Towards Exact Numerical Voronoi Diagrams (Invited Talk)

Chee K. Yap
Department of Computer Science
Courant Institute, NYU
New York, NY 10012, USA
yap@cs.nyu.edu

Vikram Sharma
Institute of Mathematical Sciences
Chennai, India, 600113
vikram@imsc.res.in

Jyh-Ming Lien
Department of Computer Science
George Mason University
Fairfax, VA 22030, USA
jmlien@cs.gmu.edu

***Abstract*—Voronoi diagrams are extremely versatile as a data structure for many geometric applications. Computing this diagram "exactly" for a polyhedral set in 3-D has been a quest of computational geometers for over two decades; this quest is still unrealized. We will locate the difficulty in this quest, thanks to a recent result of Everett et al (2009). More generally, it points to the need for alternative computational models, and other notions of exactness.**

In this paper, we consider an alternative approach based on the well-known Subdivision Paradigm. A brief review of such algorithms for Voronoi diagrams is given. Our unique emphasis is the use of purely numerical primitives. We avoid exact (algebraic) primitives because (1) they are hard to implement correctly, and (2) they fail to take full advantage of the resolution-limited properties of subdivision. We encapsulate our numerical approach using the concept of soft primitives that conservatively converge to the exact ones in the limit.

We illustrate our approach by designing the first purely numerical algorithm for the Voronoi complex of a non-degenerate polygonal set. We also discuss the critical role of filters in such algorithms. A preliminary version of our algorithm has been implemented.

***Keywords*-soft predicates; subdivision; filters**

I. What is so hard about the Voronoi Diagram of Polyhedra?

The concept of Voronoi diagrams is ubiquitous [26], with many applications. It is also a core topic in computational geometry [6], [14]. For instance, one application is in robotics where such diagrams are the basis of the retraction approach to motion planning [25], [36], [13], [32]. Much is known about Voronoi diagrams and its many generalizations. The books [26], [18], [6] focus on the planar cases. The Voronoi diagram of a point set in any dimension is well-understood. But in 3-D, already the Voronoi diagram of a set of polyhedral objects is a barrier. For more than two decades, computational geometers have been interested in an algorithm for such diagrams (e.g., [23]). For reference, call this particular problem (Voronoi diagram for polyhedral objects in 3-D) the "Voronoi Quest". This quest remains unfulfilled today. Recently, Hemmer et al. [15] announced a major milestone in this quest: they provided an algorithm for a special case, the Voronoi diagram for a collection of infinite lines. It has been implemented in CGAL.

¶1. Three Views of a Voronoi Diagram. It may sound surprising that this basic problem is still open. What are the barriers in this quest? There are three views of the "Voronoi diagram" of a polyhedral set $\Omega \subseteq \mathbb{R}^d$: a set-theoretic view, an algebraic topology view, and a combinatorial view. We use the notations

$$\overline{\mathrm{V}or}(\Omega), \quad \mathrm{V}or(\Omega), \quad \mathrm{V}or^*(\Omega)$$

to distinguish them. Briefly, $\overline{\mathrm{V}or}(\Omega)$ is just a subset of $\mathbb{R}^d \setminus \Omega$ called the **geometric Voronoi diagram**; the points p in $\overline{\mathrm{V}or}(\Omega)$ are those whose Euclidean distance to the closest point $q \in \Omega$ is achieved by two or more q's. The **Voronoi complex** $\mathrm{V}or(\Omega)$ is a cell complex in the sense of algebraic topology; each cell in $\mathrm{V}or(\Omega)$ is a subset of $\mathbb{R}^d$ that is homeomorphic to an open Euclidean ball of some dimension $i = 0, \ldots, d-1$. The support of $\mathrm{V}or(\Omega)$ is just $\overline{\mathrm{V}or}(\Omega)$. Finally, the **Voronoi graph** $\mathrm{V}or^*(\Omega)$ is a labeled combinatorial graph representing the Voronoi complex: the vertices of $\mathrm{V}or^*(\Omega)$ are in 1-1 correspondence with cells in $\mathrm{V}or(\Omega)$, and the graph edges correspond to adjacency relation between pairs of cells. A cell of dimension i in $\mathrm{V}or(\Omega)$ is also called a **Voronoi cell** or an i**-cell**; when $i = 0$ (resp., $i = 1, 2$) it is known as a **Voronoi vertex** (resp., **curve, surface**). Below, we provide a more detailed account of these concepts.

These three views give rise to (at least) three interpretations of what it means to "compute a Voronoi diagram":

- Perhaps we may call the computation of $\mathrm{V}or^*(\Omega)$ the standard Computational Geometry view. There is no explicit manipulation of numerical data, and the computation can be carried out by postulating certain abstract "geometric" operations (see [38]).
- In visualization and computer graphics, the problem amounts to computing an ε-approximation of $\overline{\mathrm{V}or}(\Omega)$. Here, a set $\widetilde{S} \subseteq \mathbb{R}^d$ is an ε-approximation of another $S \subseteq \mathbb{R}^d$ if the Hausdorff distance

$d_H(\widetilde{S}, S)$ is at most ε; the approximating set $\widetilde{S}$ might be a collection of boxes (voxels) or be some piecewise linear representation.

- In computational semi-algebraic geometry, the problem amounts to computing $\mathrm{V}or(\Omega)$. Since the cells in $\mathrm{V}or(\Omega)$ are semi-algebraic sets, their representation is a nontrivial issue. In one interpretation, each cell may be represented symbolically or implicitly by algebraic data (e.g., a set of algebraic inequalities). Another interpretation is to compute some ε-approximation of each cell (as in the visualization view). Although cell are approximated, we require these approximations to induce *exact* combinatorial information like adjacencies and isotopy-type. The result is called an ε-**approximation** of $\mathrm{V}or(\Omega)$.

Clearly the third view point is the most demanding: an ε-approximation of $\mathrm{V}or(\Omega)$ in the above sense subsumes the information for $\overline{\mathrm{V}or}(\Omega)$ and $\mathrm{V}or^*(\Omega)$. In this paper, we adopt the third viewpoint as our computational goal.

¶2. Types of Voronoi Cells. The cell complex $\mathrm{V}or(\Omega)$ has variant definitions in the literature, so we will be specific: first partition the boundary of Ω into a simplicial complex called the **boundary complex** of Ω, denoted $\Phi(\Omega)$. This is collection of simplices of each dimension $0, 1, \ldots, d-1$; cells in $\Phi(\Omega)$ are called (boundary) **features** of Ω. Our main concern is $d = 3$, where features of dimensions $0, 1, 2$ (respectively) are classified as **corners** (c), **edges** (e) and **walls** (w). In general, we classify features into "types" based solely on their dimension. Thus we have d types. In 3-space, these types may be labeled c, e, w. Given a set $T \subseteq \Phi(\Omega)$ of size $d-k+1$ ($k = 0, \ldots, d-1$), we can consider the semi-algebraic set $S(T) \subseteq \mathbb{R}^d \setminus \Omega$ comprising those points q that are simultaneously closest to each of the features in T, and such that no other feature in $\Phi(\Omega)$ is strictly closer to q. We define a k-cell of $\mathrm{V}or(\Omega)$ to be a connected component[1] of such a semi-algebraic set $S(T)$. We call T a **generator set** for cell. In general, the generator set for a k-cell will have at least $d-k+1$ features; if it could have more than $d-k+1$ features, we say the k-cell is **degenerate**. Assuming non-degeneracy, the generator set is unique. We say Ω is **non-degenerate** if none of its cells are degenerate. We will assume non-degeneracy in the current discussion. The **type** of T is just the multiset of $d-k+1$ types represented by the features in T. E.g., in $\mathbb{R}^3$, if $T = \{c_1, c_2, w\}$ has two corners and a wall, then the set $S(T)$ (if non-empty) will be a portion of a curve in space, and its connected components would be 1-cells of type $\{c, c, w\}$ (or simply ccw). Thus in 3-space, we have 10 types of Voronoi curves (i.e., 1-cells), namely

$$ccc, cce, ccw, cee, cew, cww, eee, eew, eww, www \quad (1)$$

In $\mathbb{R}^d$, the types of k-cells may be identified with the monic monomials[2] of degree $d-k+1$ over d variables. The number of such types is

$$\binom{2d-k}{d-k+1} = \binom{2d-k}{d-1}. \quad (2)$$

For instance, there are $\binom{2d-1}{d-1}$ types of Voronoi curves in d-space. For $d = 4$, there are 35 types of Voronoi curves. Any code for the exact Voronoi diagram must compute Voronoi cells of each type, and each type must handle $\binom{2k-k}{d-1}$ cases of the code.

¶3. On the Subtypes of Type eee. In 3-space, the type eee in (1) is recognized as the most demanding case. Solving the eee case amounts to determining the Voronoi curve determined by three infinite lines. The classification of such curves was only recently determined by Everett, Gillot, Lazard, Lazard and Pouget [12], [11]:

PROPOSITION 1 (Everett et al. (2009)).
The Voronoi curve determined by three lines in space are connected components of these curves:
(i) a non-singular quartic if the 3 lines are pairwise skew but not all parallel to a common plane nor on the surface of a hyperboloid of revolution;
(ii) a cubic and a line if the 3 lines are pairwise skew and lies on the surface of a hyperboloid of revolution;
(iii) a nodal quartic if the 3 lines are pairwise skew and all parallel to a common plane;
(iv) one or two parabolas or hyperbolas if there is exactly one pair of coplanar lines;
(v) Between 0 and 4 lines if there are two pairs of coplanar lines.

This theorem is the basis of the algorithm of Hemmer et al. [15]. According to this classification, the case eee gives rise to 5 **subcases**. In general, each type will give rise to exponentially many cases, and each case may give rise to further subcases. Unlike the growth rate in the number of types (see (2)), the growth rate of the number of subcases is less understood. The determination of these subcases may be quite non-trivial as illustrated by Proposition 1. This issue already shows up in the plane: see the classification of subcases in the exact algorithm for Voronoi diagrams of circles ("Apollonius diagrams") [8] and of ellipses [9].

