Constructing Two-Dimensional Voronoi Diagrams via Divide-and-Conquer of Envelopes in Space*

Ophir Setter, Micha Sharir, and Dan Halperin

School of Computer Science
Tel-Aviv University
Tel-Aviv, Israel
{ophirset,michas,danha}@post.tau.ac.il

Abstract. We present a general framework for computing Voronoi diagrams of different classes of sites under various distance functions in $\mathbb{R}^2$. Most diagrams mentioned in the paper are in the plane. However, the framework is sufficiently general to support diagrams embedded on a family of two-dimensional parametric surfaces in three-dimensions. The computation of the diagrams is carried out through the construction of envelopes of surfaces in 3-space provided by CGAL (the Computational Geometry Algorithm Library). The construction of the envelopes follows a divide-and-conquer approach. A straightforward application of the divide-and-conquer approach for Voronoi diagrams yields algorithms that are inefficient in the worst case. We prove that through randomization, the expected running time becomes near-optimal in the worst case. We also show how to apply the new framework and other existing tools from CGAL to compute minimum-width annuli of sets of disks, which requires the computation of two Voronoi diagrams of different types, and of the overlay of the two diagrams. We do not assume general position. Namely, we handle degenerate input, and produce exact results.

1 Introduction

Voronoi diagrams were thoroughly investigated, and were used to solve many geometric problems, since introduced by Shamos and Hoey to the field of computer science [1] (although their origin dates back centuries ago — see [2]). The concept of Voronoi diagrams was extended to handle various kinds of geometric sites, ambient spaces, and distance functions. Among those are power diagrams

* Work on this paper has been supported in part by the Hermann Minkowski–Minerva Center for Geometry at Tel-Aviv University. Work by Ophir Setter and Dan Halperin has also been supported in part by the Israel Science Foundation (Grant no. 236/06), and by the German-Israeli Foundation (Grant no. 969/07). Work by Micha Sharir was also partially supported by NSF Grants CCF-05-14079 and CCF-08-30272, by Grant 2006/194 from the U.S.-Israeli Binational Science Foundation, by Grants 155/05 and 338/09 from the Israel Science Fund, Israeli Academy of Sciences, and by a grant from the French-Israeli AFIRST program.

of points in the plane, multiplicatively-weighted Voronoi diagrams, additively-weighted Voronoi diagrams (also known as Apollonius diagrams), Voronoi diagrams of line segments, and many other types [2–4]. There are types of Voronoi diagrams that are defined for pairs (or larger tuples) of sites [5]. Different types of Voronoi diagrams have been unified under a generalized framework [6].

Numerous approaches for computing Voronoi diagrams were developed: the divide-and-conquer algorithm by Shamos and Hoey [1], the sweep-line algorithm by Fortune [7], randomized incremental constructions [8, 9], a dynamic algorithm for planar convex objects [10], and more. A recent approach employs a divide-and-conquer medial-axis algorithm on an augmented domain to compute the Euclidean Voronoi diagrams of various types of sites [11]. Available exact implementations of Voronoi diagram algorithms include the computation of Delaunay graphs dual to standard Voronoi diagrams, to Apollonius diagrams, and to segment Voronoi diagrams in CGAL [12–14], the Voronoi diagrams of ellipses in CGAL [15], and the construction of segment Voronoi diagrams in LEDA [16]. Prominent alternatives include the VRONI code that uses floating-point arithmetic to compute Voronoi diagrams of points, segments, and arcs in 2D [17], and other implementations that use the Graphics Processing Unit (GPU) to discretely compute Voronoi diagrams [18, 19]. Typically, the time complexity of constructing a Voronoi diagram that has linear complexity, using the above algorithms, is nearly linear.

Edelsbrunner and Seidel observed the connection between Voronoi diagrams in $\mathbb{R}^d$ and lower envelopes of the distance functions that correspond to the sites in $\mathbb{R}^{d+1}$ (see also Section 2), yielding a very general approach for computing Voronoi diagrams [20].

The Computational Geometry Algorithms Library (CGAL)[1] contains a robust and efficient implementation of a divide-and-conquer algorithm for constructing envelopes of surfaces in three dimensions [21,22]. The theoretical worst-case time complexity of constructing the envelope of n "well-behaved" surfaces in three dimensions using the algorithm is[2] $O(n^{2+\varepsilon})$ [23] (this is also an upper bound, almost tight in the worst case, on the combinatorial complexity of the envelope). As observed below, this near-quadratic running time can arise also in the case of envelopes that represent Voronoi diagrams of linear complexity. This fact poses an obstacle when this divide-and-conquer algorithm is used for computing Voronoi diagrams that have linear complexity, as we aim for algorithms that run in near-linear time. We show, in Section 3, that using randomization in the divide step eliminates the high-complexity cost of this approach, and yields an algorithm with asymptotically efficient expected running time.

We present a general framework for computing various two-dimensional Voronoi diagrams, exploiting the efficient, robust, and general-purpose code of CGAL for constructing envelopes of surfaces. Section 4 explains at a high level how to

[1] `http://www.cgal.org`

[2] A bound of the form $O(f(n) \cdot n^{\varepsilon})$ means that the actual upper bound is $C_{\varepsilon} f(n) \cdot n^{\varepsilon}$, for any $\varepsilon > 0$, where C_{ε} is a constant that depends on ε, and generally tends to infinity as ε goes to 0.

produce new types of Voronoi diagrams, and shows various examples of instantiations of our framework for computing numerous types of Voronoi diagrams, together with other experimental results. Appendix A supplies technical details on the software interface and outlines the concrete steps for producing code for new types of Voronoi diagrams. Further details on the implementation can be found in [24]. The framework is sufficiently general to support diagrams embedded on certain two-dimensional orientable parametric surfaces in $\mathbb{R}^3$, exploiting the facts that the diagrams can be represented as arrangements on the given surface, and that CGAL contains a package that facilitates operations on such arrangements [25]. Section 5 generalizes an algorithm for computing a minimum-width annulus of a set of points in the plane [26] to a set of disks in the plane, and shows how the new framework for Voronoi diagrams can be applied to successfully solve this problem.

The major strength of our approach is its completeness, robustness, and generality, that is, the ability to handle degenerate input, the agility to produce exact results, and the capability to construct diverse types of Voronoi diagrams. The code is designed to successfully handle degenerate input, while exploiting the synergy between generic programming and exact geometric computing, and the divide-and-conquer framework to construct Voronoi diagrams. From a theoretical point of view, the randomized divide-and-conquer envelope approach for computing Voronoi diagrams is shown to be efficient (in an expected sense) and to be asymptotically comparable to other (near-)optimal methods. However, the method uses constructions of bisectors and Voronoi vertices as elementary building blocks, and they must be exact, which makes the concrete running time of our exact implementation inferior to various existing implementations dedicated to specific diagram-types that do not necessarily construct bisectors and vertices (e.g., computing the Delaunay graphs dual to Voronoi diagrams does not require construction of bisectors).

Our software practically supports any kind of nearest-site Voronoi diagrams as well as the corresponding farthest-site Voronoi diagrams, provided that the user supplies a set of basic procedures for manipulating a small number of sites and their bisectors. The implementation of these procedures is an art in itself. However, recent tools for manipulating arbitrary algebraic plane-curves [27, 28] cover a wide range of bisector types, which enables the construction of different types of Voronoi diagrams with greater ease. Table 1 lists the types of diagrams that are currently supported by our implementation. Figure 1 illustrates several types of planar Voronoi diagrams computed with our software. Figure 2 shows two types of Voronoi diagrams on the sphere computed with our software. Both diagrams are composed of geodesic arcs.

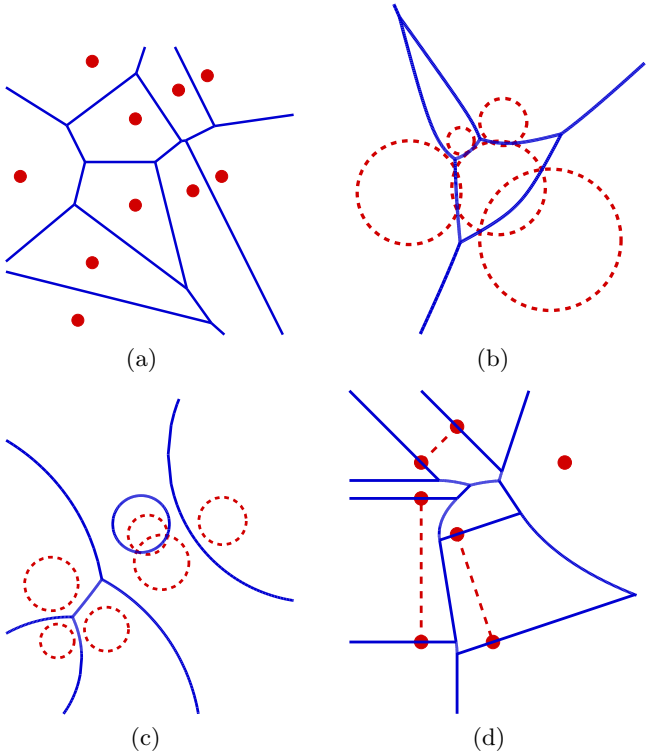


Fig. 1. Various Voronoi diagrams computed with our software. (For the parameters of sites in each diagram, see Table 1.) (a) A standard Voronoi diagram (point sites with L_2 metric). (b) An Apollonius diagram (additively-weighted Voronoi diagram) with disk centers as sites and disk radii as weights. (c) A Möbius diagram with disk centers as sites. The distance from every point on the boundary of a disk to its corresponding site is zero. (d) A Voronoi diagram of segments and points. The sites in (b), (c), and (d) are illustrated with dashed curves.

