University of West Bohemia in Pilsen
Department of Computer Science and Engineering
Univerzitni 8
30614 Pilsen
Czech Republic

Computational Geometry Applied to Modeling and Visualization of Proteins

State of the Art and Concept of Ph.D. Thesis

Martin Maňák

Technical Report No. DCSE/TR-2010-5
May, 2010

Distribution: public

Technical Report No. DCSE/TR-2010-5
May 2010

Computational Geometry Applied to Modeling and Visualization of Proteins

Martin Maňák

Abstract

The modeling of protein structures is a challenging task that closely relates to the field of computational geometry. A protein molecule is often modeled as a set of balls which represents its atoms. Since it is strongly agreed that the function of a protein is mostly determined by its shape, describing spatial relations among these balls is of a great importance for solving related problems. Many kinds of Voronoi diagrams and their duals have been used here to describe the spatial properties. The Voronoi diagram of balls, its dual and related concepts proved to be the best available choice in this area. This work gives a historical background to this area from the computational geometry point of view, provides an overview of the best available concepts in this area, i.e., the Voronoi diagram of balls, its dual structure and derived shape concepts, and shows some of their applications, such as the computation of molecular surfaces or pocket extraction. A part of this work is dedicated to an interesting extension of this kind of diagrams by allowing a ball to be inverted. This extension can be used as a convenient boundary constraint of the whole diagram or its parts. A new approach of fast construction of these diagrams is also discussed. This approach uses a three-dimensional Delaunay triangulation of atom centers and spatial filters in order to discover relevant parts of the diagram rapidly. Finally, future research directions are outlined.

This work was supported by The Grant Agency of the Czech Republic, project No. P202/10/1435.

Copies of this report are available on
http://www.kiv.zcu.cz/publications/
or by surface mail on request sent to the following address:

University of West Bohemia in Pilsen
Department of Computer Science and Engineering
Univerzitni 8
30614 Pilsen
Czech Republic

Contents

Chapter 1

Introduction

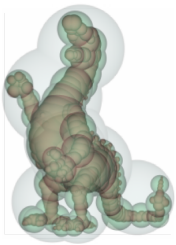
A set of spherical balls in the Euclidean metric space has been frequently used as a geometric model in various fields of computational geometry or computer graphics. For example, an object can be approximated by a set of balls, such as the bunny in Figure 1.1(a), for the purpose of collision detection or calculating Minkowski sums [53, 1]. Another example is the concept of a sphere tree, which is a hierarchy of spheres built over an object to achieve several level of details [9] as it is shown in Figure 1.1(b). Another important context is the field of biochemistry, because one of the models of a protein molecule is the model where atoms are represented as balls. Their radii are based on forces and hence the balls usually intersect. An example is shown in Figure 1.1(c). The shape of a molecule is one of the factors determining the function of a protein and hence describing the proximity relations among atoms is particularly important in this context.

From the geometric point of view, atoms are balls and the proximity relations are based on distances from the balls, e.g., on the power distance, Euclidean distance from their centers or Euclidean distance from their surface. Various kinds of 3D Voronoi diagrams (tessellation of space) or their duals (3D triangulations) are often used to represent the proximity relations.

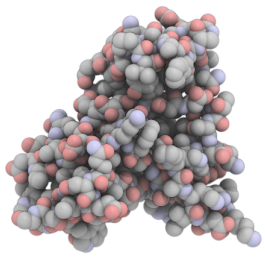
Sometimes, using an ordinary Delaunay triangulation built over the centers of balls is sufficient [43]. On the other hand, often it is necessary to use their radii as weights, e.g., to use the regular triangulation. But this triangulation does not model the proximity of non-intersecting balls correctly [27].

The best available choice is the quasi-triangulation [34], solving many geometric problems in molecules effectively [27, 33]. It is the dual for the Voronoi diagram of 3D balls. Such a diagram is useful but difficult to compute. It is usually constructed by the edge-tracing [30] or similar algorithms, such as the face-tracing [34] or the algorithm for Voronoi S-network [44]. They are based on the same principle of discovering unknown Voronoi vertices from some already known (computed) vertices. There are also some other algorithms which are more complex [25, 15, 5].

(a) Approximation by a set of balls. Picture taken from [53].



(b) Sphere tree. Pictures taken from [9] and merged together.



(c) Protein model 2RE7. Rendered by QuteMol [50].

Figure 1.1: Different contexts where a set of spherical balls is used as a geometric model.

A quasi-triangulation alone is still not strong enough when a more advanced spatial analysis of a protein molecule is required, e.g., when a molecular surface is involved in the analysis. Beta-shapes and beta-complexes [36, 32] computed from the quasi-triangulation provide the right description of spatial relations for this purpose.

This work is organized as follows. Several space tessellations that have been used for molecular systems are discussed in Chapter 2. The chapter also provides a brief historical background. Chapter 3 provides the definition of 3D Voronoi diagrams of spherical balls and its properties, including a surprising geometrical property regarding the projection of a Voronoi cell onto its defining sphere and the fascinating connection of these diagrams with power diagrams. The dual structure, i.e., the quasi-triangulation, is briefly described in Chapter 4. The ideas of some of the algorithms for the construction of these diagrams are described in Chapter 5. The fundamental edge-tracing is described more formally in Section 5.3. The theory of beta-shapes and beta-complexes is mentioned in Chapter 6. Some of the possible applications of these concepts are reviewed in Chapter 7. Our contribution is described in Chapter 8. Chapter 9 describes a perspective for the future research and concludes this work.