It should now be clear that any exact algorithm for such Voronoi diagrams is a major undertaking, just

[1] It can be shown that each connected component is homeomorphic to an open ball of some dimension.

[2] i.e., power products of (commutative) variables.

to completely enumerate all possibilities. One answer to the question posed in the title of this section can now be given: the difficulty in the Voronoi Quest lies in providing the analogue of Proposition 1 for the other nine cases of (1). Of course, the good news is that the hardest case has been cracked. Such analysis requires some non-trivial algebraic geometry. So is computational geometry destined to become a subfield of algebraic geometry? The main good news of this paper is that there is a way out. We will return to this in the next section.

¶4. On Generic Voronoi Diagrams. Proposition 1 can be mined for other insights. The first is that if we try to design Voronoi diagram algorithm using some "abstract computational model" (as is common in Computational Geometry), the subcases of Proposition 1 will never be revealed. It is another example of the pitfalls of abstract computational models discussed in [38].

Next, Proposition 1 contains an outline for implementing the primitives for constructing the Voronoi cells determined by 3 lines. To detect case (i), we need three predicates to check the following conditions:

- (a) if two lines are skew;
- (b) if three lines are parallel to a common plane;
- (c) if three lines lie on the surface of a hyperboloid of revolution.

Using these predicates, we can then take the appropriate branch into the 5 subcases. The efficient evaluation of such predicates is a central theme of "Exact Geometry Computation" (EGC) since the 1990s. Methods for their correct and automatic evaluation is today available in libraries such as [22], [5], [39]. Nevertheless, the combinatorial explosion in the number of cases remains a serious barrier to implementations.

A partial remedy is possible if we could tolerate infinitesimal perturbations of the input [7], [37]. Such perturbations ensures that you only need to handle the "generic inputs" i.e., those that do not satisfy any algebraic conditions (such as (a), (b), (c) above). Algorithms that only handle generic cases are called **generic algorithms**. Let us understand what symbolic perturbation achieves in the current setting: According to Proposition 1, infinitesimal perturbation reduces the subcases $(i) - (v)$ to just subcase (i). More generally, *symbolic perturbation amounts to getting rid of subcases.* This leads to the following corollary:

Corollary 2. *The "generic" algorithm for computing the exact Voronoi diagram of a set of polyhedral objects in d-space has exactly $\binom{2d+1}{d} - (d+1)$ cases.*

Proof. There are $\binom{2d-k}{d-1}$ cases for Voronoi cells of dimension k. Summing over all dimensions, the number of cases is $\sum_{k=0}^{d-1} \binom{2d-k}{d-1}$. But since $\sum_{k=0}^{d+1} \binom{2d-k}{d-1} = \binom{2d+1}{d}$, the corollary follows. **Q.E.D.**

Thus, the technique of symbolic perturbation does not remove the exponential dependence on d. For low dimensions, perturbation alone might be enough to render exact algorithms practical. For instance, in the Voronoi diagrams of polyhedral sets the number of cases to handle are $\binom{7}{3} - 4 = 31$ for $d = 3$ and $\binom{9}{4} - 5 = 121$ for $d = 4$.

Interestingly, there is a paradox in the application of symbolic perturbation. By avoiding the subcases, we also opted to give up advantages conferred by degeneracies. For example, suppose our input is a set of horizontal rectangles whose edges are parallel to the x- or y-directions. Then according to Proposition 1, the only Voronoi curves we need to handle falls under subcase (iv); these are parabolas and hyperbolas. But by opting for the generic situation, we cannot take advantage of this "degenerate" situation. Such observations have prompted Burnikel et al. [3] to argue that the explicit handling of degeneracies (subcases) is preferable to symbolic perturbation. In view of our discussions of the Voronoi quest, their argument is no longer persuasive for dimensions 3 or above. Indeed, we do not even know what these subcases are! On the other hand, it seems quite feasible to develop an exact but generic algorithm for our Voronoi quest.

¶5. What is new in this paper?

- In view of the long history of subdivision approaches, it may be hard to imagine anything fundamentally new except for "yet another subdivision algorithm". The key observation is that the primitives (i.e., predicates and constructors) used in subdivision algorithms can be classified as *exact* and or *(purely) numerical*. Previous subdivision algorithms for computing the Voronoi complex $\mathrm{V}or(\Omega)$ inevitably resort to some exact primitives. Besides causing an implementation gap, exact primitives do not fully exploit the power of subdivision, namely, the ability to produce **resolution-limited** outputs. We note that the use of exact predicates is unavoidable (short of using zero bounds as in EGC) *unless we weaken the notion of exactness.* Thus we advocate a paradigm-shift for subdivision algorithms, away from traditional notions of exactness, toward concepts of resolution exactness that can be achieved using purely numerical primitives (e.g., [33], [35]).
- We develop a new subdivision algorithm for the Voronoi complex of a polygonal set. It is the first complete solution based on purely numerical primitives. There is a price for purely numerical primitives: *termination is guaranteed on assuming*

non-degeneracy of the Voronoi complex. In our implementations, we use an δ-cutoff. Upon cutoff, one can invoke a weakened correctness criterion or resort to exact computation which is relatively easy in the planar setting. These aspects will be elaborated in a future paper.

- One general contribution is to introduce the idea of **soft primitives** to this domain. One could say this idea is implicit in some prior work, but to our knowledge it has never been made explicit. We believe the full exploitation of soft primitives will open up new classes of practical algorithms in computational geometry. There are two main goals in the design of soft predicates (1) simplicity of implementation and (2) effectiveness. To achieve (1), we try to reduce all predicate to computing and comparing separations between pairs of features. Such computations are very robust (see [35]) and easy to approximate to high precision. Goal (2) is achieved via **filters** that provide conditions that are sufficient (but not necessary) for a predicate to hold or to fail. Filters should be relatively cheap compared to its efficacy.

¶6. Overview of paper. In Section 2, we discuss an alternative computational model based on the Subdivision Approach. Section 3 contains a review of some key subdivision papers for Voronoi diagrams. We attempt to provide a unifying framework for this literature. In Section 4, we introduce the basic tools for our approach, in particular, the notion of soft primitives. Section 5 describes our new subdivision algorithm for the Voronoi complex of a polygonal set. Section 6 discuss our implementation and address the issue of filters. We conclude in Section 7.

II. Resolution-Bounded Solutions

There is a sense in which our Voronoi diagram quest is a solved problem: it can be reduced to a key problem in computational semi-algebraic geometry [1]: given a set Σ of polynomial equations in d variables, we can compute a cell complex K of $\mathbb{R}^d$ such that Σ is sign-invariant over each cell $c \in K$. Moreover, we can also determine the adjacencies among these cells. It is not hard to define such a set $\Sigma = \Sigma(\Omega)$ so that Voronoi diagram $\mathrm{V}or(\Omega)$ can be partitioned naturally into a union of such cells in K. A simple definition of Σ to define, for each pair $f, f' \in \Phi(\Omega)$ of features, a polynomial of the form $Dist(q, f) - Dist(q, f')$ where $Dist(q, f)$ is the squared distance from a variable point q to the affine span of f. Thus Σ is a system of polynomials in the d coordinate variables of q. After a post-processing of K, we obtain an exact representation of $\mathrm{V}or(\Omega)$. The cells in K are directly related to the combinatorial structure of the polyhedral set Ω, and the size of K can be bounded as a function of the combinatorial complexity of Ω. An algorithm for cell decomposition goes back to Tarski's procedure (1951) for deciding sentences in the first-order theory of reals, and to Collin's algorithm for Cylindrical Algebraic Decomposition (1971). Recent progress on this and related problems may be found in [1]. Unfortunately, the reduction of Voronoi diagrams to cell decomposition would be totally impractical. Thus, our Voronoi quest has just acquired a new requirement: we seek not just "any solution" but an efficient and practical one.

But for this, we must turn to a different computational model. Computational Geometers normally assume a Real RAM model augmented with suitable (problem specific) primitives. These primitives, if implemented, would be reduced to semi-algebraic computation as above. But Voronoi diagrams can also be computed using a more explicitly numerical approach. Perhaps the common form of this approach is based on the idea of (spatial) subdivision. These are the familiar quadtrees (2-D) or octrees (3-D); we call them **subdivision trees** in general. Figure 1 is a screen shot from our subdivision algorithm's output.

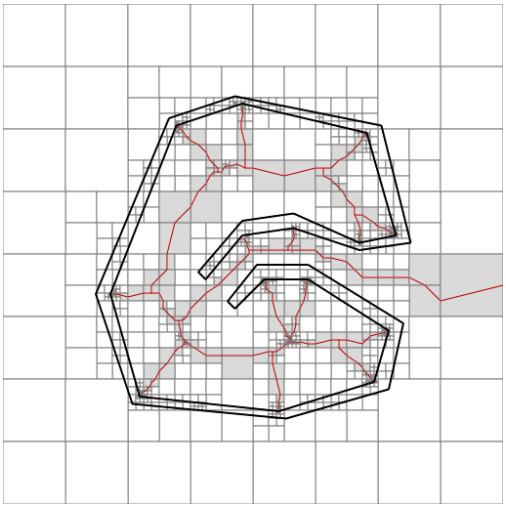


Figure 1. Subdivision output: Voronoi complex for Bugtrap Polygon

In brief, the subdivision method use an iterative process that subdivides an initial region B_0 in $\mathbb{R}^d$ into simple regions. For our purposes, we may assume these regions are d-dimensional boxes. Once these cells are sufficiently "simple" relative to Ω, the subdivision process stops. Finally, we synthesize an approximation to the set $\overline{\mathrm{V}or}(\Omega)$ from these simple boxes. To determine the correct combinatorial/topological nature of the Voronoi complex $\mathrm{V}or(\Omega)$ is much harder, especially if exact predicates are disallowed. That is the main challenge of this paper.

