Voronoi Diagrams of Algebraic Distance Fields

Ioannis Z. Emiris, Angelos Mantzaflaris, Bernard Mourrain

Voronoï Diagrams of Algebraic Distance Fields

Ioannis Emiris [a] Angelos Mantzaflaris [b] Bernard Mourrain [c]

[a] *Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece*
[b] *RICAM-Linz, Austrian Academy of Sciences, Linz, Austria*
[c] *GALAAD, INRIA Méditerranée, Sophia-Antipolis, France*

Abstract

We design and implement an efficient and certified algorithm for the computation of Voronoï Diagrams (VD's) constrained to a given domain. Our framework is general and applicable to any VD-type where the distance field is given explicitly or implicitly by a polynomial, notably the anisotropic VD or VD's of non-punctual sites. We use the Bernstein form of polynomials and DeCasteljau's algorithm to subdivide the initial domain and isolate bisector, or domains that contain a Voronoï vertex. The efficiency of our algorithm is due to a filtering process, based on bounding the field over the subdivided domains. This allows to exclude functions (thus sites) that do not contribute locally to the lower envelope of the lifted diagram. The output is a polygonal description of each Voronoï cell, within any user-defined precision, isotopic to the exact VD. Correctness of the result is implied by the certified approximations of bisector branches, which are computed by existing methods for handling algebraic curves. First experiments with our C++ implementation, based on double precision arithmetic, demonstrate the adaptability of the algorithm.

Key words: Voronoï diagram, subdivision algorithm, bisector curve, anisotropic diagram, lower envelope

1 Introduction

Voronoï Diagrams (VD's) have a surprising variety of applications, e.g. in path planning, computer vision and machine perception, meshing etc. It is a widely studied subject, and several algorithms exist for their computation.

The VD of a given set of geometric objects (sites) in the plane is the partition of the plane into regions (cells), where each site S is associated to the region consisting of all points for which S is the nearest site, compared to any other site of the (so called) generating set. If we take a set of points in the plane as generating set, and as distance between a site and an arbitrary point the length of their connecting line segment, we have the classic VD under the Euclidean metric. The distance induces a scalar *distance field* over the plane for every site in the generating set.

There are at least two ways to generalize the construction of VD's, in order to meet applications' needs; to allow non-punctual sites, e.g. circles, under the Euclidean distance field, or to attach distance fields other than the Euclidean to the sites, for example the *anisotropic diagram* (see Sect. 1.1). These generalizations lead to curved VD's, having algebraic bisectors, see [4] for an expository paper.

The bibliography on VD's is vast, and many alternative strategies have been proposed, but let us mention some of the latest developments. Boada *et al.* [3] use a subdivision approach to compute approximate VD's. The closest site to the corners of a subdivided domain is computed and used to deduce the Voronoï cell in which the box belongs to. Their output is proved to converge to the exact VD, but the result is not always topologically correct, since corner signs alone are not in one-to-one correspondence with the possible configurations of bisectors inside rectangular domains. A similar idea, using the centres of pixels as points of reference and graphics hardware is implemented in [10]. Their discrete computation has to be carried out on every pixel of the screen, and is subject to resolution errors.

Setter *et al.* [18] compute VD's using divide-and-conquer of lower envelopes and exact computations. They recursively build the diagram, by going down to two sites and then merge up a full VD. They provide good treatment of degenerate cases, and a complexity analysis for randomized inputs. They comment that computation speed may be limited due to extensive use of symbolic computations.

Divide-and-conquer techniques are also utilized by Aichholzer *et al.* [1], to propose a VD algorithm for general shapes. They exploit connections to medial axis and use a plane-sweep technique. Their method avoids computing redundant pieces of bisectors that will then be rejected.

Seong *et al.* [17] explore the VD of parametric NURBS-curves. They approximate bisectors as algebraic curves in

parameter-space and use real solving to find their intersections globally, and then trim away the unwanted parts.

Emiris *et al.* [7] present an efficient, certified algorithm for VD's, specialized to certain families of closed planar curves, notably ellipses. They use adapted numerical techniques to reduce the degree of predicates to be evaluated. Their algorithm becomes exact if they choose to use a special iterated resultant for the "InCircle" predicate.

Algebraic curves are the main objects of study when looking at VD's and their bisector sets. Meshing algebraic curves is regarded as a black-box operation in the present work. Several subdivision techniques are available for this task. An important highlight of certain approaches is that the output is guaranteed to be *isotopic* to the input curve. In [12] they give algorithms for (local and global) isotopic computation of algebraic curves in 2D and 3D. They use univariate solving, enveloping techniques and fast regularity criteria. In [13], these ideas are extended to semi-algebraic sets, notably to the arrangement of several curves.