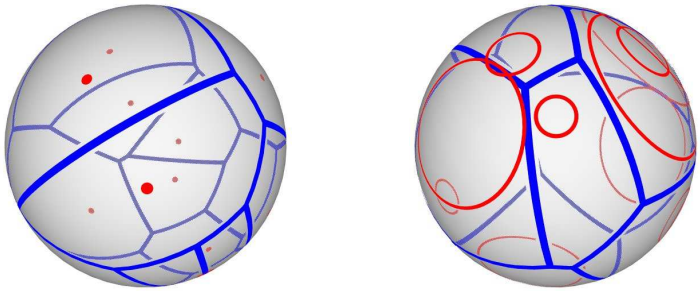


Fig. 2. Voronoi diagrams on the sphere computed with our software. (a) Spherical Voronoi diagram of 14 points. (b) Spherical power diagram of 10 circular sites.

Table 1. Types of Voronoi diagrams currently supported by our implementation, and their bisector classes.

Name	Sites	Distance function	Bisector class
Standard Voronoi diagram	points p_i	$\|\|x - p_i\|\|$	lines
Power diagram	disks (with center c_i and radius r_i)	$\sqrt{(x - c_i)^2 - r_i^2}$	
2-point triangle-area Voronoi diagram	pairs of points $\{p_i, q_i\}$	area of $\triangle x p_i q_i$	pairs of lines
Apollonius diagram	points p_i and weights w_i	$\|\|x - p_i\|\| - w_i$	hyperbolic arcs
Möbius diagram	points p_i with scalars λ_i, μ_i	$\lambda_i(x - p_i)^2 - \mu_i$	circles and lines
Anisotropic diagram	points p_i, with positive definite matrices M_i, and a scalar π_i	$(x-p_i)^t M_i (x-p_i) - \pi_i$	conic arcs
Voronoi diagram of linear objects	interior-disjoint points, segments, rays, or lines	Euclidean distance	piecewise algebraic curves composed of line segments and parabolic arcs
Spherical Voronoi diagram	points on a sphere	geodesic distance	arcs of great circles (geodesic arcs)
Power diagram on a sphere	circles on a sphere	"spherical" power distance[a]	

[a] Given a point p and a circle with center q and radius r on the sphere, the spherical power "proximity" between p and the circle is defined to be $\frac{\cos d(p,q)}{\cos r}$ where $d(p, q)$ is the geodesic distance between p and q [29].

2 Preliminaries

2.1 A Divide-and-Conquer Algorithm for Constructing Voronoi Diagrams

Definition 1 (Lower envelope). *Given a set of bivariate functions $F = \{f_1, \ldots, f_n\}$, $f_i : \mathbb{R}^2 \to \mathbb{R}$ for each i, their* lower envelope $\Psi(x, y)$ *is defined to be their pointwise minimum:*

$$\Psi(x, y) = \min_{1 \le i \le n} f_i(x, y) \ .$$

The *minimization diagram* of F is the subdivision of the xy-plane into maximal relatively-open connected cells, such that the function (or the set of functions) that attains the lower envelope over a specific cell of the subdivision is the same for all points in the cell.

Definition 2 (Voronoi diagram). *Let $O = \{o_1, \ldots, o_n\}$ be a set of n objects in the plane (also called* Voronoi sites*). The* Voronoi diagram $\mathrm{Vor}_\rho(O)$ *of O with respect to a given distance function ρ, is the partition of the plane into maximal*

relatively-open connected cells, where each cell consists of points that are closer to one particular site (or a set of sites) than to any other site.

In certain cases, the distance to a site may depend on various parameters associated with the site; see, for example, the cases of Möbius diagrams or anisotropic diagrams [4] in Table 1. The *bisector* $B(o_i, o_j)$ of two Voronoi sites $o_i, o_j \in O$ is the locus of all points that have an equal distance to both sites, that is

$$B(o_i, o_j) = \{x \in \mathbb{R}^2 \mid \rho(x, o_i) = \rho(x, o_j)\} \ .$$

From the above definitions it is clear that if we let $f_i : \mathbb{R}^2 \to \mathbb{R}$ to be $f_i(x) = \rho(x, o_i)$, for each $i = 1, \ldots, n$, then the minimization diagram of $\{f_1, \ldots, f_n\}$ is exactly the Voronoi diagram of O.

Constructing envelopes of bivariate functions in $\mathbb{R}^3$ can be done with a divide-and-conquer algorithm [23]. The algorithm is adapted to Voronoi diagrams computations as follows:[3] let S be a collection of n sites in the plane and let ρ be some distance function. We partition S into two disjoint subsets S_1 and S_2 of (roughly) equal size, recursively construct their respective Voronoi diagrams $\mathrm{Vor}_\rho(S_1)$ and $\mathrm{Vor}_\rho(S_2)$, and then merge the two diagrams to obtain $\mathrm{Vor}_\rho(S)$.

The merging step starts with overlaying $\mathrm{Vor}_\rho(S_1)$ and $\mathrm{Vor}_\rho(S_2)$. For each face f of the overlay, all its points have a fixed pair of nearest sites s_1 and s_2 from S_1 and S_2, respectively, and the bisector between s_1 and s_2 (restricted to f) partitions f into its portion of points nearer to s_1 and the complementary portion of points nearer to s_2. This results with pieces of the final Voronoi cells. Each feature of the refined overlay is labeled with the site nearest to it. Finally, redundant features are removed (these are vertices and portions of edges from one Voronoi diagram that lie closer to a site in the other Voronoi diagram), and subcells of the same cell are stitched together, to yield the combined final diagram. Figure 3 illustrates the merging step of two Voronoi diagrams.

The asymptotic worst-case time complexity of the divide-and-conquer envelope algorithm (under the natural assumption that the objects and distance function have "constant description complexity") is $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$. Indeed, there are Voronoi diagrams in the plane that have quadratic complexity for which this construction is nearly worst-case optimal. An example for such diagrams are multiplicatively-weighted Voronoi diagrams [30] (see Figure 4 for an illustration). However, in cases where the complexity of the diagrams is sub-quadratic (for most of the cases, linear), we would like the algorithm to run in sub-quadratic (or near-linear) time.

2.2 Constructing Envelopes in CGAL

CGAL is a library of efficient and reliable geometric data structures and algorithms. CGAL follows the exact geometric computation paradigm [31], and adheres to the generic programming paradigm [32] to achieve maximum flexibility without compromising efficiency.

[3] The description assumes general position, even though the algorithm, as well as its implementation, also handle degenerate cases.

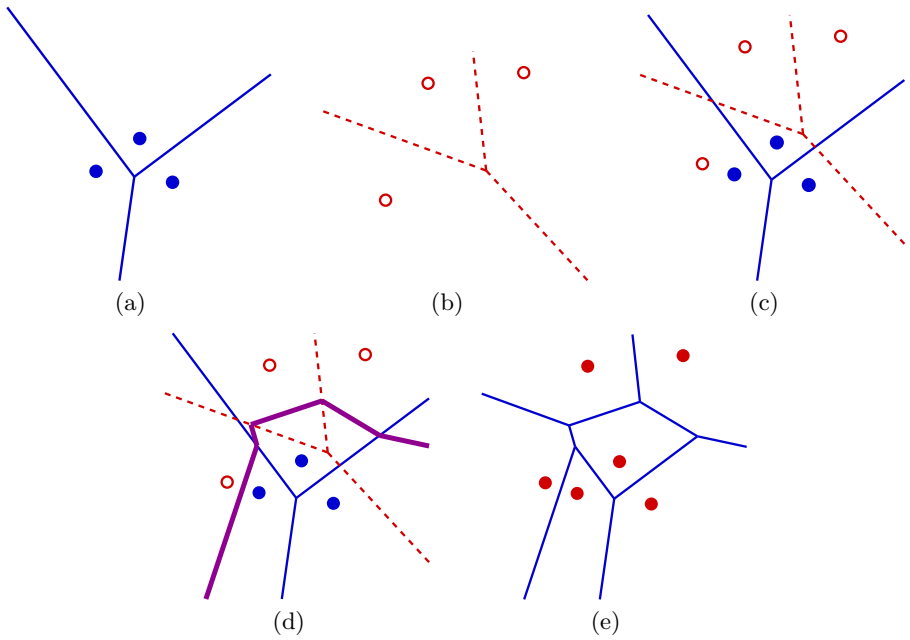


Fig. 3. The merge step of the divide-and-conquer algorithm for computing Voronoi diagrams. (a) The first Voronoi diagram, $\mathrm{Vor}_\rho(S_1)$. (b) The second Voronoi diagram, $\mathrm{Vor}_\rho(S_2)$. (c) The overlay of the two diagrams. (d) The refined overlay. Each face is partitioned to regions that are closer to sites from S_1 and region that are closer to sites from S_2. (e) The final diagram obtained after the removal of redundant features from the refined overlay, and the stitching of the remaining pieces.

The `Arrangement_on_surface_2` package of CGAL supports the construction, the maintenance, and the manipulation of arrangements embedded on certain two-dimensional oriented parametric surfaces in three dimensions, such as spheres, cylinders, tori, etc. [25, 33].

The arrangement package is the basis of the `Envelope_3` package of CGAL, which implements the algorithm mentioned in the previous section for computing the lower (or the upper) envelope of a set of surfaces in three dimensions [21]. While insuring stability, the number of calls to the exact (and slow) geometric predicates is minimized by propagating pre-computed information about the structure of the envelope to neighboring cells in the merge step of the algorithm. The package decouples the topology-related computation from the geometry-related computation, resulting in a generic software which is easy to reuse and adapt. The `Envelope_3` package handles all degenerate situations, and when used with exact numeric types, achieves robustness and produces exact results.