Thus we have two fundamentally different approaches

to computing Voronoi diagrams. The **Exact Method** and the **Subdivision Method**. The complexity of Exact Method is usually a function of the combinatorial size of the input, i.e., the number n of features of $\Phi(\Omega)$. The number of Voronoi cells is polynomial in n; the best upper bound for this size in 3-D is $O(n^{2+\varepsilon})$ [19]. In terms of abstract operation counts (but not bit complexity, the computational complexity of exact algorithms can achieve such limits. In contrast, the boxes in Subdivision Methods are not directly related to n but to the "resolution" of the geometric representation. Typically, bound the number of boxes as a function of n *and* the bit-size of the representation of Ω.

In any Subdivision Method, there ultimately must be some predicate to control the termination for subdivision, and predicates to confirm the presence of a Voronoi vertex within a box. Generally speaking, previous algorithms use a combination of exact and numerical predicates for this purpose. Synthesizing an approximate $\overline{\mathrm{V}or}(\Omega)$ is an easier problem for which purely numerical predicates are known. But to synthesize the Voronoi complex, all previous methods ultimately rely on some exact predicate. For instance, to confirm the presence of a Voronoi vertex within a given box, some authors directly compute the Voronoi vertex, while others infer this by computing the exact Voronoi points on the box boundary. Our goal is to avoid all such exact computations.

III. Literature Review

In this section, we review several key papers on subdivision algorithms for Voronoi diagrams: Lavender, Bowyer, Davenport, Wallis and Woodward (1992) Vleugels and Overmars (1995), Teichmann and Teller (1998), Etzion and Rappoport (2002), Sud, Zhang and Manocha, (2007), Boada, Coll, Madern and Sellarés (2008), and Stolpner, Whitesides and Siddiqi (2011) In our assessment, only Etzion and Rappoport, and Sud, Zhang and Manocha provide algorithms for computing the Voronoi complex, while the other algorithms mainly compute the geometric Voronoi diagram.

We first establish a common terminology for these subdivision methods, basically following [21]. A **box** (or d-box) B is a subset of $\mathbb{R}^d$ ($d \geq 1$) of the form $B = I_1 \times I_2 \times \cdots \times I_d$ where $I_j = [a_j, b_j]$, $a_j < b_j$, are closed non-degenerate intervals. The **center** of a box B is denoted $m(B)$ (or m_B). The box has 2^d **vertices** and $2d$ **faces**. Thus each face is a $(d-1)$-box. Its **radius** is the distance from $m(B)$ to any vertex, where the distance between any two points p and q is the Euclidean distance $\|p-q\|$. A collection $\mathcal{S}$ of boxes is called a **subdivision** if the boxes in $\mathcal{S}$ have pairwise disjoint interiors; we say $\mathcal{S}$ is a **subdivision of** the set $\cup\mathcal{S}$ (union of the boxes in $\mathcal{S}$). For instance, if B is a box, then there is a unique subdivision of B comprising 2^d congruent boxes. This subdivision is called the **full split** of B. A **subdivision tree** $\mathcal{T}$ rooted at a box B_0 is a finite tree whose tree nodes are boxes, and where each internal node B has 2^d congruent children that form the full split of B. Thus the set of leaves of $\mathcal{T}$ forms a subdivision of B_0. Two boxes B, B' are **adjacent** if their interiors are disjoint and $B \cap B'$ is a $(d-1)$-box. A subdivision $\mathcal{S}$ is said to be **balanced** if for every pair of adjacent boxes in $\mathcal{S}$, they either has the same radius or one has twice the radius of the other. In our development, boxes are assumed to be squares ($d = 2$) or cubes ($d = 3$).

¶7. The Standard Subdivision Framework. To help unify the literature, we borrow the "standard" framework[3] for describing subdivision algorithms from Lin and Yap [21], where it was used for isotopic approximation of surfaces. Input consists of a set Ω of geometric objects, a box B_0 containing the region of interest, an $\epsilon > 0$, and a cutoff parameter δ. Most subdivision algorithms for computing Voronoi diagram may be viewed as having three phases: Subdivision, Refinement and Construction.

The Subdivision Phase constructs a subdivision tree $\mathcal{T}$ rooted at B_0 and expands leaves until some predicate $C(B)$ holds at each leaf B. While subdividing a box B, we also propagate certain information to each of its children.

The Refinement Phase further refines $\mathcal{T}$, typically by balancing the subdivision of B_0.

The Construction Phase takes the subdivision $\mathcal{S}$ of B_0 from the Refinement Phase. For each box $B \in \mathcal{S}$, the typical aim is to find the topology of the Voronoi diagram inside the box. This involves detecting Voronoi walls, edges and vertices in a box. The output of this phase may be a collection of boxes (that form an ε-cover of the geometric Voronoi diagram $\overline{\mathrm{V}or}(\Omega)$ or an ε-approximation of the Voronoi complex $\mathrm{V}or(\Omega)$.

STANDARD SUBDIVISION FRAMEWORK
Input: A set Ω of objects, $\epsilon > 0$ and B_0.
Output: An ϵ-approximation G to $\mathrm{V}or(\Omega)$.
0. Let $Q_{in} \leftarrow \{B_0\}$ be a queue of boxes.
1. $Q_{out} \leftarrow SUBDIVIDE(Q_{in})$.
2. $Q_{ref} \leftarrow REFINE(Q_{out})$.
3. $G \leftarrow CONSTRUCT(Q_{ref})$.

We next review the literature using this framework. Many, but not all, of these algorithms can be put into this standard framework. For instance, the Refinement and Construction Phases are sometimes merged into one

[3] Actually [21] calls this the "Generic Subdivision Framework" but we call it "Standard" to avoid confusing this with the earlier discussion of "Generic Algorithms" based on infinitesimal input perturbation.

computation.

¶8. Lavender et al. [20]

This paper claims credit as the first paper to provide a subdivision algorithm for Voronoi diagrams. [4] The objects in Ω are semi-algebraic sets [1]. To reduce the complexity of estimating distances to Ω, they subdivide the objects, in addition to the usual spatial subdivision. Thus, this can be viewed as a generalization of our standard framework. It has no Refinement Phase.

Subdivision Phase: This phase consists of two subphases: Object division phase and a Voronoi division phase. In the former phase, B_0 is subdivided to obtain a sequences of boxes covering the boundary of each $f \in \Omega$. The Voronoi division phases uses this covering of f to estimate the distance from a point p to f. The distance of any point to an object is thus an interval; if two such intervals are disjoint then we know that p is closer to one object than other. We label every box B with a set of objects $\widetilde{\phi}(B)$ as follows: compute the distance of B to the objects in Ω; consider the interval that has the smallest lower and upper bound; $\widetilde{\phi}(B)$ is the set of all objects whose lower bound is in this interval. The termination criteria are either $|\widetilde{\phi}(B)| = 1$, or the size of B is smaller than δ. If both tests fail, then subdivide B. The $\widetilde{\phi}$-set of the children of B is a subset of $\widetilde{\phi}(B)$.

Construction Phase: The topology of the Voronoi diagram inside each leaf box in the Voronoi division phase is detected by solving a corresponding system of multivariate polynomial equations. For solving these system of equations, Newton-Raphson method is used.

The algorithm computes $\overline{Vor}(\Omega)$. It does not provide any topological guarantees; in fact, the Newton-Raphson method can fail if the starting point for the iteration is not carefully chosen. Moreover, the use of a box-covering for representing objects implies that boxes far away from the actual Voronoi diagram may have more than one labels, since the distance computation deviates a lot from the actual distance. The algorithm does not handle degeneracies.

¶9. Vleugels and Overmars [32] The input Ω is an indexed set containing convex compact sets. They compute the distance function to the objects exactly; this avoids the box-covering approach of Lavender et al., and hence the distance between the computed Voronoi diagram and the actual Voronoi diagram is bounded.

Subdivision Phase: Instead of labeling the box, they label the vertices of B with a unique object from Ω; in case of ties, choose the object with the smallest index. Let $\lambda(B)$ be the set of labels assigned to the vertices. If $|\lambda(B)| > 1$ then the Voronoi diagram intersects B. The termination criteria are similar to Lavender et al.: either $|\lambda(B)| = 1$, or the size of B is smaller than δ. If both these fail, then subdivide B. The children of B inherit the labels of the vertices shared with B; for the remaining vertices, we have to compute their distance from Ω; this is unlike Lavender et al.

Refinement Phase: The algorithm balances the subdivision tree.

Construction Phase: There is no explicit construction of the topology of the Voronoi diagrams inside cells with more than one label and size smaller than δ.

There is no topological guarantee. The only guarantee is that if δ is small enough then the set of output boxes forms a connected component if the Voronoi diagram is connected. Their key observation is the following: there is a δ such that for a box B whose interior contains the Voronoi diagram, if $w(B) < \delta$ then either $|\lambda(B)| > 1$ or $|\lambda(B')| > 1$, for some box B' adjacent to B. No quantitative value for δ is given. The algorithm does not handle degeneracies.

¶10. Boada et al. [2] Their set Ω of objects contains arbitrary geometric objects in 2-D and 3-D. Similar to Vleugels and Overmars, they assume that the distance function for these objects can be computed exactly. Moreover, the metric underlying the distance function need not be euclidean. We describe their algorithm in 2-D. There is no refinement phase.

Subdivision Phase: Let B be a box B in a subdivision rooted at B_0. The objects close to B are categorized into three types: objects contained in B are called an $\mathcal{I}$-objects (internal objects); objects closest to a vertex of B than any other object are called a $\mathcal{V}$-object; objects that are not $\mathcal{V}$-objects and are close to some subset of an edge of B are called $\mathcal{B}$-objects (boundary objects). The termination criteria are either the union of the $\mathcal{I}$- objects, $\mathcal{V}$-objects and $\mathcal{B}$-objects of B is a singleton set, or the size of B is smaller than δ. The subdivision process always maintains a queue Q. While subdividing the $\mathcal{I}$- objects, $\mathcal{V}$-objects and $\mathcal{B}$-objects of B are appropriately distributed amongst its children. The crucial step in subdivision is checking the coherence between two adjacent boxes B, B': the $\mathcal{V}, \mathcal{B}$-objects of B and B' should match on the shared vertices or edges. If not then the larger box is subdivided. Note that a box that was terminal can become non-terminal in this step.