Our method computes VD's by means of the lower envelope of the distance fields. The latter exhibit an algebraic degree which is usually quite low, thus handling them in an algebraic fashion is computationally advantageous. The method is inspired by, and uses tools from geometric modeling and CAGD (Computer Aided Geometric Design). Polynomial curves constitute a major branch of the research in CAGD, and flexible curve representations have been developed in this frame. Therefore, we aim at bringing data representations and algorithmic experience from CAGD to this fundamental problem of computational geometry.

Our algorithm applies on the distance field induced by the sites and does not require the sites themselves in the input. It provides polygonal approximations of the Voronoï cells and is applicable to all inputs where the distance field of every site can be expressed by a polynomial. This expression may be explicit, but also implicit. In Sect. 1.1 we give a list of distance fields for commonly encountered VD's.

The implementation is done in C++, using double precision with controlled rounding modes to certify the subdivision process. The algorithm is naturally parallelizable, since it applies locally and independently on different domains of the plane. We did not focus on insertion of new sites, or point queries, yet these operations can easily be added, without significant changes. On the other hand, we focus on genericity of the framework, the correctness of the result and the ability to treat arbitrary diagrams, if certain reasonable algorithmic prerequisites are met (cf. Sect. 2).

The rest of the text is organized as follows. We complete our introduction by a list of common VD's, as well as some details on curve representations. We present the core of our algorithm in Sect. 2, with details on the filtering, subdivision and the recovery phase. Then we extend the method to implicitly given distance fields in Sect. 3, and conclude with some experiments in Sect. 4.

1.1 Voronoï diagrams & distance fields. In this section we recall a non-exhaustive list of VD-types that we are interested in, and briefly introduce the main tools and representations that build up our algorithm.

For a point $q = (x, y)$, the distance between q and the Voronoï site attached to f_i, *as viewed by site* i, is given by $\texttt{dist}_i(q) = f_i(q)$. Now one can define the Voronoï cell of $i-$th site as:

$$\mathrm{Vor}(i) = \{q \in \mathbb{R}^2 \ : \ \texttt{dist}_i(q) \leq \texttt{dist}_j(q),\ j = 1 \dots n\}$$

Therefore the VD can be retrieved as the projection of the lower envelope of the distance fields [5], also known as the *minimization diagram*.

In the simple case of Euclidean VD the input is a list of (squared) distance functions $(f_1, \dots, f_n)$. The lower envelope changes if we work with squared fields, yet the VD (that is, its projection) remains intact under squaring, or under any other invertible and strictly increasing transformation. Different distance fields $\texttt{dist}_i(q),\ q \in \mathbb{R}^2$, give rise to different types of VD's, including the:

- VD of points $p_i = (x_i, y_i)$ under the ℓ_p-metric, p even: $\texttt{dist}_i(q) = (x - x_i)^p + (y - y_i)^p$.
- Anisotropic diagram of points $p_i = (x_i, y_i)$ with weights $w_i \in \mathbb{R}$ [11]: $\texttt{dist}_i(q) = (q - p_i)^T M_i (q - p_i) - w_i$, with $M_i \in \mathbb{R}^{2\times 2}$ symmetric positive-definite matrices.
- Power (or Laguerre) diagram of points with weights: $\texttt{dist}_i(q) = \|q - p_i\|^2 - w_i^2$.
- Möbius diagram of points: $\texttt{dist}_i(q) = v_i\|q - p_i\|^2 - w_i$, with $v_i, w_i \in \mathbb{R}$.
- Apollonius (or additively weighted) diagram of disks [6] with centers p_i and radii w_i: $\texttt{dist}_i(q) = \|q - p_i\| - w_i$.
- Euclidean VD of ellipses [7], or Euclidean VD of general closed parametric curves [9].

An algebraic function f can be represented over a bounded domain $\mathcal{D} = [a, b] \times [c, d]$ by its Bernstein coefficients over $\mathcal{D}$. A number of properties of this basis, e.g. convexity, variation diminishing, positivity etc, make it suitable for stable numerical computations. Also, DeCasteljau's algorithm can be applied to split the representation, i.e. produce the coefficients over sub-domains of $\mathcal{D}$ (cf. [8]). We use the notation $f_i|_{\mathcal{D}}$ for the restriction of $f_i(x, y)$ in $\mathcal{D}$, that is, the Bernstein representation of f_i over the domain $\mathcal{D}$. For instance, for the anisotropic diagram, the f_i's are conics, thus every $f_i|_{\mathcal{D}}$ is defined over $\mathcal{D}$ by 9 coefficients.

Note that some VD types in the previous list do not have polynomial distance fields, nor do they admit squaring to eliminate radicals. In such cases, it is usually possible to express the distance field implicitly. For instance, if we want to use our framework to compute the Apollonius diagram, i.e. the Euclidean VD of circles with centers p_i and radii w_i, we need to treat the corresponding distance function $\texttt{dist}_i(q) = \|q - p_i\| - w_i$, which may be transformed to the algebraic equation $(\texttt{dist}_i(q) + w_i)^2 = \|q - p_i\|^2$. Therefore, distance field values $z := \texttt{dist}_i(x, y)$ are given in *implicit form* by the cone

$$g_i(x, y, z) := (x - x_i)^2 + (y - y_i)^2 - (z + w_i)^2 = 0, \quad (1)$$

i.e., given (x_0, y_0), values of $\texttt{dist}_i(x_0, y_0)$ are computed by solving $g_i(x_0, y_0, z) = 0$ for the smallest root > 0 w.r.t. z.

