg i_0$, since $p_{c_i} \to p$ and $r_{c_i} \to 0$, for every $s \in \mathcal{C}_i$, (a) implies that $s \in \lambda(p) = \phi(p)$. Since $\phi(p) \subseteq \widetilde{\phi}(\mathcal{C}_i)$, result follows. □

The previous lemma is crucial in proving correctness of the algorithm. It is also worth mentioning that a preliminary version of $\widetilde{\phi}$ first appeared in [17].

One can easily verify $\widetilde{\phi}(\mathcal{C}) \subseteq \widetilde{\phi}(\mathcal{C}')$, therefore the complexity of computing $\widetilde{\phi}(\mathcal{C})$ is linear in the size of $\widetilde{\phi}(\mathcal{C}')$. The algorithm proceeds as follows: For each subdivision cell we maintain the label sets of its corner points and of its central point, and $\widetilde{\phi}$. The subdivision of a cell stops whenever at least one of the *termination criteria* below holds (checked in turn). Upon subdivision, we propagate $\widetilde{\phi}$ and the label sets of the parent cell to its children. For every child we compute the remaining label sets and refine its active set. Let M be the maximum degree of a Voronoi vertex $(M \leq 8)$.

Termination criteria. The subdivision of a cell stops whenever at least one of the following criteria holds (checked in turn):

(T1) $\mathcal{C} \subseteq V_{\mathcal{D}}(s)$ for some $s \in \mathcal{S}$

(T2) $int(\mathcal{C}) \cap \mathcal{P} = \emptyset$

(T3) $|\widetilde{\phi}(\mathcal{C})| \leq 3$

(T4) $|\widetilde{\phi}(\mathcal{C})| \leq M$ and the sites in $\widetilde{\phi}(\mathcal{C})$ define a unique Voronoi vertex $v \in \mathcal{C}$.

When (T1) holds, $\mathcal{C}$ is contained in a Voronoi region so no part of the diagram is in it. So, there is no need for further subdivision. (T2) stops the subdivision when the open cell is completely outside the polygon. If (T3) holds, we determine in $O(1)$ the diagram's topology in $\mathcal{C}$ since there are ≤ 3 Voronoi regions intersected. (T4) stops cell subdivision if it contains a single degenerate Voronoi vertex. The process is summarized in Algorithm 1.

Theorem 3.1. *Algorithm 1 halts.*

Proof. Consider an infinite sequence of boxes $\mathcal{C}_1 \supseteq \mathcal{C}_2 \supseteq \ldots$ such that none of the termination criteria holds. Since (T1) and (T2) do not hold for any $\mathcal{C}_i$ with $i \geq 1$, the sequence converges to a point $p \in \mathcal{V}_{\mathcal{D}}(\mathcal{P})$. From Lem. 3.1(b), there exists $i_0 \in \mathbb{N}$ such that $\widetilde{\phi}(\mathcal{C}_{i_0}) = \phi(p) = \lambda(p)$. Since $|\lambda(p)| \leq 8$, (T4) will hold. □

Lemma 3.2. *For a subdivision cell $\mathcal{C}$, let $v_1, \ldots, v_4$ its corner vertices. For $s \in \mathcal{S}$, $\mathcal{C} \subseteq V_{\mathcal{D}}(s)$ if and only if $v_1, \ldots, v_4 \in V_{\mathcal{D}}(s)$.*

Algorithm 1 Subdivision2D($\mathcal{P}$)

```
1: root ← bounding box of P
2: Q ← root
3: while Q ≠ ∅ do
4:     C ← pop(Q)
5:     Compute φ̃(C) and the label sets of the vertices and the central point.
6:     if (T1) ∨ (T2) ∨ (T3) ∨ (T4) then
7:         return
8:     else
9:         Subdivide C into C1, C2, C3, C4
10:        Q ← Q ∪ {C1, C2, C3, C4}
11:    end if
12: end while
```

Proof. Let $v_1, \ldots, v_4 \in V_{\mathcal{D}}(s)$ and $p \in \mathcal{C}$. Then, $p \in \mathcal{Z}^+(s)$, since $\mathcal{Z}^+(s)$ is convex in 2D and $v_1, \ldots, v_4 \in \mathcal{Z}^+(s)$. For $i = 1, \ldots, 4$ the open ball $B_i := B_\infty(v_i, \mu_\infty(v_i, s))$ is empty of sites. Since $B_\infty(p, \mu_\infty(p, \mathsf{aff}(s))) \subset \cup_{i\in[4]} B_i$ it holds that $\mu_\infty(p, \mathcal{P}) \geq \mu_\infty(p, \mathsf{aff}(s)) = \mu_\infty(p, s)$. On the other hand, $\mu_\infty(p, \mathcal{P}) \leq \mu_\infty(p, s)$. So, if $p \notin V_{\mathcal{D}}(s)$ there is a site s' s.t. $D_{s'}(p) = D_s(p)$ and $p \in \mathcal{V}_{\mathcal{D}}(\mathcal{P})$. Therefore, since Voronoi regions are simply connected and Voronoi edges are straight lines, p must be on the boundary of $\mathcal{C}$. The two possible configurations are shown in Fig. 6 and are contradictory; for the first, use an argument similar to that of Lem. 2.3(b). For the second, notice that this cannot hold since the cell is square. We conclude that $\mathcal{C} \subseteq V_{\mathcal{D}}(s)$. The other direction is trivial. □

(a) **(b)**

Figure 6: The two cases in proof of Lemma 3.2. Different colors correspond to different Voronoi regions.

Lemma 3.3. *For a subdivision cell $\mathcal{C}$ it holds that $int(\mathcal{C}) \cap \mathcal{P} = \emptyset$ if and only if $\lambda(p_c) = \emptyset$ and for every $s \in \widetilde{\phi}(C)$ it holds that $s \cap int(C) = \emptyset$.*

Proof. Suppose that $\lambda(p_c) = \emptyset$, i.e. $p_c \notin \mathcal{P}$, and that $s \cap int(C) = \emptyset$, for every $s \in \widetilde{\phi}(C)$. If is $int(\mathcal{C}) \cap \mathcal{P} \neq \emptyset$, since the center of the cell is not in $\mathcal{P}$, a segment (site) on the boundary of $\mathcal{P}$ must intersect $int(\mathcal{C})$. This site belongs to $\widetilde{\phi}(C)$, leading to a contradiction. Proof of the other direction is trivial. □

Hence one decides (T1) by checking the vertices' labels. (T2) is valid for $\mathcal{C}$ iff $\lambda(p_c) = \emptyset$ and $\forall s \in \widetilde{\phi}(\mathcal{C})$, $s \cap int(\mathcal{C}) = \emptyset$. Fot (T4), the presence of a Voronoi vertex in $\mathcal{C}$ is verified

through constructor `VoronoiVertexTest`: given $\mathcal{C}$ with $|\widetilde{\phi}(\mathcal{C})| \geq 3$, the affine bisectors of sites in $\widetilde{\phi}(\mathcal{C})$ are intersected. If the intersection point is in $\mathcal{C}$ and in $\mathcal{Z}^+(s)$ for every $s \in \widetilde{\phi}(\mathcal{C})$ then it is a Voronoi vertex. We do not need to check whether it is in $\mathcal{P}$ or not; since (T1) fails for $\mathcal{C}$, if $v \notin \mathcal{P}$, there must be s intersecting $\mathcal{C}$ such that $v \notin \mathcal{Z}^+(s)$: contradiction.

3.2 Reconstruction Phase

We take the quadtree of the subdivision phase and output a planar straight-line graph (PSLG) $G = (V, E)$ representing the Voronoi diagram of $\mathcal{P}$. G is a (vertex) labeled graph and its nodes are of two types: *bisector nodes* and *Voronoi vertex nodes*. Bisector nodes span Voronoi edges and are labeled by the two sites to which they are equidistant. Voronoi vertex nodes correspond to Voronoi vertices and so are labeled by at least 3 sites.

The reconstruction can be briefly described as follows: We visit the leaves of the quadtree and, whenever the Voronoi diagram intersects the cell, bisector or vertex nodes are introduced. By connecting them accordingly with straight-line edges, we obtain the exact Voronoi diagram and not an approximation. We process leaves with $|\widetilde{\phi}(\cdot)| \geq 2$ that do not satisfy (T1) nor (T2).

Cell with two active sites. When $\widetilde{\phi}(\mathcal{C}) = \{s_1, s_2\}$, $\mathcal{C}$ intersects $V_{\mathcal{D}}(s_1)$ or $V_{\mathcal{D}}(s_2)$ or both; at this point $\mathcal{C}$ cannot be wholly contained in $V_D(s_1)$ or $V_D(s_2)$, since (T1) is not satisfied. The intersection of $\text{bis}_{\mathcal{D}}(s_1, s_2)$ with the cell, when non empty, is part of the Voronoi diagram: for each $p \in \text{bis}_{\mathcal{D}}(s_1, s_2) \cap \mathcal{C}$ it holds that $D_{s_1}(p) = D_{s_2}(p)$ and $\lambda(p) \subseteq \widetilde{\phi}(\mathcal{C}) = \{s_1, s_2\}$. Therefore $p \in \overline{V_{\mathcal{D}}(s_1)} \cap \overline{V_{\mathcal{D}}(s_2)}$.