Construction Phase: Assuming no-degeneracies, the boxes in the partition of B_0 fall into 7 different patterns depending upon whether the number of V-objects of the box is $1, 2, 3$ or 4. Depending upon this number and the pattern of the V-objects, marching-cube type rules are given to obtain an approximation to the Voronoi diagram in each of the boxes; e.g., if there is an edge whose vertices have different V-objects then the Voronoi diagram must intersect this edge.

There is no topological guarantee. The algorithm

[4] Subdivision has been used earlier in the context of solid modeling, for instance [34].

does not work correctly if the Voronoi regions are disconnected (this is mainly governed by δ). However, they give a qualitative bound on how small δ should be to guarantee the connectedness of the Voronoi diagram. The same bound also implies that all boxes contained in the Voronoi region of an object will have their V-objects correctly labelled. The algorithm cannot handle degeneracies.

¶11. Teichmann and Teller [31] The set Ω consists of triangles in 3-D. Output is an approximation to $\overline{Vor}(\Omega)$. Unlike other approaches where the boxes are labelled while subdividing, their labeling is done after an initial subdivision phase. Also, they subdivide tetrahedrons rather than boxes.

This phase consists of three sub-phases: a covering phase, a propagation phase and a standard-subdivision phase.

- Covering phase. B_0 is subdivided until a covering of Ω is obtained such that each box B contains at most one triangle (assuming no two triangles are adjacent), and the six tetrahedrons that share the vertices of B and partition it do not contain more than one corner from Ω. Replace B by these six tetrahedrons; let T be such a tetrahedron.
- In propagation phase a label set $\phi(T) \subseteq \Omega$ is computed for each T. In this phase a conceptual wavefront emanates from each feature in Ω at time zero. The region where two wavefronts meet corresponds to the Voronoi diagram. The set $\phi(T)$ contains the input triangles whose wavefront meets T. The timeframe between which wavefronts enter and leave T is smaller than twice its radius. Also label the corners of T with the triangle nearest to it; let $\lambda(T)$ be this set.
- Standard-subdivision: Subdivide each T until either some predetermined number of labels remain (usually determined experimentally), or $|\lambda(T)| \leq |\phi(T)|$, or radius of $T < \delta$. While subdividing T, the label set ϕ of its children is a subset of $\phi(T)$, and is obtained by a proximity test.

Refinement Phase: T is subdivided by planes parallel to its sides and passing through the edge midpoints to get a balanced subdivision. Update $\phi(T)$ and $\lambda(T)$ during refinement.

Construction Phase: The Voronoi diagram is approximated inside each T depending upon the size of $\phi(T)$ and $\lambda(T)$. The canonical case is $|\lambda(T)| = |\phi(T)|$. So, e.g., if $|\phi(T)| = |\lambda(T)| = 2$ then compute the bisector of the two labels and intersect it with T; if $|\phi(T)| = |\lambda(T)| = 3$ then intersect the three bisectors and find the edge that intersect with T; similarly, find a Voronoi vertex inside T if $|\phi(T)| = |\lambda(T)| = 4$.

Their algorithm computes an approximation to $\overline{Vor}(\Omega)$. It cannot handle degeneracies, and there is no guarantee on the topology.

¶12. Etzion and Rappoport [10]: They compute the Voronoi diagram of a bounded 3-D polyhedron. They guarantee the correct topology of Voronoi Diagram when there are no degeneracies. The set Ω is the boundary complex of the 3-D polyhedron, and B_0 is a bounding box for the polyhedron. There refinement phase is called as a subroutine in the construction phase.

Subdivision Phase For every box B, the set $\phi(B)$ of objects whose Voronoi region intersects B is computed. To introduce their termination criteria, we need to define some special combinations of objects: A type-0 set of objects consists of a corner and two collinear edges such that the two edges are incident on the corner; a type-1 set of objects consists of an edge and two coplanar walls such that the edge is shared by the two walls. A box B is a leaf if one of the following predicates hold:

- $|\phi(B)| \leq 4$.
- $|\phi(B)| = 5$ and $\phi(B)$ contains either a type-0 or type-1 set.
- $|\phi(B)| = 6$ and $\phi(B)$ contains either two type-0, or two type-1, or a pair of type-0 and type-1 sets.
- All the objects in $\phi(B)$ share a vertex.
- All the objects in $\phi(B)$ except one share a vertex and a plane.

They show that if there are no degeneracies, then eventually one of these predicate will be true. The set $\phi(B)$ is computed by checking whether the bisector of a pair of objects intersects a face of B; this uses exact arithmetic. To compute the label-set of the children of B, we only need to consider the objects in $\phi(B)$.

Construction Phase: Given a box B and its set $\phi(B)$, they first detect Voronoi edge intersections with the faces of B. Since a bisector of two objects is a quadratic surface, its intersection with a face of B is a conic. Thus checking if a Voronoi edge intersects B reduces to intersecting two conic sections. To detect a Voronoi vertex inside a box, they use the following characterization: A box B contains a Voronoi vertex iff there is an edge that intersects its boundary an odd number of times. Since the edge-intersections are known, this test can be easily implemented; they assume, though, that every box has at most one Voronoi vertex. To construct Voronoi edges, they further refine the subdivision until no edge intersects the box more than twice; if an edge intersects a box B exactly twice then connect the two edge-intersections on the boundary of B. A Voronoi face is determined by a sequence of edges that have the same generator set.

Their algorithm is the first complete algorithm that constructs a topologically correct Voronoi diagram of a 3-D polyhedron assuming no degeneracies. To handle degeneracy, the δ cutoff parameter is used to stop the subdivision process. The topology of the Voronoi

diagram is guaranteed in boxes of size greater than δ. Their algorithm differs from our algorithm in the use of exact arithmetic, since they need to intersect two conic sections exactly, thus introducing algebraic predicates. Moreover, they do not use soft predicates.

¶13. Sud et al. [30] They consider the same problem as Etzion and Rappoport and build upon their work. Two key difference are the use of some soft predicates, and computing an approximation to the Voronoi diagram that is homotopy equivalent to the exact Voronoi diagram.

Subdivision Phase: Similar to Lavender et al., a label set $\widetilde{\phi}(B)$ is computed for B. To compute this set they use a soft predicate first introduced by Milenkovic [23]. Let m be the center and r the radius of B. Let $d(m, \Omega)$ be the distance from m to the nearest object in Ω. An object $f \in \Omega$ belongs to the label set $\widetilde{\phi}(B)$ if the distance from f to m is smaller than $d(m, \Omega)+2r$. They also use two exclusion tests based upon upper bounds on Voronoi regions for objects. A box B is a leaf if one of the following holds:

- $|\widetilde{\phi}(B)| = 1$.
- (The Homotopy Criterion) the intersection of the Voronoi region of a $f \in \widetilde{\phi}(B)$ with the boundary of B is homeomorphic to a disc, which ensures that the Voronoi diagram inside B can be retracted to a point.
- If B intersects a set $S \subseteq \Omega$ of objects, then $\widetilde{\phi}(B) \subseteq S$, and there is at most one corner in S and all the other objects must be incident on this corner. Such boxes are called boundary boxes.

To test the homotopy criterion they first compute the arrangements of the conics obtained by intersecting the Voronoi regions of the objects in $\widetilde{\phi}(B)$ with the faces of the box B. The edges and the vertices in the arrangement are labelled with the objects generating them. A connected sequence of edges sharing a common label $f \in \widetilde{\phi}(B)$ form the boundary of the intersection of the Voronoi region corresponding to the object f with the faces of B. For computing an arrangement of conics, they use an algorithm by Keyser et al. [16]. The termination criteria for boundary boxes implies that the Voronoi diagram of the objects in S always intersects the boxes. For completeness, they show that a box will eventually satisfy one of the three criteria.

Construction Phase: They first construct an approximate Voronoi graph $\mathrm{V}or^*(\Omega)$. For each box satisfying the homotopy criterion, they place a node at its center and connect it to the intersections of the conics on the faces of the box. The rest of the construction phase is similar to Etzion and Rappoport.

Their homotopic approach can handle near-degenerate situations, i.e., vertices with more than 4 generators, since these configurations can be retracted to a point. Their algorithm is also complete, though, again it uses exact arithmetic.

¶14. Stolpner et al. [?] They also consider the problem of computing the medial axis of a 3-D polyhedron Ω; let $\partial(\Omega)$ be the boundary of the polyhedron. The box B_0 is a bounding box for Ω.

Subdivision Phase: Let $D(p) := \inf_{q \in \partial(\Omega)} \|p - q\|$, $p \in \mathbb{R}^3$, be the distance function associated with Ω. Consider the gradient $\nabla D : \mathbb{R}^3 \to \mathbb{R}^3$ of D. For all points on the medial axis, ∇D is multivalued; for all points not on the medial axis, it has a unique value. Let S be the boundary of a sphere inside Ω. The average outward flux of ∇D through S is defined as $\mathrm{AOF}_S(\nabla D) := \frac{\int\int_S \nabla \cdot N_s dS}{\int\int_S dS}$, where N_S is the outward normal of S. It is not hard to see that if the medial axis does not intersect S then as the area of S tends to zero $\mathrm{AOF}_S(\nabla D)$ tends to zero. To compute AOF_S, a set of N points is chosen from S and the discretized version AOF'_S of AOF_S is computed. Given an initial resolution $\sigma < 0$, partition the interior of Ω into boxes of size σ. The algorithm will refine these boxes. A box B is a leaf in one of the following hold:

- $\mathrm{AOF}'_S(\nabla) \geq 0$, where S is the circumscribing sphere around B.
- Size of B is less than δ and $\mathrm{AOF}'_S(\nabla) < 0$.

Construction Phase: To find the medial axis inside a box B from the subdivision phase, the following observation is crucial: Let $q := p+\gamma\nabla D(p)$, for a scalar γ such that the $p + \gamma\nabla D(p) \in \Omega$; the segment $[p, q]$ intersects the medial axis iff $\nabla D(p) \neq \nabla D(q)$. Thus to find a point near the medial axis inside B a binary search can be done. By choosing more points p on the boundary of B, a pointwise approximation to the medial axis inside B can be obtained.