Chapter 2

Proximity Relations via Space Tessellation

This chapter provides a brief insight to the historical development of space tessellation used for molecules. Section 2.1 explains some basic concepts and Section 2.2 gives the reasons why these concepts have been used for molecules.

2.1 Preliminaries

Consider we are given a finite nonempty set of sites, e.g., points or weighted points in a space, and also that a function that allows us to measure the distance between a point and a site. Then each point of the space can be assigned to some of the sites such that the distance is minimal. This assignment of all points of the space creates a tessellation of the space, where each site gets a subset of the space - a cell. If the cell is nonempty, all points in its interior are closer to the defining site than to any other site. Points on the boundary of a cell are equidistant to more than one site.

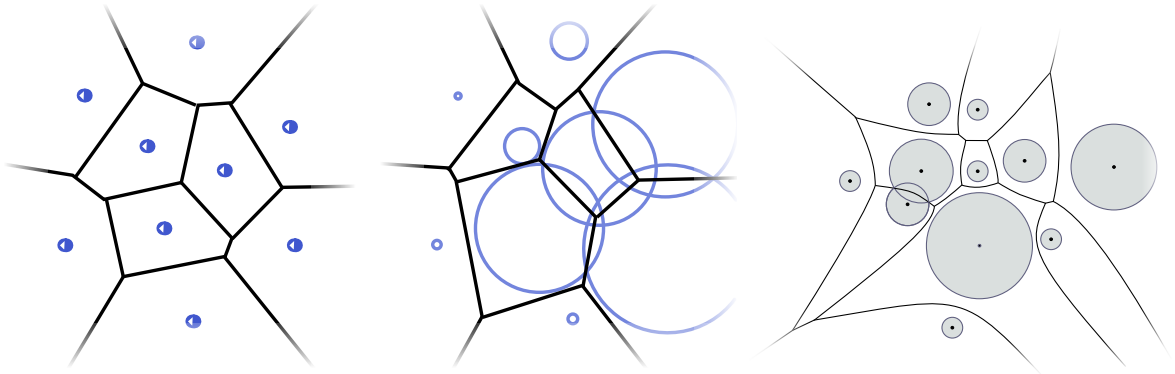
The set of points equidistant to a pair of sites is often called a bisector or a *separator*, because it creates a boundary between the space assigned to one site and the space assigned to the other site. Note that the boundary of a cell is determined by separators.

When the sites are points and the distance is the Euclidean distance, this tessellation is called Voronoi diagram after the Russian mathematician who defined this structure in 1907 [52]. Voronoi used this tessellation in his study of positive definite quadratic forms and considered it for a general d-dimensional case. The history of this concept, its properties and applications can be found in the work of Okabe, Boots, Sugihara and Chiu [46].

When the sites are weighted points and the function is the power distance, this tessellation is called the power diagram or radical plane tessellation. The weight of a point affects the size of the cell as the separator is shifted toward the site with greater weight.

Another function for measuring the distance with respect to weighted points is the additively weighted distance. When the weight of a point is considered as the radius of a ball centered at the weighted point, this function describes how far a point is from the boundary of the ball. The separator is not necessarily planar in this case and the corresponding tessellation is called the additively weighted Voronoi diagram.

Figure 2.1 shows two-dimensional examples of the aforementioned space tessellations. Figure 2.1(a) is the ordinary Voronoi diagram of points, Figure 2.1(b) is the power diagram of weighted points. Circles correspond to weights - the radius of a circle is the square root of the weight. Figure 2.1(c) is the additively weighted Voronoi diagram, where the radii of the circles represent the weights.



(a) Voronoi diagram of points (b) Power diagram of weighted points (c) Additively weighted Voronoi diagram

Figure 2.1: 2D examples of different kinds of space tessellations.

2.2 Tessellations in the Context of Molecules

It was John Bernal in 1959, who first suggested to study the properties of liquids in terms of the packing of irregular polyhedra [3]. In 1967, Bernal and Finney used a standard *Voronoi procedure* for the analysis of the density in a system of random close-packed spheres [4]. This packing represented the model of a simple liquid. In the Voronoi procedure, each sphere is represented by its center and the procedure tessellates the entire space into cells. Each cell is the locus of points closer to the corresponding center than to any other center. An example of a cell is shown in Figure 2.2. Each cell represents the space allocated to the corresponding sphere. Cells are bounded by separating planes. The density of a sphere is the ratio of the sphere volume inside the cell and the cell volume. The separator used by the Voronoi procedure is shown in Figure 2.4(a) as the locus of points equidistant to both centers.

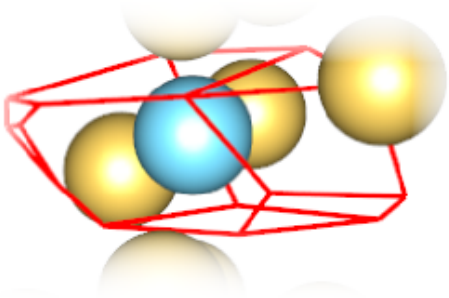


Figure 2.2: 3D example of a Voronoi cell with some of its neighbors.