Remark 3.1. *If there is no Voronoi vertex in $\mathcal{C}$ and $p_1, p_2 \in \text{bis}_{\mathcal{D}}(s_1, s_2) \cap \mathcal{C}$ for $s_1, s_2 \in \widetilde{\phi}(\mathcal{C})$, then $p_1p_2 \subset \text{bis}_{\mathcal{D}}(s_1, s_2)$.*

Since $\text{bis}_{\mathcal{D}}(s_1, s_2) \subset \text{affbis}(s_1, s_2)$ we intersect the affine bisector with the boundary of the cell. An intersection point $p \in bis_\infty(\text{aff}(s_1), \text{aff}(s_2))$ is in $bis_{\mathcal{D}}(s_1, s_2)$ iff $p \in \mathcal{Z}^+(s_1) \cap \mathcal{Z}^+(s_2)$ (Lemma 3.1). If intersection points are both in $\mathcal{Z}^+(s_1) \cap \mathcal{Z}^+(s_2)$ (Figure 7a), we introduce a bisector node in the middle of the line segment joining them, labeled by $\{s_1, s_2\}$. When only one intersection point is in $\mathcal{Z}^+(s_1) \cap \mathcal{Z}^+(s_2)$, then s_1, s_2 must intersect in $\mathcal{C}$ (Figure 7b). We insert a bisector node at their intersection point labeled by $\{s_1, s_2\}$.

Cell with 3 active sites or more. When $|\widetilde{\phi}(\mathcal{C})| = 3$ and the `VoronoiVertexTest` finds a vertex in $\mathcal{C}$ or when $|\widetilde{\phi}(\mathcal{C})| \geq 4$ (a vertex has already been found), we introduce a Voronoi vertex node at the vertex, labeled by corresponding sites. In the presence of corners of $\mathcal{P}$ in $\mathcal{C}$, bisector nodes are introduced and connected to the vertex node (Fig. 8).

If no Voronoi vertex is in $\mathcal{C}$, we repeat the procedure described in previous paragraph for each pair of sites. Even if a bisector node is found, it is not inserted at the graph if it is closer to the third site.

Figure 7: The intersection of the affine bisector and the cell. The circled points are in $\mathcal{Z}^+(s_1) \cap \mathcal{Z}^+(s_2)$, whereas the crossed point is not. Square nodes are the bisector nodes inserted to the output graph.

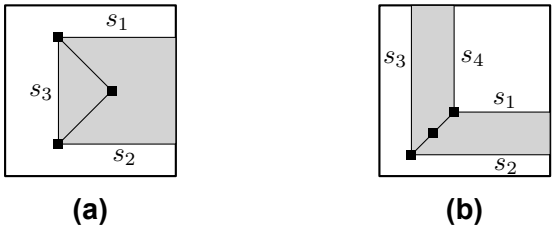


Figure 8: (a) A Voronoi vertex node labeled with $\{s_1, s_2, s_3\}$ connected with two bisector nodes labeled with $\{s_1, s_3\}$ and $\{s_2, s_3\}$. (b) A (degenerate) Voronoi vertex node labeled with $\{s_1, s_2, s_3, s_4\}$ connected with two bisector nodes.

Connecting the graph nodes. Once the graph nodes are fixed they have to be connected appropriately. The only graph edges added so far are those that are completely contained in a subdivision cell. The remaining graph edges must cross two subdivision cells. We apply "dual marching cubes" [19] to enumerate pairs of neighboring cells in time linear in the size of the quadtree: cells are neighboring if they share a facet.

The traversal method involves two recursive functions: the `faceProc` and the `edgeProc` (see Figure 10). At first we call the `faceProc` for the bounding box of the input polygon. This function, for every internal node of the quadtree, recursively calls itself for each of its children and the `edgeProc` for every pair of neighboring children. When the `faceProc` reaches a leaf it terminates. When the `edgeProc` reaches two adjacent leaves, then the corresponding cells share a facet. Let v_1, v_2 be graph nodes in neighboring cells. We connect them if and only if:

- v_1, v_2 are bisector nodes and $\lambda(v_1) = \lambda(v_2)$.
- v_1 is a bisector node, v_2 is a Voronoi vertex node and $\lambda(v_1) \subset \lambda(v_2)$.
- v_1, v_2 are Voronoi vertex nodes, $\lambda(v_1) \cap \lambda(v_2) = \{s, s'\}$ and $v_1v_2 \subset \mathcal{P}$.

See Fig. 9 for an example where v_1, v_2 are Voronoi vertex nodes with $\lambda(v_1) \cap \lambda(v_2) = \{s, s'\}$ and $v_1v_2 \not\subset \mathcal{P}$.

Theorem 3.2. *The output graph is isomorphic to $\mathcal{V}_{\mathcal{D}}(\mathcal{P})$.*

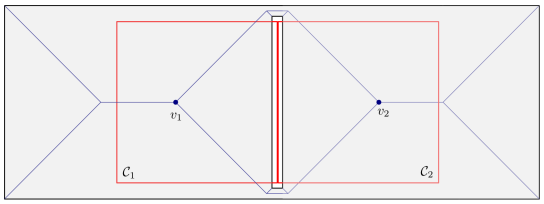
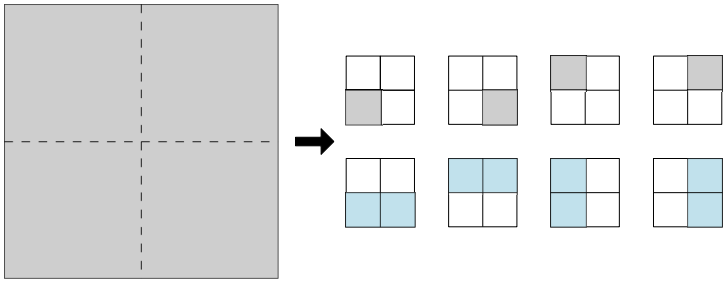
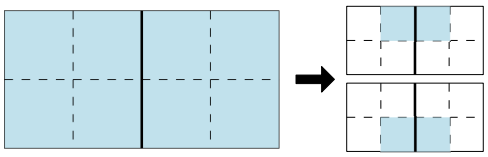


Figure 9: $\mathcal{C}_1, \mathcal{C}_2$ are two neighboring subdivision cells and the Voronoi vertices v_1, v_2 have two common sites as labels but are not connected with a Voronoi edge.



(a)

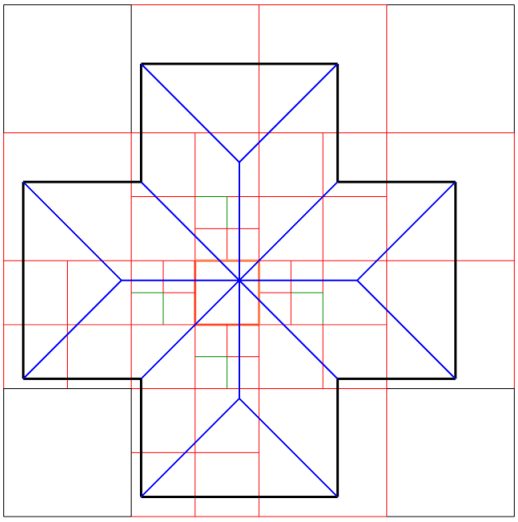


(b)

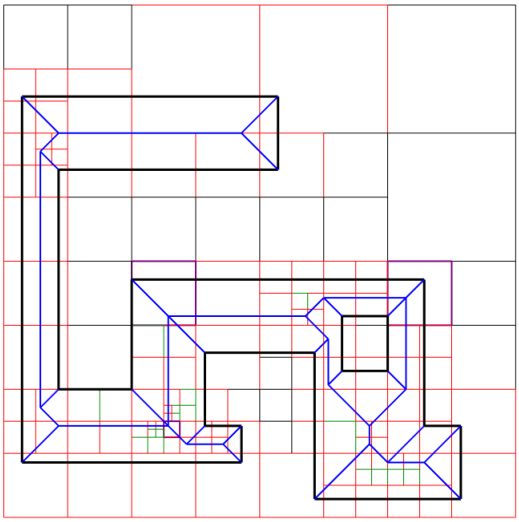
Figure 10: Illustration of (a) the `facetProc` function (b) the `edgeProc` function in the dual Marching Cubes method.

Proof. We need to prove that the nodes in the graph are connected correctly. Let neighboring cells $\mathcal{C}_1, \mathcal{C}_2$ and v_1, v_2 graph nodes in each of them respectively. If v_1, v_2 are bisector nodes and $\lambda(v_1) = \lambda(v_2)$, then the line segment v_1v_2 is in $bis_{\mathcal{D}}(s_1, s_2)$, for $s_1, s_2 \in \lambda(v_1)$, and on the Voronoi diagram (Rem. 3.1). If v_1 is a bisector node and v_2 is a Voronoi vertex node s.t. $\lambda(v_1) \subseteq \lambda(v_2)$, then $v_1v_2 \subset \text{bis}_\infty(s_1, s_2)$. If the segment v_1v_2 is not on the Voronoi diagram then, there is a Voronoi vertex node different than v_2 in $\mathcal{C}_1$ or $\mathcal{C}_2$; contradiction. At last, let v_1 and v_2 be Voronoi vertex nodes such that their labels have two sites in common, say s, s', and the edge $v_1v_2 \subset \mathcal{P}$. Vertices v_1, v_2 are both on the boundary of $\overline{V_{\mathcal{D}}(s)} \cap \overline{V_{\mathcal{D}}(s')}$. Since $v_1v_2 \subset \mathcal{P}$, if it does not coincide with the Voronoi edge equidistant to s, s', then both v_1, v_2 must also be on the boundary of a Voronoi region other than $\overline{V_{\mathcal{D}}(s)}$ and $\overline{V_{\mathcal{D}}(s')}$. This leads to a contradiction. □

See Figure 16 for some examples computed with our software. Notably, in Figure 11a a degenerate Voronoi vertex of the maximum possible degree is found.