Usually, the input consists of an angle α, and the output consists of points on the medial axis that have exactly two nearby points on the boundary of Ω and the angle between these nearby points is 2α. In this case, the test $\mathrm{AOF}'_S(\nabla) < 0$ is replaced by $\mathrm{AOF}'_S(\nabla) < -c \cdot \sin\alpha$, for some positive constant $c < 1$. Thus vertices on the medial axis cannot be detected, since they have more than two points on the boundary nearest to them. Their algorithm only provides a one-sided approximation to $\overline{\mathrm{V}or}(\Omega)$ and no guarantee on the topology.

Milenkovic [23] has described another algorithm for approximating Voronoi diagrams of 3-D polyhedron The algorithm does use subdivision, but only as an optimization step. The main emphasis of his approach is to trace the Voronoi edges starting from Voronoi vertices. He uses linear programming to detect Voronoi vertices.

IV. Voronoi Diagram of Polyhedral Sets

To begin the development our new algorithm, we now introduce some basic concepts. Let $\Omega \subseteq \mathbb{R}^3$. Call it a **polyhedral set** if it is a closed set whose boundary can be partitioned into a finite set of vertices, open line segments and open triangles. Our goal is to compute an ε-approximation of $\mathrm{V}or(\Omega)$. For most of this paper, Ω may be assumed to be fixed.

¶15. Geometric Voronoi diagram. To define the geometric Voronoi diagram $\overline{\mathrm{V}or}(\Omega)$, we begin with the well-known concept[5] of separation. For $A, B \subseteq \mathbb{R}^d$, the **separation** between A and B is

$$\mathrm{S}ep(A, B) := \inf\{\|a - b\| : a \in A, b \in B\}.$$

This function is symmetric: $\mathrm{S}ep(A, B) = \mathrm{S}ep(B, A)$. If $A = \{p\}$, we simply write $\mathrm{S}ep(p, B)$ instead of $\mathrm{S}ep(\{p\}, B)$. We also introduce the "set extension" of $\mathrm{S}ep$, denoted $\square\mathrm{S}ep(A, B)$ where

$$\square\mathrm{S}ep(A, B) := \{\mathrm{S}ep(a, B) : a \in A\}.$$

Note that $\square\mathrm{S}ep(A, B)$ is non-symmetric as $\square\mathrm{S}ep(A, B) \neq \square\mathrm{S}ep(B, A)$ in general. If A is a connected set, then $\square\mathrm{S}ep(A, B)$ would be a real interval. The **clearance** of a point $q \in \mathbb{R}^d$ is just its separation from Ω,

$$C\ell(q) = C\ell_\Omega(q) := \mathrm{S}ep(q, \Omega).$$

The **clearance ball** of q is the closed ball $D(q) = D_\Omega(q)$ centered at q with radius $C\ell(q)$. Since Ω is non-empty, $C\ell(q) < \infty$ and $D_\Omega(q) \cap \Omega$ is non-empty. Call $q \in \mathbb{R}^d$ a **Voronoi point** of Ω if $D_\Omega(q) \cap \Omega$ contains more than one point. Finally, the **Voronoi diagram** $\overline{\mathrm{V}or}(\Omega)$ is defined to be the set of Voronoi points of Ω. Observe that Voronoi points must lie in the complement of Ω, and hence $\overline{\mathrm{V}or}(\Omega) \in \mathbb{R}^d \setminus \Omega$.

¶16. Voronoi complex. Next we define the Voronoi complex $\mathrm{V}or(\Omega)$. Recall that the boundary $\partial(\Omega)$ is partitioned into the boundary complex $\Phi(\Omega)$ of vertices, edges and triangles where edges are open line segments, and triangles do not include their boundaries. Note that the usual notion of faces of Ω are maximum connected planar subsets of $\partial(\Omega)$, and these are polygons. In order to get a simplicial complex, we arbitrarily split such polygons into triangles. The simplices in $\Phi(\Omega)$ are called **features**. For any point $q \in \mathbb{R}^3$, define $\phi(q)$ to be the set of those features in $\Phi(\Omega)$ that intersect $D(q)$. For $B \subseteq \mathbb{R}^d$, the **feature set** $\phi(B)$ of B is $\bigcup\{\phi(q) : q \in B\}$.

[5] In the Voronoi diagram literature, $\mathrm{S}ep(A, B)$ is frequently called a "distance function" and denoted $d(A, B)$. Unfortunately, it fails the triangular inequality expected of distances: $\mathrm{S}ep(A, C) \leq \mathrm{S}ep(A, B) + \mathrm{S}ep(B, C)$. For instance, the relations fails if $\mathrm{S}ep(A, C) > 0$ and $B = A \cup C$. So for Euclidean sets, we generally let $d(A, B)$ denote the Hausdorff distance.

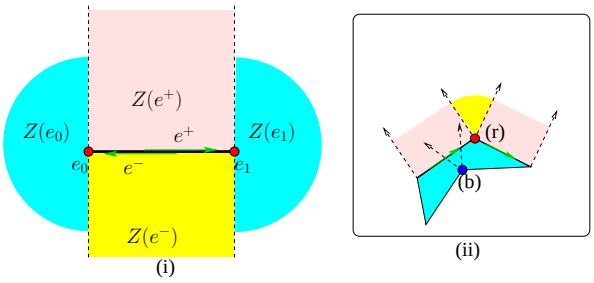


Figure 2. (i) Zones of the oriented features $S = \{e_0, e_1, e^+, e^-\}$. (ii) Corner and Edge Features

For simplicity, we first focus on the planar setting. Generalization of the basic definitions to 3-D is then straightforward. As motivation, consider the case where Ω is just the closed line segment Ω_0 whose boundary complex has three features:

$$\Omega_0 = \{(x, 0) : 0 \leq x \leq 1\}, \qquad \Phi(\Omega) = \{e, e_0, e_1\} \tag{3}$$

where $e = \{(x, 0) : 0 < x < 1\}$ is an open line segment, and e_0, e_1 are its two end points. The zone $Z(e)$ of an open line segment e in the above example was an infinite strip (the strip is an open set, not including its boundary). We thus obtain a partition of $\mathbb{R}^2$ into three simple regions called **zones**, $Z(e_0), Z(e_1), Z(e)$. $Z(e)$ can be further refined as follows: regard the segment e as two oriented segments e^+ and e^-. These are called **oriented features**, and their respective zones $Z(e^+), Z(e^-)$ are semi-infinite strips whose union is the original strip, and whose intersection is e, where a zone lies to the right of its directed segment. These zones are illustrated in Figure 2(a).

The zones in the above example can be neatly captured if we introduce a variation of our separation function $\mathrm{S}ep(A, B)$: define

$$\mathrm{S}ep^*(A, B) := \begin{cases} \mathrm{S}ep(A, B) & \text{if } [\|a - b\| = \mathrm{S}ep(A, B)] \\ & \text{for some } a \in A, b \in B, \\ \infty & \text{else.} \end{cases}$$

Call $\mathrm{S}ep^*(A, B)$ the **$*$-separation of A from B**. Clearly, if A, B are closed sets then $\mathrm{S}ep^*(A, B) = \mathrm{S}ep(A, B)$. Relative to $(\Omega, \Phi(\Omega))$, we define the **Voronoi region** of $f \in \Phi(\Omega)$ to be the set $V(f)$ comprising those points q in the complement of Ω whose $*$-separation from f is at most the $*$-separation from any other feature g:

$$V(f) := \{q \in \mathbb{R}^2 \setminus \Omega : (\forall g \in \Phi(\Omega))\mathrm{S}ep^*(q, f) \leq \mathrm{S}ep^*(q, g)\}$$

In the example Ω_0 in (3) above, we now see the zone of $f \in \{e, e_0, e_1\}$ is just the Voronoi region of f relative to $\Phi(\Omega_0)$. More generally, for any feature f in a arbitrary set Ω, the zone $Z(f)$ can be viewed as the Voronoi region of f relative to a subset $\Omega_0(f) = \{f, f_0, f_1\}$ defined as follows: if f is an edge, then f_0, f_1 are the

two endpoints of f; if f is a corner, then f_0, f_1 are the two edges that shares f as endpoint.

Suppose $F \subseteq \Phi(\Omega)$ is a set with at least two features. Let $V(F) := \cap\{V(f) : f \in F\}$. We define a **Voronoi cell** to be a connected component of a set of the form $V(F)$. The simplicial nature of our boundary complex $\Phi(\Omega)$ ensures that each cell is either a point or homeomorphic to an Euclidean ball of some dimension $i = 1, \ldots, d-1$. Moreover, for each Voronoi cell C we associate the maximal set F of features that define C. The collection of these cells constitute our Voronoi complex $\mathrm{V}or(\Omega)$.

The zone idea originated in Kirkpatrick [17]. Traditionally, it is viewed as a method to split a Voronoi cell into simpler "subcells". But in in our subdivision setting, we view Kirkpatrick's trick as an effective "filtering" mechanism in computing soft predicates. We next address a refinement of this filtering.

Henceforth, we assume Ω is **regularized** meaning that it is non-empty and Ω is equal to the closure of its interior. The latter requirement says that Ω does not contain isolated points or line segments or slits, for instance. Thus the set Ω_0 in (3) is not regularized. Recall that the geometric Voronoi diagram $\overline{\mathrm{V}or}(\Omega)$ lies in the complement of Ω, i.e., $\overline{\mathrm{V}or}(\Omega) \cap \Omega$ is empty. We shall now refine the zone idea to take advantage of a regularized Ω. For each feature $f \in \Phi(\Omega)$, we have defined its zone $Z(f)$. We now define the **oriented zone** $Z^*(f)$ which has the property that $Z^*(f) \subseteq Z(f)$. Computationally, replacing $Z(f)$ by $Z^*(f)$ will greatly improve the effectiveness of zone filtering.

For a regularized $\Omega \subseteq \mathbb{R}^2$, we can replace each segment e by its oriented version e^+ where we adopt the convention that these e^+ result in a counter-clockwise orientation around the boundary of each polygon of Ω. We define $Z^*(e)$ to be the zone of e^+. Note that locally, around e, $Z^*(e)$ lies in the complement of Ω.