In 1974, Richards first used this method with proteins to the computation of their atomic volume and to determine their packing density [47]. The standard Voronoi procedure ignores radii of atoms but proteins usually consist of several types of atoms with different radii. Using the Voronoi procedure has a drawback that it may allocate more volume to small atoms, less volume to greater atoms and some volume may be even lost in the total sum. Richards was aware of this and incorporated the radii as weights into the procedure. This method is called Richards' *method B*. The idea is to shift the planes that will bound a cell in order to balance the volume allocation with respect to the size of the atoms.

An unpleasant side-effect of this method is that some space is left unassigned. This neglected volume is called a *vertex error* and shown in Figure 2.4(b) as the tiny yellow triangle. Although Richards has shown the vertex error is small when compared to the atomic volume, this method has been criticized by the later authors.

The *radical plane method* introduced by Gellatly and Finley in 1982 [18], which produces a tessellation known as the *power diagram*, also incorporates atomic radii as weights. Contrary to the Richards' B, it does not suffer from the vertex error anymore. Furthermore, it has the advantage that the separation plane of any two intersecting atoms passes through their intersection circle. This is a useful property, because the atomic volume of the entire protein is then analytically equal to the sum of the atomic volumes of all atoms, which was not guaranteed by any of the previous methods. Figure 2.4(c) shows an example of this separating plane. This plane is shifted from the middle position between the centers away from the greater sphere, hence more space is allocated to the greater sphere. Furthermore, the plane passes through the common intersection of the spheres (in the 2D example the intersection results in two points). Note that both centers are on the same side with respect to the their common separator, which means that the center of the smaller atom is within the space allocated to the greater atom. This is one of the reasons why this method have been later criticized by Gerstein [21] and Goede [20], stating that the radical plane method is not as chemically reasonable as method B.

In 1995, Gerstein et al. discussed a generalization of method B to non-planar separators in order to get a better space partitioning scheme [21]. The idea was to keep the distance ratio of the two neighboring atoms constant. A two-dimensional analogy of this spherical separator is shown in Figure 2.4(d). It is the circle passing through the intersection points.

One year later, Goede et al. have compared the previous approaches (standard Voronoi, Richards' B, radical plane, Gerstein spheres), showed their disadvantages and proposed a new partitioning scheme using additively weighted Voronoi cells. A single cell is shown in Figure 2.3 and an example of the separator for this scheme is shown in Figure 2.4(e). This method unifies the advantages of the previous approaches. The partitioning is geometrically reasonable, such as the Richards' B, and it avoids the vertex error. The separator between two intersecting atoms meets their circle of intersection, such as in the case of the radical plane method, but non-planar boundaries are used and the atom centers are within their corresponding cells.

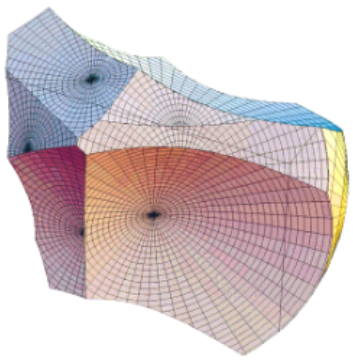
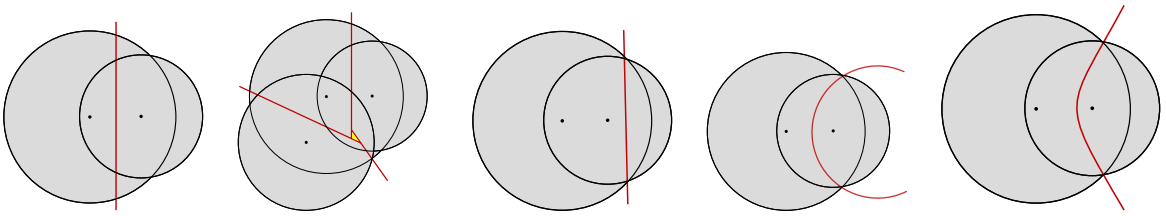


Figure 2.3: 3D example of an Additively weighted Voronoi cell. Picture taken from [20].

It might seem that after the proposal of using additively weighted Voronoi diagrams, the former methods have lost on their importance. This is not true. Even if ordinary Voronoi diagram is not a good choice for density calculation, its dual, i.e., the Delaunay triangulation, can be very useful in solving



(a) Standard Voronoi procedure (b) Richards' method B and the vertex error (c) Radical plane (d) Gerstein spheres (e) Additively weighted

Figure 2.4: Several types of Voronoi cell separators (2D analogies)

other problems, such as computing empty channels in a molecule [43]. Also the radical plane method finds its use - its dual, i.e., the regular triangulation, can be used for computing the total volume of a molecule effectively [13].

Chapter 3

Voronoi Diagram of Balls

Voronoi diagram of 3D balls is defined in this chapter and some of its important properties are mentioned.