The aforementioned definitions (Section 2.1) can be generalized to lower envelopes (or Voronoi diagrams) projected onto two-dimensional parametric sur-

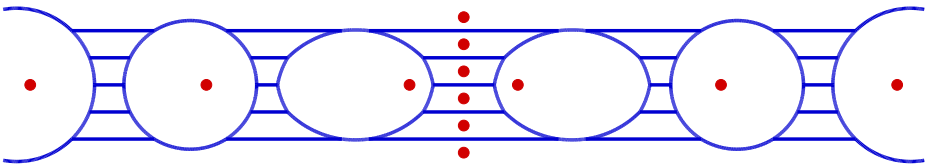


Fig. 4. A worst-case quadratic-size example of a multiplicatively-weighted Voronoi diagram with 12 sites, based on an example by Aurenhammer and Edelsbrunner [30]. The diagram was computed with our software.

faces. Furthermore, the divide-and-conquer algorithm can be used to compute lower (or upper) envelopes over these surfaces.

3 Near-Optimal Bound on the Expected Construction Time

The complexity of the merge step of the algorithm described in Section 2.1 directly depends on the complexity of the overlay of the two partial diagrams. (The cost of the best general algorithm for constructing the overlay is larger by a logarithmic factor than the combined complexity of the input diagrams and of the overlay.) Careless partition of the input sites into two subsets can dramatically slow down the computation. For example, consider the standard L_2-diagram for the following point-set in the plane, $S = \{(i,i)\}_{i=1}^{n/2} \cup \{(-i,i)\}_{i=1}^{n/2}$; see Figure 5(a) for an illustration. If we partition the set into two subsets, to the left and to the right of the y-axis ("deterministic" partitioning strategy), then in the final merge step, the overlay of the two sub-diagrams has $\Theta(n^2)$ complexity. Hence the algorithm runs in $\Omega(n^2)$ time, even though the complexity of the final diagram is only $\Theta(n)$. We argue that when the partitioning is done *randomly*, the expected complexity of the overlay is comparable with the maximum complexity of the diagram for essentially any kind of sites and distance function, and for any possible input. The proof assumes general position, but the results remain true for degenerate cases as well.

Theorem 1. *Consider a specific type of two-dimensional Voronoi diagrams, so that the worst-case complexity of the diagram of any set of at most n sites is $F(n)$. Let S be a set of n sites. If we randomly split S into two subsets S_1 and S_2, by choosing at random for each site, with equal probability, the subset it belongs to, then the expected complexity of the overlay of the Voronoi diagram of S_1 with the Voronoi diagram of S_2 is $O(F(n))$.*

Proof. Each vertex of the overlay is either a vertex of $\mathrm{Vor}_\rho(S_1)$, a vertex of $\mathrm{Vor}_\rho(S_2)$, or a crossing between an edge of $\mathrm{Vor}_\rho(S_1)$ and an edge of $\mathrm{Vor}_\rho(S_2)$ (in degenerate cases, this is a non exclusive disjunction). The number of vertices of the first two kinds is $O(F(n))$, so it suffices to bound the expected number of

crossings between edges of $\mathrm{Vor}_\rho(S_1)$ and of $\mathrm{Vor}_\rho(S_2)$. Such a crossing is a point u that is defined by four sites, $p_1, q_1 \in S_1$ and $p_2, q_2 \in S_2$, so that u lies on the Voronoi edge $e(p_1, q_1)$ of $\mathrm{Vor}_\rho(S_1)$ that bounds the cells of p_1 and q_1, and on the Voronoi edge $e(p_2, q_2)$ of $\mathrm{Vor}_\rho(S_2)$ that bounds the cells of p_2 and q_2. Without loss of generality, assume that $\rho(u, p_1) = \rho(u, q_1) < \rho(u, p_2) = \rho(u, q_2)$.

A simple but crucial observation is that u must also lie on the Voronoi edge between the cells of p_1, q_1 in the *overall* diagram $\mathrm{Vor}_\rho(S)$. Indeed, if this were not the case then there must exist another site $s \in S$ so that u is nearer to s than to p_1, q_1. But then s cannot belong to S_1, for otherwise it would prevent u from lying on the Voronoi edge of p_1, q_1. For exactly the same reason, s cannot belong to S_2 — it would then prevent u from lying on the Voronoi edge of p_2, q_2 in that diagram. This contradiction establishes the claim.

Define the *weight* k_u of u to be the number of sites s satisfying

$$\rho(u, p_1) = \rho(u, q_1) < \rho(u, s) < \rho(u, p_2) = \rho(u, q_2).$$

Clearly, all these k_u sites must be assigned to S_1.

In other words, for any crossing point u between two Voronoi edges $e(p_1, q_1)$, $e(p_2, q_2)$, with weight k_u (with all the corresponding k_u sites being farther from u than p_1, q_1 and nearer than p_2, q_2), u appears as a crossing point in the overlay of $\mathrm{Vor}_\rho(S_1)$, $\mathrm{Vor}_\rho(S_2)$ if and only if the following three conditions (or their symmetric counterparts, obtained by reversing the roles of S_1 and S_2) hold: (i) $p_1, q_1 \in S_1$; (ii) $p_2, q_2 \in S_2$; and (iii) all the k_u sites that contribute to the weight are assigned to S_1. This happens with probability $\dfrac{1}{2^{k_u+3}}$.

Hence, if we denote by N_w (resp., $N_{\le w}$) the number of crossings of weight w (resp., of weight at most w), the expected number of crossings in the overlay is

$$\sum_{w \ge 0} \frac{N_w}{2^{w+3}} = O\left(\sum_{w \ge 0} \frac{N_{\le w}}{2^w} \right), \tag{1}$$

where the right-hand side is obtained by substituting $N_w = N_{\le w} - N_{\le w-1}$, and by a simple rearrangement of the sum.

We can obtain an upper bound on $N_{\le w}$ using the Clarkson-Shor technique [34]. Specifically, denote by $N_w(n)$ (resp., $N_{\le w}(n)$) the maximum value of N_w (resp., $N_{\le w}$), taken over all sets of n sites in the class under consideration. Then, since a crossing is defined by four sites, we have

$$N_{\le w}(n) = O\left(w^4 N_0(n/w)\right).$$

Note that if a crossing u, defined by p_1, q_1, p_2, q_2, has weight 0 then p_1, q_1, p_2, q_2 are the four nearest sites to u. The number of such quadruples is thus upper bounded by the complexity of the *fourth-order* Voronoi diagram of some set (actually, a random sample) S_0 of n/w sites.

We claim that the complexity of the fourth-order Voronoi diagram of n sites is $O(F(n))$. Indeed, any quadruple p_1, q_1, p_2, q_2 of four nearest sites to some

point u can be charged (essentially, in a unique manner) to a face of the fourth-order diagram (the one containing u). Each such face can in turn be charged either to one of its vertices, or to its rightmost point, or to a point at infinity on one of its edges. Assuming general position, each such boundary point can be charged at most $O(1)$ times. Now, another simple application of the Clarkson-Shor technique shows that the number of these vertices and boundary points is $O(F(n))$ — each of them becomes a feature of the (0-order) Voronoi diagram if we remove a constant number of sites, which happens with large probability when we sample a constant fraction of the sites.

In other words, we have $N_{\leq w}(n) = O(w^4 F(n/w)) = O(w^4 F(n))$. Substituting this into (1), we obtain an upper bound of $O(F(n))$ on the complexity of the overlay, as claimed. □

As we aim to compute Voronoi diagrams of a large variety of types, we use a sweep-line based overlay algorithm that exhibits good practical performance, and incurs a mere logarithmic factor over the optimal computing time, namely $O(F(n)\log n)$. In particular we use the overlay operation provided by the `Arrangement_on_surface_2` package. Combined with the "master recurrence" $T(n) = 2T(\frac{n}{2}) + O(F(n)\log n)$ for the expected running time $T(n)$ of the algorithm, we obtain the following special cases:

Corollary 1. *For any specific type of two-dimensional Voronoi diagrams, so that the worst-case complexity of the diagram of any set of at most n sites is $O(n)$, the divide-and-conquer envelope algorithm computes the diagram in expected $O(n\log^2 n)$ time. If the worst-case complexity $F(n)$ is $\Omega(n^{1+\varepsilon})$ then the expected running time is $O(F(n)\log n)$.*

When the diagram is a convex subdivision, one can carry out the merge step more efficiently, in linear $O(F(n))$ expected time using the procedure described by Guibas and Seidel [35]. In particular, we have:

Corollary 2. *The L_2-Voronoi diagram of n points in the plane, or the power diagram of n disks in the plane, can be computed using the randomized divide-and-conquer envelope algorithm in expected optimal $O(n\log n)$ time.*

Remark 1. The analysis used to prove Theorem 1 can easily be extended to the case of lower envelopes of arbitrary collections of bivariate functions (of constant description complexity). As a result, we get the following:

Corollary 3. *Let $\mathcal{G}$ be a collection of n bivariate functions of constant description complexity each, and let $F(m)$ be an upper bound on the complexity of the lower envelope of any subcollection of at most m functions. Then the expected complexity of the overlay of the minimization diagrams of two subcollections $\mathcal{G}_1$ and $\mathcal{G}_2$, obtained by randomly partitioning $\mathcal{G}$, as above, is $O(F(n))$. Consequently, the lower envelope of $\mathcal{G}$ can be constructed by the above randomized divide-and-conquer technique, in expected time $O(F(n)\log n)$, provided that $F(n) = \Omega(n^{1+\varepsilon})$, for some $\varepsilon > 0$. The expected running time is $O(n\log^2 n)$ when $F(n) = O(n)$.*

4 Examples and Experimental Results

Figure 5 demonstrates the effect of executing the recursion by randomly partitioning the sites into two subsets of equal size, when constructing standard Voronoi diagrams with the worst-case inputs mentioned in Section 3. If we partition the set into two subsets, to the left and to the right of the y-axis ("deterministic" partitioning strategy), the overlay of the two Voronoi diagrams has $\Theta(n^2)$ complexity. Figure 5(b) illustrates the fact, established above, that partitioning the sites randomly, results in running time nearly-linear in the number of sites.