This tri-variate function can be represented similarly over a cuboid, by tensoring 3 Bernstein bases. In Sect. 3, we shall adapt our algorithm to use implicit representations.

2 The algorithm

Our algorithm consists of two phases: The subdivision phase (Alg. 1), followed by the reconstruction phase (Alg. 2). Two algorithmic ingredients must be provided in order to apply the method on a specific VD type:

(i) **Field bound:** A way to bound the value range of a distance field over a given domain.

(ii) **Bisector tracking:** A way to compute an approximation of bisector(s) or vertices in a given domain.

These computations need to be carried out on axis-aligned boxes of the subdivision. Note that at this level we do not emphasize on efficiency, e.g. the bounds could be bad, or the approximation very loose. Nevertheless, it is expected that these computations converge to the actual distance value or bisector, when the size of the boxes becomes smaller.

The first box that is computed as soon as the algorithm is launched is the one corresponding to the initial domain $\mathcal{D}_0$, carrying all distance fields of the input, in Bernstein form.

In the subdivision phase we compute a graph of boxes that span the VD. To do so, the *field bound* is used in a filtering process in order to exclude most boxes and reach down to boxes intersecting the VD. These boxes contain Bernstein representations of sites that contribute to the part of the diagram inside the box.

In the reconstruction phase the boxes of the subdivision are traversed and the part of the VD that intersects the box is meshed locally using the second ingredient. These bisector segments are finally stitched together to recover the full VD.

2.1 Subdivision phase. The idea of a space-subdivision scheme is typical in CAGD. A big domain of interest is divided into smaller ones until some (fast computed) conditions are fulfilled. Then every small item is treated independently.

We denote by $B(\mathcal{D}) = \{f_i|_{\mathcal{D}}, f_j|_{\mathcal{D}}, \dots\}$ the list of control grids that are held for a subdivided domain $\mathcal{D}$. Also, by signature of a box $\mathtt{sig}(B(\mathcal{D}))$ we mean the site labels present in the list, i.e. $\mathtt{sig}(B(\mathcal{D})) = \{i \ : \ f_i|_{\mathcal{D}} \in B(\mathcal{D})\}$.

The main loop in Alg. 1 starts by filtering the box $B(\mathcal{D})$. This allows to remove large valued f_i's, i.e. those that correspond to sites that are "away" from the box. Then, we check if the remaining fields are at most three. In this case the box is not subdivided anymore. Otherwise the box is split in two sub-domains, along the longest of its sides. The new boxes are pushed in the stack and the next loop begins.

If $B(\mathcal{D})$ is left with only one $f_i|_{\mathcal{D}}$, then $\mathcal{D} \subseteq \mathrm{Vor}(i)$, hence this box does not span the VD. If there are 2 active functions then the bisector locus $f_i|_{\mathcal{D}} - f_j|_{\mathcal{D}} = 0$ possibly intersects the box. If there are 3 or more functions then there probably exists a Voronoï vertex in $\mathcal{D}$. Boxes that contain degenerate Voronoï vertices, i.e. vertices that are equi-distant from more than 3 sites will always hold > 3 functions. Hence they will be subdivided until reaching threshold size $\varepsilon > 0$.

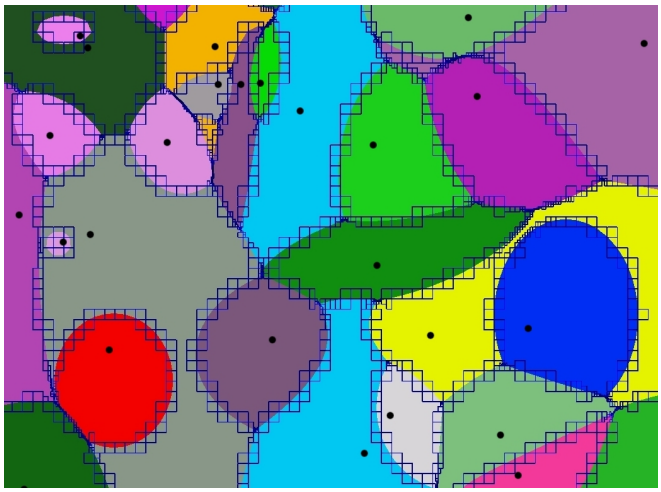

Fig. 1. Anisotropic diagram. Subdivision box span by Alg. 1 is shown.

