Pacific Graphics 2020
E. Eisemann, A. Jacobson, and F.-L Zhang
(Guest Editors)

Volume 39 (2020), Number 7

Robust Computation of 3D Apollonius Diagrams

Peihui Wang [1,*], Na Yuan [1,*], Yuewen Ma [2], Shiqing Xin [1]
Ying He [3], Shuangmin Chen [†4], Jian Xu [5], Wenping Wang [6]

[1]School of Computer Science and Technology, Shandong University, Qingdao, China
[2]Ant Technology Group Co Ltd, China
[3]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[4]School of Information and Technology, Qingdao University of Science and Technology, Qingdao, China
[5]Ningbo Institute of Materials Technology and Engineering, Chinese Academy of Sciences, Ningbo, China
[6]Department of Computer Science, The University of Hong Kong, China
[*]Peihui Wang and Na Yuan equally contribute to this paper

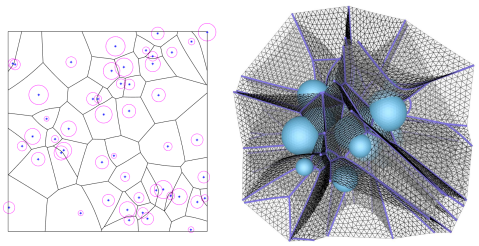
Abstract

Apollonius diagrams, also known as additively weighted Voronoi diagrams, are an extension of Voronoi diagrams, where the weighted distance is defined by the Euclidean distance minus the weight. The bisectors of Apollonius diagrams have a hyperbolic form, which is fundamentally different from traditional Voronoi diagrams and power diagrams. Though robust solvers are available for computing 2D Apollonius diagrams, there is no practical approach for the 3D counterpart. In this paper, we systematically analyze the structural features of 3D Apollonius diagrams, and then develop a fast algorithm for robustly computing Apollonius diagrams in 3D. Our algorithm consists of vertex location, edge tracing and face extraction, among which the key step is to adaptively subdivide the initial large box into a set of sufficiently small boxes such that each box contains at most one Apollonius vertex. Finally, we use centroidal Voronoi tessellation (CVT) to discretize the curved bisectors with well-tessellated triangle meshes. We validate the effectiveness and robustness of our algorithm through extensive evaluation and experiments. We also demonstrate an application on computing centroidal Apollonius diagram.

1. Introduction

Voronoi diagrams, named after Russian mathematician Georgy Voronoi [Vor08], have theoretical and practical applications in many fields, such as motion planning [GMAM06], material science [FWZL04], molecular biology [Wil99], crystallography [BBR*08] and computer graphics [LWL*09]. A Voronoi diagram is defined by a set of sites $\{\mathbf{x}_i\}_{i=1}^n$. In the simplest case, a 2D Voronoi diagram partitions the 2D plane into a set of disjoint cells $\{\Omega_i\}_{i=1}^n$ and each Voronoi cell Ω_i consists of every point in the plane whose distance to $\mathbf{x}_i$ is less than or equal to its distance to any other site. Voronoi edges are all the points in the plane that are equidistant to two nearest sites. Voronoi vertices are the points equidistant to three or more nearest sites.

There are many variants of Voronoi diagrams. First, the sites can be points, line segments, curves, disks, balls or other convex shapes. Second, the distance measurement $d_{\mathbf{x}_i}(\mathbf{x})$ between a point $\mathbf{x}$ in the domain of interest and the site $\mathbf{x}_i$ has various forms. Commonly used distances include the Euclidean distance $\|\mathbf{x}-\mathbf{x}_i\|_2$ in the conventional Voronoi diagram, L_p distance $\|\mathbf{x}-\mathbf{x}_i\|_p$ in L_p Voronoi diagram [Lee80], power distance $\|\mathbf{x}-\mathbf{x}_i\|_2^2 - w_i$ in power diagram [Aur87], and $\|\mathbf{x}-\mathbf{x}_i\|_2 - w_i$ in Apollonius diagram [EK06]. In fact, the distance function may be an arbitrary convex function of $\|\mathbf{x}-\mathbf{x}_i\|$ with a minimal value at $\mathbf{x}=\mathbf{x}_i$ [Ma00]. Last but not least, the diagrams can be defined in a curved space (e.g., a 2-manifold surface) by replacing the Euclidean distance with the geodesic distance [LCT10].



(a) 2D (b) 3D

Figure 1: *Examples of 2D and 3D Apollonius diagrams, where the radius of every circle/sphere denotes the additive weight of the corresponding site.*

In this paper we study the 3D Apollonius diagram problem that uses $d_{\mathbf{x}_i}(\mathbf{x}) = \|\mathbf{x}-\mathbf{x}_i\|_2 - w_i$ to measure the distance from $\mathbf{x}$ to the site $\mathbf{x}_i$, where w_i is $\mathbf{x}_i$'s weight. Roughly speaking, the Apollonius diagram can be viewed as an extension of Voronoi diagram by re-

† The corresponding author: csmqq@163.com

placing the point-type sites with circles (in $\mathcal{R}^2$) or balls (in $\mathcal{R}^3$). See Figure 1 for examples of 2D and 3D Apollonius diagrams. Apollonius diagrams are widely used in analysis and visualization of large scale structures such as liquids/amorphous solids [VBM02] and proteins [Wil98, KKS02].

Though robust solvers are available for computing a 2D Apollonius diagram [KY02], there is no robust solver for the 3D version since the latter is more complicated - the structure of a 3D Apollonius diagram has a higher combinatorial complexity and each bisector is a curved hyperbolic surface. Will [Wil99] studied the computation of the 3D Apollonius diagram (i.e. the Voronoi diagram of three-dimensional spheres), which shows each cell has an $O(n^2)$ combinatorial complexity and it costs $O(n^2 \log n)$ expected time to extract a single Apollonius cell based on their lower envelope algorithm. Boissonnat and Karavelas [BK02] reported an $O(n^3)$-time algorithm using the convex hull of spheres transformed by inversion. Kim and his co-authors presented many research works [KCKC04, KCK*05c, KKC05, KCK05a, KK06] on this problem, where the region-expansion algorithm [KCK05a, KK06], also of an $O(n^3)$ time complexity, runs fast in practice and becomes the state of the art on the computation of 3D Apollonius Diagrams, to our best knowledge. Beside the sequential algorithms, GPU based parallelized algorithms [HLK*17, LKHM19] have been proposed in very recent years.

In this paper, we first systematically analyze the structural features of a 3D Apollonius diagram. After that, we give a three-phase algorithm, i.e., vertex location, edge tracing and face extraction, among which the key step is to adaptively subdivide the initial large box into a set of sufficiently small-size boxes such that each box contains at most one Apollonius vertex. We also use the tool of 2D centroidal Voronoi tessellation (CVT) to yield well-tessellated triangle meshes of the curved bisectors to facilitate common digital geometry processing tasks. Finally, we provide a set of statistics to show the advantages of our algorithm, followed by discussing its application in centroidal Apollonius diagram.

This paper makes the following contributions:

1. A systematical analysis of the topological and geometric features of 3D Apollonius diagrams and a full discussion on degenerate cases.
2. A fast algorithm for computing 3D Apollonius diagrams. The key idea is to adaptively subdivide an initial large box into a set of sufficiently small boxes such that each box contains at most one Apollonius vertex.
3. A computational tool for representing 3D Apollonius diagrams using well-tessellated triangle meshes that can benefit the downstream applications in digital geometry processing.

2. Related Works

2.1. Voronoi Tessellations

Generally speaking, the Voronoi diagram defines the partitioning by a given domain $\Omega \in \mathbb{R}^n$ into convex regions based on the Euclidean distance to a set of points $\{\mathbf{x}_i \in \Omega\}_{i=1}^n$, called ***sites*** or ***generators***, where $\mathbf{x}_i$ dominates the subregion

$$\Omega_i = \{\mathbf{x} \in \Omega \mid \|\mathbf{x}-\mathbf{x}_i\| \leq \|\mathbf{x}-\mathbf{x}_j\|, i \neq j\}. \quad (1)$$

Due to its nice geometric properties, the Voronoi diagram has been widely applied to many science and engineering fields [Aur91, Sen93], including computer graphics, image processing, robot navigation, computational chemistry, materials science, climatology, etc. There is a large body of literature on computing Voronoi diagrams including the divide-and-conquer scheme [SH75], the incremental construction [GS78], and Fortune's elegant sweep line algorithm [For87], etc. Some later parallelized algorithms include randomization [RR94], divide-and-conquer [CGD90], and graphics hardware based implementations [XWX*13, HIKL*99, RLW*10]. The computation of Voronoi diagrams can also be implemented based on its dual, i.e., the Delaunay triangulation [Dev98], which is an equally important problem. Note that the Delaunay triangulation can be constructed by a lifting technique [For95].

2.2. Power Diagrams

The power diagram extends the Voronoi diagram by associating each site $\mathbf{x}_i$ with a weighting scalar w_i such that the proximity is measured using a different metric. In a power diagram, the cell belonging to $\mathbf{x}_i$ is

$$\Omega_i = \{\mathbf{x} \in \Omega \mid \|\mathbf{x}-\mathbf{x}_i\|^2 - w_i \leq \|\mathbf{x}-\mathbf{x}_j\|^2 - w_j, i \neq j\}. \quad (2)$$

The power diagram reduces to the Voronoi diagram when all the weights are equal [SU00]. It was proposed and widely studied since 1980s [IIM85, Aur87]. Gavrilova et al. [GR96] proposed an efficient algorithm to construct the power diagram from the Voronoi diagram. It was later shown that the power diagram is helpful for solving the semi-discrete optimal mass transport problem [AHA98]. Similar with the relationship between Voronoi diagram and Delaunay triangulation, there is a strong duality between power diagram and regular triangulation [Gli05]. Furthermore, Dou et al. [DXX*19] showed that the power diagram w.r.t. medial balls is able to reconstruct the original boundary surface.