For each corner feature c, we define $Z^*(c)$ as follows: suppose c is the endpoint shared by two edges e and e'. Intuitively, we would like to define the zone of c to be $R \cap R'$ where R (resp., R') is the zone of c when viewed as an endpoint of e (resp., e'). The set $R \cap R'$ is a cone pointed at c. But we can do better: the cone, locally at c, either lies in Ω or lies in the complement of Ω. Since the Voronoi diagram lies only in the complement of Ω, we may therefore define $Z^*(c)$ to be the empty set in the former case, and $Z^*(c) = R \cap R'$ in the latter case.

The definition of oriented zones is illustrated by the non-convex pentagon (colored cyan) in Figure 2(ii). There are 5 corner features and 5 edge features in the boundary complex $\Phi(\Omega)$. For an edge e, $Z^*(e)$ is an semi-infinite strip (two such zones are colored pink in Figure 2(ii)). For a corner c, the oriented zone $Z^*(c)$ is empty if c is a concave corner, and otherwise it is a cone with apex c. Figure 2(ii) illustrates these two possibilities (convex corner in red, concave corner in blue).

¶17. Soft Predicates. The most basic predicate we consider is the following: given a box B, does B intersect $\mathrm{V}or(\Omega)$? This is encapsulated by the logical[6] predicate $C(B)$ that returns true iff $B \cap \mathrm{V}or(\Omega) = \emptyset$. But our true interest is in the 3-valued predicate $\widetilde{C}(B)$,

$$\widetilde{C} : B \mapsto \{\texttt{IN}, \texttt{ON}, \texttt{OUT}\}$$

that approximates $C(B)$ in the following sense:
(1) Conservative: $\widetilde{C}(B) = \texttt{IN}$ implies $C(B)$ is true, and $\widetilde{C}(B) = \texttt{OUT}$ implies $C(B)$ is false.
(2) Convergent: for any sequence $\{B_i : i \geq 0\}$ of boxes that converges to a point p, $\widetilde{C}(B_i)$ will converge to $\texttt{IN}$ (resp., $\texttt{OUT}$) if $p \in \mathrm{V}or(\Omega)$ (resp., $p \notin \mathrm{V}or(\Omega)$).

Thus the $\texttt{ON}$-answer is viewed as an indecisive answer, as opposed to the decisive $\texttt{IN}$- or $\texttt{OUT}$-answers. We used the "distributional technique" to evaluate such soft predicates. Recall the notion of the feature set $\phi(B)$, comprising those features that are closest to some point B. But for soft predicates, we prefer to define the set $\widetilde{\phi}(B)$ that is an approximation of $\phi(B)$. The **influence disc** $D(B)$ of B is centered at m_B with radius $C\ell(m_B) + 2r_B$. We say that feature f **belongs** to a box B if the $D(B)Z^*(f)$ is non-empty. In particular, we have

$$Sep(m(B), f) \leq C\ell(m(B)) + 2r(B). \tag{4}$$

This conservative predicate was first used by Milenkovic [23]. Let $\widetilde{\phi}(B)$ denote the set of features that belongs to B. For example, in Figure 3(a), the corner f and two edges g, h belongs to the box B. Thus $\widetilde{\phi}(B) = \{f, g, h\}$. But in Figure 3(b), the corner f does not belong to B because $Z^*(f)$ is empty. We say a set of features $\widetilde{\phi}(B)$ is **inseparable** if $|\widetilde{\phi}(B)| = 1$ or there exists a corner $c \in \widetilde{\phi}(B)$ such that for each feature $g \in \widetilde{\phi}(B)$, $g \neq c$ implies g is an edge with an endpoint at c. Thus, the set $\widetilde{\phi}(B)$ is inseparable in Figure 3(a) but separable in Figure 3(b).

The key properties are:

LEMMA 3.
(i) Conservative: $\phi(B) \subseteq \widetilde{\phi}(B)$.
(ii) Convergent: Let $(B_i : i \geq 0)$ *be an infinite sequence of boxes that strictly converges to a point* q: $B_i \to q$ *as* $i \to \infty$. *Then* $\widetilde{\phi}(B_i) \to \phi(q)$, *i.e.,* $\widetilde{\phi}(B_i) = \phi(q)$ *for* i *sufficiently large.*
(iii) Distributional: If B' *is a child of* B, *then* $\widetilde{\phi}(B') \subseteq \widetilde{\phi}(B)$.

This lemma implies that no feature outside of $\widetilde{\phi}(B)$ has any influence on the Voronoi diagram restricted to

[6] By "logical predicate", we mean a standard 2-valued truth function. Geometric predicates tend to be 3-valued.

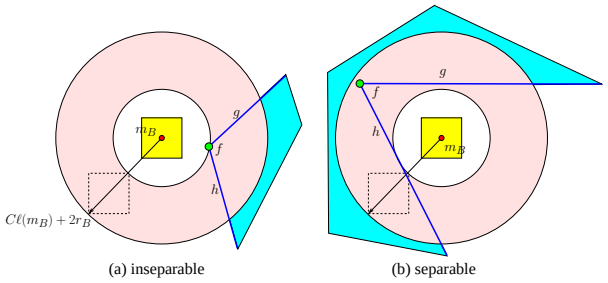


Figure 3. (a) $\widetilde{\phi}(B) = \{f, g, h\}$ is inseparable, (b) $\widetilde{\phi}(B) = \{g, h\}$ is separable

B, and that when B is small enough, $\widetilde{\phi}(B) = \phi(B)$. The distributional property ensures that when we split a box, the children only need to inherit features from its parent. The basic idea for defining the soft predicate $\widetilde{C}(B)$ is defined to be ON as long as $\widetilde{\phi}(B)$ is not "simple". When $\widetilde{\phi}(B)$ is "simple", we will determine if $\widetilde{C}(B)$ is IN or OUT. Moreover, by repeated subdivision of boxes, eventually $\widetilde{\phi}(B)$ will be "simple".

V. Subdivision Algorithm for the Voronoi Complex

We now describe a new algorithm for computing an ε-approximation of the Voronoi complex $\mathrm{V}or(\Omega)$ for a non-degenerate polygonal set $\Omega \subseteq \mathbb{R}^2$. The input is the polygonal set Ω represented by its boundary complex $\Phi(\Omega)$, a resolution parameter $\varepsilon > 0$ and an initial box B_0. The output will be an ε-approximation of $\mathrm{V}or(\Omega)$ restricted to B_0. To avoid the issues of boundary processing (cf. [4]), we assume that ∂B_0 is nice (has no Voronoi vertices or tangential intersection with Voronoi curves). Our algorithm follows the standard subdivision algorithm of ¶7 with its three Phases.

For simplicity, our description below assumes $\varepsilon = \infty$, i.e., we are only concerned with topological correctness and not geometric accuracy. A simple method to enforce a finite ε-bound is to further subdivide any box that potentially contains Voronoi points during the Refinement Phase. More efficient methods can be devised, as a post-processing Phase. We address this in the full paper.

¶18. Subdivision Phase. We only have to describe the box predicate $C(B)$ which provides the termination criterion. We use the standard distributional approach: for each box B in the subdivision tree, we maintain its feature set $\widetilde{\phi}(B)$ and labels for each vertex of B. In general, for any point q, its **label** $\lambda(q)$ is the feature f which is closest to q subject to $q \in Z^*(f)$. Note that $\lambda(q)$ is undefined if q belongs to no $Z^*(f)$; labels need not be unique, but we break ties arbitrarily. The predicate $C(B)$ is a conjunction of three simpler predicates, $C(B) \equiv C_0(B) \wedge\ C_1(B) \wedge\ C_2(B)$ where

- $C_0(B)$ says $|\widetilde{\phi}(B)| \leq 3$.
- $C_1(B)$ says: if $C\ell(m_B) < r_B$ then $\widetilde{\phi}(B)$ is an inseparable set.
- $C_2(B)$ says that if B is "(f, g)-special" then $C\ell(m_B) \leq (3r_B^2)/(2\alpha)$ where $\alpha = \mathrm{S}ep(f, g)$.

We must explain the concept of a "special" box here in $C_2(B)$. A pair (f, g) of features is called a **parabola generator** if f is a corner and g an edge, and $Z^*(f) \cap Z^*(g)$ is non-empty. A box B is said to be (f, g)-**special** (or simply **special**) if the following three conditions hold:

- (i) [B is g-monochromatic] The four vertices of B are all labeled with g.
- (ii) [B is free of features] $C\ell(m_B) > r_B$
- (iii) [B has potential incursion] $f, g \in \widetilde{\phi}(B)$ where (f, g) is a parabola generator. It generates the **special parabola** of B.

One might naively conclude from condition (i) that B belongs to the Voronoi region of g. Vleugels-Overmars [32] points out this fallacy, and invoked a lemma of Siersma to show that if the boxes are subdivided "sufficiently", then Voronoi bisectors must be detected in a box adjacent to B. Our analysis provides explicit bounds for detecting this condition and this is encoded in our definition of special boxes.

LEMMA 4. *If Ω is non-degenerate, then the Subdivision Phase halts.*

¶19. Refinement Phase. Following Vleugels-Overmars [32], we balance the subdivision in this Phase (i.e., split a box if it more than twice the size of a neighbor). More precisely, at the start of this phase, boxes satisfy $|\widetilde{\phi}(B)| \in \{1, 2, 3\}$. The **candidate boxes** are those with $|\widetilde{\phi}(B)| > 1$, as these may contain Voronoi points. It suffices to balance candidate boxes; this is easily done using a priority queue.

¶20. Construction Phase. Our goal is to introduce **nodes** which are either points in the interior of a box (representing Voronoi vertices) or the interior of box edges (representing via points for Voronoi curves). We also need to introduce **arcs** which connect these nodes, representing portions of Voronoi curves. The resulting graph $G = (N, A)$ where N is the node set and A the arc set will be an ε-approximation of $\mathrm{V}or(\Omega)$. Figure 4 illustrates such nodes and arcs.