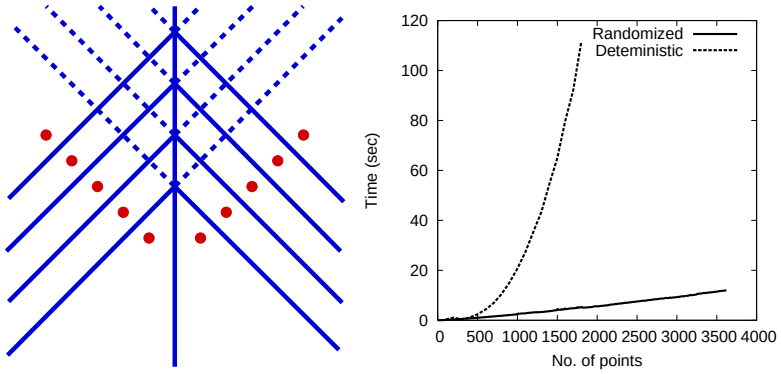


Fig. 5. Effect of randomization. (a) A worst-case example of 10 point sites in the case of the standard Voronoi diagram. The dashed segments are segments that exist in the overlay (if we use the deterministic partition strategy) but will not be present in the final diagram. (b) The graph shows the running time in seconds as a function of the number of sites (arranged in a worst case constellation as in (a)) in both the worst-case deterministic and the randomized partitioning strategies.

A prime advantage of our framework is the ability to compute new types of Voronoi diagrams with relative ease. Despite the fact that the envelope code is used to compute Voronoi diagrams, the user does not have to define the surfaces explicitly, and does not have to know the algorithmic details of constructing envelopes of surfaces and their minimization diagrams. Creating a new type of Voronoi diagrams amounts only to the provision of a small set of geometric types and operations on a small number of sites and bisectors (e.g., determine which is the closer site, among two given sites, to a given point, construct the bisector of two sites, etc.), gathered in a traits class; see Appendix A for further details.

The traits class of a new type of diagrams is based on a traits class for the arrangement package, which supplies the handling and manipulation of bisector curves of pairs of sites. A Voronoi diagram, the bisectors of which can be represented with an existing traits class for the arrangement package can be easily developed. Existing traits classes support bounded or unbounded lin-

ear curves (line segments, rays, and lines), circles, circular and linear segments, conic arcs [33], Bézier curves, and real algebraic plane curves of arbitrary degree [27, 28]. These traits classes enable the construction of all the Voronoi diagrams shown in Figure 1 and listed in Table 1, and of further types of Voronoi diagrams with bisectors that can be represented by curves supported by these traits classes.

The framework supports a vast variety of Voronoi diagrams with different properties. Existing frameworks for computing Voronoi diagrams usually support a specific family of Voronoi diagrams, e.g., planar Voronoi diagrams of points under the L_p metric, planar Voronoi diagrams of general sites under the Euclidean metric, etc. Using an envelope-construction based algorithm allows our framework to support many families of two-dimensional Voronoi diagrams; linear Voronoi diagrams as well as Voronoi diagrams with quadratic complexity, and Voronoi diagrams with two-dimensional bisectors can be implemented. The diagrams do not have to conform with the definition of abstract Voronoi diagrams [6]. For example, anisotropic diagrams, which violate the theoretical requirement from an abstract Voronoi diagram that all bisectors divide the plane into two unbounded regions, and easily constructed by our approach.

The implementation of the `Envelope_3` package mainly makes use of two operations supported by the arrangement package: (i) sweep-based overlay operation, which is used to overlay two planar minimization diagrams, and (ii) zone computation-based insertion operation, which is used to insert projected intersection curves (of the surfaces) into the current planar diagram, which refine it by partitioning some of its cells. The new `Arrangement_on_surface_2` package extends the aforementioned operations, that is, the sweep-line and zone-computation, to support arrangements on two-dimensional parametric surfaces [25]. Thus, we extensively reuse code developed for planar Voronoi diagrams to compute Voronoi diagrams embedded on the sphere by extending the `Envelope_3` code to work together with the new `Arrangement_on_surface_2` package, and handle minimization diagrams that are embedded on two-dimensional parametric surfaces. We compute Voronoi diagrams of points on a sphere and power diagrams on a sphere using the framework together with the traits class for arcs of great circles on a sphere (see [36, 37] for more information, and Figures 2 and 6(d) for illustrations).

Our framework, like the `Arrangement_on_surface_2` package of CGAL, handles all degenerate situations that arise in the computation of Voronoi diagrams, as long as the supplied traits class handles a small number of degenerate cases. For example, the traits class should be able to detect whether two curves overlap, when trying to intersect them. Figure 6 depicts various degenerate scenarios that are properly handled by our framework and the traits classes. Figure 7 shows scenarios of Voronoi diagrams of points, linear segments, and lines containing certain degeneracies.

Given a traits class for nearest-site Voronoi diagrams, our framework can also be used to compute the respective farthest-site Voronoi diagrams by constructing

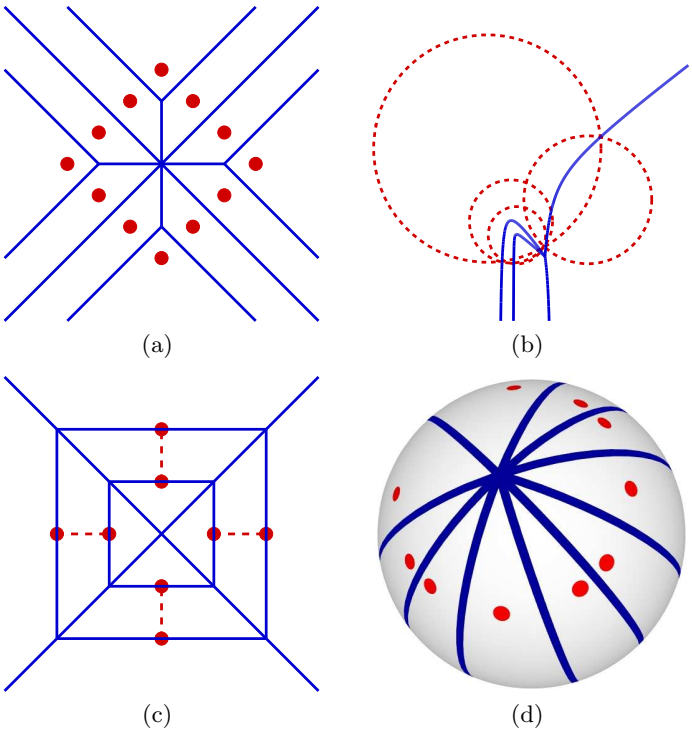


Fig. 6. Degenerate Voronoi diagrams computed with our software. (a) A degenerate standard Voronoi diagram. (b) A degenerate Apollonius (additively-weighted Voronoi) diagram. (c) A degenerate Voronoi diagram of segments and points. (d) A degenerate spherical Voronoi diagram. The sites in (b) and (c) are illustrated with dashed curves.

upper, rather than lower, envelopes. Figure 8 shows various farthest-site Voronoi diagrams computed with our framework.

Voronoi diagrams are represented as exact CGAL arrangements, and their vertices, edges, and faces can easily be traversed while obtaining the coordinates of the vertices of the diagram to any desired precision, if the situation so requires.

The computed diagrams can be passed as input to consecutive operations supported by the `Arrangement_on_surface_2` package and its derivatives. We get plenty of additional functionalities for free. Among those are (i) point-location queries and vertical-ray shooting, (ii) the ability to perform aggregated insertion of additional curves, (iii) computing the zone of a curve inside a Voronoi diagram and inserting curves in an incremental fashion to an existing diagram, (iv) the ability to remove existing edges of the diagram, (v) overlaying two (or more) Voronoi diagrams or arrangements, (vi) the ability to attach user-defined data to all the geometric objects (i.e., points and curves) and to all the topological primitives (i.e., vertices, edges, and faces) required in certain applications, and (vii) the ability to use the BOOST graph library to apply various graph algorithms

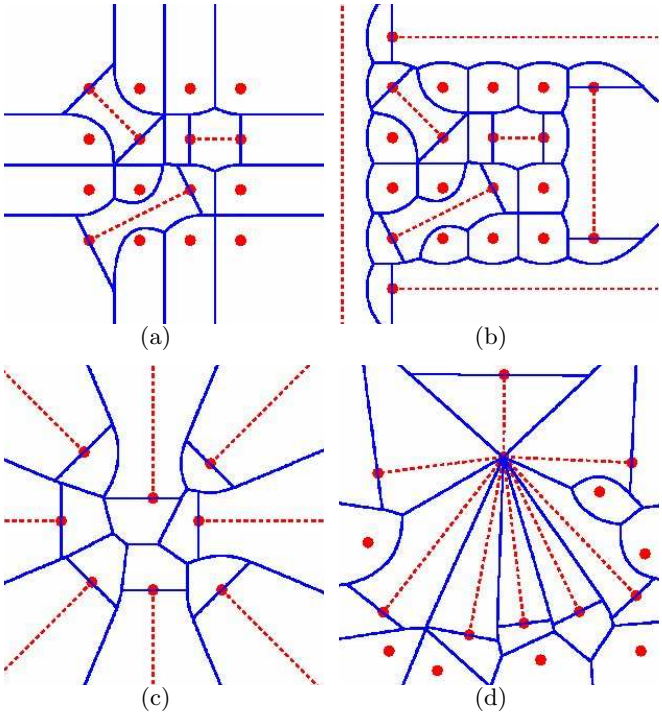


Fig. 7. Voronoi diagrams of various sets of linear objects as induced by the Euclidean metric computed with our framework. The sites are illustrated with dashed curves. (a) The Voronoi diagram of 4×4 grid with three line segments connecting 6 grid points. (b) The Voronoi diagram of a set of sites identical to the set in (a) with a line, a segment, and two rays around the grid points. (c) The Voronoi diagrams of 8 rays in the plane. (d) The Voronoi diagram of 8 segments and 7 isolated points. All segments intersect at one point, which is one of their endpoints.

on the diagram and on its dual structure [33]. For example, we can use the various point-location algorithms that come with the arrangement package to solve the corresponding post-office problems [38].