2.3. 3D Apollonius Diagrams

The Apollonius diagram, also named additively weighted Voronoi diagram, is also an extension of the Voronoi diagram but much different from the power diagram. The Apollonius diagram also needs to be equipped with a weight set $\{w_i\}_{i=1}^n$, and correspondingly the cell belonging to $\mathbf{x}_i$ is

$$\Omega_i = \{\mathbf{x} \in \Omega \mid \|\mathbf{x}-\mathbf{x}_i\| - w_i \leq \|\mathbf{x}-\mathbf{x}_j\| - w_j, i \neq j\}. \quad (3)$$

The Apollonius diagram has a more complicated combinational structure than Voronoi diagrams. Furthermore, the common boundaries between two adjacent cells have a hyperbolic form and may consist of disconnected segments. It can be obtained by projecting the lower envelope of a set of cones rooted at $\{(\mathbf{x}_i, -w_i)\}_{i=1}^n$ [Wil99, Wil98, ABMY02]. Emiris and Karavelas [EK06] discussed the necessary predicates for computing a 2D Apollonius diagram. CGAL includes a robust solver for computing a 2D Apollonius diagram. Wang et al. [WFX*18] proposed to trace high-quality geodesic isolines by using the 2D Apollonius diagram technique.

The computation of 3D Apollonius diagrams is of our interest. There are quite a few research works [Wil99, KCKC04, KCK*05c,

KKC05, KCK05a, KK06] on the topic. Among them, Kim and his co-authors made a great progress and their region-expansion algorithm [KCK05a, KK06], of an $O(n^3)$ time complexity, runs fast in practice and becomes the state of the art on the computation of 3D Apollonius Diagrams, to our best knowledge. It can be seen that the challenges include at least two aspects: (1) although the Apollonius vertices can be found by checking $O(n^4)$-many site quadruples one by one, it is too computationally expensive - the key lies in developing an effective strategy for quickly selecting the $O(n)$-many helpful site quadruples, and (2) the bisectors of Apollonius diagrams have a hyperbolic form, which is unfriendly to general digital geometry processing tasks. In this paper, our motivation is to overcome the couple of challenges and develop a robust solver for 3D Apollonius Diagrams.

2.4. Centroidal Voronoi/Power Diagrams

The centroidal Voronoi tessellation (CVT) w.r.t. a set of sites $X = \{\mathbf{x}_i\}_{i=1}^n$ is a special Voronoi diagram for which each site $\mathbf{x}_i \in \Omega$ coincides with the centroid of $\mathbf{x}_i$'s cell [DFG99]. Mathetically, CVT can be computed by minimizing the following objective function:

$$E(X) = \sum_{i=1}^{n} \int_{\Omega_i} \|\mathbf{x} - \mathbf{x}_i\|^2 \rho(\mathbf{x}) d\sigma. \tag{4}$$

Where $\rho(\mathbf{x})$ is the density function defined on the domain Ω. CVT is central to compute an optimal partition [LSZ*14, ZJW15, BH08], and thus can be applied to a range of occasions including sampling, remeshing and super-pixel generation. In the past research, prevailing methods for solving CVTs include linearly convergent Lloyd's method [DEJ06] and the super-linearly convergent quasi-Newton method [LWL*09].

When the mass constraints are further enforced, the CVT can be promoted to the centroidal power diagram (CPD) [DGBOD12]. Beside the nice property that each site coincides with the centroid of its power cell, the mass of each site has to exactly meet the given mass constraints. de Goes et al. [DGBOD12] developed a Lloyd method, while Xin et al. [XLC*16] proposed to use a L-BFGS method to solve this problem. Most of the past research assumes the 2-Wasserstein metric which takes $\|\mathbf{x} - \mathbf{x}_i\|^2$ as the kernel. In this paper, we change the kernel from $\|\mathbf{x} - \mathbf{x}_i\|^2$ to $\|\mathbf{x} - \mathbf{x}_i\|$. It is observed that the formulation leads to a new kind of mass-controllable tessellations, i.e. centroidal Apollonius diagrams.

3. Preliminaries

In this section, we systematically discuss the basic properties of a 3D Apollonius diagram, which provides a base for further studying the combinatoric structure of Apollonius vertices, Apollonius edges and Apollonius cells. Some theoretical results are built on the existing research literature [KCK05b, GR03].

Given a set of sites $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^3$, each site $\mathbf{x}_i$ equipped with a weight w_i, we shall give a set of preliminaries in the following. As shown in Figure 2, the site $\mathbf{x}_i$ may be hidden from another site $\mathbf{x}_j$, i.e.,

$$\|\mathbf{x}_i - \mathbf{x}\| - w_i \geq \min_{\exists j} \{\|\mathbf{x}_j - \mathbf{x}\| - w_j\} \tag{5}$$

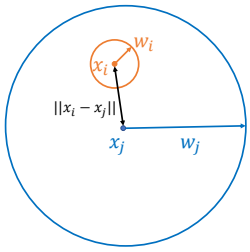


Figure 2: *The site x_i is hidden by x_j (Theorem 1).*

holds for any point $\mathbf{x} \in \mathbb{R}^3$. In this case, $\mathbf{x}_i$ doesn't contribute to the Apollonius diagram $\mathcal{A}$ at all. In the following, we elaborate the sufficient and necessary condition of the existence of hidden sites.

Theorem 1 The Apollonius diagram $\mathcal{A}$ doesn't include a hidden site if and only if any pair of sites $\mathbf{x}_i, \mathbf{x}_j$ meets

$$\|\mathbf{x}_i - \mathbf{x}_j\| > |w_i - w_j|. \tag{6}$$

Proof We suppose that $\|\mathbf{x}_i - \mathbf{x}_j\| > |w_i - w_j|$ for any $j \neq i$, then the site $\mathbf{x}_i$ dominates the point itself (assuming $\mathbf{x} = \mathbf{x}_i$), i.e.,

$$\|\mathbf{x}_i - \mathbf{x}_i\| - w_i < \min_{j \neq i} \{\|\mathbf{x}_j - \mathbf{x}_i\| - w_j\}. \tag{7}$$

Therefore, there exists a non-empty point set, including at least $\mathbf{x}_i$, which is closer to $\mathbf{x}_i$ than any other sites (in the sense of additively weighted distance), which shows that $\mathbf{x}_i$ is not a hidden site, and vice versa. □

For the convenience of statement, we assume that there is no hidden site at the moment. (In spite this, our algorithm will naturally support the existence of hidden sites.) Furthermore, we exclude two spatial coincidence cases including:

- Any triple $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ cannot lie on a common straight line.
- Any quadruple $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k, \mathbf{x}_l)$ cannot lie on a common plane.

First, we shall show for each non-hidden site $\mathbf{x}_i$, the region dominated by $\mathbf{x}_i$ forms a star shape, which is the following theorem.

Theorem 2 Let $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^3$ be a set of sites, each site $\mathbf{x}_i$ equipped with a weight w_i. The cell belonging to the site $\mathbf{x}_i$ forms a star shape w.r.t. $\mathbf{x}_i$.

Proof Let $\mathbf{x}$ be a point dominated by $\mathbf{x}_i$. We have

$$\|\mathbf{x}_i - \mathbf{x}\| - w_i < \min_{j \neq i} \{\|\mathbf{x}_j - \mathbf{x}\| - w_j\}. \tag{8}$$

For any point $p \in \overline{\mathbf{x}\mathbf{x}_i}$, $\|\mathbf{x}_i - p\| - w_i = \|\mathbf{x}_i - \mathbf{x}\| - w_i - \|\overline{\mathbf{x}p}\|$. Considering $\|\mathbf{x}_j - p\| - w_j \geq \|\mathbf{x}_j - \mathbf{x}\| - w_j - \|\overline{\mathbf{x}p}\|$, for any $j \neq i$, we have

$$\|\mathbf{x}_i - p\| - w_i < \min_{j \neq i} \{\|\mathbf{x}_j - p\| - w_j\}, \tag{9}$$

which implies that the cell belonging to the site $\mathbf{x}_i$ forms a star shape w.r.t. $\mathbf{x}_i$. □

Theorem 3 Given two sites $\mathbf{x}_1, \mathbf{x}_2$, respectively with a weight w_1, w_2 and $0 < w_1 - w_2 < \|\mathbf{x}_1 - \mathbf{x}_2\|$, the equal-distance point set forms a hyperbolic surface $\mathcal{H}$. Let Π be the perpendicular bisector plane of $\mathbf{x}_1$ and $\mathbf{x}_2$. If we map a point $h \in \mathcal{H}$ to h's projection p onto

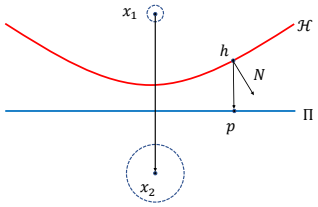


Figure 3: *Theorem 3*

Π, we get a 1-to-1 mapping. The ratio of the infinitesimal area on Π to that on $\mathcal{H}$ is

$$\frac{|\mathbf{N}\cdot(\mathbf{x}_2-\mathbf{x}_1)|}{\|\mathbf{N}\|\cdot\|\mathbf{x}_2-\mathbf{x}_1\|}, \tag{10}$$

where the normal vector $\mathbf{N}$ at h is given by

$$\mathbf{N}=\frac{h-\mathbf{x}_1}{\|\mathbf{x}_1-h\|}-\frac{h-\mathbf{x}_2}{\|\mathbf{x}_2-h\|}. \tag{11}$$

Proof For any point $h\in\mathcal{H}$, $\|\mathbf{x}_1-h\|-w_1=\|\mathbf{x}_2-h\|-w_2$. The projection of h on Π can be represented by

$$p=h+\lambda\cdot(\mathbf{x}_2-\mathbf{x}_1). \tag{12}$$

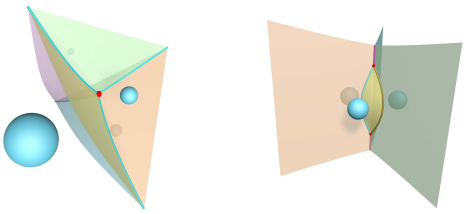
Since p is also located on the perpendicular bisector plane of $\mathbf{x}_1$ and $\mathbf{x}_2$, we have

$$(p-\frac{\mathbf{x}_1+\mathbf{x}_2}{2})\cdot(\mathbf{x}_2-\mathbf{x}_1)=0. \tag{13}$$