Let $\mathcal{S}$ be the subdivision of B_0 at the end of the Refinement Phase. A **segment** refers to a side of a box in $\mathcal{S}$ that contains no vertices. Because of balancing, the side of a box in $\mathcal{S}$ is either a segment or the union of two segments. A segment is **monochromatic** or **bichromatic** depending on whether its endpoints have identical or different labels. Recall that a box is monochromatic if all its vertices have the same label; we now say it is **fully monochromatic** if every segment in its boundary

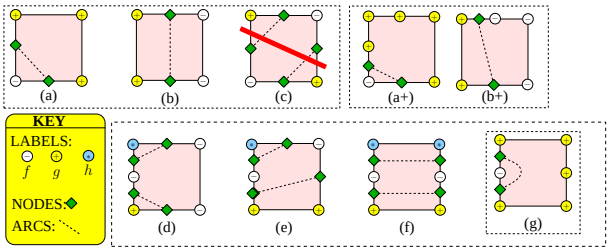


Figure 4. Some connection rules based on Labels.

is monochromatic. The main basis for introducing nodes is the presence of bichromatic segments. The simplest examples are shown Figure 4(a,b,c) when the box has only four segments. Case(c) cannot arise when there are only kinds of labels. Figure 4(a+,b+) are similar to Cases(a,b) (respectively) but in situations where the box has more than 4 segments. As a default, we place the nodes in the middle of the segment, but simple interpolations can produce more accurate placements (we do not do exact computations to place the nodes as they may have irrational coordinates). Besides bichromatic segments, there are two other basis for introducing nodes: in the middle of a box for Voronoi vertices, and there is also a possibility to introduce two nodes in a segment.

Arcs are introduced inside each box, connecting nodes on its boundary or its interior. As a default, we use a straightline segment for an arc; the exception is when we connect two nodes on the same side, as in Figure 4(g). Observe that each box has at most two arcs, unless there is an interior node in which case there are three arcs.

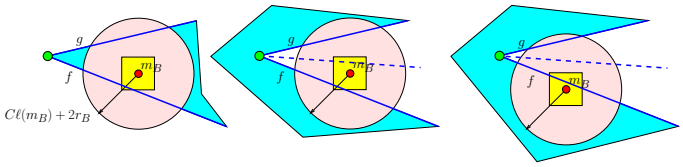


Figure 5. $\widetilde{\phi}(B)$ is separable: (a) $\Omega \cap D(B)$ is connected. (b,c) $\Omega \cap D(B)$ is disconnected.

¶21. Boxes with two features. Let us first process the boxes B with only two features, say f and g.

- Discard B if $\widetilde{\phi}(B)$ is inseparable or ($C\ell(B) < r$ and $D(B) \cap \Omega$ is a connected set). See Figure 5(a).
- Suppose $C\ell(B) < r$ and $D(B) \cap \Omega$ is disconnected (see Figure 5(b) and (c)). Intersect the bisector of f, g with the boundary of B: if there are two intersection points (this happens only in Figure 5(b)), introduce two nodes and connect them with a straightline segment as arc.
- Otherwise, we introduce a node in each bichromatic segment of B. Either two or no node will be introduced. In the former case, we introduce an arc connecting them. The arc is generally a line segment, but if two nodes belong to one side of B, we use two line segments. See Figure 5(b).

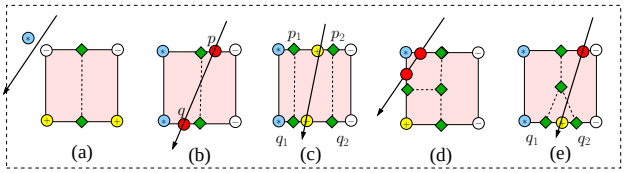


Figure 6. Voronoi Vertex Test: (a) Fails (reduced to 2 features) (b) Fails (three distinct labels) (c) Fails (two distinct labels) (d) Fails (three distinct labels) (e) Passes (two distinct labels)

¶22. Boxes with three features. Next assume $\widetilde{\phi}(B)$ has three features, f, g and h. Wlog, assume that f, g have the same type (both corners or both edges). The following computation constitutes the *Voronoi Vertex Test:*

- Check if the straightline bisector of f, g intersects B. If it does not, the Test **fails**. We can discard either f or g. Then the construction is reduced to the previous case where $|\widetilde{\phi}(B)| = 2$. See Figure 6(a).
- Otherwise, suppose the bisector intersect B at the points p, q. To be specific, we may direct the bisector from p towards q with the property that the approach begins from points which are closer to f and g than to h. We may further assume that the labels of p and q are never g (that is, if the label could be g, we simply choose f as tie breaker). We have three possibilities:
- If $\lambda(p) = \lambda(q) = f$, the Test **fails**. In this case, we introduce two nodes in the segments containing p and q respectively, and also an arc to connect them. The feature h has no role in this box (i.e., $h \notin \phi(B)$). See Figure 6(b).
- If $\lambda(p) = \lambda(q) = h$, the Test **fails**. Consider the segment containing p: this segment is split into two subsegments by p; similarly, q introduces two subsegments. We check the subsegments and any original segments of B lying to one side of pq, to see if any is bichromatic (where p, q have label h). If so, we introduce nodes as usual. This will introduce either no node or two nodes p_1, q_1. We connect p_1, q_1. Similarly, we may introduce no nodes or two nodes p_2, q_2 on the other side of pq. Figure 6(c) illustrates the case where we introduced four nodes in this way.
- If $\lambda(p) = f, \lambda(q) = h$, the Test **passes**. This means we have confirmed the presence of a Voronoi

vertex inside B. We introduce a node v in interior of B to represent the Voronoi vertex. There are now two possibilities:

(1) B has three bichromatic segments, representing the three Voronoi curves generated by f, g, h in pairs: $(f, g), (g, h), (f, h)$. We introduce three nodes in the usual way, and join each of them to the v. See Figure 6(d).

(2) B has two bichromatic segments. These are necessarily labeled by f and g. Moreover, one of them contains the point q. We introduce a node in the segment containing p and two nodes q_1, q_2 on either side of the segment containing q. See Figure 6(d).

The correctness of our algorithm comes from the following properties. Let $G = (N, A)$ be the PSLG constructed above. The nodes in N are called **via nodes** if they lie in the interior of a segment, and they are called **vertex nodes** if in the interior of a box. A **maximal path** in the graph $G = (N, A)$ refers to a path in G whose endpoints have degrees different than 2.

LEMMA 5. *(a) Each via node has degree 2 and each vertex node has degree 3.*
(b) Each Voronoi vertex in $\mathrm{Vor}(\Omega)$ is isolated in some box that passes the Voronoi Vertex Test.
(c) Each Voronoi curve c in $\mathrm{Vor}(\Omega)$ is associated with a maximal path $\widetilde{c}$. Moreover, there is an isotopy $I_c : [0, 1] \to \mathbb{R}^2$ from c to $\widetilde{c}$ which respects the vertices of boxes. I_c also respect the sides of non-special boxes.

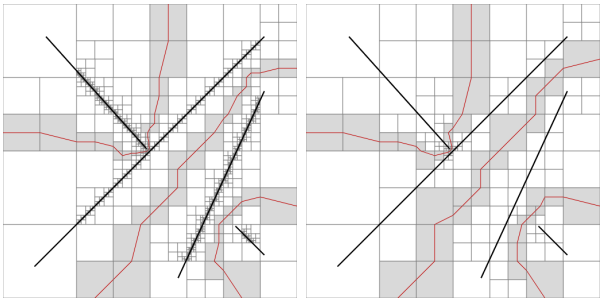
THEOREM 6. *(Correctness) Assuming Ω is non-degenerate, our algorithm halts and the output graph $G = (N, A)$ is an approximation to the Voronoi complex $\mathrm{Vor}(\Omega)$.*

¶23. Remarks. Although the basic primitives are simple, the correctness arguments are intricate and will appear in the full paper (and our webpages). An interesting remark is that our algorithm is almost exclusively relying on local isotopy (i.e., the arc connections within each subdivision box is isotopic to the actual Voronoi curves in the box). A mild exception is the case of special boxes; non-local isotopy was first exploited by Plantinga-Vegter [27], [21].

VI. Implementation and the Role of Filters

We have a preliminary `C++` implementation of our new algorithm, freely available in our open-source Core Library [39]. Our input files and test results are archived there. The full paper will include experimental comparisons with known exact algorithms.

¶24. What is a Purely Numerical Algorithm? It is important to understand the claim that our algorithm is "purely numerical". It means that all our predicates can be reduced to comparisons of the form $x < y$ (or $x \leq y$) where x, y are computed quantities. Typically, x, y are clearances or separations between features. But distances are irrational functions (involving square-roots) of the input parameters. For exposition, we write them in exact terms. But we do not (should not) compute them exactly. One should think of all these quantities x, y as intervals. We only require the widths of these intervals to approach 0 as $r(B) \to 0$. An consequence of this interpretation is that "$x < y$" is a one-sided test; failure of the test does not imply "$x \leq y$". Our current implementation is based on naive machine arithmetic, and suffices for our demos. We plan to use the number types of `Core Library` [39] that can provide such interval functionalities. The key difference between soft predicates and usual exact predicates is termination is based on non-degeneracy (or cut-offs), not zero-bounds.



(a) using the original C_1 filter (b) using the new C_1 filter

Figure 7. The Voronoi complex of line segments. Each line segment consists of two half-edges. (a) Original with 11,932 boxes, (b) New with 169 boxes.

In our Construction Phase, we prescribe certain operations by intersecting pairs of lines. This could be done exactly if we are willing to use rational arithmetic. But even this should be avoided as we wish to use dyadic numbers (BigFloats) only. A correct implementation can easily replace these by approximations.