3.1 Definition

There are several names of this kind of diagram. It is also known as *additively weighted Voronoi diagram*. In this case, the input is a set of weighted points. Other authors name it *Voronoi diagram of spheres*, *balls* or *atoms* and understand the input set as a set of spheres, balls or atoms, respectively, in order to emphasize the geometrical meaning or the application. In 2D, *Voronoi diagram of circles* or *discs* is used and sometimes it is called *Johnson-Mehl tessellation* since in 1939 Johnson and Mehl introduced it as a model of crystal growth process, or as *Apollonius diagram* in the honor of Apollonius of Perga (262 BC - 190 BC, Greek).

In this work, we will use the nomenclature of balls from [30] and often leave the word "Voronoi" from terms, e.g., we will use "diagram" instead of Voronoi diagram.

Definition 3.1.1. Let $S = \{b_1, \dots b_n\}$ be a set of *balls* $b_i = (c_i, r_i)$ with centers $c_i \in \mathbb{R}^3$ and radii $r_i \in \mathbb{R}$. Then the *signed distance* of a point $x \in \mathbb{R}^3$ to a ball b_i is defined as

$$d(x, b_i) = \|x - c_i\| - r_i. \tag{3.1}$$

The *Voronoi cell* or *Voronoi region* for a ball b_i is the set of points

$$C_i = \{x \in \mathbb{R}^3 \mid \forall b_j \in S, j \neq i : d(x, b_i) \leq d(x, b_j)\}. \tag{3.2}$$

The *Voronoi diagram* for S is the set of Voronoi cells

$$VD(S) = \{C_1, \dots, C_n\}. \tag{3.3}$$

Voronoi cells are bounded by 2-dimensional *Voronoi faces*. The intersection of two faces constitutes 1-dimensional *Voronoi edges* and the intersection of two edges creates 0-dimensional *Voronoi vertices*.

Definition 3.1.2 provides a nomenclature for the elements of a Voronoi diagram. It will be useful in the context of a dual structure, which has different elements of similar names.

Definition 3.1.2. $\mathrm{VD} = (V^V, E^V, F^V, C^V)$ is the Voronoi diagram, where V^V, E^V, F^V and C^V are the Voronoi vertices (V-vertices), Voronoi edges (V-edges), Voronoi faces (V-faces) and Voronoi cells (V-cells), respectively.

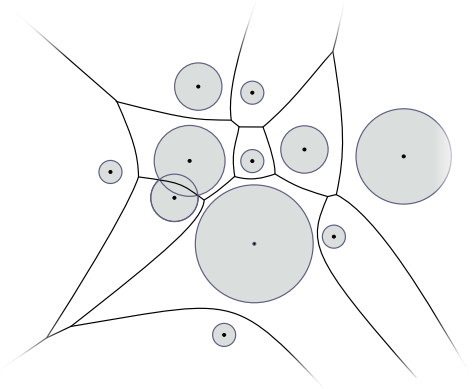


Figure 3.1: 2D analogy to the Voronoi diagram of a set of ball

A simple 2D example of a diagram is shown in Figure 3.1.

We assume all balls are in a general position. This assumption simplifies algorithms for the construction and representation of diagrams. Degeneracy can be removed by small changes of positions of balls. We further assume non-negative radii, since adding a constant value to all radii does not change the diagram (it is apparent from equations 3.1 and 3.2).

Note that the function $d(x, b_i)$ in the equation 3.1 can be negative. For variables $x \in \mathbb{R}^3$ and $i \in \{1, \dots n\}$ it is $d(x, b_i) \in [-r_{max}, \infty)$, where $r_{max} = \max\{r_i \mid i \in \{1 \dots n\}\}$. Negativity is not an issue in our case. Otherwise, another distance function defined as

$$u(x, b_i) = 1 + \max(d(x, b_i), 0) + \frac{\min(d(x, b_i), 0)}{r_{max}} \tag{3.4}$$

can be used instead of 3.1. The function u maps all negative values of the function d from $[-r_{max}, 0]$ to $[0, 1]$ and shifts all positive values of d by one, i.e., behind the interval $[0, 1]$. Equations 3.4 and 3.1 provide the same ordering functionality required in 3.2 but 3.4 is always non-negative.

Note that if all radii are zero, then all input balls degenerate to points, the distance function is the standard Euclidean distance and the resulting diagram becomes the ordinary Voronoi diagram. Therefore, the class of ordinary Voronoi diagrams is a subclass of Voronoi diagrams of balls.

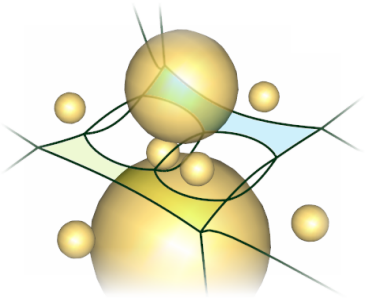
3.2 Geometrical and Topological Properties

Voronoi cells, faces and edges are 3-, 2- and 1-dimensional connected subsets of $\mathbb{R}^3$, respectively.