Algorithm 1: Subdivision phase

Input: A set of distance fields $f_1, f_2, \dots, f_n \in \mathbb{R}[x, y]$, a threshold $\varepsilon > 0$ and a rectangle domain $\mathcal{D}_0$.
Output: A graph G of boxes that span the minimization diagram of $f_1, \dots, f_n$.

```
Compute B(D_0) := {f_1|_{D_0}, ..., f_n|_{D_0}};
Initialize stack Q and add B(D_0) on top of it;
Initialize empty box graph G;
while Q is not empty do
    Pop a box B(D) from Q;
    Apply filter (Sect. 2.2) on B(D);
    if |B(D)| ≤ 3 or |D| < ε then
        Add B̃(D) := {(f_i − f_j)|_D : i < j ∈ sig(B(D))} to G ;
    else
        Split B(D) into B(D_1) and B(D_2) ;
        Update adjacency graph G with B(D_1), B(D_2) ;
        Push B(D_1) and B(D_2) into Q ;
return G;
```

Whenever we arrive to a box with $|B(\mathcal{D})| \leq 3$, or if the box reaches the threshold size ε, the box is *mutated*: This means that we form differences $f_i|_{\mathcal{D}} - f_j|_{\mathcal{D}}$, $i < j$ and we store them in the place of $f_i|_{\mathcal{D}}$, $i \in \mathtt{sig}(B(\mathcal{D}))$. Thus the box is transformed into *bisector box*. This box will be further treated in the next phase.

Three main operations constitute every iteration of the subdivision loop: First, an application of the filter of Sect. 2.2 on the subdivided boxes. Then, an execution of DeCasteljau's algorithm to split the representation of the distance functions. Finally, an update of the adjacency graph, i.e. connection of newly created boxes with their surrounding neighbors. These operations are $O(n)$ for a box with $|B(\mathcal{D})| = n$. The overall time of Alg. 1 depends on the efficiency of the filter, since it affects the stopping condition $|B(\mathcal{D})| \leq 3$.

2.2 Upper bounds and filtering. Not all bisectors are Voronoï edges. Furthermore, only a specific piece of the bisector curve of two sites is contained in the VD, the one connecting two adjacent Voronoï vertices. For instance, in the classic VD of points, only a linear amount (w.r.t. the

number of sites) of line segments out of the totality of (line) bisectors contribute to the VD. This implies that, locally, only a few sites contribute to the VD and therefore the majority of sites should be filtered out. This idea leads to the filtering process described herein.

The filter is based on the following fact: If the control grids of any two $f_i|_{\mathcal{D}}$, $f_j|_{\mathcal{D}}$ do not intersect, then the one that is superior to the other, say the $i-$th, does not contribute to the lower envelope over $\mathcal{D}$, or equivalently $\mathrm{Vor}(i) \cap \mathcal{D} = \varnothing$. This is a direct consequence of the variation diminishing property of Bernstein representation [8].

Doing the previous test directly for a set of n sites requires $O(n^2)$ time to check all pairs. But we can do better: we compute an upper bound $U_{\mathcal{D}}$ on the lower envelope over $\mathcal{D}$, and then compare every control grid $f_i|_{\mathcal{D}}$ against this bound, excluding those that are found to be over it. This leads to a linear time filter, w.r.t. $n = \#B(\mathcal{D})$.

Before computing $U_{\mathcal{D}}$, we need bounds on every $f_i|_{\mathcal{D}}$. For this we use the minimum (resp. maximum) Bernstein coefficient of $f_i|_{\mathcal{D}}$ to bound every f_i from below (resp. above). These extreme coefficients yield two supporting planes, parallel to $xy-$plane, that enclose the values $f_i(\mathcal{D})$. More sophisticated bounds exist, see [14–16] and references therein. We choose to employ constant time bounds, since they are quite efficient in practice. Indeed, the control grid is known to converge with quadratic speed to the function it represents [16], thus a small number of refinements is needed to separate the control grids of two non-intersecting patches.

Now, to compute $U_{\mathcal{D}}$, an upper bound on the lower envelope over $\mathcal{D}$, consider $(x, y) \in \mathcal{D}$ and let $L(x, y)$ be the value of the lower envelope at $(x, y) \in \mathcal{D}$. We have:

$$L(x, y) = \min_i\{f_i(x, y)\} \leq \min_i\{\texttt{maxcoef}(f_i|_{\mathcal{D}})\}. \qquad (2)$$

This directly proposes to take $U_{\mathcal{D}} = \min\{\texttt{maxcoef}(f_i|_{\mathcal{D}})\}$. Having $U_{\mathcal{D}}$, we can check which distance fields are bounded over this limit, and filter them out.