(a) Input consists of 12 sites and 46 cells are generated. Total time is 3.0 ms. The Voronoi diagram contains a degenerate Voronoi vertex of degree 8.



(b) Input consists of 22 sites and 136 cells are generated. Total time is 8.6 ms.

Figure 11: The 1-skeleton of the Voronoi diagram is shown in blue.

3.3 Primitives and Data-structures

Assuming the input vertices are rational, Voronoi vertices are rational [18]. Computing Voronoi vertices, and intersections between affine bisectors and cell facets require linear operations, distance evaluations and comparisons. Therefore, they are exact. The above operations, computing $\widetilde{\phi}$ and deciding site-cell intersection are formulated to allow for a direct extension to 3D. In the sequel we discuss design of predicates, computation of label sets and construction of a Bounding Volume Hierarchy.

3.3.1 Primitives

Membership in $\mathcal{H}(s)$ is trivial to decide, thus we focus on predicates that decide membership in $\mathcal{Z}(s)$. Given $p \in \mathbb{R}^2$ and $s \in \mathcal{S}$, let $pr_{\mathsf{aff}(s)}(p)$ the projection of p to $\mathsf{aff}(s)$ and $I_{p,s}$ the $1d-$interval on $\mathsf{aff}(s)$ centered at $pr_{\mathsf{aff}(s)}(p)$ with radius $\mu_\infty(p, \mathsf{aff}(s))$. `inZone`$(p, s)$ decides if $p \in \mathcal{Z}(s)$; this holds if and only if $I_{p,s} \cap s \neq \emptyset$ (Figure 12).

Given $\mathcal{C}$,$s \in \mathcal{S}$, `ZoneInCell`$(s, \mathcal{C})$ decides if $\mathcal{Z}(s) \cap \mathcal{C} \neq \emptyset$. For this evaluation see Lem-

Figure 12: Test performed by `inZone`(p, s).

(a) (b)

Figure 13: Illustration of test performed by `ZoneInCell`

mma 3.4 and Figure 13.

Lemma 3.4. *Let $s \in \mathcal{S}$, f_1, f_2 the two facets of $\mathcal{C}$ parallel to aff(s), $\rho_i = \mu_\infty(f_i, \text{aff}(s))$ for $i = 1, 2$ and $p'_c = pr_{\text{aff}(s)}(p_c)$. Then, $\mathcal{Z}(s) \cap \mathcal{C} \neq \emptyset$ iff $\exists i \in \{1, 2\}$ s.t. $B_\infty(p'_c, r_c + \rho_i) \cap s \neq \emptyset$.*

Proof. $\mathcal{Z}(s) \cap \mathcal{C} \neq \emptyset$ iff $\mathcal{Z}(s) \cap f_i \neq \emptyset$ for at least one $i \in \{1, 2\}$: Let $p \in \mathcal{Z}(s) \cap C$ s.t. $p \notin f_1 \cup f_2$ and pr$_{f_i}(p)$ be the projection of p on f_i. There exists $i \in \{1, 2\}$ s.t. $\mu_\infty(\text{pr}_{f_i}(p), \text{aff}(s)) > \mu_\infty(p, \text{aff}(s))$. Then, pr$_{f_i}(p) \in \mathcal{Z}(s)$. It holds that $\mathcal{Z}(s) \cap f_i \neq \emptyset$ iff $B_\infty(p'_c, r_c + \rho_i) \cap s \neq \emptyset$: Let $q \in \mathcal{Z}(s) \cap f_i$ and q' its projection on aff(s). Then $\mu_\infty(q', s) \leq \mu_\infty(q, \text{aff}(s)) = \rho_i$ and $\mu_\infty(p'_c, q') \leq r_c$. We deduce that $B_\infty(p'_c, r_c + \rho_i) \cap s \neq \emptyset$, since $\mu_\infty(p'_c, s) \leq \mu_\infty(p'_c, q') + \mu_\infty(q', s) \leq r_c + \rho_i$. For the inverse direction, let $B_\infty(p'_c, r_c + \rho_i) \cap s \neq \emptyset$ and q' in s s.t. $\mu_\infty(p'_c, q') \leq r_c + \rho_i$. Let q be its projection on aff(f_i). If $q \in f_i$ we are done. Otherwise, q is at L_∞ distance from f_i equal to $\mu_\infty(p'_c, q') - r_c$, attained at a boundary point $q'' \in f_i$. Then, $\rho_i \leq \mu_\infty(q'', s) \leq \mu_\infty(q'', q') = \max\{\rho_i, \mu_\infty(p'_c, q') - r_c\} = \rho_i$. It follows that $q'' \in \mathcal{Z}(s)$. □

To decide if $s \cap \mathcal{C} \neq \emptyset$ and if $s \cap int(\mathcal{C}) \neq \emptyset$, we use `isIntersecting`$(s, \mathcal{C})$ and `isStrictlyIntersecting`$(s, \mathcal{C})$ respectively. Design is trivial. All these predicates are computed in $O(1)$ time.

Therefore, deciding whether a site belongs to $\widetilde{\phi}(\mathcal{C})$, is done using the predicates:

- `isIntersecting`: to decide whether the site intersects the cell centered at p_c with radius $2 \cdot r_c + \delta_c$.
- `ZoneInCell`: to decide whether the zone of the site intersects $\mathcal{C}$.

Computing label sets. If $p \in \mathcal{P} \cap \mathcal{C}$ then its closest sites are in $\widetilde{\phi}(\mathcal{C})$. Deciding if $p \in \mathcal{P}$ is done by `LocationTest`, which identifies position based on the sites that intersect $\mathcal{C}$: among these we select those with minimum L_∞ distance to p and for whom `inZone`(p, s) is true. If a convex (resp. concave) corner with respect to the interior of $\mathcal{P}$ is formed by these sites then $p \in \mathcal{P}$ iff it belongs to the intersection (resp. union) of the oriented zones. If no corner is formed or even if $\mathcal{C}$ is not intersected by any site, decision is trivial. This takes $O(|\widetilde{\phi}(\mathcal{C})|)$ time.

In more details, we define:

- $T := \{s \in \mathcal{S} \mid s \cap \mathcal{C} \neq \emptyset\}$
- $d(p, T) := \min\{\mu_\infty(p, s) \mid s \in T \text{ and } p \in \mathcal{Z}(s)\}$

Figure 14: The two possible configurations of a corner.

- $R(p,T) := \{s \in T \mid \mu_\infty(p,s) = d(p)$ and $p \in \mathcal{Z}(s)\}$
- $R'(p,s,T) = \{s' \in R(p,T) \mid s \cap s' \neq \emptyset\}$, $s \in R(p,T)$.

If $T = \emptyset$ then the cell is contained in $\mathcal{P}$. Otherwise, even if $p \notin \mathcal{P}$ there always exists a site s intersecting $\mathcal{C}$ with $p \in \mathcal{Z}(s)$. Therefore $d(p,T)$ is well defined and $|R(p,T)| \geq 1$ for every $p \in \mathcal{C}$. We pick a site $s \in R(p,T)$ and then perform the `LocationTest` for $R'(p,s,T)$ that returns true if and only if $p \in \mathcal{P}$; this test identifies the position of p relative to $\mathcal{P}$ using one site if $|R'(p,s,T)| = 1$ or a corner formed by two sites in $R'(p,s,T)$ ortherwise. Note that $R'(p,s,T)$ may contain 1 or 2 sites.

`LocationTest`$(p, R'(p,s,T))$

1: **if** $|R'(p,s,T)| = 1$ **then** ▷ $R'(p,s,T) = \{s\}$
2: **return** $p \in \mathcal{Z}^+(s_1)$
3: **else if** $|R'(p,s,T)| = 2$ **then** ▷ $R'(p,s) = \{s, s'\}$
4: **if** $\angle s's$ is convex **then**
5: **return** $p \in \mathcal{Z}^+(s) \cap \mathcal{Z}^+(s')$
6: **else**
7: **return** $p \in \mathcal{Z}^+(s) \cup \mathcal{Z}^+(s')$
8: **end if**
9: **end if**

Lemma 3.5. *Let $p \in \mathcal{C}$. The following equivalence holds:*

$p \in \mathcal{P}$ if and only if $T = \emptyset$ or `LocationTest`$(p, R'(p,s,T))$ *returns* `true` *for some $s \in R(p,T)$*

Proof. When $T = \emptyset$ the equivalence is obvious. When $T \neq \emptyset$, we distinguish two cases according to the cardinality of $R'(p,s,T)$: When $|R'(p,s,T)| = 1$ it is straightforward that $p \in \mathcal{P} \Leftrightarrow p \in \mathcal{Z}^+(s)$. When $|R'(p,s,T)| = 2$, the configuration of sites in $R'(p,s)$ is like in Figures 14a and 14b. The test returns `true` if and only if $p \in \mathcal{P}$ in any case. □

3.3.2 Rectangular Decomposition and Bounding Volume Hierarchy

We decompose $\mathcal{P}$ into a collection of rectangles such that any two of them have disjoint interior. The resulting decomposition may not be optimal with respect to the number of rectangles but allows for an immediate construction of a data structure (a Bounding Volume Hierarchy) that can answer point and rectangle-intersection queries on the decomposition's rectangles efficiently. This data structure can accelerate the 2D algorithm for certain inputs and is essential for efficiency of the 3D version of the algorithm.