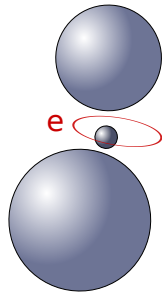
Each cell is star-shaped relative to the center of the corresponding ball (i.e. straight line segments drawn from the center to points in the corresponding cell are fully contained in the cell), but it does not have to be convex (see Figure 2.3). Each face is a part of a hyperboloid of two sheets or a plane, edge is a part of a hyperbola (see Figure 3.2(c)), ellipse (see Figures 3.2(b) and 3.2(e)) or a straight line and each vertex is the center of an *empty sphere*[1] externally tangent to 4 balls (see Figure 3.2(d)).

[1]The interpretation of a vertex as a center of an empty sphere is taken from [30]. This interpretation might be a little bit confusing for a first-time reader considering the four balls intersect each other, because then there would be a negative radius of the empty sphere. For the empty sphere, the authors used the definition from Gavrilova's Ph.D. thesis [16], where she assumes only non-intersecting balls.

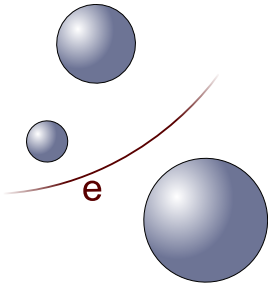
Each face is defined by a pair, edge by a triplet and each vertex by a quadruplet of balls, respectively. On the other hand, a pair of balls can define more than one face (see Figure 3.2(a)), a triplet of balls can define more than one edge (see Figures 3.2(f) and 3.3), and the maximum number of vertices defined by a ball quadruplet is two (see Figures 3.2(e) and 3.2(f)). Furthermore, it is possible for an elliptic edge to be without any vertices (see Figure 3.2(b)).



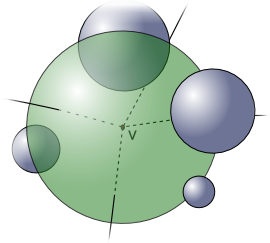
(a) Two faces are defined by the same pair of balls



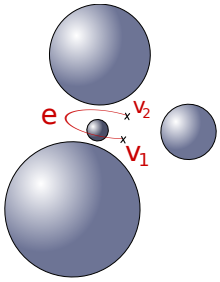
(b) An elliptic edge



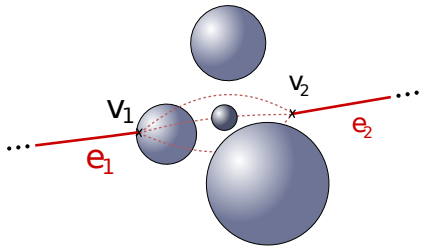
(c) An edge as a part of a hyperbola



(d) A vertex as the center of an empty sphere



(e) Two vertices v_1 and v_2 are defined by the same balls via elliptic edge e



(f) Two vertices v_1 and v_2 are defined by the same balls via non-elliptic edges. Two edges e_1 and e_2 are defined by the same balls.

Figure 3.2: Geometric and topological interpretation of diagram elements

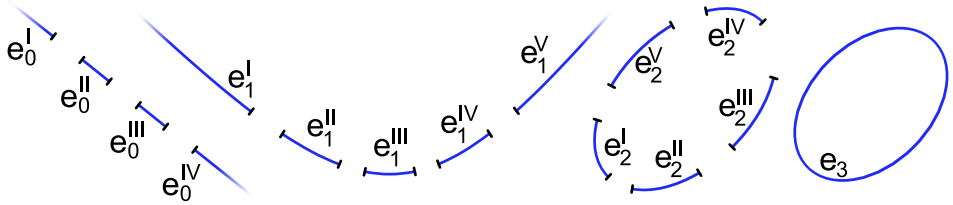


Figure 3.3: The edges $e_0^I, \dots e_0^{IV}$ are linear and defined by the same triplet of balls. The edges $e_1^I, \dots e_1^V$ are hyperbolic and defined by the same triplet of balls. The edges $e_2^I, \dots e_1^{IV}$ are elliptic and defined by the same triplet of balls. The edge e_3 is elliptic without any vertex on it.

3.3 Projection onto a Sphere

Since Voronoi cells are star-shaped, it is possible to bijective project the boundary of a cell onto a unit sphere concentric with the ball defining the cell. The projection is $\pi : p \to p/\|p\|$ for $p \in \mathbb{R}^3$ when

considering the center of the unit sphere located at the origin. An interesting property is that the edges of the cell, which are hyperbolic or elliptic, project as circles or circular arcs onto the unit sphere. Similarly, the faces bounding the cell, which are hyperbolic or planar, project as spherical patches. In other words, a single cell can be equivalently represented as a subdivision of the unit sphere. The subdivision can be realized as the intersection of a polyhedron and a sphere. These properties have been proven and published by Will [55]. An example of a spherical subdivision representing a cell is shown in Figure 3.4.

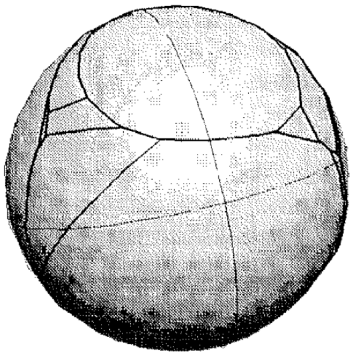


Figure 3.4: Voronoi cell projected onto a unit sphere creates a subdivision. The edges of the cell project as circular arcs. Picture taken from [55].