Fig. 2 illustrates the filtering process in 1D. Three point-sites a, b, c yield the red distance fields, that are squared Euclidean metrics. Domain $\mathcal{D}_1$ holds all three f_a, f_b, f_c, and the dotted line is the level of $U_{\mathcal{D}_1}$, equal to $\texttt{maxcoef}(f_b|_{\mathcal{D}_1})$. Since $f_c|_{\mathcal{D}_1}$ is over the dotted line, it will be filtered out of the list $B(\mathcal{D}_1)$. Similarly, $\texttt{maxcoef}(f_c|_{\mathcal{D}_2})$ defines $U_{\mathcal{D}_2}$. Consequently, $f_b|_{\mathcal{D}_2}$ is excluded and $\mathcal{D}_2$ is dominated by site c, i.e. $\mathcal{D}_2 \subset \mathrm{Vor}(c)$.

2.3 Cell reconstruction phase. The box-graph computed by Alg. 1 spans the VD, having edges between adjacent boxes. Traversing this graph, by navigating based on signature and sign of the bisector leads to the cells of the VD. In particular, we shall traverse all boundary-contours of the Voronoï cells in counter-clock wise (CCW) order, completing open cells with pieces of the boundary of the initial domain $\mathcal{D}_0$.

As soon as the process starts, there is a pre-processing of the graph G. In this step, all the boxes of the graph G that touch the boundary of $\mathcal{D}_0$ are connected in a CCW-directed loop. This loop constrains the reconstruction phase in $\mathcal{D}_0$.

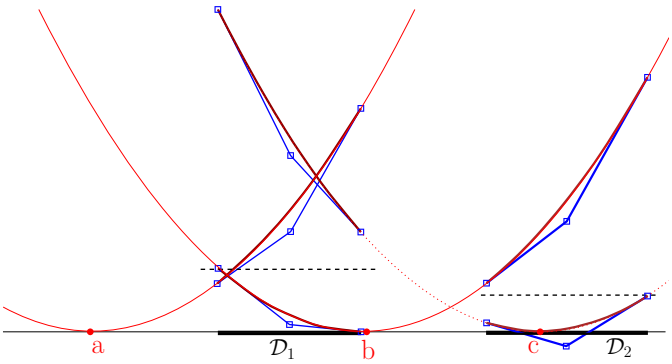

Fig. 2. Filtering process applied to domains $\mathcal{D}_1$ and $\mathcal{D}_2$, in computing the VD of tree points $a, b, c \in \mathbb{R}$.

First we discuss how every single box is treated, when encountered in the traversal. Every bisector in the box is approximated inside a domain $\mathcal{D}$ by line segments connecting its intersections with $\partial\mathcal{D}$, or other bisectors. For getting the arrangement inside $\mathcal{D}$, we use intersection points and the sign of derivatives (cf. [13]). Computation of intersections of bisectors is carried out by a Bernstein solver [14]. Then, signature information that is attached to bisectors is used to identify Voronoï vertices.

If there are no degenerate vertices, and an adequetely small tolerance, the reported diagram is isotopic to the exact one. In practice, subdivision around a degenerate vertex goes down to $\varepsilon-$precision. At this point the vertex is considered to be unique and is attached to all involved bisectors. Therefore, it is possible that two (or more) close vertices are reported as one degenerate vertex. The impact of this change on the global structure of the diagram is small: we may neglect some edges of the dual graph (ie. the generalization of Delaunay's graph, see for instance [7]), but no false edges are introduced.

The same holds even in the case of a tiny cell of size lower than the working tolerance that may be discarded, due to agglomeration of close vertices. The cell is reported empty, therefore some edges attached to the respective site are missing in the dual graph.

The defining polynomial $f_i - f_j$ of a bisector (i, j) is negative over the (open) Voronoï cell of i and positive over the cell of j. The traversal of the spanning boxes uses this sign information: The signs of the bisector on the corners of $\mathcal{D}$ give us the correct orientation for tracking the cells corresponding to i and j.

Alg. 2 summarizes the reconstruction process. Starting from an arbitrary box, the boundary of the cell of site i is tracked by traversing the box-span G. The navigation is done based on the sign of the bisector $f_i - f_j$. When a vertex is reached, j is updated, so that we follow the next edge of $Vor(i)$, and the process continues until the recovery of the whole partition V.

The output of this algorithm is a polygonal approximation of each Voronoï cell. Fig. 3 demonstrates bisector computation. The control polygon in $\mathcal{D}_1$ of the equation $f_a - f_b = 0$ is shown in green. Its intersection with the do-

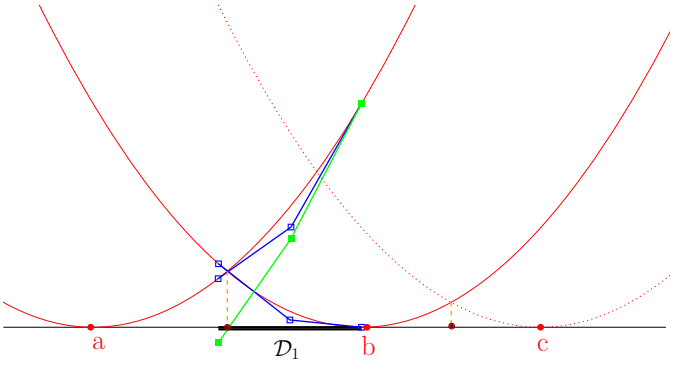

Fig. 3. Approximating the bisector of a and b.

main defines the bisector point (in 1D) of sites a, b.