Putting them together, we have

$$\lambda=\frac{(\frac{\mathbf{x}_1+\mathbf{x}_2}{2}-h)\cdot(\mathbf{x}_2-\mathbf{x}_1)}{\|\mathbf{x}_2-\mathbf{x}_1\|^2}, \tag{14}$$

which implies that for any point $h\in\mathcal{H}$, we can find a projection p. Furthermore, different h's lead to different projections p's.
On the other hand, let p be a point on the perpendicular bisector plane. Let $h=p+\mu(\mathbf{x}_1-\mathbf{x}_2)$ be the intersection point between the straight line $\mathbf{x}_1\mathbf{x}_2$ and the hyperbolic surface $\|\mathbf{x}_1-h(\mu)\|-w_1=\|\mathbf{x}_2-h(\mu)\|-w_2$. μ can be solved by finding the roots to a quadratic equation, and exactly only one of them makes sense. Finally, the area ratio can be quickly obtained from Lambert's cosine law. □



(a) Type I (b) Type II

Figure 4: *Given 4 sites $\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4\in\{\mathbf{x}_i\}_{i=1}^n$, we show two typical situations on how they determine the Apollonius vertices. (a) There is only one Apollonius vertex. (b) There are two Apollonius vertices.*

Theorem 4 Let $\{\mathbf{x}_i\}_{i=1}^n\in\mathbb{R}^3$ be a set of sites, any four of which are non-coplanar. Each site $\mathbf{x}_i$ is equipped with a weight w_i. Denote by $\mathcal{A}$ the Apollonius diagram of $\{\mathbf{x}_i\}_{i=1}^n$. Suppose that $\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4\in\{\mathbf{x}_i\}_{i=1}^n$ are one of the quadruples, which contributes to $\mathcal{A}$ in one of the following situations:

1. No point $v\in\mathbb{R}^3$ is able to meet $\|v-\mathbf{x}_1\|-w_1=\|v-\mathbf{x}_2\|-w_2=\|v-\mathbf{x}_3\|-w_3=\|v-\mathbf{x}_4\|-w_4<\|v-\mathbf{x}_i\|-w_i, i=5,6,\cdots,n$; In this case, x_1,x_2,x_3,x_4 cannot give an Apollonius vertex.
2. Only one point $\mathbf{v}$ exists such that $\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4$ are equally distant to v and the weighted distance to $\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4$ is less than that to the other sites; v is called the Apollonius vertex given by $\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4$.
3. Two points, say, v_1,v_2, are determined by $\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4$; Both of them are Apollonius vertices.

Proof Let (x,y,z) be the coordinates of Apollonius vertex v. We have

$$\begin{cases}\sqrt{(x-x_1)^2+(y-y_1)^2+(z-z_1)^2}-w_1=r\\ \sqrt{(x-x_2)^2+(y-y_2)^2+(z-z_2)^2}-w_2=r\\ \sqrt{(x-x_3)^2+(y-y_3)^2+(z-z_3)^2}-w_3=r\\ \sqrt{(x-x_4)^2+(y-y_4)^2+(z-z_4)^2}-w_4=r\end{cases}, \tag{15}$$

where (x_i,y_i,z_i) are the coordinates of $\mathbf{x}_i$ and r is the common weighted distance. Rewriting the above equations leads to

$$\begin{cases}(x-x_1)^2+(y-y_1)^2+(z-z_1)^2=(w_1+r)^2\\ (x-x_2)^2+(y-y_2)^2+(z-z_2)^2=(w_2+r)^2\\ (x-x_3)^2+(y-y_3)^2+(z-z_3)^2=(w_3+r)^2\\ (x-x_4)^2+(y-y_4)^2+(z-z_4)^2=(w_4+r)^2\end{cases}. \tag{16}$$

Subtracting the first three equations by the fourth one yields

$$\mathbf{A}_{3\times3}\begin{bmatrix}x\\y\\z\end{bmatrix}=r\begin{bmatrix}b_1\\b_2\\b_3\end{bmatrix}+\begin{bmatrix}c_1\\c_2\\c_3\end{bmatrix}, \tag{17}$$

where $b_i,c_i,i=1,2,3$, are constants and matrix $\mathbf{A}_{3\times3}$ is

$$\mathbf{A}=\begin{bmatrix}x_1-x_4 & y_1-y_4 & z_1-z_4\\ x_2-x_4 & y_2-y_4 & z_2-z_4\\ x_3-x_4 & y_3-y_4 & z_3-z_4\end{bmatrix}. \tag{18}$$

If $\mathbf{A}$ is invertible, x, y, and z linearly depend on r (multiplying $\mathbf{A}^{-1}$ on both sides of Eq. (17)):

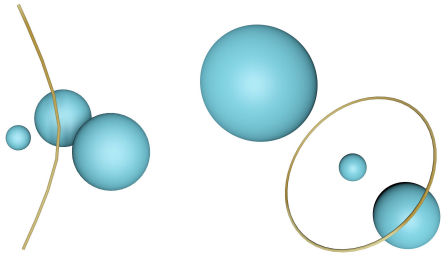
$$\begin{cases}x=\alpha_1 r+\beta_1\\ y=\alpha_2 r+\beta_2\\ z=\alpha_3 r+\beta_3\end{cases}, \tag{19}$$

where $\alpha_i,\beta_i,i=1,2,3$, are constants. Taking them into the fourth equation of Eq. (16), we get a quadratic equation of r. Therefore r has at most two solutions r_1,r_2. Eq. (19) implies that if $\mathbf{A}$ is invertible, then $\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4$ give at most two Apollonius vertices. There are three cases to consider:

- If both r_1 and r_2 satisfy Eq. (15), and the corresponding vertices are closer to $\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4$ than any other site, then they give two Apollonius vertices at the same time.
- If only one of r_1,r_2 satisfies Eq. (15) but the corresponding vertex is closer to another site different from $\mathbf{x}_1$,$\mathbf{x}_2$,$\mathbf{x}_3$,$\mathbf{x}_4$ or both the vertices given by r_1,r_2 are occupied by another site, then they give no Apollonius vertex.
- Otherwise, they give only one Apollonius vertex.

Figure 4 shows the two typical situations. For the case where A

is non-invertible, the tetrahedron $\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4$ has zero volume, since $\frac{1}{6}\det|\mathbf{A}| = 0$. Therefore, $\mathbf{x}_1$,$\mathbf{x}_2$,$\mathbf{x}_3$,$\mathbf{x}_4$ are co-planar, which contradicts to our non-coplanarity assumption. □



(a) Open (b) Closed

Figure 5: *When there are three sites, without any hidden sites or degenerate cases, they determine an open or closed Apollonius edge.*

Theorem 5 Let $\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3$ be three sites (no other site exists), each site equipped with a weight w_i. Under the case that no hidden site exists and they are not co-linear, $\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3$ determine a planar Apollonius edge in the following two situations (see Figure 5):

1. A curve both of whose endpoints stretching to infinity (Figure 5(a)).
2. A closed Apollonius curve (Figure 5(b)).

Furthermore, for each point h on the Apollonius edge, the tangent direction at h is given by

$$\left(\frac{h-\mathbf{x}_1}{\|\mathbf{x}_1-h\|}-\frac{h-\mathbf{x}_2}{\|\mathbf{x}_2-h\|}\right)\times\left(\frac{h-\mathbf{x}_2}{\|\mathbf{x}_2-h\|}-\frac{h-\mathbf{x}_3}{\|\mathbf{x}_3-h\|}\right).$$

Proof Generally, for a point (x,y,z) on the Apollonius edge, we have

$$\begin{cases}\sqrt{(x-x_1)^2+(y-y_1)^2+(z-z_1)^2}-w_1=r\\ \sqrt{(x-x_2)^2+(y-y_2)^2+(z-z_2)^2}-w_2=r\\ \sqrt{(x-x_3)^2+(y-y_3)^2+(z-z_3)^2}-w_3=r\end{cases}, \tag{20}$$

where r is the common weighted distance. Similarly, we can reshape the above equations as follows:

$$\begin{cases}(x-x_1)^2+(y-y_1)^2+(z-z_1)^2=(w_1+r)^2\\ (x-x_2)^2+(y-y_2)^2+(z-z_2)^2=(w_2+r)^2\\ (x-x_3)^2+(y-y_3)^2+(z-z_3)^2=(w_3+r)^2\end{cases}. \tag{21}$$

We subtract the first two equations by the third one, yielding

$$\begin{bmatrix}x_1-x_3 & y_1-y_3 & z_1-z_3\\ x_2-x_3 & y_2-y_3 & z_2-z_3\end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix}=r\begin{bmatrix}b_1\\b_2\end{bmatrix}+\begin{bmatrix}c_1\\c_2\end{bmatrix}, \tag{22}$$

where $b_i,c_i,i=1,2$, are constants. Since $\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3$ are not co-linear, the cross product γ between $(x_1-x_3,y_1-y_3,z_1-z_3)$ and $(x_2-x_3,y_2-y_3,z_2-z_3)$ is a non-zero vector. Let ξ,η be two unit vectors that are orthogonal to γ, meeting $\xi\cdot\eta=\xi\cdot\gamma=\eta\cdot\gamma=0$, and we can decompose $(x_1-x_3,y_1-y_3,z_1-z_3)$ and $(x_2-x_3,y_2-y_3,z_2-z_3)$ into the linear combinations w.r.t. ξ and η. Therefore, the point (x,y,z) spans a one-dimensional space:

$$\begin{pmatrix}x\\y\\z\end{pmatrix}=r\begin{pmatrix}\alpha_1\\\alpha_2\\\alpha_3\end{pmatrix}+\begin{pmatrix}\beta_1\\\beta_2\\\beta_3\end{pmatrix}+t\gamma, \tag{23}$$

where $\alpha_i,\beta_i,i=1,2,3$ are constants (depending on ξ and η), and t is a parameter ranging from $-\infty$ to ∞.
Let ς be the cross product between $(\alpha_1,\alpha_2,\alpha_3)^{\mathrm{T}}$ and γ. Then

$$\begin{pmatrix}x\\y\\z\end{pmatrix}\cdot\varsigma=\begin{pmatrix}\beta_1\\\beta_2\\\beta_3\end{pmatrix}\cdot\varsigma=\text{const}, \tag{24}$$

which shows that the intersection curve $(x(t),y(t),z(t))$ is planar. Finally, for each point h on the Apollonius edge, the tangent direction at h can be inferred based on Theorem 3. □

4. Structures of Vertices, Edges and Faces

In this section, we discuss the structures of vertices, edges and faces respectively under the assumption that no degenerate case occurs.