Rectangular decomposition.

It is known [20] that the minimum number of rectangles in a partition of a polygon with n vertices and h holes is $n/2+h-g-1$, where g is the maximum number of nonintersecting chords that can be drawn either horizontally or vertically between reflex vertices. To the direction of minimizing the number of rectangles, the optimal algorithm has $O(n^{3/2}\log n)$ time complexity [15].

However, for our purpose, the decomposition does not need to be optimal. We observe that drawing an axis-parallel edge, from every reflex vertex results to a rectangular decomposition. Thus, we construct a kd-tree on the reflex vertices of the polygon, splitting always at a vertex. Assuming the polygon has r reflex vertices, the kd-tree subdivides the plane into at most $r+1$ regions. Every terminal region contains a disjoint collection of rectangles (nonempty). We denote by t the maximum number of rectangles in a terminal region.

Lemma 3.6. *For an orthogonal polygon on n vertices with h holes, let r be the number of its reflex vertices. It holds that:*

$$r = \frac{n}{2} + 2(h-1)$$

Proof. A simple orthogonal polygon without holes on N vertices has $N/2-2$ reflex vertices. So, if we denote by n_0 the number of the outer contour's vertices, and by n_i the respective number for every hole ($i \in [h]$) we have that: $r = n_0/2 - 2 + \sum_{i=1}^{h}\left(n_i/2+2\right) = \frac{n}{2} + 2(h-1)$. □

Bounding Volume Hierarchy.

A *Bounding Volume Hierarchy* (BVH) [11] is a tree structure on a set of objects stored at the leaves along with their bounding volume while internal nodes store information of their descendants' bounding volume. Two important properties are *minimal volume* and *small overlap*.

In our setting, the geometric objects are the axis aligned rectangles obtained by the decomposition. Consequently, the most appropriate bounding shape is the Axis Aligned Bounding Box (AABB) offering a good tradeoff between minimal volume and simplicity of representation. As for minimizing the overlap of bounding boxes at the same level in the hierarchy, we group rectangles in a specific way in order to accomplish that:

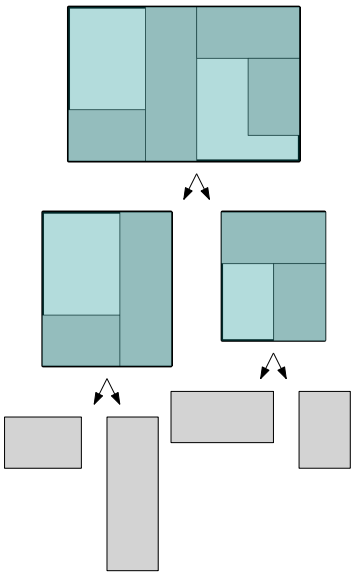


Figure 15: A rectangular decomposition for an orthogonal polygon and the corresponding BVH tree

The BVH is built in a bottom-up manner, by traversing the kd-tree previously constructed and adding some additional information to its nodes. At every leaf of the kd-tree we compute the AABB of its rectangles (namely a *terminal bounding box*) and for every internal node we find the AABB of its two children. In that way, the bounding volumes of a node's children intersect only at their boundary (see Figure 15 for a simple illustration). Space complexity is linear in tree size.

Rectangle-Intersection queries: Given query rectangle Q the data structure reports all rectangles in the decomposition overlapping with Q. Starting from the root, for every internal node, we check whether Q intersects its bounding rectangle or not. In the latter case the data structure reports no rectangles. In the former, we check the position of Q relative to the bounding boxes of the node's children so as to decide for each one if it should be traversed or not. We continue similarly: when we reach a terminal bounding box, we check the position of Q relative to every rectangle in it. Let k be the number of terminal bounding boxes intersected by Q. Following [1], we count the number of internal nodes visited on each level of the tree and show:

Theorem 3.3. *Rectangle intersection queries are answered in* $O(k \lg r + kt)$.

Proof. Let Q a rectangle-intersection query and v an internal node of the BVH tree visited during the query. We distinguish two cases; in the first case the subtree rooted at v contains a terminal bounding box that intersects Q. There are $O(k)$ such nodes at each level. Otherwise, Q intersects with the bounding rectangle V stored at v but does not intersect any terminal bounding box of the subtree rooted at v . There are at least two such terminal bounding boxes, say b and b'. Since Q does not intersect b there is a line ℓ passing through a facet of Q separating Q from b. Similarly, there exists a line ℓ' passing through a facet of

Q that separates it from b'. W.l.o.g. there is a choice of b, b' such that ℓ and ℓ' are distinct -if all the terminal bounding boxes of the subtree can be separated by the same line, then V cannot intersect Q-. If ℓ, ℓ' are perpendicular, then their intersection also intersects V. Since the bounding boxes of each level are strictly non-overlapping, every vertex of Q intersects a constant number of them (up to 4). So, there is a constant number of such nodes at a given level. When ℓ, ℓ' are parallel and no vertex of Q intersects V, then the terminal bounding rectangles of the subtree can be partitioned to those separated by ℓ from Q and to those separated by ℓ' from Q. For these distinct sets of terminal bounding boxes to be formed, there must occur a split of V by a line parallel and in between ℓ, ℓ'. So there is a reflex vertex of the polygon in $V \cap Q$, causing this split. But $V \cap Q \cap \mathcal{P} = \emptyset$; a contradiction.

So there are $O(k)$ internal nodes visited at each level of tree. The visited leaf nodes correspond to the $O(k)$ terminal bounding boxes that intersect Q and since each of them encloses at most t rectangles, the additional amount of performed operations equals $O(kt)$. Summing over all levels of the tree yields a total query complexity of $\sum_{i=0}^{\lceil \lg r \rceil} O(k) + O(kt) = O(k \lg r + kt)$. □

Point queries: Given $p \in \mathbb{R}^2$, we report on the rectangles of the decomposition in which p lies inside. The number of reported rectangles can be at most 4; for the case of four rectangles meeting at a vertex. When zero, the point lies outside the polygon. Since it is a special case of a rectangle-intersection query, the query time complexity is $O(\lg r + t)$.

3.4 Complexity analysis

Analysis requires a bound on the height of the quadtree. The edge length of the initial bounding box is supposed to be 1 under appropriate scaling. Let *separation bound* Δ be the maximum value such that for every cell of edge length $\geq \Delta$ at least one termination criterion holds. Then, the maximum tree height is $L = O(\lg(1/\Delta))$. Let β be the minimum distance of two Voronoi vertices, and γ the relative thickness of $\mathcal{P}^c$, i.e. the minimum diameter of a maximally inscribed L_∞-ball in $\mathcal{P}^c$, where $\mathcal{P}^c$ is the complement of $\mathcal{P}$.

Lemma 3.7. *Separation bound Δ is $\Omega(\min\{\gamma, \beta\})$, where the asymptotic notation is used to hide some constants.*

Proof. The algorithm mainly subdivides cells that intersect $\mathcal{V}_\mathcal{D}(\mathcal{P})$, since a cell inside a Voronoi region or outside $\mathcal{P}$ is not subdivided (Termination criteria (T1), (T2)). Most subdivisions occur as long as non neighboring Voronoi regions are "too close". Consider $\mathcal{C}$ centered at $p_\mathcal{C} \in V_\mathcal{D}(s)$ and $s' \in \widetilde{\phi}(\mathcal{C}) \setminus \phi(\mathcal{C})$, with $V_\mathcal{D}(s), V_\mathcal{D}(s')$ non neighboring. For $r_\mathcal{C} < \frac{\mu_\infty(p_\mathcal{C}, s') - \mu_\infty(p_\mathcal{C}, s)}{2}$ site s' is not in $\widetilde{\phi}(\mathcal{C})$. It holds that $\mu_\infty(p_\mathcal{C}, s') - \mu_\infty(p_\mathcal{C}, s) \geq \zeta(s, s')$, where $\zeta(s, s') = \min\{\mu_\infty(p, q) \mid p \in \overline{V_\mathcal{D}(s)}, q \in \overline{V_\mathcal{D}(s')}\}$, i.e. the minimum distance of the closure of the two Voronoi regions. When $\overline{V_\mathcal{D}(s)}, \overline{V_\mathcal{D}(s')}$ are connected with a Voronoi edge, $\zeta(s, s') = \Omega(\beta)$. When a minimum cell size of $\Omega(\beta)$ is not sufficient for s' to not belong in $\widetilde{\phi}(\mathcal{C})$, then there is a hole between $\overline{V_\mathcal{D}(s)}, \overline{V_\mathcal{D}(s')}$ and Δ is $\Omega(\gamma)$ in this case. □

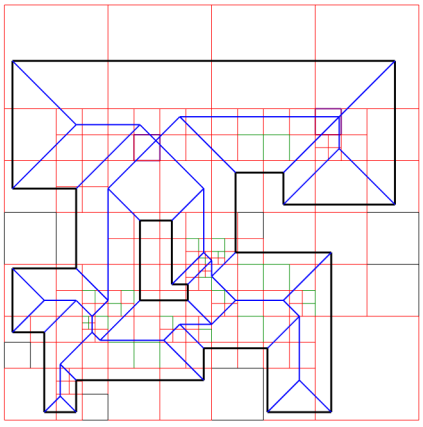
This lower bound is tight: in Figure 16a for $\Delta = 0.8125\beta$, and in Figure 16b for $\Delta = \gamma$. Next we target a realistic complexity analysis rather than worst-case. For this, assume the site distribution in $\mathcal{C}_0$ is "sufficiently uniform". Formally:

Uniform Distribution Hypothesis (UDH). For L_∞ balls $A_1 \subseteq A_0 \subset \mathcal{C}_0$, let N_0 (resp. N_1) be the number of sites intersecting A_0 (resp. A_1). We suppose $N_1/N_0 = O(vol(A_1)/vol(A_0))$, where $vol(\cdot)$ denotes the volume of a set in $\mathbb{R}^d$, d being the dimension of $\mathcal{C}_0$.