Algorithm 2: Cell reconstruction phase

Input: A graph G of boxes that span the minimization diagram of $f_1, \ldots, f_n$.
Output: A partition V of $\mathcal{D}_0$, defined by meshing the minimization diagram of $f_1, \ldots, f_n$.
Join border boxes of G in a CCW loop ;
Initialize $V[i] = \varnothing$, $i = 1, \ldots, n$;
foreach *unvisited node (box)* $B(\mathcal{D})$ *of* G **do**
  foreach *label* $i \in \mathtt{sig}(B(\mathcal{D}))$ **do**
    foreach *label* $j \in \mathtt{sig}(B(\mathcal{D}))$, $j \neq i$ **do**
      • Traverse G, starting from $B(\mathcal{D})$, and track bisector (i, j) along boxes with label i ;
      • If some $B'(\mathcal{D})$ contains a vertex, set $j := j'$, $(i, j') \in B'(\mathcal{D})$ and continue, until $B(\mathcal{D})$ is reached for the second time ;
      • Add tracked cell component to $V[i]$.
return V;

3 The case of implicit distance fields

In this section we sketch how the framework is extended to implicitly defined distance fields.

There are distance fields that cannot be made polynomial by a mere squaring, or by another suitable transformation. A nice polynomial formula even for Euclidean distance fields no longer exists when the sites are not points. In such cases it is natural to represent the function in implicit form, that is, as a polynomial $g \in \mathbb{R}[x, y, z]$, s.t. a triple satisfying $g(x, y, z) = 0$, with $z > 0$ and minimal among the solutions implies that (x, y) is at distance z from the site. Some diagrams that fall in this class are the Apollonius diagram [6], the VD of ellipses [7], or the VD of closed curves given by support function representation [9].

In order to have a representation of $g_i(x, y, z)$ over the initial domain $\mathcal{D}_0 \subset \mathbb{R}^2$, first we compute an interval I_0 for the third variable. This must contain the $z-$values of zeros of g. For this we use an upper bound (e.g. Cauchy's bound) for the roots of the (univariate) interval polynomial $g_i(\mathcal{D}, z)$. Then we feed the subdivision Alg. 1 with $g_i|_{\mathcal{D}_0 \times I_0}$, for every site i in the generating set.

The same ingredients, i.e. the "field bound" and "bisector tracking" (Sect. 2) are needed, in order to run the scheme. The subdivision and reconstruction is done the same way, on 3D boxes that enclose implicit space-curve branches.

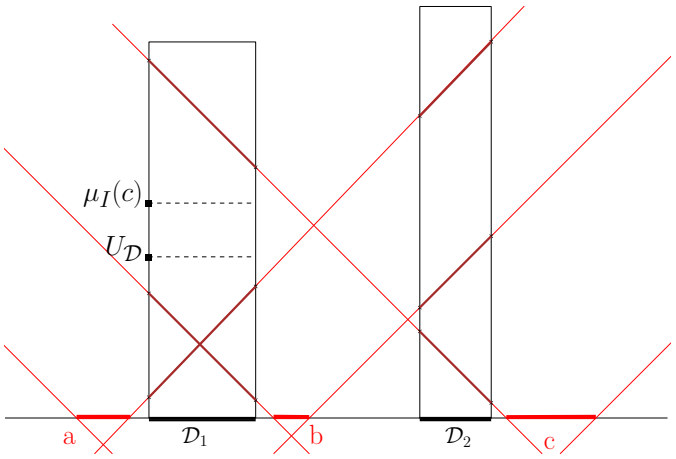

Fig. 4. Instance of Apollonius diagram in 1D: sites a, b, c are line segments and distance fields are bi-variate (implicit) cones.

3.1 Field bound. For $g|_{\mathcal{D}\times I} = \sum \gamma_{ijk} B^i_{d_x}(x)\, B^j_{d_y}(y)\, B^k_{d_z}(z)$ we set $m_k = \min_{i,j}\{\gamma_{ijk}\}$ and $M_k = \max_{i,j}\{\gamma_{ijk}\}$. Let

$$m_g(z) := \sum_{k=0}^{d_z} m_k\, B^k_{d_z}(z), \quad M_g(z) := \sum_{k=0}^{d_z} M_k\, B^k_{d_z}(z). \quad (3)$$

These polynomials are defined over the $z-$interval I and enclose the range of $z-$values, since, for $(x, y, z) \in \mathcal{D} \times I$,

$$g(x, y, z) \geq \sum_{k=0}^{d_z} m_k B^k_{d_z}(z) \sum_{j=0}^{d_x} B^i_{d_x}(x) \sum_{j=0}^{d_y} B^j_{d_y}(y) \;= m_g(z),$$

and similarly for $M_g(z)$. Consequently, a lower bound is given by $\mu_I := \begin{cases} \text{smallest root of } M_g(z) \text{ in } I & \text{if } M_0 < 0 \\ \text{smallest root of } m_g(z) \text{ in } I & \text{if } m_0 > 0 \\ 0 & \text{otherwise} \end{cases}$, and similarly for the upper bound $\mathcal{M}_I$, using the last real roots of $m_g(z)$ or $M_g(z)$ in I. Finally, using these enclosures we get $U_{\mathcal{D}} = \min\{\mathcal{M}_I(i) \,:\, g_i \in \mathcal{D}\}$, and Sect. 2.2 applies.