4.1. Apollonius Vertex

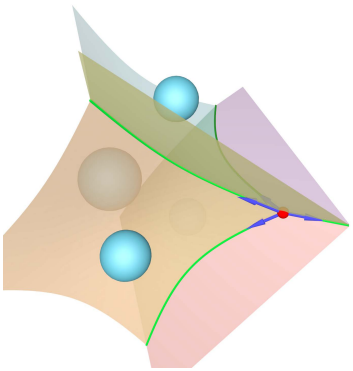


Figure 6: *A typical 4-fork Apollonius vertex.*

As mentioned above, each Apollonius vertex v is given by four different weighted sites $(\mathbf{x}_1,w_1)$, $(\mathbf{x}_2,w_2)$, $(\mathbf{x}_3,w_3)$, $(\mathbf{x}_4,w_4)$. Furthermore, there are 6 Apollonius faces, as well as 4 Apollonius edges, rooted at v, as Figure 6 shows. For example, $\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3$ are able to define an Apollonius edge e. Based on Theorem 5, the tangent direction at v along the Apollonius edge e (see the arrows in Figure 6) is given by

$$\mathbf{T}=\left(\frac{v-\mathbf{x}_1}{\|\mathbf{x}_1-v\|}-\frac{v-\mathbf{x}_2}{\|\mathbf{x}_2-v\|}\right)\times\left(\frac{v-\mathbf{x}_2}{\|\mathbf{x}_2-v\|}-\frac{v-\mathbf{x}_3}{\|\mathbf{x}_3-v\|}\right). \tag{25}$$

Next, we discuss whether the marching direction of e is in the same direction of $\mathbf{T}$ or in the reverse direction. We build a function

$$F(t)=\|\mathbf{x}_4-(v+t\mathbf{T})\|-w_4-\|\mathbf{x}_1-(v+t\mathbf{T})\|+w_1, \tag{26}$$

which meets $F(0)=0$. Considering that

$$F'(t)=\frac{(v+t\mathbf{T}-\mathbf{x}_4)\cdot\mathbf{T}}{\|\mathbf{x}_4-(v+t\mathbf{T})\|}-\frac{(v+t\mathbf{T}-\mathbf{x}_1)\cdot\mathbf{T}}{\|\mathbf{x}_1-(v+t\mathbf{T})\|}, \tag{27}$$

we have

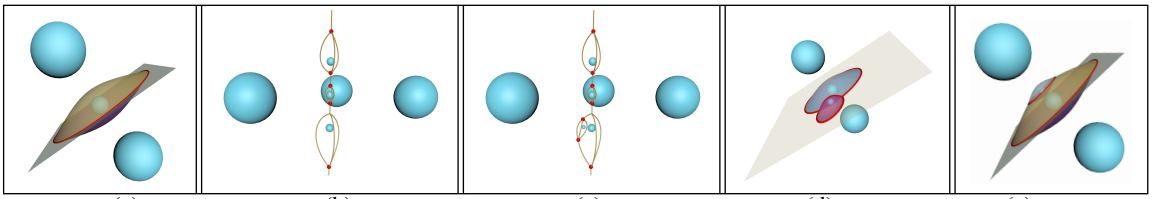


Figure 7: *Combinatorial structures of Apollonius cells. (a) Apollonius-face hump, (b) Apollonius-edge hump, (c) Apollonius-edge hump sitting on an Apollonius edge, (d) Apollonius-edge hump sitting on a closed Apollonius edge, and (e) Apollonius-face hump on an Apollonius-face hump.*

$$F'(0) = \left(\frac{v-\mathbf{x}_4}{\|\mathbf{x}_4-v\|} - \frac{v-\mathbf{x}_1}{\|\mathbf{x}_1-v\|}\right) \cdot \left(\left(\frac{v-\mathbf{x}_1}{\|\mathbf{x}_1-v\|} - \frac{v-\mathbf{x}_2}{\|\mathbf{x}_2-v\|}\right) \times \left(\frac{v-\mathbf{x}_2}{\|\mathbf{x}_2-v\|} - \frac{v-\mathbf{x}_3}{\|\mathbf{x}_3-v\|}\right)\right) \neq 0, \quad (28)$$

since the three vectors, each being the normal vector of a hyperbolic surface at v, are impossible to be co-planar (otherwise, the four sites must be co-planar). If $F'(0) > 0$, then the marching direction of the Apollonius edge e should be in the same direction $\mathbf{T}$. Otherwise, we trace the Apollonius edge e in the reverse direction of $\mathbf{T}$. In Figure 6, we mark the four directions at v.

4.2. Apollonius Edge

In fact, there are totally four types of Apollonius edges including

- A curve both of whose endpoints stretching to infinity;
- A closed Apollonius curve without any Apollonius vertex;
- A curved segment both of whose endpoints are vertices;
- A curve starting from an Apollonius vertex v to infinity.

The first two types are shown in Figure 5, and the last two types are formed by interception. For the third type, we suppose that the Apollonius edge connects two Apollonius vertices v_1, v_2. Let v_1 be given by $X_1 = \left(\mathbf{x}_1^{(v_1)}, \mathbf{x}_2^{(v_1)}, \mathbf{x}_3^{(v_1)}, \mathbf{x}_4^{(v_1)}\right)$ and v_2 be given by $X_2 = \left(\mathbf{x}_1^{(v_2)}, \mathbf{x}_2^{(v_2)}, \mathbf{x}_3^{(v_2)}, \mathbf{x}_4^{(v_2)}\right)$. Then it can be shown that X_1 and X_2 are either identical or share three common sites.

4.3. Apollonius Face

The simplest case of an Apollonius face is a complete and unbounded hyperbolic surface. But generally, an Apollonius face may have some boundaries, which is induced by clipping a complete hyperbolic surface with other hyperbolic surfaces. However, what is different from Voronoi diagrams is that an Apollonius face f may not be simply connected. To be more detailed, there may exist some humps inside f or across f's boundary; See Figure 7 for an illustration.

In Voronoi/power diagrams, there is a unique edge between two adjacent vertices. But the structure of an Apollonius diagram is very different. Specially, the following two cases distinguish itself from Voronoi/power diagrams:

1. There may exist three Apollonius edges between two adjacent vertices.
2. There may exist a closed Apollonius edge without any Apollonius vertices.

The two cases lead to Apollonius-face humps and Apollonius-edge humps, which are two special forms in Apollonius diagrams; See Figure 7(a-b). Therefore, there are many possible combinations of such humps in one Apollonius diagram. Figure 7(c-e) show (1) Apollonius-edge humps on open Apollonius edges, (2) Apollonius-edge humps on closed Apollonius edges, and (3) Apollonius-face humps on Apollonius faces.

5. Algorithm

5.1. Overview

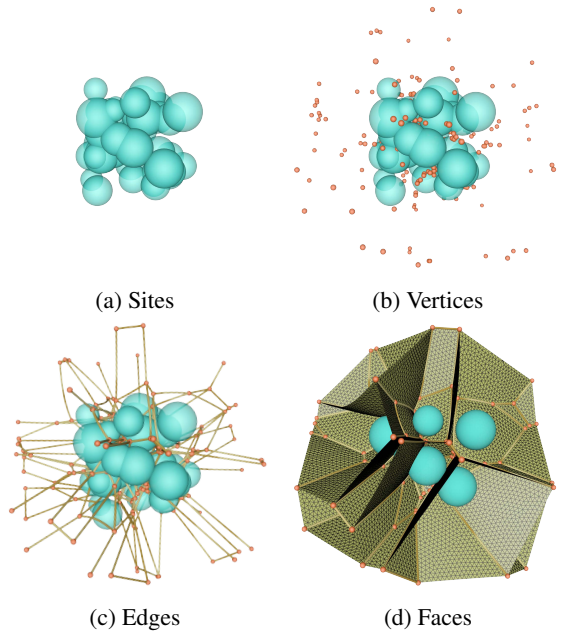


Figure 8: *Algorithmic pipeline. Given a set of sites (a), we first compute the Apollonius vertices (b), then compute the Apollonius edges (c), and finally compute the Apollonius faces (d).*

Given a set of sites $\{\mathbf{x}_i\}_{i=1}^n$ and the associate weights $\{w_i\}_{i=1}^n$,

our task is to build the proximity between the sites using the additively weighted distance. As Figure 8 shows, we extract the Apollonius vertices, edges and faces step by step. We first locate the Apollonius vertices by a fast sweeping technique. After that, we connect the Apollonius vertices into Apollonius edges. In the last step, we build a high-quality meshing representation of each Apollonius faces.

5.2. Apollonius Vertices

5.2.1. Finding candidate sites for each cube

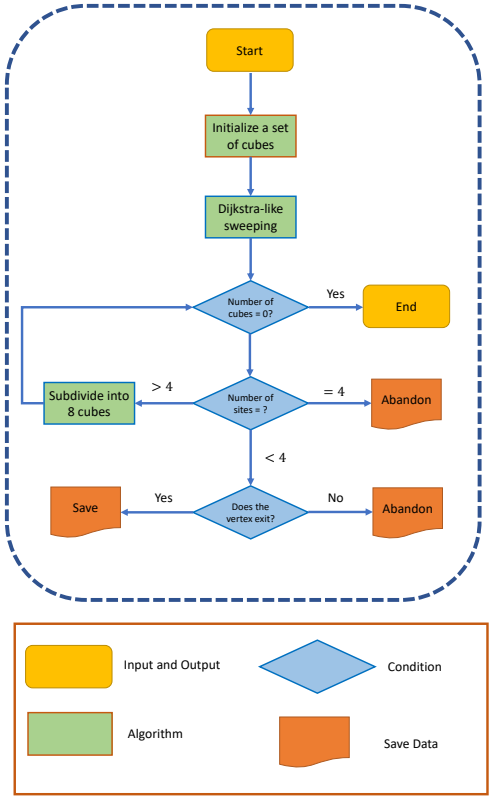


Figure 9: *Algorithm flowchart for Apollonius vertex location*

Our strategy is to adaptively subdivide the region of interest into small cubes such that each cube $\mathcal{C}$ contains at most one Apollonius vertex.