3.4 Connection to Apollonius

Recall that a vertex in a Voronoi diagram of 3D balls is the center of an empty sphere externally tangent to four defining balls. This generalizes the 2D case, where a vertex is the center of an empty circle externally tangent to three defining circles. This 2D problem dates back to the ancient Greek.

Apollonius of Perga (circa 262 to 190 B.C.) was a famous Greek mathematician known as "The Great Geometer". In his work *Tangencies*, he left the following problem: given any three points, straight lines or circles, construct a circle which contains the points or is tangent to the lines and circles. Apollonius enumerated all ten possible cases of this problem. The last one, finding a circle tangent to three other circles, is considered to be the most complicated and is known as the *Apollonius* 10^{th} *problem* and it has up to eight solutions. It should be noted that the tools allowed for the construction of the circle were only the compass and a straightedge.

Since no copy of Tangencies has survived, we do not know the original solutions of these problems. The formulation of these problems survived thanks to the work of Pappus of Alexandria. The possible original reconstruction (involving only a compass and straightedge) come from Francois Viete. Many other solution methods are known, from purely geometrical to analytical.

Further historical details can be found, e.g., in [19].

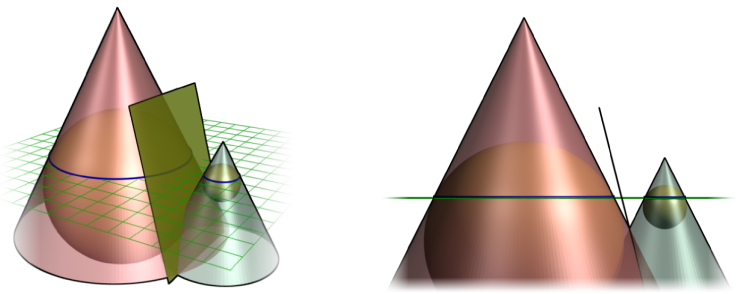
Our 3D case of finding a sphere externally tangent to four other spheres is a special case of the generalization of the Apollonius 10^{th} problem to higher dimensions. Gavrilova showed an analytical solution for the general d-dimensional case and the description of this solution can be found, e.g., in [17]. The balls are first reduced by the radius of the minimal ball and translated such that the center of the minimal ball becomes the origin. Now we are left with finding a sphere externally tangent

to d balls and a point (the origin). This is then solved as a system of d linear equations in $d + 1$ variables and a quadratic equation in terms of the free variable. We use this approach in our edge tracing implementation.

3.5 Connection to Power Diagrams

There is an interesting connection between Voronoi diagram of spheres in $\mathbb{R}^d$ and power diagrams in $\mathbb{R}^{d+1}$ shown by Aurenhammer in 1987 [2]. Basically, each Voronoi cell in $\mathbb{R}^d$ can be obtained by intersecting a $(d + 1)$-dimensional cone with the corresponding $(d + 1)$-dimensional (polyhedral) power cell and projecting the result orthogonally back to $\mathbb{R}^d$. This section briefly reviews the idea. Figure 3.5 shows the geometric interpretation for $d = 2$.

Let us have a set T of d-dimensional spheres and denote $h_0 : x_{d+1} = 0$ in $\mathbb{R}^{d+1}$. Put T into h_0. In Figure 3.5, h_0 is the plane shown as a grid and T are the two discs in the plane. Then each $s \in T$ can be covered by a cone $\kappa(s)$, which has its axis aligned to the $(d + 1)$-coordinate axis and apex on the positive[2] side. All $\kappa(s)$ have the same apex angle and $\kappa(s) \cap h_0 = s$. Considering another sphere $t \in T$, then the set of points in h_0 equidistant to s and t is the hyperboloid $hyp(s, t) = \{x \in h_0 \mid d(x, s) = d(x, t)\}$. It is a d-dimensional hyperboloid embedded in h_0. The important property is that $hyp(s, t)$ is the orthogonal projection of $\kappa(s) \cap \kappa(t)$ onto h_0, which is the same as the vertical projection of $\kappa(s) \cap R(\sigma(s), \sigma(t))$, where $R(.,.)$ is the radical plane in $\mathbb{R}^{d+1}$ between two spheres and $\sigma(s)$ is the $(d + 1)$-dimensional sphere inscribed to $\kappa(s)$ such that $\sigma(s) \cap \kappa(s) = s$. Analogically, $\sigma(t)$ is inscribed to $\kappa(t)$ such that $\sigma(t) \cap \kappa(t) = t$. The boundary of a power cell for $\sigma(s)$ is constituted by radical planes $R(s, .)$ and therefore intersecting the power cell with $\kappa(s)$ and projecting the result orthogonally to h_0 yields the Voronoi cell of s.



(a) Perspective (b) Side view

Figure 3.5: 2D analogy of the connection to power diagrams. Two circles are embedded into 3D space and covered by cones. Spheres tangent to the cones are lifted from the circles and their radical plane is created. The orthogonal projection of the intersection between the radical plane and the first cone back to the 2D space constitutes the boundary of the Voronoi cell corresponding to the first circle.