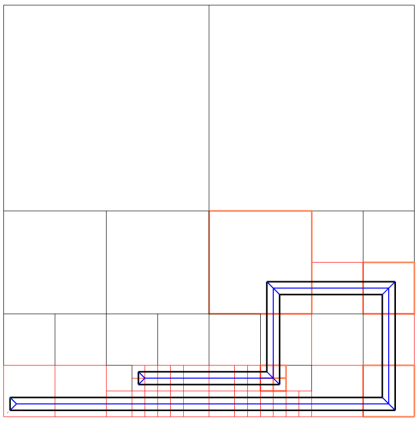
Theorem 3.4. *Under UDH the algorithm's complexity is $O(n/\Delta + 1/\Delta^2)$, where n is the total number of boundary edges (including any holes).*

Proof. At each node, refinement and checking the termination criteria run in time linear in the size of its parent's active set. At the root $|\widetilde{\phi}(\mathcal{C}_0)| = n$. The cardinality of active sets decreases as we move to the lower levels of the quadtree: Let $A(p, d, R) = \{q \in \mathbb{R}^2 \mid d \leq \mu_\infty(p, q) \leq 2R + d\}$. For cell $\mathcal{C}$ and $s \in \widetilde{\phi}(\mathcal{C})$, $s \cap A(p_c, \delta_c, r_c) \neq \emptyset$. Let $E = vol(A(p_c, \delta_c, r_c))$. For $\mathcal{C}_1$ a child of $\mathcal{C}$ and $s_1 \in \widetilde{\phi}(\mathcal{C}_1)$, $s_1 \cap A(p_{c_1}, \delta_{c_1}, r_{c_1}) \neq \emptyset$. Since $B_\infty(p_c, \delta_c)$ is empty of sites and may intersect with $A(p_{c_1}, \delta_{c_1}, r_{c_1})$, we let $E_1 = vol(A(p_{c_1}, \delta_{c_1}, r_{c_1}) \setminus (A(p_{c_1}, \delta_{c_1}, r_{c_1}) \cap B_\infty(p_c, \delta_c)))$. We prove that in any combination of $\delta_c, \delta_{c_1}, r_c$ it is $E_1 \leq E/2$. Under Hypothesis 1, a cell at tree level i has $|\widetilde{\phi}(\mathcal{C}_i)| = O(n/2^i)$. Computation per tree level, is linear in sum of active sets' cardinality, therefore summing over all levels of the tree, we find the that complexity of the subdivision phase is $O(n/\Delta)$. The complexity of the reconstruction phase is $O(\tilde{n})$, where $\tilde{n}$ is the number of leaf nodes in the quadtree, which is in turn $O(1/\Delta^2)$. This allows to conclude. □

Queries in the BVH can be used to compute label sets and the active set of a cell. Assume the number of segments touching a rectangle's boundary is $O(1)$, which is the typical case. Then, we prove the following.



(a) Input consists of 28 sites and 164 cells are generated. Minimum cell size is $0.8125 \cdot \beta$.



(b) Input consists of 12 sites and 64 cells are generated. Minimum cell size is γ.

Figure 16: The 1-skeleton of the Voronoi diagram is shown in blue.

Lemma 3.8. *We denote by $\mathcal{C}'$ the parent of $\mathcal{C}$ in the subdivision. Using BVH accelerates the refinement of $\mathcal{C}$ if $|\widetilde{\phi}(\mathcal{C}')|/|\widetilde{\phi}(\mathcal{C})| = \Omega(\lg n + t)$.*

Proof. A label set $\lambda(p)$ is determined by performing a point and a rectagle-intersection query; once the point is detected to lie inside a rectangle R_0 of a leaf T_0 we find an initial estimation d_0 of $\mu_\infty(p, \mathcal{P})$. Since the closest site to p may be on another leaf, we do a rectangle-intersection query centered at p with radius d_0. The closest site(s) to p are in the intersected leaves. Thus, finding $\lambda(p)$ takes $O(k(p, d_0) \lg r + k(p, d_0)t)$ (Thm. 3.3), where $k(p, d_0) = O(1)$ is the number of BVH leaves intersected by $\overline{B_\infty(p, d_0)}$. Computing the sites in $\widetilde{\phi}(\mathcal{C})$ is accelerated if combined with a rectangle intersection query to find segments at L_∞-distance $\leq 2r_c + \delta_c$ from p_c. Let k_c be the maximum number of BVH leaves intersected by this rectangle-intersection query. We obtain a total refinement time for the cell equal to $O(k_c \lg r + k_c t)$. Since $k_c = O(|\widetilde{\phi}(\mathcal{C})|)$ and $r = O(n)$ the lemma follows. □

4. SUBDIVISION ALGORITHM IN THREE DIMENSIONS

Let $\mathcal{P}$ be a manifold orthogonal polyhedron: every edge of $\mathcal{P}$ is shared by exactly two and every vertex by exactly 3 facets. For non-manifold input we first employ trihedralization of vertices, discussed in [16, 4]. Input consists of $\mathcal{S}$ and bounding box $\mathcal{C}_0$ of $\mathcal{P}$. An octree is used to represent the subdivision.

The main difference with the 2D case is that Voronoi sites can be nonconvex. As a consequence, for site s, $\mathcal{Z}^+(s)$ is not necessarily convex and therefore the distance function $D_s(\cdot)$ cannot be computed in $O(1)$ time: it is not trivial to check membership in $\mathcal{Z}^+(s)$. It is direct to extend the 2D algorithm in three dimensions. However, we examine efficiency issues.

For an efficient computation of the basic predicates (of Chapter 3.3), we **preprocess** every facet of the polyhedron and decompose it to a collection of rectangles. Then a BVH on the rectangles is constructed. The basic operation of all these predicates in 2D is an overlap test between an interval and a segment in 1D. In 3D, the analog is an overlap test between a 2D rectangle and a site (rectilinear polygon). Once the BVH is constructed for each facet, the rectangle-intersection query takes time logarithmic in the number of facet vertices (Theorem 3.3).

4.1 Subdivision Phase

The active set $\widetilde{\phi}$, ϕ and the label set of a point are defined as in to 2D. Most importantly, Lem. 3.1 is valid in 3D as well. The algorithm proceeds as follows: We recursively subdivide $\mathcal{C}_0$ into 8 identical cells. The subdivision of a cell stops whenever at least one of the termination criteria below holds. For each cell of the subdivision we maintain the label set of its central point and $\widetilde{\phi}$. Upon subdivision, we propagate $\widetilde{\phi}$ from a parent cell to its children for further refinement. We denote by M the maximum degree of a Voronoi vertex $(M \leq 24)$.

3D Termination criteria. The subdivision of a cell stops whenever at least one of the following criteria holds (checked in turn):

(T1') $int(\mathcal{C}) \cap \mathcal{P} = \emptyset$,

(T2') $|\widetilde{\phi}(\mathcal{C})| \leq 4$,

(T3') $|\widetilde{\phi}(\mathcal{C})| \leq M$ and the sites in $\widetilde{\phi}(\mathcal{C})$ define a unique Voronoi vertex $v \in \mathcal{C}$.

Subdivision is summarized in Algorithm 2. (T1') is valid for $\mathcal{C}$ if and only if $\lambda(p_c) = \emptyset$ and $\forall s \in \widetilde{\phi}(C)$ it holds that $s \cap int(\mathcal{C}) = \emptyset$. Detecting a Voronoi vertex in $\mathcal{C}$ proceeds like in

Algorithm 2 Subdivision3D($\mathcal{P}$)

```
1: root ← bounding box of P
2: Q ← root
3: while Q ≠ ∅ do
4:     C ← pop(Q)
5:     Compute the label set of central point and φ̃(C).
6:     if (T1') ∨ (T2') ∨ (T3') then
7:         return
8:     else
9:         Subdivide C into C_1, ..., C_8
10:        Q ← Q ∪ {C_1, ..., C_8}
11:    end if
12: end while
```

2D. A Voronoi vertex is equidistant to at least 4 sites and there is a site parallel to each coordinate hyperplane among them.

(T1) used in 2D is omitted, for it is not efficiently decided: labels of the cell vertices cannot guarantee that $\mathcal{C} \subseteq V_{\mathcal{D}}(s)$. However, as the following lemma indicates, termination of the subdivision is not affected.