Vertex location. Initially, we partition the bounding box into a set of cubes. After that, we propagate the distance from the given sites into the cubes, in a Dijkstra-like style. For each cube $\mathcal{C}$, we keep the sites that may dominate a part of the cube $\mathcal{C}$ (See Section 5.2.2). Next, we push the cubes into a queue for further processing. Suppose $\mathcal{C}$ is the cube popped from the queue. We visit the sites kept in $\mathcal{C}$ and filter out those useless sites based on a Filtering rule (which will be given later). If the number of remaining sites in $\mathcal{C}$ is less than 4, then $\mathcal{C}$ cannot contain an Apollonius vertex. If the number is exactly 4, then we compute the coordinates of the Apollonius vertex and check its validity. Otherwise, we continue subdividing $\mathcal{C}$ into 8 sub-cubes and push them into the queue for further processing. See Figure 9 for the workflow of our algorithm.

Data structure. In implementation, the basic data structure is to describe a cube element:

```
Struct Cube
{
    int  cube_id;
    double cube_edge_length;
    vector candidate_sites;
    point center;
    double minimum_weighted_distance_to_center;
};
```

During the sweeping process, we need to repeatedly generate distance propagation events that are processed with a priority queue, where we encode an event by

```
Struct Event
{
    int source_site;
    int cube_id;
    double weighted_distance_to_center;
};
```

At the end of the distance propagation process, we keep several sites in each cube - the sites kept in $\mathcal{C}_i$ may be helpful in determining the weighted distance to at least one point in $\mathcal{C}_i$.

5.2.2. Filtering Rules

First, we shall build the sufficient condition for the case that $\mathcal{C}$ doesn't include an Apollonius vertex.

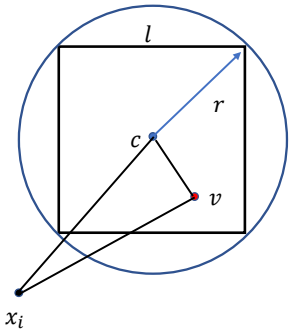


Figure 10: *Proof to Theorem 6*

Theorem 6 Let l be the edge length of a cube $\mathcal{C}$ and c be the center of $\mathcal{C}$. The weighted distance from the site $\mathbf{x}_i$ to the points in $\mathcal{C}$ ranges in $[\|\mathbf{x}_i-c\|-w_i-\frac{\sqrt{3}}{2}l, \|\mathbf{x}_i-c\|-w_i+\frac{\sqrt{3}}{2}l]$. If the number of sites satisfying

$$\left\{\mathbf{x}_i \;\middle|\; \|\mathbf{x}_i-c\|-w_j-\frac{\sqrt{3}}{2}l < \min_j\{\|\mathbf{x}_j-c\|-w_j+\frac{\sqrt{3}}{2}l\}\right\} \tag{29}$$

is less than 4, then there is no Apollonius vertex in $\mathcal{C}$.

Proof Let us assume that there is an Apolonius vertex v inside the sphere of radius $r=\frac{\sqrt{3}}{2}l$ enclosing the cube $\mathcal{C}$. Then there are four sites $\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4$ such that

$$\|\mathbf{x}_i-v\|-w_i = const < \|\mathbf{x}_j-v\|-w_j, i=1,2,3,4, j>4. \tag{30}$$

Plugging the following two inequalities into the left side

$$\|\mathbf{x}_i - c\| - \|v - c\| \le \|\mathbf{x}_i - v\|, \quad \|v - c\| \le r, \tag{31}$$

we have

$$\|\mathbf{x}_i - c\| - w_i - r \le \|\mathbf{x}_i - v\| - w_i. \tag{32}$$

Similarly, based on the triangle inequality, we have

$$\|\mathbf{x}_j - v\| - w_j \le \|\mathbf{x}_j - c\| - w_j + r. \tag{33}$$

Putting Eq. (30), Eq. (32) and Eq. (33) together, we have

$$\|\mathbf{x}_i - c\| - w_i - r \le \|\mathbf{x}_j - c\| - w_j + r, \tag{34}$$

where $r = \frac{\sqrt{3}}{2}l$. □

The above theorem enables us to greatly filter out those sites far from the cube $\mathcal{C}$. But when there are two sites $\mathbf{x}_1, \mathbf{x}_2$ that are very close to each other, it is very difficult to exclude one of the sites using the other based on the theorem. Therefore, we introduce one more theorem to further improve the filtering effect.

Theorem 7 Let l be the edge length of a cube $\mathcal{C}$ and c be the center of $\mathcal{C}$. There are two sites $(\mathbf{x}_1, w_1), (\mathbf{x}_2, w_2)$ and c has a projection c' on the hyperbolic bisector surface. Then they cannot contribute to determining an Apollonius vertex in $\mathcal{C}$ at the same time if

$$\|c - c'\| \ge \frac{\sqrt{3}}{2}l \tag{35}$$

and the eight corners of $\mathcal{C}$, as well as the center c', lie on the same side of the hyperbolic surface.

Proof Due to the assumption that the eight corners of $\mathcal{C}$ lie on the same side of the hyperbolic surface $\mathcal{H}$, we consider the problem in two cases based on the fact that $\mathcal{H}$ divide the whole Euclidean space into two parts, one convex part and one concave part.
Case1 If the eight corners are all located in the convex part, the whole cube $\mathcal{C}$ is also inside the convex part, which is due to the convexity. This shows that the hyperbolic surface $\mathcal{H}$ cannot go through the interior of the cube $\mathcal{C}$.
Case2 If the eight corners are located in the concave part, we make a tangent plane at c' and the tangent plane is totally inside the concave part. Since $\|c - c'\| \ge \frac{\sqrt{3}}{2}l$, the entire cube $\mathcal{C}$ is on the same side of the tangent plane and thus $\mathcal{C}$ cannot intersect the hyperbolic surface $\mathcal{H}$. To summarize, $(\mathbf{x}_1, w_1), (\mathbf{x}_2, w_2)$ cannot contribute to determining an Apollonius vertex in $\mathcal{C}$ at the same time, which enables us to filter out one site using the other. □

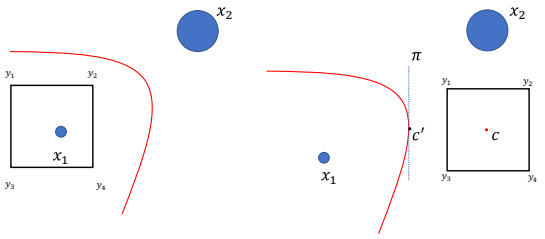


Figure 11: *Two situations discussed in Theorem 8*

Furthermore, it is not difficult to validate the following filtering theorem.

Theorem 8 There are two sites $(\mathbf{x}_1, w_1), (\mathbf{x}_2, w_2), w_1 \le w_2$. The cube $\mathcal{C}$ has 8 corners $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_8$. Let c be the center of $\mathcal{C}$, c' be the projection on the bisector, and π be the tangent plane of the bisector at c'.

1. If $\|\mathbf{y}_i - \mathbf{x}_1\| - w_1 < \|\mathbf{y}_i - \mathbf{x}_2\| - w_2$ for $i = 1, 2, \cdots, 8$, the site $\mathbf{x}_1$ can give smaller distance to the cube $\mathcal{C}$ than $\mathbf{x}_2$.
2. Considering that $\mathbf{x}_1$ and $\mathbf{x}_2$ are on different sides of π, if the 8 corners lies on the same side with $\mathbf{x}_2$, the site $\mathbf{x}_2$ can give smaller distance to the cube $\mathcal{C}$ than $\mathbf{x}_1$.

Let v be an Apollonius vertex given by $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$. In a sufficiently small neighborhood around v, the weighted distance to $\mathbf{x}_i, i > 4$ must be larger than that given by $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$. Based on the observation, we develop a sweeping algorithm, resembling the scheme of Dijkstra's algorithm, to adaptively subdivide an initial cube set, each one divided into eight, until the cubic elements that may contain a unique Apollonius vertex are found.

In implementation, the data structure of an Apollonius vertex can be represented by

```
Struct ApolloniusVertex
{
    int idOfSite1, idOfSite2, idOfSite3, idOfSite4;
    vector tangent1, tangent2, tangent3, tangent4;
    point vertex;
};
```

5.3. Apollonius Edges

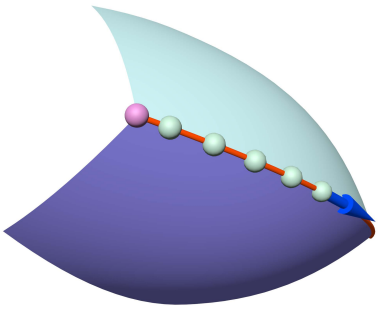


Figure 12: *Tracing an Apollonius edge.*

Each Apollonius vertex v is given by four different sites, e.g., $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$, where any three of them are able to define an open or closed curve that is equally distant to the three sites. We call the curve an ***primitive Apollonius edge*** at this moment. Note that each primitive Apollonius edge may contain many Apollonius edges, as is shown in Figure 7(b). There are at most $4m$ primitive Apollonius edges by checking the list of Apollonius vertices, where m is the total number of Apollonius vertices. We need to perform the following steps to accomplish tracing Apollonius edges:

1. Find all the primitive Apollonius edges by checking the Apollonius vertex list.
2. Link each Apollonius vertex v to the four primitive Apollonius edges that go through v.