[2]The choice of the side can be arbitrary but consistent for all $\kappa(s)$.

3.6 Complexity

The relationship with power diagrams from section 3.5 leads to the worst case complexity of the Voronoi diagram of spheres. Aurenhammer [2] proved that the upper bound to the worst case complexity of a Voronoi diagram of n spheres in $\mathbb{R}^d$ is $O(n^{\lfloor \frac{d}{2} \rfloor + 1})$. This is tight in odd dimensions. At first he showed an algorithm for the construction of a power diagram in any dimension and analyzed its worst case time and space complexities in terms of the respective complexities of algorithms for the construction of a convex hull of a set of points. Then he showed the complexity for the Voronoi diagrams of spheres using the above mentioned relationship.

Aurenhammer [2] has also shown that the complexity of a single cell is $O(n^{\lceil \frac{d}{2} \rceil})$. This has been known worst case optimal only for even dimension. Will [55] was the first one who published an example where an arrangement of spheres in $\mathbb{R}^3$ creates a cell with the complexity $\Omega(n^2)$. Hence these cells can be significantly different from the cells of an ordinary 3D Voronoi diagram of points, where the worst case complexity of a cell is $\Theta(n)$. But it was Boissonnat and Karavelas [6] in 2003 who showed that the worst case combinatorial complexity of a single cell is $\Theta(n^{\lceil \frac{d}{2} \rceil})$ for any dimension d. Their approach uses the equivalence relationship between a single Voronoi cell in $\mathbb{R}^d$ and a *Möbius diagram* in $\mathbb{R}^{d-1}$. This kind of diagrams is the generalization of both power diagrams and the multiplicatively weighted diagrams. These authors have also shown an equivalence between a single Voronoi cell and the convex hull of a set of spheres, which yields the same worst case complexity of the convex hull, i.e. $\Theta(n^{\lceil \frac{d}{2} \rceil})$. They have also pointed out to the open problems, namely the worst case complexity of the whole diagram in even dimensions $d > 2$ and the complexity of a single cell, the whole diagram or the convex hull when the spheres have constant number of different radii.

Chapter 4

Quasi Triangulation

The duality between d-dimensional Voronoi diagrams and triangulations is an important and powerful concept in computational geometry. A duality maps between diagram elements and simplices. The dual for a Voronoi diagram of points is a Delaunay triangulation, the dual for a power diagram is a regular triangulation. Both these triangulations satisfy the Definition 4.1.2 of a simplicial complex. Working with simplices is often more convenient than with diagram elements and the representation of a triangulation in a data structure can be very efficient. But what about the dual of a Voronoi diagram of spheres - is there any triangulation?

With an exception to the work of Kim et al. [34], the duality for Voronoi diagrams of spheres has been only marginally mentioned in the literature, e.g., Gavrilova mentions that the dual can contain duplicate or intersecting faces as a consequence of the possible non-convexity of diagram cells and shows an example in her thesis [16, Section 3.2.3]. Medvedev et al. in [44] have also dealt with the duality, calling the elements *Delaunay S-simplices* forming an *S-covering* of space (i.e. not necessarily a tessellation). Okabe et al. [46, Section 3.1.2] only state that it is possible to define the dual and refer the reader to Mirzaian's work [45]. Andy Mirzaian has considered only the 2D case and defined the dual as a multigraph, where nodes are the centers of discs and edges are either straight line segments or two connected straight line segments. An edge in the graph represents the common face of two neighboring Voronoi cells. Because there can be more faces between two cells, there can be multi-edges in the graph. He calls the planar drawing as *quasi-straight-line triangulation*. In 2006, Kim et al. [34] explored this duality into depth. They named the dual structure as a *quasi triangulation*, emphasizing that the dual does not have to be a simplicial complex nor a valid triangulation (caused by so called *anomalies*). The authors further suggested how to represent the dual in a data structure and they also published traversing algorithms [31]. Definition 4.2.1 of a quasi triangulation comes from [34, 31].

The definitions of simplices, complexes and triangulations are in Section 4.1. The quasi-triangulation is defined from a Voronoi diagram of balls via a duality operator in Section 4.2. Peculiarities of a diagram cause anomalies making the quasi-triangulation a non-simplicial complex or not a triangulation of all. These are mentioned in Section 4.3. Components of topological connectivity are discussed in Section 4.4. The representation of topology in data structures IWDS and eIWDS is briefly reviewed in Section 4.5.

4.1 From Simplicial Complex to Triangulation

This short section provides the definitions of a simplex, a simplicial complex and triangulation. The definitions come from [7, 34].

Definition 4.1.1. A *simplex* of dimension $k \leq d$ in d-dimensional Euclidean space (also called k-simplex) is a k-polytope with $k + 1$ vertices. Equivalently, it is the convex hull of $k + 1$ affinely independent points.

Any subset of $l+1 \leq k+1$ points of a k-simplex defines an l-simplex, called a *face*. Often 0-simplices are called *points*, 1-simplices are *segments*, 2-simplices are *triangles* and 3-simplices are *tetrahedra*.