Lemma 4.1. *Let $\mathcal{C} \subseteq V_{\mathcal{D}}(s)$. There exists $r^* > 0$ s.t. if $r_c < r^*$ it holds that $\widetilde{\phi}(\mathcal{C}) = \{s\}$.*

Proof. Let $s' \in \mathcal{S} \setminus s$. We will prove that if $\mathcal{Z}^+(s') \cap \mathcal{C} \neq \emptyset$, it holds that $\delta_c < \mu_\infty(p_c, s')$. Therefore there is $r(s') > 0$ such that $2r(s') + \delta_c < \mu_\infty(p_c, s')$. Let r^* be the minimum of these radii for every site different than s. When $r_c < r^*$, it holds that $\widetilde{\phi}(\mathcal{C}) = \{s\}$.

Suppose that $\delta_c = \mu_\infty(p_c, s') = \mu_\infty(p_c, q)$ for $q \in s'$. Then $q \in \text{aff}(s)$ and s' cannot be a subset of aff(s). So, s' is adjacent to s. If $\mathcal{Z}^+(s') \cap \mathcal{C} = \emptyset$ then $s' \notin \widetilde{\phi}(\mathcal{C})$. If $\mathcal{Z}^+(s') \cap \mathcal{C} \neq \emptyset$, since for adjacent sites it holds that $\mathcal{Z}^+(s) \cap \mathcal{Z}^+(s') = \text{bis}_{\mathcal{D}}(s, s')$, bisector intersects $\mathcal{C}$ which is a contradiction. Thus, there is no site s' with $\mathcal{Z}^+(s') \cap \mathcal{C} \neq \emptyset$ and $\mu_\infty(p_c, s') = \delta_c$. □

Theorem 4.1. *Algorithm 2 halts.*

4.2 Reconstruction Phase

We construct a graph $G = (V, E)$, representing the 1-skeleton of the Voronoi diagram. The nodes of G are of two types, *skeleton nodes* and *Voronoi vertex nodes*, and are labeled by their closest sites. Skeleton nodes span Voronoi edges and are labeled by 3 or 4 sites. We visit the leaves of the octree and process cells with $|\widetilde{\phi}(\mathcal{C})| \geq 3$ and that do not satisfy (T1'). We introduce the nodes to the graph as in 2D. Graph edges are added between corners and Voronoi vertex nodes inside a cell. We run dual marching cubes (linear in the octree size) and connect graph nodes v_1, v_2 located in neighboring cells, if and only if:

- v_1, v_2 are skeleton nodes and $\lambda(v_1) = \lambda(v_2)$, or
- v_1 is a skeleton node, v_2 is a Voronoi vertex node and $\lambda(v_1) \subset \lambda(v_2)$, or
- v_1, v_2 are Voronoi vertex nodes, $\lambda(v_1) \cap \lambda(v_2) = \{s, s', s''\}$ and segment $v_1v_2 \subset \mathcal{P}$.

Theorem 4.2 (Correctness)**.** *The output graph is isomorphic to the 1-skeleton of the Voronoi diagram.*

An example computed with our software is shown in Figure 17.

Primitives. Deciding membership in $\mathcal{H}(\cdot)$ is trivial. The predicates of Chapter 3.3 extend to 3D and the runtime of each is that of a rectangle-intersection query on the BVH constructed for the corresponding site at preprocessing: Let $pr_{\mathsf{aff}(s)}(p)$ be the projection of p to $\mathsf{aff}(s)$ and $B_{p,s}$ the $2d-$box on $\mathsf{aff}(s)$ centered at $pr_{\mathsf{aff}(s)}(p)$ with radius $\mu_\infty(p, \mathsf{aff}(s))$. Then, $p \in \mathcal{Z}(s)$ iff $B_{p,s} \cap s \neq \emptyset$. A query with $B_{p,s}$ is done by `inZone`(p, s). For `ZoneInCell`$(p, \mathcal{C})$ we do a query with $\overline{B_\infty(p'_c, r_c + \rho_i)}$ where $p'_c = pr_{\mathsf{aff}(s)}(p_c)$, $\rho_i = \mu_\infty(f_i, \mathsf{aff}(s))$ and f_1, f_2 the two facets of $\mathcal{C}$ parallel to $\mathsf{aff}(s)$. Queries with $\overline{B_\infty(p'_c, r_c)}$ are also performed by `isIntersecting`$(s, \mathcal{C})$ and `isStrictlyIntersecting`$(s, \mathcal{C})$. When computing *label sets*, `LocationTest` is slightly modified, since the corners used to identify the position of a point can also be formed by 3 sites.

4.3 Complexity analysis

Under appropriate scaling so that the edge length of $\mathcal{C}_0$ be 1, if Δ is the separation bound, then the maximum height of the octree is $L = O(\lg(1/\Delta))$. The algorithm mainly subdivides cells intersecting $\mathcal{P}$, unlike the 2D algorithm that mainly subdivides cells intersecting $\mathcal{V}_\mathcal{D}(\mathcal{P})$, because a criterion like (T1) is missing. This absence does not affect tree height, since by Lem. 4.1 the minimum cell size is same as when we separate sites whose regions are non-neighboring (handled by Lem. 3.7). If β is the minimum distance of two Voronoi vertices and γ the relative thickness of $\mathcal{P}^c$, taking $\Delta = \Omega(\min\{\gamma, \beta\})$ suffices, as in 2D (Lem. 3.7).

Theorem 4.3. *Under UDH and if n is the number of polyhedral facets, α the maximum number of vertices per facet and t_α the maximum number of rectangles in a BVH leaf, the algorithm's complexity is $O(n\,\alpha(\lg\alpha + t_\alpha)/\Delta^2 + 1/\Delta^3)$.*

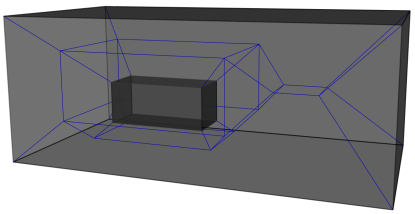


Figure 17: The 1-skeleton of the Voronoi diagram is shown in blue. Input consists of 12 sites and 386 cells are generated (not shown). Total time is 94.8 ms.

Proof. We sum active sets' cardinalities of octree nodes, since refining a cell requires a number of rectangle-intersection queries linear in the size of its parent's active set. Let $\mathcal{C}$ and its child $\mathcal{C}_1$. Any $s \in \widetilde{\phi}(\mathcal{C})$ satisfies $\delta_c \leq \mu_\infty(p_c, s) \leq 2r_c + \delta_c$. We denote by E the volume of the annulus $\{q \in \mathbb{R}^2 \mid \delta_c \leq \mu_\infty(p_c, q) \leq 2r_c + \delta_c\}$ and by E_1 the volume of the respective annulus for $\mathcal{C}_1$, minus the volume of the annulus' intersection with $B_\infty(p_c, \delta_c)$. It is easy to show $E_1 \leq E/2$. Under Hypothesis 1, we sum all levels and bound by $O(4^L n)$ the number of rectangle-intesection queries. Using Theorem 3.3, we find that the complexity of the subdivision phase is $O(n\,\alpha(\lg\alpha + t_\alpha)/\Delta^2)$. The complexity of the reconstruction phase is $O(\tilde{n})$, where $\tilde{n}$ is the number of leaf nodes in the octree, which is in turn $O(1/\Delta^3)$. This allows to conclude. □

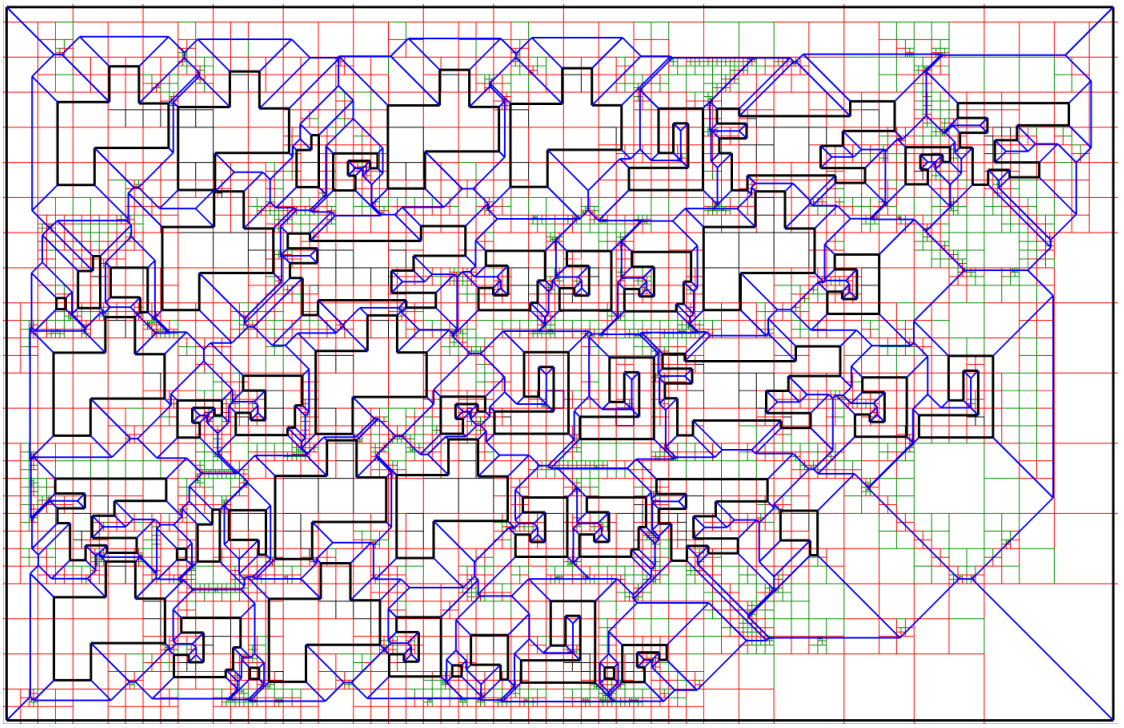
Remark 4.1. $t_\alpha = O(\alpha)$ *so the bound of Theorem 4.3 is* $O(n\alpha^2/\Delta^2 + 1/\Delta^3)$. *Let* V *be the number of input vertices. It is expected that* $n\alpha = O(V)$*; also* α *is usually constant. In this case, the complexity simplifies to* $O(V/\Delta^2 + 1/\Delta^3)$.