3. For those Apollonius vertices on the primitive Apollonius edge $\mathcal{E}$, we label each vertex with "empty". For an arbitrary empty vertex, we trace the edge until the next empty vertex is encountered (See Section 4 for how to infer the marching direction). We connect the vertex pair into an Apollonius edge, and then mark them as "non-empty". (A slight modification is required for those Apollonius edges that are infinite or pass through the restricted region.)

5.4. Apollonius Faces

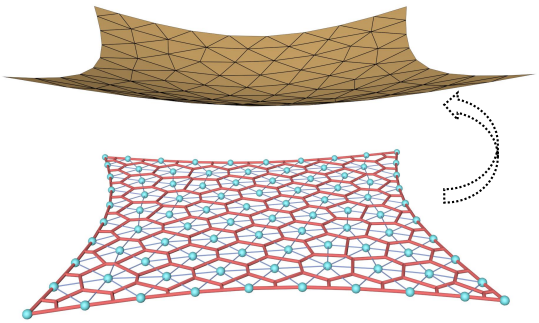


Figure 13: *We generate the high-quality triangulation of Apollonius faces by 2D CVT. The lower part shows the parametrization domain.*

To this end, we have a list of Apollonius edges at hand. The next step is to find the Apollonius vertex pairs that are helpful for determining an Apollonius face. Recall that each primitive Apollonius edge $\mathcal{E}$ is given by three different sites, e.g., $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$, and any two of them are able to define a hyperbolic surface. For example, $\mathbf{x}_1$ and $\mathbf{x}_2$ defines one branch of hyperbolic surface:

$$\|\mathbf{x}_1 - \mathbf{x}\| - w_1 = \|\mathbf{x}_2 - \mathbf{x}\| - w_2, \tag{36}$$

which is called a ***primitive Apollonius face*** $\mathcal{F}$. After that, we need to find all the Apollonius edges $\{\mathcal{E}_i\}$ that are located on $\mathcal{F}$. The Apollonius edge set $\{\mathcal{E}_i\}$ may enclose multiple connected regions $\{f_i\}$ on $\mathcal{F}$, each of which defines a real Apollonius face.

As Figure 13 shows, an Apollonius face f can be projected, by a one-to-one mapping, onto the perpendicular bisector plane Π of the corresponding two sites (See Theorem 3). We shall generate the triangulation of the planar region first and then map the triangulation back to the f. In order to achieve this, we first project the bounding curves of f onto Π. According to Theorem 3, we define a non-uniform density function ρ on Π:

$$\rho(p) = \frac{\| \frac{h(p)-\mathbf{x}_1}{\|\mathbf{x}_1-h(p)\|} - \frac{h(p)-\mathbf{x}_2}{\|\mathbf{x}_2-h(p)\|} \| \times \|\mathbf{x}_2 - \mathbf{x}_1\|}{|(\frac{h(p)-\mathbf{x}_1}{\|\mathbf{x}_1-h(p)\|} - \frac{h(p)-\mathbf{x}_2}{\|\mathbf{x}_2-h(p)\|}) \cdot (\mathbf{x}_2 - \mathbf{x}_1)|}, \tag{37}$$

where $h = h(p)$ is the inverse image point of p under the one-to-one mapping, and h can be computed by solving a quadratic equation. In this way, we can achieve a uniform triangulation of f with the help of ρ; See Figure 13.

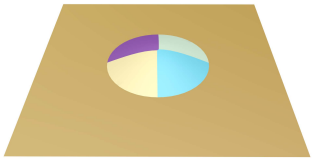


Figure 14: *Replication of Apollonius-face-hump sites - transform any Apollonius-face-hump site into 4 copies by replication and displacements. Note that in [KK06], the authors used the term "infinite edge" to name the closed Apollonius edges that need extra handling. Our strategy, in its spirit, is to break closed edges into open ones such that the Apollonius diagram can be computed in a simpler and more uniform algorithmic paradigm.*

5.5. Apollonius-face Humps

The above discussions begin with a step of finding Apollonius vertices and end with Apollonius face triangulation. In fact, there may exist Apollonius-face humps, as mentioned in Section 4, that have a closed boundary without any Apollonius vertex (see Figure 7). However, such closed boundaries fail to be captured by the sweeping algorithm. In the following, we need to distinguish Apollonius-face-hump sites from hidden sites when all the Apollonius vertices are located. Finally, we transform any Apollonius-face-hump site into 4 copies by replication and displacements.

5.5.1. Distinguishing Apollonius-face-hump Sites

The key criterion for checking if a site $\hat{\mathbf{x}}$ is helpful is whether $\hat{\mathbf{x}}$ can provide smaller weighted distance to itself than any other sites.

Let τ be the global minimum gap between sites and Apollonius faces. It is easy to show that if the size of a cube containing $\hat{\mathbf{x}}$ is less than τ, then the cube cannot intersect any Apollonius face. The observation inspires us to distinguish Apollonius-face-hump sites from hidden sites by making the subdivision sufficiently small. At the end of the sweeping process of finding Apollonius vertices, we can enforce an additional subdivision operation - keep subdividing those small cubes containing a site $\hat{\mathbf{x}}$ that doesn't help determining an Apollonius vertex until the following two cases happen:
Case 1: $\hat{\mathbf{x}}$ cannot contribute to determining an Apollonius vertex in $\hat{\mathbf{x}}$'s cube - $\hat{\mathbf{x}}$ is a hidden site.
Case 2: $\hat{\mathbf{x}}$ itself can provide smaller weighted distance for points in $\hat{\mathbf{x}}$'s cube than any other sites - $\hat{\mathbf{x}}$ is an Apollonius-face-hump site; See Figure 7(a).

5.5.2. Replication of Apollonius-face-hump Sites

As Figure 14 shows, for an Apollonius-face-hump site $\hat{\mathbf{x}}$, we generate four copies and perturb them in $\hat{\mathbf{x}}$'s neighborhood, and the resulting four alternative sites are respectively

$$\begin{cases} \mathbf{x}^{(1)} = \hat{\mathbf{x}} + \varepsilon(1,1,1) \\ \mathbf{x}^{(2)} = \hat{\mathbf{x}} + \varepsilon(-1,-1,1) \\ \mathbf{x}^{(3)} = \hat{\mathbf{x}} + \varepsilon(-1,1,-1) \\ \mathbf{x}^{(4)} = \hat{\mathbf{x}} + \varepsilon(1,-1,-1), \end{cases} \tag{38}$$

where ε is a sufficiently small constant. We take $\varepsilon = 10^{-5}$ in our experimental setting. It can be seen from Figure 14 that the four

cells dominated by $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \mathbf{x}^{(4)}$, after being merged, are able to recover the cell given by the original site $\hat{\mathbf{x}}$.

We skip the details for brevity.

6. Experimental Results

We implemented our algorithm in C++ and evaluated it on a 64-bit workstation with a 3.07 GHz Intel(R) Core(TM) i5 CPU and 16 GB memory.

6.1. Performance

We test our algorithm on several typical protein structures from the protein data bank (https://www.rcsb.org/), including "1al1" (92 atoms), "1crn" (327 atoms), "1bh8" (1074 atoms), "1isn" (2617 atoms) and "2hhb" (4384 atoms). The total time consists of consists of three parts: (a) locating vertices, (b) tracing edges, and (c) generating the well-tessellated triangle mesh of each curved facet. The timing statistics are available in Figure 15. It can be seen that the total timing cost of our algorithm basically increases linearly w.r.t. the number of sites.

Besides, after one turn of subdivision, some cubes are removed because the number of remaining sites is less than 4, some cubes are fixed because the number is exactly 4, while some cubes need further subdivision because the number is larger than 4. After 4 cycles of subdivision, the cubes that contain 4 or more sites become quite few.

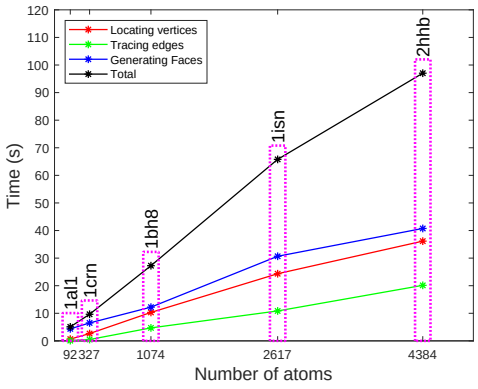
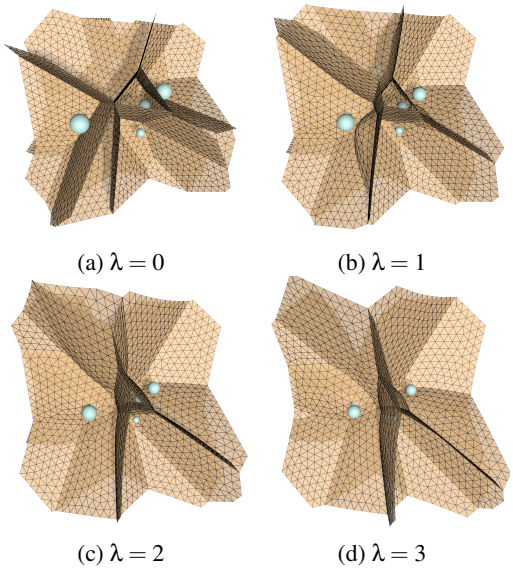


Figure 15: *The running time cost statistics of our algorithm on several typical protein structures. It consists of three parts: (a) locating vertices, (b) tracing edges, and (c) generating the well-tessellated triangle mesh of each curved facet.*

6.2. Scalability

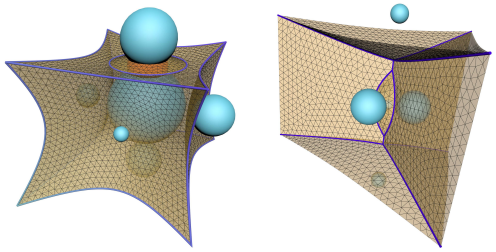
An Apollonius diagram, unlike a Voronoi diagram, may contain complicated structures. Suppose that we multiply $\{w_i\}_{i=1}^n$ by λ. When $\lambda = 0$, it degenerates into a Voronoi diagram. When $\lambda > 1$, the entire structure is simplified - some small cells are hidden by other sites with a larger weight. When λ is infinitely large, the whole space is dominated by the largest-weight site. We use Figure 16 to observe the interesting change.