3.2 Bisector tracking. Bisectors are defined by the (projection of the) intersection of two implicit surfaces, thus they are implicit spatial curves. One way to get the projection is using resultants, but this would be costly to do in every box, and would increase the degree of the polynomial to handle. Isotopic meshing of 3D curves, given in Bernstein form, has been implemented in [12], and yields a suitable way to derive the projection. Fig. 4 shows a snapshot of the implicit method. Sites are segments on the real axis and distance fields are implicit cones. The box $\mathcal{D}_1 \times I_1$ contains 3 bi-variate Bernstein polynomials. Based on $U_{\mathcal{D}_1}$ and the lower bound $\mu_{I_1}(c)$, the $f_c|_{\mathcal{D}_1}$ is filtered out.

3.3 Deriving the implicit equation. The offset curve of a site is closely related to the distance function. The $z-$offset is the set of points at distance z from the site. Therefore, we need to compute an implicit equation containing the $z-$offset, having z as a parameter.

For the Apollonius diagram, the $z-$offset of a circle-site centered at p_i and with radius w_i is the circle of the same center and radius $z + w_i$ (see (1)).

diagram / # sites		50	100	200	400	800
Euclidean	time	0.87	1.9	3.9	7.5	13.9
	boxes	2008	2981	4210	5975	8222
Anisotropic	time	1.1	2.4	4.2	8.7	16.1
	boxes	2213	3328	4364	6380	8702
ℓ_4 metric	time	1.1	2.1	4.2	7.8	16.7
	boxes	2283	2965	4178	5866	8105
ℓ_8 metric	time	1.1	2.5	4.9	9.3	19.4
	boxes	2050	2971	4298	5974	8037

Table 1
Execution details for random inputs in $[-2,2]^2$.

Let $n(t) : \mathbb{R} \to \mathbb{S}^1$ be a parametrization of the circle $\mathbb{S}^1$. A supported curve (cf. [9]) is a curve given by a parametrization $h(t)$ over the unit circle, which associates to every point $n(t)$ of $\mathbb{S}^1$ the point on the curve that has normal vector $n(t)$. The offset curve at distance z has support $h(t) + z$. After elimination of t from $(h(t) + z) \cdot n(t) - (x, y) = 0$ we arrive to the implicit form of the $z-$offset, ie. the distance field of the supported curve.

Anton *et al.* [2] compute, using resultants, the offset curve of radius z of a conic, e.g. an ellipse $\mathcal{E}$. This is a polynomial of degree 4 w.r.t. z and of degree 8 w.r.t. x, y. For any specific point $q = (x, y)$ outside of the ellipse, we have an irreducible univariate polynomial that has exactly one real positive solution, corresponding to $\texttt{dist}(q, \mathcal{E})$. We note that this equation can also be derived if we consider a support function parametrization of the ellipse.

4 Experimentation

We run experiments on Euclidean, anisotropic and VD under the ℓ_p metric, for $p = 4, 8$. For these tests, we set $\varepsilon = 0.001$ and $\mathcal{D}_0 = [-2, 2]^2$. Also, for the sake of a smooth result, we set a maximum admissible box-size, i.e. all boxes are subdivided until their longest side is at most $\epsilon = 0.05$. Table 1 reports execution time (in secs) and number of boxes produced, for random inputs of 50 up to 800 sites in $\mathcal{D}_0$. The timings are of the same order for all VD types. This can be explained by the fact that the process discards the linear nature of the Euclidean VD. On the other hand, it is applied to the non-linear, anisotropic case, where bisectors are general conics, with the same performance. The reason is that the total time depends mostly on the number of boxes that are produced as output of Alg. 1. This adaptability to general topologies, or mixed diagrams (see Fig. 5), seems to be an advantage of our approach.

The experiments showed a stable method. For example, nearly touching bisectors, as in the cell on the top right of Fig. 1, were correctly separated. Near-degenerate Voronoï vertices were reported as a single vertex of high valency. We note that for computing bounds for univariate root-finding or for the filtering, the rounding mode in the floating point operations is controlled, so that we always enclose the exact values. Nevertheless, exact treatment of all degenerate cases would require manipulation of algebraic numbers, something that we refrain from, for the sake of efficiency.