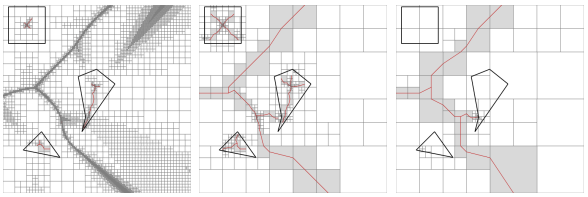
¶25. Filters. The performance of exact numerical algorithms depends critically on the design of efficacious filters. This theme is well-known from the Exact Geometric Computation [38]. In our setting, filters are aimed at preventing unnecessary subdivision. We provide two examples here. To measure the effectiveness of the filters, we count number of subdivision boxes. Both examples show two orders of magnitude improvements. (a) The above definition of the predicate $C_1(B)$ is sufficient for correctness. But we can redefine $C_1(B)$

as:

$$C\ell(m_B) < r_B \text{ implies } \begin{cases} \widetilde{\phi}(B) \text{ is an inseparable set}, \textit{or} \\ \widetilde{\phi}(B) = \{e_1, e_2\} \text{ where each } e_i \text{ is an edge} \end{cases}.$$

The extra filter condition, where $\widetilde{\phi}(B) = \{e_1, e_2\}$ is illustrated in Figure 5(a,b,c). The advantages of using this extra filter is seen in Figure 7(a,b).

(b) As another example, we show the effects of turning on the zone filter $Z(f)$ and the oriented zone filter $Z^*(f)$. The former filter says that if $B \cap Z(f) = \emptyset$ then $f \notin \widetilde{\phi}(B)$. The latter is similar, except that we use $Z^*(f)$ instead of $Z(f)$. We can see the dramatic reduction in the size of the subdivision for a relatively simple input in Figure 8(a,b,c).



(a) (b) (c)

Figure 8. (a) Without zone filter (13777 boxes), (b) With $Z(f)$ filter (493 boxes) and (c) With $Z^*(f)$ filter (112 boxes)

VII. Conclusions

The 20-year old quest of Computational Geometers for an exact Voronoi Diagram offers an object lesson about computational models and their limitations. The exact viewpoint of geometric computation offered by Semi-Algebraic Geometry and Computational Geometry is the "gold standard", and rightly so. But that does not mean that our computational goals must always be these ideal objects. Most practical applications have no need for these ideal objects. What is often needed is a resolution-bounded approximation of such ideals. Instead of exact geometric primitives in standard computational models, we can use soft primitives that converge to the exact primitives in the limit. Highlighting such primitives is a necessary step towards laying a sound foundation for deeper algorithmic analysis in this field. Moreover, non-trivial complexity analysis of such algorithms remain largely undeveloped.

In this paper, we designed a new algorithm for the Voronoi complex of polygonal sets. Our preliminary implementation of the algorithm (available from [39]) shows great promise. There remain many fundamental issues to be addressed even for this algorithm: weakened correctness in δ-cutoff, analysis of numerical approximations, and complexity analysis.

We have also applied our viewpoint to the problem of motion planning [33], [35], isotopic approximation of curves [21] and even root isolation [28]. The resolution-bounded view point has been around for a long time and permeates many subjects. For instance, convergence is the central to numerical computing. But it has made few in-roads into Computational Geometry so far. We believe that exciting new and practical algorithms will emerge when the breakthrough comes.

Acknowledgment

Yap's work is supported in part by NSF Grant CCF-0917093. Lien's work is supported in part by NSF Grant IIS-096053.

References

[1] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry*. Algorithms and Computation in Mathematics. Springer, 2003.

[2] I. Boada, N. Coll, N. Madern, and J. A. Sellarés. Approximations of 2d and 3d generalized Voronoi diagrams. *Intl. J. of Computer Mathematics*, 85(7):1003–1022, 2008.

[3] C. Burnikel, K. Mehlhorn, and S. Schirra. On degeneracy in geometric computations. In *Proc. 5th ACM-SIAM Symp. on Discrete Algorithms*, pages 16–23, 1995.

[4] M. Burr, S. Choi, B. Galehouse, and C. Yap. Complete subdivision algorithms, II: Isotopic meshing of singular algebraic curves. *J. Symbolic Computation*, 47(2):131–152, 2012. Special Issue for ISSAC 2008.

[5] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[6] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[7] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9:66–104, 1990.

[8] I. Z. Emiris and M. I. Karavelas. The predicates of the Apollonius diagram: Algorithmic analysis and implementation. *Comput. Geometry: Theory and Appl.*, 33(1–2):18–57, 2006. Special Issue on Robust Geometric Algorithms and their Implementations.

[9] I. Z. Emiris, E. P. Tsigaridas, and G. M. Tzoumas. The predicates for the Voronoi diagram of ellipses. *22nd ACM Symp. on Comp. Geom.*, 2006.

[10] M. Etzion and A. Rappoport. Computing Voronoi skeletons of a 3-d polyhedron by space subdivision. *Computational Geometry*, 21:87–120, 2002.

[11] H. Everett, C. Gillot, D. Lazard, S. Lazard, and M. Pouget. The Voronoi diagram of three arbitrary lines in r^3. In *25th European Workshop on Computational Geometry (EuroCG'09)*, 2009. Mar 2009, Bruxelles, Belgium.

[12] H. Everett, D. Lazard, S. Lazard, and M. S. E. Din. The Voronoi diagram of three lines. *Discrete and Comp. Geom.*, 42(1):94–130, 2009. See also 23rd SoCG, 2007. pp.255–264.

[13] S. J. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[14] J. E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry*. CRC Press LLC, 1997.

[15] M. Hemmer, O. Setter, and D. Halperin. Constructing the exact Voronoi diagram of arbitrary lines in three-dimensional space. In *Algorithms ESA 2010*, volume 6346 of *Lecture Notes in Computer Science*, pages 398–409. Springer Berlin / Heidelberg, 2010.

[16] J. Keyser, T. Culver, D. Manocha, and S. Krishnan. Mapc: A library for efficient and exact manipulation of algebraic points and curves. *ACM Symp. on Comp. Geom.*, 15:360–369, 1999.

[17] D. G. Kirkpatrick. Efficient computation of continuous skeletons. *IEEE Foundations of Comp. Sci.*, 20:18–27, 1979.

[18] R. Klein. *Concrete and abstract Voronoi diagrams*. Lecture Notes in Computer Science, No. 400. Springer-Verlag, Berlin, 1989.

[19] V. Koltun and M. Sharir. Polyhedral Voronoi diagrams of polyhedra in three dimensions. *Discrete and Comp. Geom.*, 31:83–124, 2004.

[20] D. Lavender, A. Bowyer, J. Davenport, A. Wallis, and J. Woodwark. Voronoi diagrams of set-theoretic solid models. *IEEE Computer Graphics and Applications*, 12(5):69–77, 1992.

[21] L. Lin and C. Yap. Adaptive isotopic approximation of nonsingular curves: the parameterizability and nonlocal isotopy approach. *Discrete and Comp. Geom.*, 45(4):760–795, 2011.

[22] K. Mehlhorn and S. Näher. LEDA: a platform for combinatorial and geometric computing. *Comm. of the ACM*, 38:96–102, 1995.

[23] V. Milenkovic. Robust construction of the Voronoi diagram of a polyhedron, 1993. Univ.of Waterloo, Ontario, Canada. Aug 5-9, 1993.

[24] J. R. Munkres. *Elements of Algebraic Topology*. The Benjamin/Cummings Publishing Company, Inc, Menlo Park, CA, 1984.

[25] C. Ó'Dúnlaing, M. Sharir, and C. K. Yap. Retraction: a new approach to motion-planning. *ACM Symp. Theory of Comput.*, 15:207–220, 1983.

[26] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations — Concepts and Applications of Voronoi Diagrams*. John Wiley and Sons, 2nd edition edition, 2000.

[27] S. Plantinga and G. Vegter. Isotopic approximation of implicit curves and surfaces. In *Proc. Eurographics Symposium on Geometry Processing*, pages 245–254, New York, 2004. ACM Press.

[28] M. Sagraloff and C. K. Yap. A simple but exact and efficient algorithm for complex root isolation. In I. Z. Emiris, editor, *36th Int'l Symp. Symbolic and Alge. Comp. (ISSAC)*, pages 353–360, 2011. June 8-11, San Jose, California.

[29] S. Stolpner and S. Whitesides. Medial axis approximation with bounded error. In F. Anton, editor, *ISVD*, pages 171–180. IEEE Computer Society, 2009.

[30] A. Sud, M. Foskey, and D. Manocha. Homotopy-preserving medial axis simplification. *Int. J. Comput. Geometry Appl.*, 17(5):423–451, 2007.

[31] M. Teichmann and S. Teller. Polygonal approximation of voronoi diagrams of a set of triangles in three dimensions. LCS Technical Report 766, MIT, 1998.

[32] J. Vleugels and M. Overmars. Approximating generalized Voronoi diagrams in any dimension. Technical Report UU-CS-1995-14, Department of Computer Science, Utrecht University, 1995.

[33] C. Wang, Y.-J. Chiang, and C. Yap. On soft predicates in subdivision motion planning, 2012. Submitted, April 20, 2012: Symp. on Geometric Processing (SGP).

[34] J. Woodwark and A. Boyer. Better and faster pictures from solid models. *Computer-Aided Eng.*, 3(1):17–24, February 1986.

[35] C. Yap. Theory of soft subdivision search and motion planning, 2012. Submitted.

[36] C. K. Yap. An $O(n \log n)$ algorithm for the Voronoi diagram for a set of simple curve segments. *Discrete and Comp. Geom.*, 2:365–394, 1987. Also: NYU-Courant Institute, Robotics Lab., No. 43, May 1985.

[37] C. K. Yap. A geometric consistency theorem for a symbolic perturbation scheme. *J. Computer and System Sciences*, 40(1):2–18, 1990.

[38] C. K. Yap. In praise of numerical computation. In S. Albers, H. Alt, and S. Näher, editors, *Efficient Algorithms*, volume 5760 of *Lect. Notes in C.S.*, pages 308–407. Springer-Verlag, 2009.

[39] J. Yu, C. Yap, Z. Du, S. Pion, and H. Bronnimann. Core 2: A library for Exact Numeric Computation in Geometry and Algebra. In *3rd Proc. Int'l Congress on Mathematical Software (ICMS)*, pages 121–141. Springer, 2010. LNCS No. 6327.