(a) $\lambda = 0$ (b) $\lambda = 1$ (c) $\lambda = 2$ (d) $\lambda = 3$

Figure 16: *We multiply the weight by a scalar to observe the change of the Apollonius diagram. When $\lambda = 0$, it degenerates into a Voronoi diagram. When $\lambda > 1$, the entire structure is simplified - some small cells are hidden by other sites with a larger weight.*

6.3. Meshing quality



(a) (b)

	$Q_{\alpha_{min}}$	Q_{Ll}	Q_{ALS}	Q_{Rr}	Q_{lr}
(a)	0.890776	0.875221	0.946824	0.973214	0.955525
(b)	0.903214	0.934154	0.913413	0.905613	0.891646

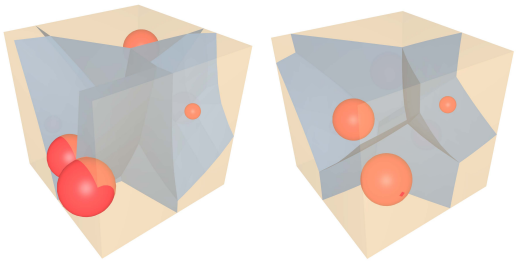
Table 1: *We use five indicators to measure the meshing quality.*

We use five indicators to measure the meshing quality [RPH03], which are

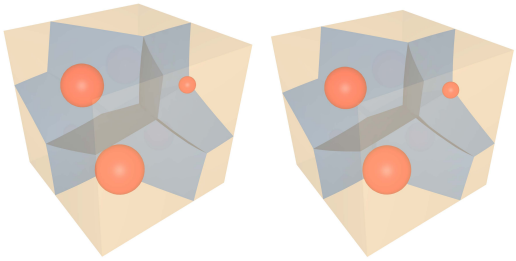
- $Q_{\alpha_{min}} = \frac{3\alpha_{min}}{\pi}$, where α_{min} is the smallest interior angle of the triangle.
- $Q_{Ll} = \frac{l_{min}}{l_{max}}$, where l_{min} and l_{max} are respectively the minimum and maximum edge lengths.
- $Q_{ALS} = \frac{4\sqrt{3}S}{l_1^2+l_2^2+l_3^2}$, where S is the area of the triangle and l_1, l_2, l_3 are the edge lengths.
- $Q_{Rr} = \frac{2r}{R}$, where r is the inscribed circle radius and R is the circumscribed circle radius.
- $Q_{lr} = \frac{2\sqrt{3}r}{l_{min}}$, where r is the inscribed circle radius and l_{min} is the minimum edge length.

The common feature of the five indicators is that when the value amounts to 1, the triangle quality reaches the best (a regular triangle). From the statistics in Table 1, we can see that the tool of centroidal Voronoi tessellation is able to yield a high-quality meshing representation at the end. In implementation, after the Apollonius edges are traced, we perform sampling along the edges at a step of 2% of the bounding box size. We denote the step by t. The number of points (mesh vertices) inside an Apollonius face is thus set to the total area divided by $\frac{\sqrt{3}}{4}t^2$.

7. Centroidal Apollonius Diagrams



(a) Initial (b) Iteration 1



(a) Iteration 5 (b) Iteration 20

Figure 17: *Centroidal Apollonius diagram.*

Given a set of movable sites $X = \{\mathbf{x}_i\}_{i=1}^n \in \Omega$, the centroidal Voronoi tessellation (CVT) is to find a special Voronoi diagram where each site $\mathbf{x}_i$ coincides with the centroid of $\mathbf{x}_i$'s cell [DFG99]. Mathematically, CVT can be computed by minimizing the following objective function:

$$E_1(X) = \sum_{i=1}^{n} \int_{\Omega_i} \|\mathbf{x} - \mathbf{x}_i\|^2 \rho(\mathbf{x}) d\sigma, \tag{39}$$

where $\{\Omega_i\}_{i=1}^n$ defines the Voronoi diagram w.r.t. $X = \{\mathbf{x}_i\}_{i=1}^n$.

If the sites are fixed but enforced by a set of mass constraints such that the site $\mathbf{x}_i$ has a mass M_i, then the partitioning of the domain Ω with the minimum transport cost in the sense of 2-Wassertein metric must come from some power diagram [AHA98]. Therefore, one can define a set of weights $W = \{w_i\}_{i=1}^n$ and then find the optimal partitioning by maximizing

$$E_2(W) = \sum_{i=1}^{n} \int_{\mathrm{PW}_i} \|\mathbf{x} - \mathbf{x}_i\|^2 \mathrm{d}\sigma - w_i(m_i - M_i), \tag{40}$$

where $\{\mathrm{PW}_i\}_{i=1}^n$ defines the power diagram w.r.t. $W = \{w_i\}_{i=1}^n$ and m_i is the real mass of i-th cell.

The centroidal power diagram is to find the stationary point of Eq. (40) w.r.t. X and W at the same time. By defining $E_3(X,W) = E_2(W)$, we need to find a pair (X^*, W^*) such that $\nabla_X E_3|_{X^*} = 0$ and $\nabla_W E_3|_{W^*} = 0$.

In this paper, we extend the above objective function by substituting the 1-Wassertein metric for the 2-Wassertein metric, i.e.,

$$E_4(X,W) = \sum_{i=1}^{n} \int_{\mathrm{AW}_i} \|\mathbf{x} - \mathbf{x}_i\| \mathrm{d}\sigma - w_i(m_i - M_i), \tag{41}$$

where AW_i is the Apollonius cell belonging to $\mathbf{x}_i$, $m_i = \int_{\mathrm{AW}_i} \mathrm{d}\sigma$ is the real mass of AW_i, and M_i is the target mass for $\mathbf{x}_i$ specified by users. In order to find the stationary point to Eq. (41), we can adopt an alternatively iterative scheme by inheriting the algorithmic scheme of [DGBOD12]. Due to the difference of the objective function, we re-define the three key actions:

1: Initial $\{M_i\}_{i=1}^n$ (specified by users);
2: **while** the gradient norm is greater than the specified tolerance **do**
3: Compute the Apollonius diagram; //Action 1
4: Compute the pseudo-centroids: $\mathbf{x}_i^* = \arg\min_{\mathbf{x}_i} \int_{\mathrm{PW}_i} \|\mathbf{x} - \mathbf{x}_i\| \mathrm{d}\sigma$;
5: Move each site $\mathbf{x}_i$ to $\mathbf{x}_i^*$; //Action 2
6: Compute $m_i = \int_{\mathrm{PW}_i} \mathrm{d}\sigma$;
7: Compute the gradient $\nabla_{w_i} E_4 = M_i - m_i$;
8: Update $W = \{w_i\}_{i=1}^n$ by the gradient descent method; //Action 3
9: **end while**

Figure 17 shows an example of 3D centroidal Apollonius diagram with 6 sites whose specified volumes are respectively $\frac{M}{12}$, $\frac{M}{12}$, $\frac{2M}{12}$, $\frac{2M}{12}$, $\frac{3M}{12}$, and $\frac{3M}{12}$, where M is the total volume. It runs in 20 iterations and takes our algorithm 3 seconds to finish the algorithm by setting the accuracy tolerance to be 10^{-6}.

8. Conclusions and Future Works

Computing 3D Apollonius diagrams is a challenging problem in computational geometry. To our knowledge, there is no practical solver for this problem. In this paper, we systematically analyze the structures of cells, faces, edges, vertices of 3D Apollonius diagrams, and develop a three-phase algorithm consisting of vertex location, edge tracing and face extraction. The key step of our algorithm is to adaptively subdivide the initial large box into a set of sufficiently small boxes such that each box contains at most one Apollonius vertex. We use the tool of 2D centroidal Voronoi tessellation (CVT) to yield well-tessellated triangle meshes of the curved bisectors. We evaluate our algorithm through numerous experiments and demonstrate the efficacy on centroidal Apollonius diagram. In the near future, we will investigate incremental computation and parallelization, as well as seeking more applications of 3D Apollonius diagrams.

9. Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by National Natural Science Foundation of China (61772016, 62002190,

52075526, 91860204), Singapore MOE AcRF (RG20/20), Hong Kong RGC GRF grant (17263316), National Young Talents Program of China (XDA21010205), and the Dalian University of Technology Fundamental Research Fund (DUT19RC(3)037).