Overlaying two (or more) Voronoi diagrams is immediate, a fact that we used to implement the algorithm for computing a minimum-width annulus of a set of disks in the plane, described in detail in Section 5. There are also specific applications for the overlay of Voronoi diagrams, for example, representing the local zones of two competing telecommunication operators [39].

5 Minimum-Width Annulus of Disks

In this section we describe an application of our framework that demonstrates its generality and usefulness. We use our tools to compute a minimum-width

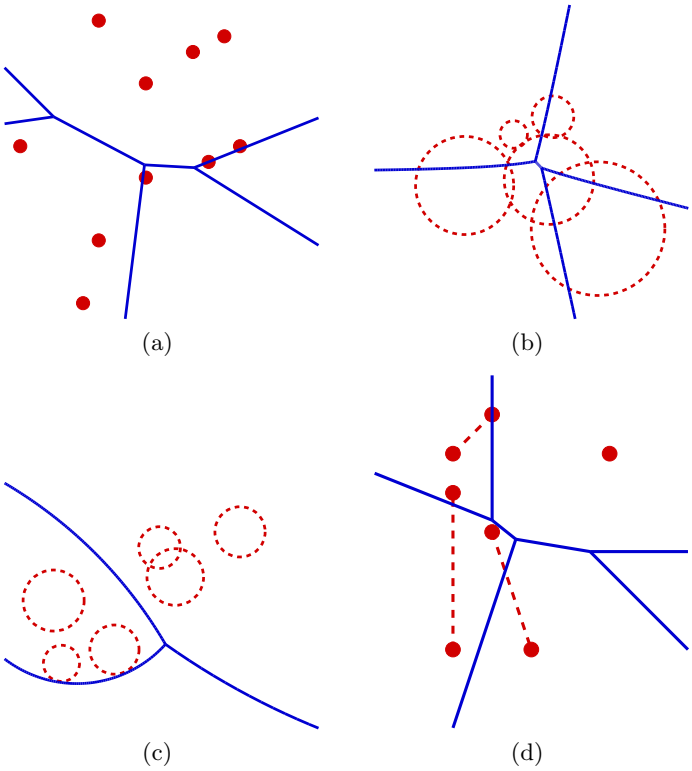


Fig. 8. Farthest-site Voronoi diagrams of sites identical to the sites in Figure 1, computed with our software. (a) A farthest Euclidean Voronoi diagram of points. (b) A farthest Apollonius diagram (additively-weighted Voronoi diagram). (c) A farthest Möbius diagram. (d) A farthest Euclidean Voronoi diagram of segments. The sites in (b), (c), and (d) are illustrated with dashed curves.

annulus containing a set of disks in the plane, by constructing and overlaying two different Voronoi diagrams defined on the input disks.

An *annulus* is the bounded area between two concentric circles. The width of an annulus is the difference between the radius of its outer circle and the radius of its inner circle. Given a set of objects in the plane (disks in our case) the objective is to find a minimum-width annulus (MWA) containing those objects, if one exists.[4] For point sets, a minimum-width annulus can be found by computing the overlay of the nearest-site and farthest-site Voronoi diagrams of the points. The center of the desired annulus must be a vertex of the overlay, and, in fact, an intersection point of the two diagrams [26]. A similar approach handles the case of disks but the diagrams have to be defined more carefully.

[4] Generally, when the width of the set is smaller than the width of any containing annulus there is no minimum-width annulus. For example, four collinear points in the plane have no minimum-width annulus.

Consider the following farthest-point distance function ρ from a point $p \in \mathbb{R}^2$ to a set of points $S \subset \mathbb{R}^2$:

$$\rho(p, S) = \sup_{x \in S} ||p - x|| \ ,$$

which measures the farthest distance from the point p to the set S. Consider the farthest-site Voronoi diagram with respect to this distance function. We call this diagram the "farthest-point farthest-site" Voronoi diagram. The distance function $\rho(p, S)$ becomes the Euclidean distance when the set S consists of a single point. However, this is not the case when the set S is a disk in the plane, say.

The following observation establishes a connection between the farthest-point farthest-site Voronoi diagram of a set of disks and the Apollonius diagram, and comes in handy below.

Observation 1. *Let $\mathcal{D}$ be a set of disk in $\mathbb{R}^2$. The farthest-point farthest-site Voronoi diagram of $\mathcal{D}$ is identical to the farthest-site Apollonius diagram of the disks with negated radii.*

Indeed, for a disk d with center c and radius r, the farthest-point distance function is $\rho(p, d) = ||p - c|| + r$. On the other hand, the Apollonius distance function is defined to be $\rho(p, d) = ||p - c|| - r$. Hence the farthest-point distance function is identical to the Apollonius distance with "negative radii."

We prove that there is a minimum-width annulus (in case one exists), the center of which is a vertex of the overlay of the Apollonius diagram and the farthest-point farthest-site Voronoi diagram of the disks.

Recall that in the case of point sets the center of a minimum-width annulus is an intersection point of an edge of the nearest-site Voronoi diagram and an edge of the farthest-site Voronoi diagram of the points. In the case of disks, this is not always true. Figure 9 shows a set of disks located at $(2, 0)$, $(0, 3)$, $(-4, 0)$, and $(0, -5)$ with radii 1, 2, 3, and 4, respectively. In this setting, the annulus centered at the origin with radii 1 and 9 is a minimum-width annulus, as its width is equal to a diameter of one of the disks. The origin does not lie on an edge of the farthest-point farthest-site Voronoi diagram, as the outer circle touches only one disk. Nevertheless, the origin is a vertex of the Apollonius diagram, and thus also of the overlay.

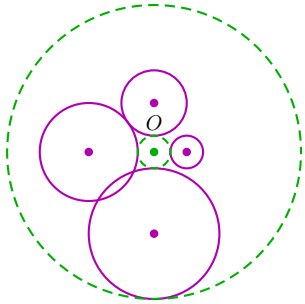


Fig. 9. The outer circle of a MWA that touches one disk.

Let $\mathcal{D} = \{d_1, \ldots, d_n\}$ be a collection of disks in the plane such that non of the disks is fully contained in the union of the others, and formally, for all i $d_i \not\subseteq \bigcup_{j \neq i} d_j$. For simplicity of exposition, we assume here that $n \geq 3$; the case of $n < 3$ is simple to handle. We show that there is a minimum-width annulus, the bounding circles of which intersect the disks of $\mathcal{D}$ in at least 4 points, arguing as in [40].

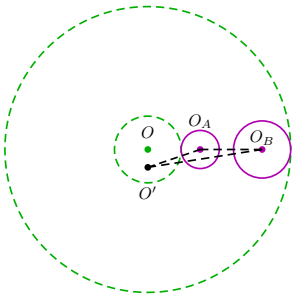
Observation 2. *If N is a minimum-width annulus containing $\mathcal{D}$ then each of the inner circle and the outer circle of N touches a disk of $\mathcal{D}$.*

Indeed, if this is not the case, then we can shrink the outer circle of the annulus or expand the inner circle until each of them touches a disk. Thus, fixing a point p in the plane as the center of a containing annulus fixes an annulus bounding $\mathcal{D}$, of smallest width, for this center. We call this annulus the *tight annulus* fixed at p, denote it as N_p and its width as W_{N_p}. Let $\mathcal{I}_p, \mathcal{O}_p \subseteq \mathcal{D}$ denote the set of disks that are touched by the inner and outer circles of N_p, respectively.

Definition 3 (Neighboring center in direction $\boldsymbol{d}$). *For a point p and a direction $\boldsymbol{d}$ in the plane, consider the ray r emanating from p in direction $\boldsymbol{d}$. r can be divided into maximal contiguous cells, such that every point x in a cell (regarded as the center of a tight annulus) obtains the same $\mathcal{I}_x$ and $\mathcal{O}_x$. Let p' be an interior point of the cell of p, or, if p comprises a single-point cell, an interior point of the neighboring cell in the direction of the ray. We call p' a neighboring center of p in direction $\boldsymbol{d}$.*

Lemma 1. *There is no minimum-width annulus N_O containing $\mathcal{D}$ and two different disks $A, B \in \mathcal{D}$ such that $\mathcal{I}_O = \{A\}$, $\mathcal{O}_O = \{B\}$, and the centers of A, B and N_O are collinear.*

Proof. Suppose to the contrary that such a minimum-width annulus N_O exists. Denote by O_A and O_B the respective centers of A and B and by R_A and R_B their respective radii. O_A is on the segment OO_B; see the figure to the right for an illustration. Choose O' to be a neighboring center of O in a direction perpendicular to OO_B. Then, by the triangle inequality, $|O_AO_B| > |O'O_B| - |O'O_A|$, and

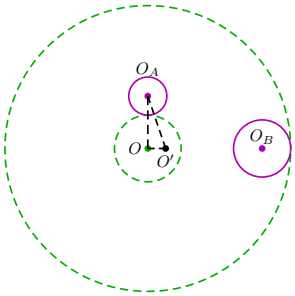


$$\begin{aligned} W_{N_O} &= (|OO_B| + R_B) - (|OO_A| - R_A) \\ &= |O_AO_B| + R_B + R_A > |O'O_B| - |O'O_A| + R_B + R_A \ . \end{aligned}$$

Thus, $N_{O'}$ has a smaller width, which is a contradiction. □

Lemma 2. *If N_O is a minimum-width annulus containing $\mathcal{D}$ then either $|\mathcal{I}_O| > 1$, or $|\mathcal{O}_O| > 1$, or there is another annulus $N_{O'}$ of minimum-width containing $\mathcal{D}$ such that $|\mathcal{I}_{O'}| > 1$ or $|\mathcal{O}_{O'}| > 1$.*

Proof. Suppose that $\mathcal{I}_O = \{A\}$ and $\mathcal{O}_O = \{B\}$. Let R_A and R_B denote the respective radii of A and B, and let O_A and O_B denote their respective centers.