5. IMPLEMENTATION AND EXPERIMENTAL RESULTS

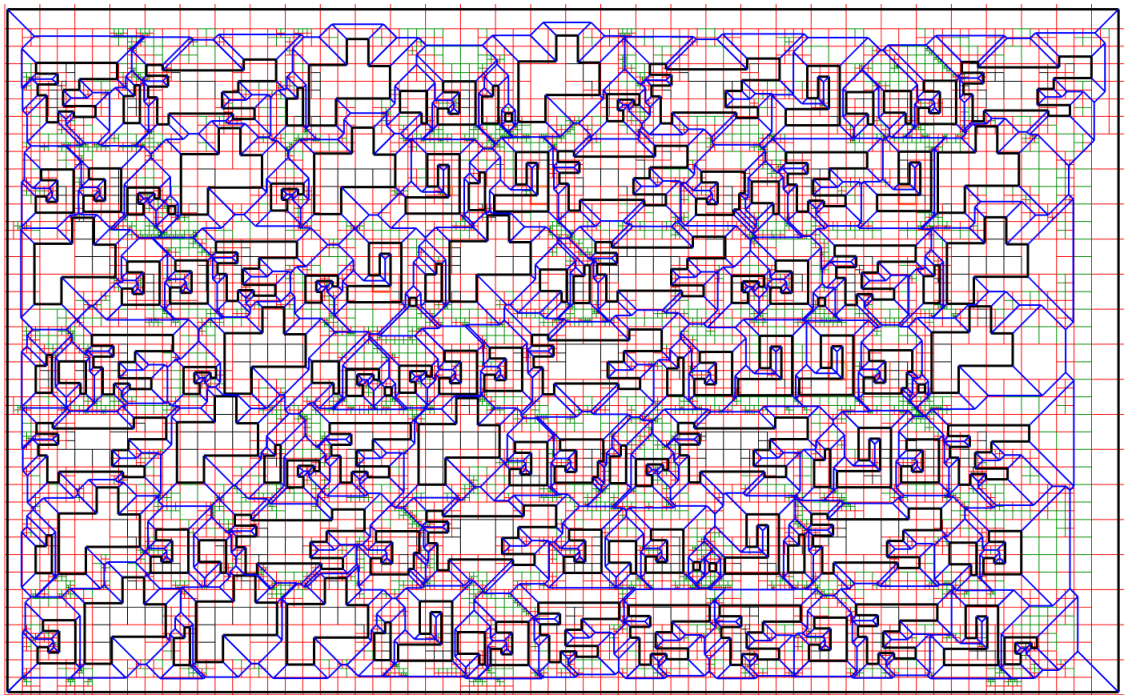
Our algorithms are implemented in Julia and are available in [`https://gitlab.inria.fr/ckatsama/L_infinity_Voronoi/`]. We use the algebraic geometric modeler `Axl` for visualization. All experiments conducted for the purpose of this thesis were run on a 64-bit machine with an Intel(R) Core(TM) i7-8550U CPU @1.80GHz and 8.00 GB of RAM.

We tested the runtimes of the *2D version* of the algorithm for 4000 input polygons, that consist of up to approximately 2500 sites. The input is generated by using a sample of 10 different rectilinear polygons as holes inside a bounding rectangle. We use BigFloats only, which suffices for the numerical correctness of our demos. Some large-scale examples are shown in Figure 18.

As it can be seen in the diagrams of Figures 19,20 the experimental evaluation of runtimes indicates a linear behaviour of the subdivision phase with respect to the number of sites; the Pearson correlation coefficient is 0.97. Theoretically, the complexity of the reconstruction phase is linear in the tree size and the Pearson correlation coefficient of the reconstruction times and the number of leaf cells is 0.93. We anticipate that the magnitude of the coefficient would be increased with a more careful implementation and by using a bigger sample.

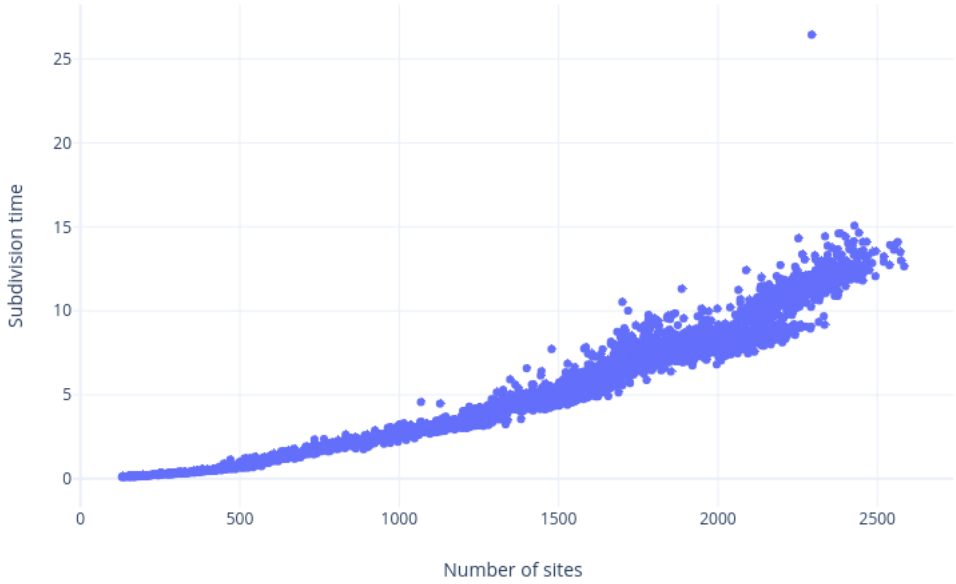


(a)

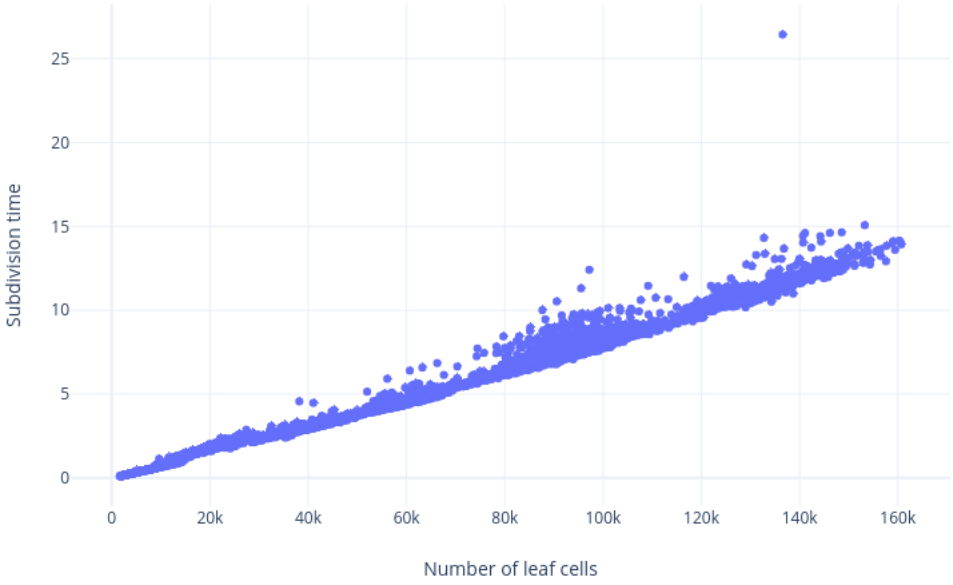


(b)

Figure 18: (a) Input consists of 538 sites and 7468 cells are generated. Subdivision and reconstruction runtimes are 495ms and 363ms respectively. (b) Input consists of 1308 sites and 17044 cells are generated. Subdivision and reconstruction runtimes are 1156ms and 1179ms respectively.