References

[ABMY02] ANTON F., BOISSONNAT J.-D., MIOC D., YVINEC M.: An exact predicate for the optimal construction of the additively weighted Voronoi diagram. In Europ. Workshop Comput. Geom (2002). 2

[AHA98] AURENHAMMER F., HOFFMANN F., ARONOV B.: Minkowski-type theorems and least-squares clustering. Algorithmica 20, 1 (1998), 61–76. 2, 11

[Aur87] AURENHAMMER F.: Power diagrams: properties, algorithms and applications. SIAM Journal on Computing 16, 1 (1987), 78–96. 1, 2

[Aur91] AURENHAMMER F.: Voronoi diagrams—a survey of a fundamental geometric data structure. ACM Computing Surveys (CSUR) 23, 3 (1991), 345–405. 2

[BBR*08] BERNAUER J., BAHADUR R. P., RODIER F., JANIN J., POUPON A.: DiMoVo: a Voronoi tessellation-based method for discriminating crystallographic and biological protein–protein interactions. Bioinformatics 24, 5 (2008), 652–658. 1

[BH08] BALZER M., HECK D.: Capacity-constrained Voronoi diagrams in finite spaces. In Voronoi Diagrams in Science and Engineering (2008). 3

[BK02] BOISSONNAT J.-D., KARAVELAS M. I.: On the combinatorial complexity of Euclidean Voronoi cells and convex hulls of d-dimensional spheres. Tech. Rep. RR-4504, INRIA, July 2002. URL: https://hal.inria.fr/inria-00072084. 2

[CGD90] COLE R., GOODRICH M. T., DÚNLAING C. Ó.: Merging free trees in parallel for efficient Voronoi diagram construction. In International Colloquium on Automata, Languages, and Programming (1990), Springer, pp. 432–445. 2

[DEJ06] DU Q., EMELIANENKO M., JU L.: Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations. SIAM journal on numerical analysis 44, 1 (2006), 102–119. 3

[Dev98] DEVILLERS O.: Improved incremental randomized Delaunay triangulation. In Proceedings of the fourteenth annual symposium on Computational geometry (1998), pp. 106–115. 2

[DFG99] DU Q., FABER V., GUNZBURGER M.: Centroidal Voronoi tessellations: Applications and algorithms. SIAM review 41, 4 (1999), 637–676. 3, 11

[DGBOD12] DE GOES F., BREEDEN K., OSTROMOUKHOV V., DESBRUN M.: Blue noise through optimal transport. ACM Transactions on Graphics (TOG) 31, 6 (2012), 171. 3, 11

[DXX*19] DOU Z., XIN S., XU R., XU J., ZHOU Y., CHEN S., WANG W., ZHAO X., TU C.: Top-down shape abstraction based on greedy pole selection. In Computational Visual Media Conference (2019). 2

[EK06] EMIRIS I. Z., KARAVELAS M. I.: The predicates of the Apollonius diagram: algorithmic analysis and implementation. Computational Geometry 33, 1-2 (2006), 18–57. 1, 2

[For87] FORTUNE S.: A sweepline algorithm for Voronoi diagrams. Algorithmica 2, 1-4 (1987), 153. 2

[For95] FORTUNE S.: Voronoi diagrams and Delaunay triangulations. In Computing in Euclidean geometry. World Scientific, 1995, pp. 225–265. 2

[FWZL04] FAN Z., WU Y., ZHAO X., LU Y.: Simulation of polycrystalline structure with Voronoi diagram in Laguerre geometry based on random closed packing of spheres. Computational materials science 29, 3 (2004), 301–308. 1

[Gli05] GLICKENSTEIN D.: Geometric triangulations and discrete laplacians on manifolds. arXiv preprint math/0508188 (2005). 2

[GMAM06] GARRIDO S., MORENO L., ABDERRAHIM M., MARTIN F.: Path planning for mobile robot navigation using Voronoi diagram and fast marching. In 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (2006), IEEE, pp. 2376–2381. 1

[GR96] GAVRILOVA M., ROKNE J.: An efficient algorithm for construction of the power diagram from the Voronoi diagram in the plane. International journal of computer mathematics 61, 1-2 (1996), 49–61. 2

[GR03] GAVRILOVA M. L., ROKNE J. G.: Updating the topology of the dynamic voronoi diagram for spheres in euclidean d-dimensional space. Comput. Aided Geom. Des. 20 (2003), 231–242. 3

[GS78] GREEN P. J., SIBSON R.: Computing Dirichlet tessellations in the plane. The computer journal 21, 2 (1978), 168–173. 2

[HIKL*99] 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 conference on Computer graphics and interactive techniques (1999), ACM Press/Addison-Wesley Publishing Co., pp. 277–286. 2

[HLK*17] HU Z., LI X., KRISHNAMURTHY A., HANNIEL I., MCMAINS S.: Voronoi cells of non-general position spheres using the gpu. Computer-Aided Design and Applications 14, 5 (2017), 572–581. 2

[IIM85] IMAI H., IRI M., MUROTA K.: Voronoi diagram in the Laguerre geometry and its applications. SIAM Journal on Computing 14, 1 (1985), 93–105. 2

[KCK05a] KIM D., CHO Y., KIM D.-S.: Region expansion by flipping edges for euclidean voronoi diagrams of 3d spheres based on a radial data structure. In Computational Science and Its Applications – ICCSA 2005 (Berlin, Heidelberg, 2005), Gervasi O., Gavrilova M. L., Kumar V., Laganà A., Lee H. P., Mun Y., Taniar D., Tan C. J. K., (Eds.), Springer Berlin Heidelberg, pp. 716–725. 2, 3

[KCK05b] KIM D.-S., CHO Y., KIM D.: Euclidean Voronoi diagram of 3D balls and its computation via tracing edges. Computer-Aided Design 37, 13 (2005), 1412–1424. 3

[KCK*05c] KIM D.-S., CHO Y., KIM D., KIM S., BHAK J., LEE S.: Euclidean voronoi diagrams of 3d spheres and applications to protein structure analysis. Japan Journal of Industrial and Applied Mathematics 22 (2005), 251–265. 2

[KCKC04] KIM D.-S., CHO Y., KIM D., CHO C.-H.: Protein structure analysis using euclidean voronoi diagram of atoms. Proc. International Workshop on Biometric Technologies (BT 2004) (01 2004), 125–129. 2

[KK06] KIM D., KIM D.-S.: Region-expansion for the voronoi diagram of 3d spheres. Computer-Aided Design 38, 5 (2006), 417 – 430. 2, 3, 9

[KKC05] KIM D.-S., KIM D., CHO Y.: Euclidean voronoi diagrams of 3d spheres: Their construction and related problems from biochemistry. In Mathematics of Surfaces XI (Berlin, Heidelberg, 2005), Springer Berlin Heidelberg, pp. 255–271. 2

[KKS02] KIM D.-S., KIM D., SUGIHARA K.: Unifying method for computing the circumcircles of three circles. International journal of CAD/CAM 2, 1 (2002), 45–54. 2

[KY02] KARAVELAS M. I., YVINEC M.: Dynamic additively weighted voronoi diagrams in 2d. In European Symposium on Algorithms (2002). 2

[LCT10] LIU Y.-J., CHEN Z., TANG K.: Construction of iso-contours, bisectors, and voronoi diagrams on triangulated surfaces. IEEE Transactions on Pattern Analysis and Machine Intelligence 33, 8 (2010), 1502–1517. 1

[Lee80] LEE D. T.: Two-dimensional voronoi diagrams in the lp-metric. J. ACM 27, 4 (Oct. 1980), 604–618. 1

[LKHM19] LI X., KRISHNAMURTHY A., HANNIEL I., MCMAINS S.: Edge topology construction of voronoi diagrams of spheres in non-general position. Computers and Graphics 82 (2019), 332 – 342. 2

[LSZ*14] LU L., SHARF A., ZHAO H., WEI Y., FAN Q., CHEN X., SAVOYE Y., TU C., COHEN-OR D., CHEN B.: Build-to-last: strength to weight 3D printed objects. ACM Transactions on Graphics (TOG) 33, 4 (2014), 97. 3

[LWL*09] LIU Y., WANG W., LÉVY B., SUN F., YAN D.-M., LU L., YANG C.: On centroidal Voronoi tessellation—energy smoothness and fast computation. ACM Transactions on Graphics (ToG) 28, 4 (2009), 101. 1, 3

[Ma00] MA L.: Bisectors and Voronoi diagrams for convex distance functions. PhD thesis, Fernuniv., Fachbereich Informatik, 2000. 1

[RLW*10] RONG G., LIU Y., WANG W., YIN X., GU D., GUO X.: GPU-assisted computation of centroidal Voronoi tessellation. IEEE transactions on visualization and computer graphics 17, 3 (2010), 345–356. 2

[RPH03] RAMOS J., PALAU J., HUERTA A.: Numerical representation of the quality measures of triangles and triangular meshes. Communications in Numerical Methods in Engineering 19 (07 2003). 10

[RR94] RAJASEKARAN S., RAMASWAMI S.: Optimal parallel randomized algorithms for the Voronoi diagram of line segments in the plane and related problems. In Proceedings of the tenth annual symposium on Computational geometry (1994), ACM, pp. 57–66. 2

[Sen93] SENECHAL M.: Spatial tessellations: Concepts and applications of Voronoi diagrams. Science 260, 5111 (1993), 1170–1173. 2

[SH75] SHAMOS M. I., HOEY D.: Closest-point problems. In 16th Annual Symposium on Foundations of Computer Science (sfcs 1975) (1975), IEEE, pp. 151–162. 2

[SU00] SACK J., URRUTIA G.: Voronoi diagrams. handbook of computational geometry. Ottawa: Elsevier Science 290 (2000). 2

[VBM02] VOLOSHIN V., BEAUFILS S., MEDVEDEV N.: Void space analysis of the structure of liquids. Journal of molecular liquids 96 (2002), 101–112. 2

[Vor08] VORONO G.: Nouvelles applications des param tres continus la th orie des formes quadratiques I: Sur quelques propri ts des formes quadratiques positives parfaites. J. reine angew. Math 133, 97 (1908), 178. 1

[WFX*18] WANG W., FANG Z., XIN S., HE Y., ZHOU Y., CHEN S.: Tracing high-quality isolines for discrete geodesic distance fields. In Computational Visual Media Conference (2018). 2

[Wil98] WILL H.-M.: Fast and efficient computation of additively weighted Voronoi cells for applications in molecular biology. In Scandinavian Workshop on Algorithm Theory (1998), pp. 310–321. 2

[Wil99] WILL H.-M.: Computation of additively weighted Voronoi cells for applications in molecular biology. PhD thesis, ETH Zurich, 1999. 1, 2

[XLC*16] XIN S.-Q., LÉVY B., CHEN Z., CHU L., YU Y., TU C., WANG W.: Centroidal power diagrams with capacity constraints: Computation, applications, and extension. ACM Transactions on Graphics (TOG) 35, 6 (2016), 244. 3

[XWX*13] XIN S.-Q., WANG X., XIA J., MUELLER-WITTIG W., WANG G.-J., HE Y.: Parallel computing 2D Voronoi diagrams using untransformed sweepcircles. Computer-Aided Design 45, 2 (2013), 483–493. 2

[ZJW15] ZHOU Y., JU L., WANG S.: Multiscale superpixels and supervoxels based on hierarchical edge-weighted centroidal Voronoi tessellation. IEEE Transactions on Image Processing 24, 11 (2015), 3834–3845. 3