In case that $A = B$, we can move O along the line that goes through O and O_A away from O_A. The new annulus, centered at O', has the same width but the radii of its bounding circles are larger. We keep moving O until one of the circles intersects another disk.

If $A \neq B$ then from Lemma 1, O_A, O_B, and O are not collinear (see the figure on the right for an illustration). Choose O' to be a neighboring center of O in the direction of O_B. Then, by the triangle inequality, $|OO_A| < |OO'| + |O'O_A|$ and

$$\begin{aligned} W_{N_O} &= |OO_B| + R_B - (|OO_A| - R_A) = |OO'| + |O'O_B| + R_B - |OO_A| + R_A \\ &> |O'O_B| - |O'O_A| + R_B + R_A \ . \end{aligned}$$

Thus the annulus $N_{O'}$ has a smaller width then N_O, which is a contradiction. □

Theorem 2. *If there is a minimum-width annulus containing $\mathcal{D}$, then there is a minimum-width annulus N_O such that $|\mathcal{I}_O| + |\mathcal{O}_O| \geq 4$.*

Proof. Suppose to the contrary that there is a minimum-width annulus N_O containing the disks of $\mathcal{D}$, but there is no minimum-width annulus for which the total number of contact points of its bounding circles with the disks of $\mathcal{D}$ is at least 4.

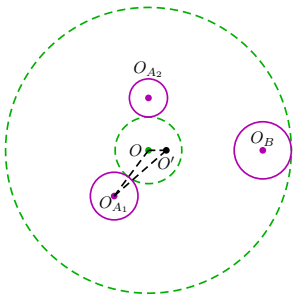
From Lemma 2, we may assume that either $|\mathcal{I}_O| > 1$, or $|\mathcal{O}_O| > 1$. The only cases that we need to rule out are $|\mathcal{I}_O| = 2$ and $|\mathcal{O}_O| = 1$, or $|\mathcal{I}_O| = 1$ and $|\mathcal{O}_O| = 2$.

Case I: Assume that $\mathcal{I}_O = \{A_1, A_2\}$ and $\mathcal{O}_O = \{B\}$. Let R_{A_1}, R_{A_2}, and R_B denote the respective radii of A_1, A_2, and B, and let O_{A_1}, O_{A_2}, and O_B denote their respective centers.

If $B \in \mathcal{I}_O$ then we can move O along the bisector of A_1 and A_2 (which is one branch of a hyperbola), increasing its distance from A_1 and A_2. W_{N_O} (which equals to $2 \cdot R_B$) will stay the same. As the radii of the bounding circles grows one of the circles will eventually intersect another disk.

If $B \notin \mathcal{I}_O$ then there are two sub-cases:

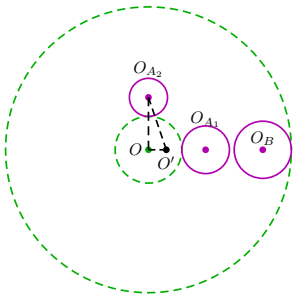
Case I(a): Neither of O_{A_1} and O_{A_2} lies on the segment OO_B (see the figure to the right for an illustration). Choose O' to be the neighboring center near O in the direction of O_B. Then again from the triangle inequality we get $|OO'| > |OO_{A_i}| - |O'O_{A_i}|$, for $i = 1, 2$, and



$$\begin{aligned} W_{N_O} &= |OO_B| + R_B - (|OO_{A_i}| - R_{A_i}) \\ &= |OO'| + |O'O_B| - |OO_{A_i}| + R_B + R_{A_i} \\ &> |O'O_B| - |O'O_{A_i}| + R_B + R_{A_i} \ , \end{aligned}$$

for $i = 1, 2$. Since $W_{N_{O'}}$ is the maximum of the last two expressions, it follows that $W_{N_O} > W_{N_{O'}}$, which is a contradiction.

Case I(b): Only one of the points O_{A_1} and O_{A_2} can lie on the segment OO_B, since if both of them lie on the segment then one of the respective disks is fully contained in the other, which contradicts our basic assumption. Without loss of generality, assume that O_{A_1} is on the segment OO_B (see the figure on the right for an illustration). For every point P on the open segment OO_{A_1} we get



$$\begin{aligned} |OP| + |PO_{A_2}| - R_{A_2} &> |OO_{A_2}| - R_{A_2} \\ &= |OO_{A_1}| - R_{A_1} = |OP| + |O'O_{A_1}| - R_{A_1} \ , \end{aligned}$$

and therefore

$$|PO_{A_2}| - R_{A_2} > |PO_{A_1}| - R_{A_1} \ .$$

This means that neighboring center O' in the direction of O_{A_1} from O satisfies $\mathcal{O}_{O'} = \mathcal{O}_O$ and $\mathcal{I}_{O'} = \mathcal{I}_O \setminus \{A_2\}$. The annulus $N_{O'}$ has the same width as N_O and therefore is also a minimum-width annulus, the center of which is collinear with O_{A_1} and O_B. This, however, is a contradiction to Lemma 1.

Case II: $|\mathcal{I}_O| = 1$ and $|\mathcal{O}_O| = 2$. The proof of this case is similar to the first case and is therefore omitted.

□

From Theorem 2 it is clear that if there is a minimum-width annulus then there is a minimum-width annulus centered at a point O that falls into one of the following three cases:

1. $|\mathcal{I}_O| \geq 3$ and $|\mathcal{O}_O| = 1$. This means that O coincides with a vertex of the Apollonius diagram of the disks.
2. $|\mathcal{I}_O| = 1$ and $|\mathcal{O}_O| \geq 3$. This means that O coincides with a vertex of the farthest-point farthest-site Voronoi diagram of the disks.

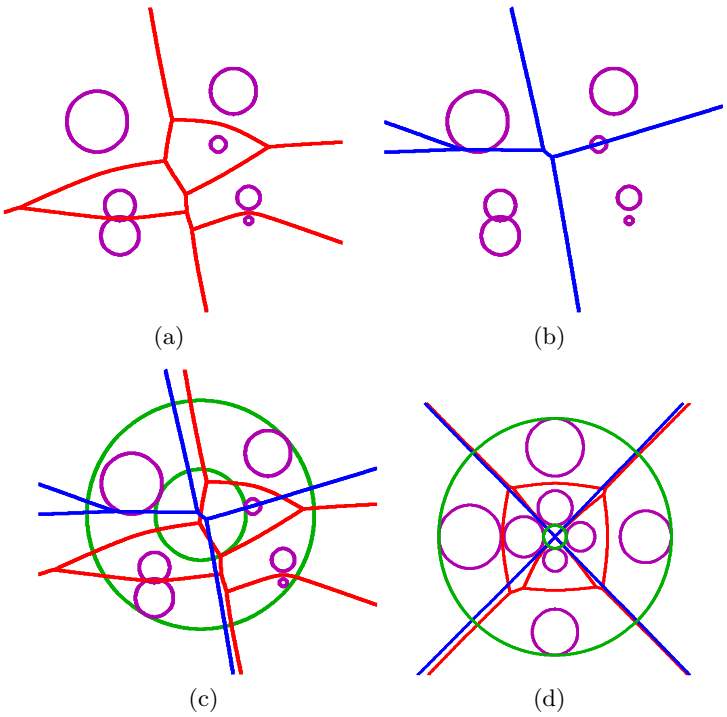


Fig. 10. Computing a minimum-width annulus of a set of disks. (a) The Apollonius diagram of the set of disks. (b) The farthest-point farthest-site Voronoi diagram of the set of disks. (c) A minimum-width annulus of the set of disks. The center of the annulus is a vertex in the overlay of the Apollonius and the farthest-point farthest-site Voronoi diagrams. (d) A highly degenerate scenario for computing a minimum-width annulus of a set of disks. Four disks induce the vertex of the farthest-point farthest-site Voronoi diagram and other four disks induce a vertex of the Apollonius diagram at the same point.

3. $|\mathcal{I}_O| \geq 2$ and $|\mathcal{O}_O| \geq 2$. In this case O lies on a Voronoi edge of the Apollonius diagram of the disks and on a Voronoi edge of the farthest-point farthest-site Voronoi diagram of the disks.

We therefore construct each of the diagrams (the Apollonius and the farthest-point farthest-site Voronoi diagrams), using our general divide-and-conquer machinery, and then overlay them. For each vertex O of the overlay, we retrieve four relevant disks (either three touching the inner circle and the one touching the outer circle, in case 1, or three touching the outer circle and the one touching the inner circle, in case 2, or two pairs of disks touching respectively the inner and outer circles, in case 3), and compute the width of the resulting annulus for those four disks. We output the annulus of the smallest width. Figures 10(a)–(c) illustrate various steps of the algorithm for computing a minimum-width annulus of a set of disks. Figure 10(d) depicts a highly degenerate input that is being handled properly by our implementation.