Acknowledgment This research received funding from [FP7/2007-2013], Marie Curie ITN SAGA, [PITN-GA-2008-214584].

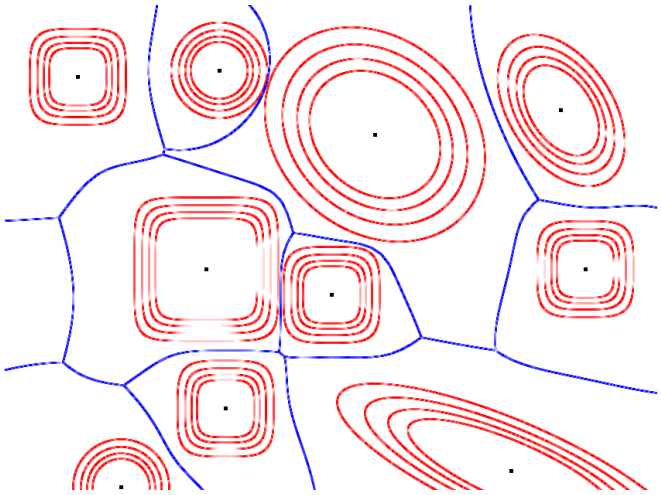

Fig. 5. A mixed VD of anisotropic, and sites with ℓ_p distance function for $p = 2$, 4 or 6. Offset curves (in red) reveal the nature of each site.

References

[1] O. Aichholzer,W. Aigner,F. Aurenhammer,T. Hackl,B. Jüttler, E. Pilgerstorfer & M. Rabl. Divide & conquer algorithms for Voronoi diagrams revisited. *CGTA*, 43:688–699, 2010.
[2] F. Anton, I. Emiris, B. Mourrain, & M. Teillaud. The offset to an algebraic curve and an application to conics. *I*CCSA 2005, v. 3480 of *LNCS*, p. 1–21. Springer, 2005.
[3] I. Boada, N. Coll, N. Madern, & J. Antoni Sellares. Approximations of 2d and 3d generalized voronoi diagrams. *Int. J. Comput. Math.*, 85(7):1003–1022, 2008.
[4] J.-D. Boissonnat, C. Wormser, and M. Yvinec. Curved Voronoï diagrams. *Effective Computational Geometry for Curves and Surfaces*, p. 67–116. Springer, 2006.
[5] H. Edelsbrunner & R. Seidel. Voronoï diagrams and arrangements. *Disc. and Comp. Geom.*, 1:25–44, 1986.
[6] I. Z. Emiris & M. I. Karavelas. The predicates of the apollonius diagram: Algorithmic analysis and implementation. *Comput. Geom. Theory Appl.*, 33:18–57, January 2006.
[7] I. Z. Emiris, E. P. Tsigaridas, & G. M. Tzoumas. Exact Delaunay graph of smooth convex pseudo-circles: general predicates, and implementation for ellipses. *SPM '09*, p. 211–222, 2009. ACM.
[8] G. Farin. *Curves and surfaces for CAGD*. SF, CA, USA, 2002.
[9] J. Gravesen, Z. Šír, & B. Jüttler. Curves and surfaces represented by polynomial support functions. *Th. Co. Sci.*,392:141–157,2008.
[10] K. E. Hoff, III, J. Keyser, M. Lin, D. Manocha, & T. Culver. Fast computation of generalized Voronoï diagrams using graphics hardware. *SIGGRAPH '99*, p. 277–286, 1999. ACM.
[11] F. Labelle & J. R. Shewchuk. Anisotropic Voronoï diagrams and guaranteed-quality anisotropic mesh generation. In *SCG '03*, p. 191–200, 2003. ACM.
[12] C. Liang, B. Mourrain, & J.-P. Pavone. Subdivision methods for the topology of 2d and 3d implicit curves. *Geom. Mod. & Alg. Geom.*, p. 199–214. Springer, 2008.
[13] A. Mantzaflaris & B. Mourrain.A subdivision approach to planar semi-algebraic sets. *LNCS* v. 6130, p. 104–123. Springer, 2010.
[14] B. Mourrain & J. Pavone. Subdivision methods for solving polynomial equations. *J. of Sym. Comp.*, 44(3):292 – 306, 2009.
[15] J. Peters & X. Wu. Sleves for planar spline curves. *Computer Aided Geometric Design*, 21(6):615 – 635, 2004.
[16] U. Reif. Best bounds on the approximation of polynomials and splines by their control structure. *CAGD*, 17(6):579–589, 2000.
[17] J.-K. Seong, E. Cohen, & G. Elber. Voronoï diagram computations for planar nurbs curves. *SPM '08*, p. 67–77, 2008. ACM.
[18] O. Setter, M. Sharir, & D. Halperin. Constructing two-dimensional Voronoï diagrams via divide-and-conquer of envelopes in space. *Trans. Com. Sci. IX*, p. 1–27. Springer, 2010.