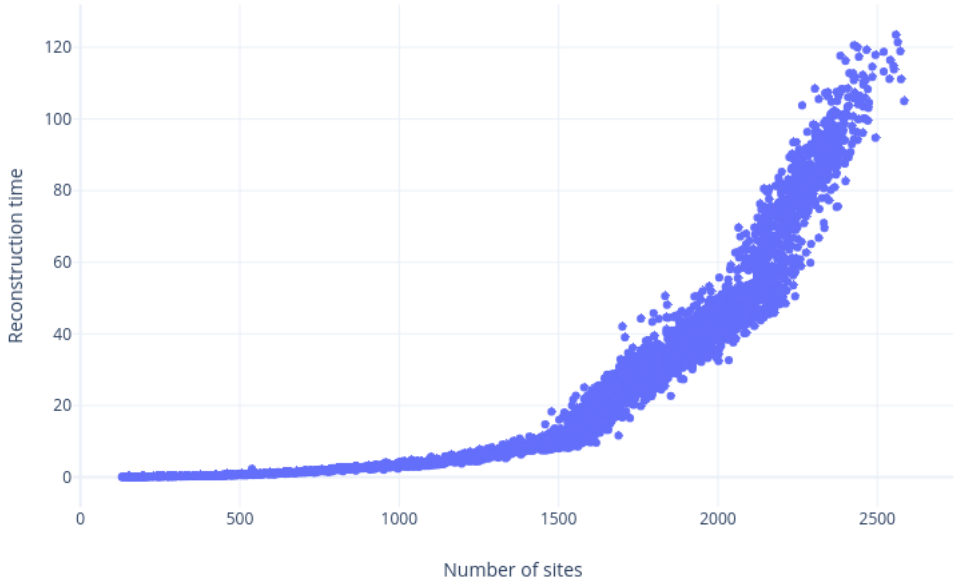


(a)

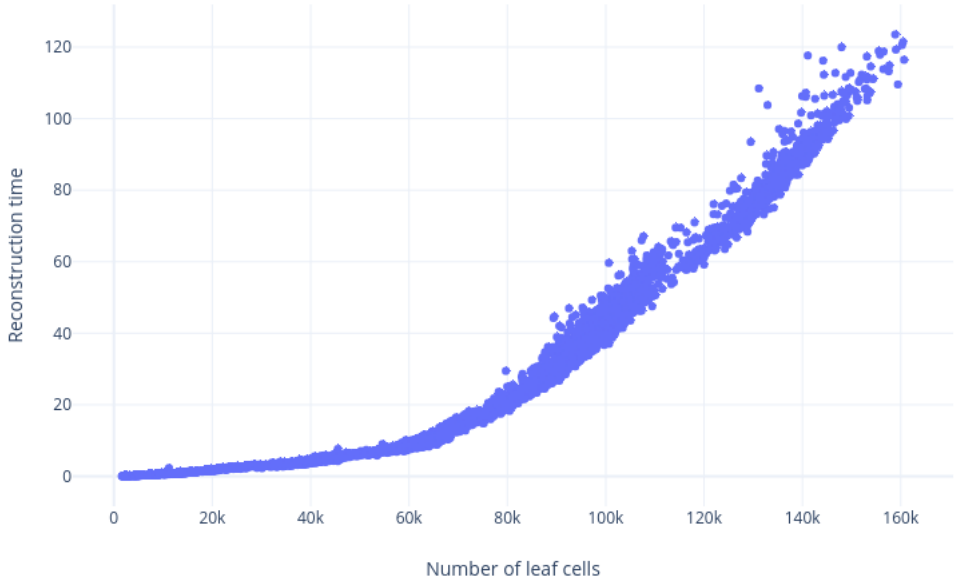


(b)

Figure 19: The abscissa denotes the number of (a) edges of $\mathcal{P}$ (sites), (b) leaf cells of the quadtree. The ordinate denotes the running time in seconds of the subdivision phase. Every blue point depicts the runtime for a single dataset.



(a)



(b)

Figure 20: The abscissa denotes the number of (a) edges of $\mathcal{P}$ (sites), (b) leaf cells of the quadtree. The ordinate denotes the running time in seconds of the reconstruction phase. Every blue point depicts the runtime for a single dataset.

6. CONCLUSION AND FUTURE WORK

We have provided algorithms for the Voronoi diagram computation of othogonal polyhedra in 2D and 3D and under the L_∞ norm. Our algorithms follow the subdivision paradigm which is an original approach to the problem. The 2D and 3D versions of the algorithm share the basic ideas. However, for efficiency reasons, extending the 2D version in three dimensions is not done in a straightforward manner; we build an hierarchical data structure of bounding volumes for every site (facet) of the input polyhedron and formulate efficient predicates relying on it.

Apart from our theoretical results we have provided a numerically stable implementation of our algorithms in Julia. It is open-source, so that other researchers will benefit from it and use it for their experimental studies. We are assured that our results will attract the interest of researchers working on computational geometry and computer aided geometric design.

Complexity analysis of adaptive subdivision algorithms is non trivial and remains largely undeveloped. Towards this difficult task, we provided complexity bounds that are output-sensitive and are comparable to that of previous methods, since linear with respect to the number of facets. However, our bounds rely on a hypothesis of ‘‘sufficiently uniform” input. Even though the Uniform Distribution Hypothesis might seem a strong assumption, note that our analysis does not encounter the width of the subdivision tree and considers it as a complete tree. This does not exploit to the fullest the local nature of subdivision algorithms; they perform additional subdivisions only near difficult features of the input.

We anticipate that in instances where our hypothesis does not hold, the ‘‘non-uniform” distribution of sites causes sudden increases or decreases in the width of the tree passing from one level of the tree to another. We expect that there is a trade-off between the number of cells at level i of the subdivision tree variating from 4^i and the cardinality of the active set of a cell being independent of the current level, so that the total complexity remains linear in the number of sites (facets). We plan to extend our research towards proving non trivial complexity bounds that exploit to the maximum degree the adaptivity of subdivision algorithms and do not rely on assumptions.

BIBLIOGRAPHY

[1] Pankaj K. Agarwal, Mark de Berg, Joachim Gudmundsson, Mikael Hammar, and Herman J. Haverkort. Box-trees and R-trees with near-optimal query time. In *Proc. Symposium on Computational Geometry*, 2001.

[2] Oswin Aichholzer, Franz Aurenhammer, David Alberts, and Bernd Gärtner. A novel type of skeleton for polygons. *J. Univ. Comp. Science (Springer)*, 1:752–761, Jan. 1995.

[3] Franz Aurenhammer and Gernot Walzl. Straight skeletons and mitered offsets of nonconvex polytopes. *Discrete & Computational Geometry*, 56(3):743–801, Oct 2016.

[4] Gill Barequet, David Eppstein, Michael Goodrich, and Amir Vaxman. Straight skeletons of three-dimensional polyhedra. 05 2008.

[5] H. Bennett, E. Papadopoulou, and C. Yap. Planar minimization diagrams via subdivision with applications to anisotropic Voronoi diagrams. *Computer Graphics Forum*, 35, 08 2016.

[6] Panagiotis Cheilaris, Sandeep Kumar Dey, Maria Gabrani, and Evanthia Papadopoulou. Implementing the l_∞ segment Voronoi diagram in CGAL and applying in VLSI pattern analysis. In Hoon Hong and Chee Yap, editors, *Proc. Math. Soft: ICMS*, pages 198–205. Springer, 2014.

[7] Günther Eder, Martin Held, and Peter Palfrader. Computing the straight skeleton of an orthogonal monotone polygon in linear time. In *Europ. Workshop Comput. Geom., Utrecht*, 03 2019.

[8] Ioannis Z. Emiris, Angelos Mantzaflaris, and Bernard Mourrain. Voronoi diagrams of algebraic distance fields. *J. Comp.-Aided Design*, 45(2):511–516, 2013. Symp. Solid Physical Modeling 2012.

[9] David Eppstein and Jeff Erickson. Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. *Discrete & Comput. Geometry*, 22:58–67, 1998.

[10] Michal Etzion and Ari Rappoport. Computing Voronoi skeletons of a 3-d polyhedron by space subdivision. *Comput. Geometry: Theory & Appl.*, 21(3):87–120, January 2002.

[11] Herman Johannes Haverkort. *Results on geometric networks and data structures*. PhD thesis, Utrecht University, 2004.

[12] Martin Held and Stefan Huber. Topology-oriented incremental computation of Voronoi diagrams of circular arcs and straight line segments. *J. Computer-Aided Design*, 41:327–338, 05 2009.

[13] Martin Held and Peter Palfrader. Straight skeletons and mitered offsets of polyhedral terrains in 3D. *J. Comp.-Aided Design & Applications*, 16:611–619, 11 2018.

[14] Vladlen Koltun and Micha Sharir. Polyhedral voronoi diagrams of polyhedra in three dimensions. In *Proc. Symp. Computational Geometry*, pages 227–236, New York, 2002. ACM.

[15] Witold Lipski, Jr. An o(n log n) manhattan path algorithm. *Inf. Process. Lett.*, 19(2):99–102, September 1984.

[16] Jonàs Martínez, Núria Pla Garcia, and Marc Vigo Anglada. Skeletal representations of orthogonal shapes. *Graphical Models*, 75(4):189–207, 2013.

[17] Victor Milenkovic. Robust construction of the voronoi diagram of a polyhedron. In *In Proc. 5th Canad. Conf. Comput. Geom*, pages 473–478, 1993.

[18] Evanthia Papadopoulou and D.T. Lee. The L_∞ Voronoi diagram of segments and VLSI applications. *Intern. J. Comput. Geom. & Applications*, 11(05):503–528, 2001.

[19] Scott Schaefer and Joe Warren. Dual marching cubes: Primal contouring of dual grids. *Computer Graphics Forum*, 24, 2005.

[20] Valeriu Soltan and Alexei Gorpinevich. Minimum dissection of a rectilinear polygon with arbitrary holes into rectangles. *Discrete & Computational Geometry*, 9(1):57–79, Jan 1993.

[21] Vijay Srinivasan and Lee R. Nackman. Voronoi diagram for multiply-connected polygonal domains I: Algorithm. *IBM J. Research & Development*, 31:361–372, May 1987.

[22] Chee K. Yap, Vikram Sharma, and Lien Jyh-Ming. Towards exact numerical Voronoi diagrams. In *IEEE Intern. Symp. Voronoi Diagrams in Science and Engineering (ISVD)*, New Brunswick, NJ, June 2012.