Apollonius diagrams are of linear complexity in the number of sites and conform to the *abstract Voronoi diagram* definition [6]. Hence, farthest-site Apollonius diagrams are *farthest abstract Voronoi diagrams* [41], and are, therefore, of linear complexity too. Thus Corollary 1 is applied to yield an expected construction time of $O(n \log^2 n)$ for both diagrams. Overlaying the two diagrams with the sweep-based algorithm has $O((n+k) \log n)$ worst-case time complexity where k is the number of intersections between edges of the diagrams. The total expected running time of the algorithm is therefore $O(n \log^2 n + k \log n)$, where k can be[5] $\Theta(n^2)$.

The implementation of the Apollonius diagram is based on the algebraic curves traits for the arrangement package [27, 28], as the bisector of two sites is, in general, one branch of a hyperbola. In order to determine which of the branches of the hyperbola constitutes the bisector of the two disks, we first lexicographically sort the x-monotone sub-curves of the hyperbola. Then, by considering the coordinates of the centers of the disks and their radii, we determine which of the two branches is the bisector. For example, in the case of a vertical asymptote, if the center of the disk of the smaller radius is to the left of the center of the disk of the larger radius, the desired branch is the left branch of the hyperbola, otherwise the desired branch is the right branch.

Given this implementation of the Apollonius traits class, the general framework allows us to easily compute the farthest-point farthest-site Voronoi diagram by computing the farthest Voronoi diagram of the disks with "negative radii." The overlay operation is readily available in the arrangement package.

Table 2 shows the sizes of the constructed diagrams in each phase of the algorithm and the time consumption (in seconds) of the execution of each phase. The vertices of the overlay are the candidates for the center of the annulus. The experiments were carried out on an Intel® Core™2 Duo 2GHz processor with 1GB memory running Linux operating system.

6 Conclusions and Future Work

Our framework together with the existing traits classes of the arrangement package provide the means to produce various Voronoi diagrams in an exact and robust manner. Moreover, the theoretical bound on the expected running time of our algorithm is nearly optimal.

Future work is to enrich the variety of Voronoi diagrams that can be computed with our software to include Voronoi diagrams of circular arcs, Bregman Voronoi diagrams [43], Voronoi diagrams in the hyperbolic Poincaré halfplane [44] or in the Poincaré hyperbolic disk [45], Hausdorff Voronoi Diagrams [2], and Voronoi diagrams on different surfaces, for example, cylinders or tori. Some of the above diagrams have bisectors that can be handled with existing traits

[5] It is reasonable to assume that the expected time complexity is, in many cases, smaller. Though not proven for disks, it is known that, for random point sets the expected number of intersections between the farthest-site and the nearest-site Voronoi diagrams is sub-linear [42].

Table 2. Time consumption (in seconds) of minimum-width annuli computation and sizes of the corresponding constructed diagrams. FPFS — farthest-point farthest-site, V, E, F — the number of vertices, edges, and faces of the diagram, T — the time in seconds consumed by the respective phase, C — the time in seconds consumed during the comparison of the candidates for the annulus center. "*dgn_**" are input data in a degenerate setting (see Figure 10(d) for an illustration), and "*rnd_**" are input data in a random setting.

Input	Apollonius				FPFS VD				Overlay				C	Total Time
	V	E	F	T	V	E	F	T	V	E	F	T		
rnd_50	84	128	45	6.74	10	21	12	2.40	126	213	88	0.44	0.57	10.16
rnd_100	162	242	81	17.47	17	35	19	5.36	238	395	158	0.81	1.46	25.12
rnd_200	317	460	144	44.07	15	31	17	11.16	416	659	244	1.28	1.33	57.86
rnd_500	672	967	296	136.46	16	33	18	29.30	775	1174	400	1.85	1.97	169.59
dgn_50	59	106	48	5.74	1	25	25	1.94	100	213	114	0.84	0.30	8.83
dgn_100	134	232	99	16.57	1	50	50	5.83	239	492	254	2.93	1.11	26.46
dgn_200	304	502	199	42.60	1	100	100	15.77	581	1156	576	7.60	2.78	68.76
dgn_500	785	1262	478	154.37	1	239	239	56.21	1645	3221	1577	37.93	7.94	256.47

classes available by the arrangement package. Another objective is to generalize the computation of minimum-width annuli to other types of objects, for example, ellipses.

Acknowledgements. The authors are grateful to Efi Fogel and Eric Berberich for many helpful suggestions and fruitful cooperation, and to Ron Wein, Michal Meyerovitch, and Barak Raveh for their cooperation and support of this work.

References

1. Shamos, M.I., Hoey, D.: Closest-point problems. In: Proceedings of the 16th IEEE Symposium on the Foundations of Computer Science. (1975) 151–162
2. Okabe, A., Boots, B., Sugihara, K., Chiu, S.N.: Spatial Tessellations: Concepts and Applications of Voronoi Diagrams. 2nd edn. John Wiley & Sons, NYC (2000)
3. Aurenhammer, F., Klein, R.: Voronoi diagrams. In Sack, J.R., Urrutia, J.B., eds.: Handbook of Computational Geometry. Elsevier Science Publishers, B.V. North-Holland (2000) 201–290
4. Boissonnat, J.D., Wormser, C., Yvinec, M.: Curved Voronoi diagrams. In Boissonnat, J.D., Teillaud, M., eds.: Effective Computational Geometry for Curves and Surfaces. Springer-Verlag (2006)
5. Barequet, G., Dickerson, M.T., Drysdale, R.L.S.: 2-point site Voronoi diagrams. Discrete Applied Mathematics **122**(1-3) (2002) 37–54
6. Klein, R.: Concrete and Abstract Voronoi Diagrams. Volume 400 of LNCS. Springer-Verlag (1989)
7. Fortune, S.J.: A sweepline algorithm for Voronoi diagrams. Algorithmica **2**(1-4) (1987) 153–174

8. Guibas, L.J., Knuth, D.E., Sharir, M.: Randomized incremental construction of Delaunay and Voronoi diagrams. Algorithmica **7**(1-6) (1992) 381–413
9. Klein, R., Mehlhorn, K., Meiser, S.: Randomized incremental construction of abstract Voronoi diagrams. Computational Geometry: Theory and Applications **3**(3) (1993) 157–184
10. Karavelas, M.I., Yvinec, M.: The Voronoi diagram of planar convex objects. In: Proceedings of the 11th Annual European Symposium on Algorithms (ESA). Volume 2832 of LNCS. (2003) 337–348
11. Aichholzer, O., Aigner, W., Aurenhammer, F., Hackl, T., Jüttler, B., Pilgerstorfer, E., Rabl, M.: Divide-and-conquer for Voronoi diagrams revisited. In: Proceedings of the 25th Annual ACM Symposium on Computational Geometry (SoCG), Association for Computing Machinery (ACM) Press (2009) 189–197
12. Boissonnat, J.D., Devillers, O., Pion, S., Teillaud, M., Yvinec, M.: Triangulations in CGAL. Computational Geometry: Theory and Applications **22**(1-3) (2002) 5–19
13. Emiris, I.Z., Karavelas, M.I.: The predicates of the Apollonius diagram: Algorithmic analysis and implementation. Computational Geometry: Theory and Applications **33**(1-2) (2006) 18–57
14. Karavelas, M.I.: A robust and efficient implementation for the segment Voronoi diagram. In: Proceedings of the 1st International Symposium on Voronoi Diagrams in Science and Engineering (ISVD). (2004) 51–62
15. Emiris, I.Z., Tsigaridas, E.P., Tzoumas, G.: Voronoi diagram of ellipses in CGAL. In: Abstracts of the 24th European Workshop on Computational Geometry. (2008) 87–90
16. Burnikel, C., Mehlhorn, K., Schirra, S.: How to compute the Voronoi diagram of line segments: Theoretical and experimental results. In: Proceedings of the 2nd Annual European Symposium on Algorithms (ESA). Volume 855 of LNCS., Springer-Verlag (1994) 227–239
17. Held, M.: VRONI: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments. Computational Geometry: Theory and Applications **18**(2) (2001) 95–123
18. Hoff III, K.E., Keyser, J., Lin, M., Manocha, D., Culver, T.: Fast computation of generalized Voronoi diagrams using graphics hardware. In: Proceedings of the 26th Annual International Conference on Computer Graphics and Interactive Techniques, Association for Computing Machinery (ACM) Press (1999) 277–286
19. Nielsen, F.: An interactive tour of Voronoi diagrams on the GPU. In: ShaderX^6: Advanced Rendering Techniques. Charles River Media (2008)
20. Edelsbrunner, H., Seidel, R.: Voronoi diagrams and arrangements. Discrete & Computational Geometry **1**(1) (1986) 25–44
21. Meyerovitch, M.: Robust, generic and efficient construction of envelopes of surfaces in three-dimensional space. In: Proceedings of the 14th Annual European Symposium on Algorithms (ESA). Volume 4168 of LNCS. (2006) 792–803
22. Meyerovitch, M., Wein, R., Zukerman, B.: 3D envelopes. In CGAL Editorial Board, ed.: CGAL User and Reference Manual. 3.4 edn. (2008)
23. Agarwal, P.K., Schwarzkopf, O., Sharir, M.: The overlay of lower envelopes and its applications. Discrete & Computational Geometry **15**(1) (1996) 1–13
24. Setter, O.: Constructing two-dimensional Voronoi diagrams via divide-and-conquer of envelopes in space. M.Sc. thesis, The Blavatnik School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel (May 2009) Electronic version can be found at: `http://arxiv.org/abs/0906.2760`.

25. Berberich, E., Fogel, E., Halperin, D., Mehlhorn, K., Wein, R.: Arrangements on parametric surfaces I: General framework and infrastructure. (2010) Accepted for publication in Mathematics in Computer Science.
26. Ebara, H., Fukuyama, N., Nakano, H., Nakanishi, Y.: Roundness algorithms using the Voronoi diagrams. In: Abstracts of the 1st Canadian Conference on Computational Geometry. (1989) 41
27. Eigenwillig, A., Kerber, M.: Exact and efficient 2D-arrangements of arbitrary algebraic curves. In: Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Society for Industrial and Applied Mathematics (SIAM) (2008) 122–131
28. Berberich, E., Emeliyanenko, P.: CGAL's curved kernel via analysis. ACS Technical Report ACS-TR-123203-04, MPI (2008)
29. Sugihara, K.: Laguerre Voronoi diagram on the sphere. Journal for Geometry and Graphics **6**(1) (2002) 69–81
30. Aurenhammer, F., Edelsbrunner, H.: An optimal algorithm for constructing the weighted Voronoi diagram in the plane. Pattern Recognition **17**(2) (1984) 251–257
31. Yap, C.K., Dubé, T.: The exact computation paradigm. In Du, D.Z., Hwang, F.K., eds.: Computing in Euclidean Geometry. Volume 1 of LNCS. 2nd edn. World Scientific, Singapore (1995) 452–492
32. Austern, M.H.: Generic Programming and the STL. Addison-Wesley (1999)
33. Wein, R., Fogel, E., Zukerman, B., Halperin, D.: 2D arrangements. In CGAL Editorial Board, ed.: CGAL User and Reference Manual. 3.4 edn. (2008)
34. Sharir, M.: The Clarkson-Shor technique revisited and extended. Combinatorics, Probability & Computing **12**(2) (2003)
35. Guibas, L.J., Seidel, R.: Computing convolutions by reciprocal search. Discrete & Computational Geometry **2**(1) (1987) 175–193
36. Berberich, E., Fogel, E., Halperin, D., Kerber, M., Setter, O.: Arrangements on parametric surfaces II: Concretization and applications. (2010) Accepted for publication in Mathematics in Computer Science.
37. Fogel, E., Setter, O., Halperin, D.: Movie: Arrangements of geodesic arcs on the sphere. In: Proceedings of the 24th Annual ACM Symposium on Computational Geometry (SoCG), New York, NY, USA, Association for Computing Machinery (ACM) Press (2008) 218–219
38. Knuth, D.E.: Art of Computer Programming, Volume 3: Sorting and Searching. 2nd edn. Addison-Wesley (April 1998)
39. Baccelli, F., Gloaguen, C., Zuyev, S.: Superposition of planar Voronoi tessellations. Stochastic Models **16** (2000) 69–98
40. Rivlin, T.J.: Approximating by circles. Computing **21** (1979) 93–104
41. Mehlhorn, K., Meiser, S., Rasch, R.: Furthest site abstract Voronoi diagrams. International Journal of Computational Geometry and Applications **11**(6) (2001) 583–616
42. Bose, P., Devroye, L.: Intersections with random geometric objects. Computational Geometry: Theory and Applications **10**(3) (1998) 139–154
43. Nielsen, F., Boissonnat, J.D., Nock, R.: On Bregman Voronoi diagrams. In: Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Society for Industrial and Applied Mathematics (SIAM) (2007) 746–755
44. Onishi, K., Takayama, N.: Construction of Voronoi diagram on the upper half-plane. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **79**(4) (1996) 533–539

45. Nilforoushan, Z., Mohades, A.: Hyperbolic Voronoi diagram. In: Proceedings of the 2006 International Conference on Computational Science and Its Applications. (2006) 735–742
46. Stroustrup, B.: The C++ Programming Language. 3 edn. Addison-Wesley, Boston, MA, USA (1997)
47. Myers, N.: "Traits": A new and useful template technique. In Lippman, S.B., ed.: C++ Gems. Volume 5 of SIGS Reference Library. (1997) 451–458
48. Berberich, E., Kerber, M.: Exact arrangements on tori and Dupin cyclides. In: Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling (SPM), Association for Computing Machinery (ACM) Press (2008) 59–66

A Software Interface: Adding New Diagrams

This appendix contains technical details on what needs to be supplied by the user of the framework in order to implement a new type of Voronoi diagrams. The interface consists of several functions, each operating on a small number of user-defined types (sites or bisector curves). Providing these functions does not require knowledge of the underlying algorithms. In this appendix we assume some familiarity of the reader with generic programming [32] and the C++ programming-language [46].

The `Envelope_3` package is fairly general and can deal with surfaces that have two-dimensional intersections [22]. Therefore, the `Envelope_3` can also compute Voronoi diagrams with two-dimensional bisectors.[6] However, most types of two-dimensional Voronoi diagrams have only one-dimensional bisectors. We used this fact to reduce and simplify the interface that is used by the code for computing Voronoi diagrams with one-dimensional bisectors via envelopes. In order to construct Voronoi diagrams with two-dimensional bisectors we refer the user to [24].

Generic programming [32] manifests a formal hierarchy of polymorphic abstract requirements on data types referred to as *concepts*, and a set of classes that conform precisely to the specified requirements, referred to as *models*. Models that describe behaviors are referred to as *traits classes* [47]. Following the generic-programming approach, we implemented a generic template-function called `voronoi_2<GeometryTraits,TopologyTraits>` that computes Voronoi diagrams via envelope constructions.

When the function-template is instantiated, the `GeometryTraits` parameter must be substituted with a geometry-traits class — a model of the concept *EnvelopeVoronoiTraits_2*, and the `TopologyTraits` parameter must be substituted with a topology-traits class. The latter adapts the underlying data-structure that maintains the incidence relations on the arrangement features to the embedding surface, which is parameterized by two parameters u and v. (In case of the plane, for example, u and v map to x and y, respectively.) Thus, the developer of a new diagram type must choose a topology-traits class that handles the

[6] For example, two point sites in the L_1-Voronoi diagram can have a two-dimensional bisector.

embedding surface of the desired diagram; for more information on topology-traits and the surfaces they support refer to [25]. The `Arrangement_on_surface_2` package currently includes topology-traits classes that handle the bounded or unbounded plane, topology-traits classes that handle elliptic quadrics and ring Dupin cyclides that generalize tori [48], and a specially tailored topology-traits class that handles the sphere.

The `Arrangement_on_surface_2` package supports robust construction of arrangements embedded on two-dimensional parametric surfaces in 3D. We use such arrangements to represent Voronoi diagrams embedded on corresponding surfaces. The `Arrangement_on_surface_2` package defines the *ArrangementTraits_2* concept, the models of which are geometry-traits classes that contain geometric primitives used by the package. The *EnvelopeVoronoiTraits_2* concept refines the *ArrangementTraits_2* concept. The predicates and operations defined by the *ArrangementTraits_2* are necessary for the manipulation of bisector curves of Voronoi sites.

The *EnvelopeVoronoiTraits_2* concept defines additional types and operations on top of those defined by the *ArrangementTraits_2* concept. The `Site_2` type represents a Voronoi site. Given two variables of type `Site_2` and an output iterator, the `Construct_bisector_2` functor[7] returns a sequence of objects of type `X_monotone_curve_2` that together form the bisector of the two Voronoi sites. If the bisector between the two sites does not exist, the function returns an empty sequence.[8]

Other required functors are used to determine which side of the bisector, if there is one, each site dominates. The `Compare_distance_above_2` functor accepts two site objects and a u-monotone curve, which is part of their bisector, and indicates which site dominates the region above the u-monotone curve, where "above" is defined to be the region to the left of the u-monotone curve when it is traversed from the uv-lexicographically smaller end to the uv-lexicographically larger end. The framework utilizes the fact that each of the sites dominates one side of the bisector to implement the "below" version of the functor. If there is no bisector between the two sites then the `Compare_dominance_2` functor is used to indicate which of the sites dominates the whole domain.

Given two input sites and a point, the `Compare_distance_at_point_2` functor indicates which site dominates the point in the two-dimensional domain. The functor is used together with the `Construct_point_on_x_monotone_curve_2` functor that constructs an interior point on a given u-monotone curve.

The `voronoi_2` function uses the `Envelope_3` package to compute the desired Voronoi diagram. It uses the class template `Voronoi_2_to_Envelope_3_adaptor` to adapt models of *EnvelopeVoronoiTraits_2* concept to classes that conform to the

[7] Functor is a synonym for "function object" and is invoked or called as if it were an ordinary function with the same syntax.

[8] There are cases where there is no bisector between two Voronoi sites. For example, two Apollonius sites where one is completely contained inside the other have no bisector.

requirements of the `Envelope_3` package. A similar function `farthest_voronoi_2` is used to compute farthest-site Voronoi diagrams.

Table 3 summarizes the types and functors that are needed in order to implement a new Voronoi diagram type.

Table 3. Required types and functors by the *EnvelopeVoronoiTraits_2* concept.

Name	*Input*	*Output*	*Description*
`Site_2`	—	—	A type that represents a Voronoi site.
`Construct_bisector_2`	Two `Site_2` objects	An output iterator with values of type of `X_monotone_curve_2`	Returns `X_monotone_curve_2` objects that together form the bisector of the two input sites.
`Compare_distance_above_2`	Two `Site_2` objects and an `X_monotone_curve_2`	`Comparison_result`	Determines which of the given Voronoi sites is closer to the area "above" the given u-monotone curve, where "above" is the area that lies to its left when the curve is traversed from its uv-lexicographically smaller end to its uv-lexicographically larger end.
`Compare_distance_at_point_2`	Two `Site_2` objects and a `Point_2`	`Comparison_result`	Determines which of the given Voronoi sites is closer to the given point.
`Compare_dominance_2`	Two `Site_2` objects	`Comparison_result`	Determines which of the sites dominates the domain in case that there is no bisector between the two sites.
`Construct_point_on_x_monotone_curve_2`	`X_monotone_curve_2`	`Point_2`	Constructs an interior point on the given u-monotone curve.