Definition 4.1.2. A (*simplicial*) *complex* C is a finite set of simplices satisfying these properties:

1. any face of a simplex in C is also a simplex in C
2. if two simplices in C intersect, they intersect at a simplex of smaller dimension which is their common face of maximal dimension.

A complex C is *homogeneously d-dimensional* if and only if any lower-dimensional simplex in C constitutes a face of some d-dimensional simplex in C.

Definition 4.1.3. A *d-triangulation* is a homogeneously d-dimensional complex, which is connected and without singular faces.

4.2 Definition

In this section, a quasi triangulation is defined. The definition originates from [34, 31].

Definition 4.2.1. Let S be the set of spheres in $\mathbb{R}^3$ and $\mathrm{VD} = (V^V, E^V, F^V, C^V)$ the corresponding Voronoi diagram from Definition 3.1.2. The *quasi-triangulation* of the set S is defined as $\mathrm{QT} = (V^Q, E^Q, F^Q, C^Q)$, where $V^Q = \{v_1^Q, v_2^Q, \dots\}$ denotes the set of vertex simplices (q-vertices), $E^Q = \{e_1^Q, e_2^Q, \dots\}$ the set of edge simplices (q-edges), $F^Q = \{f_1^Q, f_2^Q, \dots\}$ the set of face simplices (q-faces) and $C^Q = \{c_1^Q, c_2^Q, \dots\}$ the set of cell simplices (q-cells), respectively. Let a dual operator D be defined as follows. Then D maps a Voronoi diagram VD to a quasi-triangulation QT.

- $\forall c^V \in C^V : D(c^V) \equiv v^Q$ maps a V-cell to a dual q-vertex. It is a point and corresponds to the center of the sphere defining c^V.
- $\forall f^V \in F^V : D(f^V) \equiv e^Q$ maps a V-face to a dual q-edge. It is a topological line segment connecting two q-vertices.
- $\forall e^V \in E^V : D(e^V) \equiv f^Q$ maps a V-edge to a dual q-face. It is a topological triangle over three q-vertices.
- $\forall v^V \in V^V : D(v^V) \equiv c^Q$ maps a V-vertex to a dual q-cell. It is a topological tetrahedron over four q-vertices.

An example of a VD of seven spheres and its corresponding QT is shown in Figure 4.1. Note the q-face formed by spheres a_1, a_2 and a_3. It is not a part of any tetrahedral q-cell, which means that there can be anomalies in the triangulation.

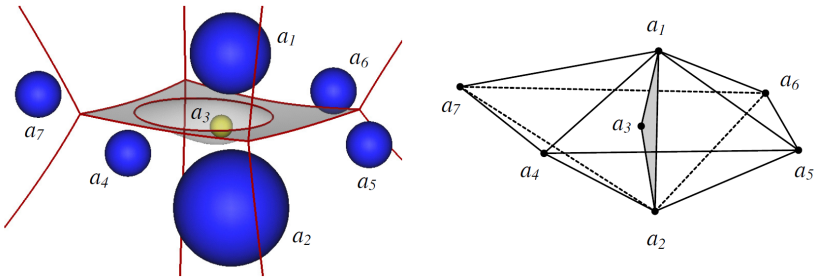


Figure 4.1: Voronoi diagram of seven spheres and its corresponding quasi-triangulation (taken from [31])

4.3 Anomalies

According to [34, 31], there can be three anomalies in a quasi triangulation violating the properties of a simplicial complex or a triangulation. The authors observed that in general (particularly for data such as atoms in a molecule), only a very few simplices causes the anomalies.

Degeneracy anomaly is caused by elliptic V-edge without any V-vertex. This means that there can be q-faces in QT(S) which are not part of any q-cell. This causes the complex to be non-homogeneous.

Another anomaly is called a *multiplicity anomaly*, which corresponds to the case of doubled V-vertices. In the dual space, two q-cells may share more than one q-face. The violation of Definition 4.1.2 is the following. In the case the intersection is formed by only two q-faces, then it is not a simplex. When the intersection consists of all three q-faces, then it is a simplex but not lower-dimensional.

The last one is a *singularity anomaly*, which is caused by topological holes in V-faces. In the dual space it occurs between q-cells sharing a specific q-edge, called a *singular edge*. This anomaly breaks the definition of a triangulation.

4.4 Connectedness and Worlds

This section briefly reviews the concept of worlds as connectedness-components in a QT. Details and proofs can be found in [31].

VD is face-connected and therefore QT is edge-connected. On the other hand, there can be topological holes in V-faces making a VD edge-disconnected, and because V-faces are mapped to q-edges by the dual operator, the corresponding QT can be face-disconnected. Considering the face-connectedness as an equivalence relation, the whole QT can be divided into (maximal) face-connected components. These components are called *worlds*.

Disconnectedness could be a problem for traversal algorithms, therefore it is important to devise some connectivity among them. This connectivity is realized via q-edges, thanks to the edge-connectedness of QT. There can be only one q-edge connecting two worlds, called a *gate edge* between two worlds. Although a gate edge in [31] is defined as the intersection of two neighboring worlds, resulting in a pair of q-vertices, we want to mention here that there can be more than one q-edge defined on the same pair of q-vertices, each one can be a q-edge to some small worlds.

There is a hierarchy among worlds. A world containing other worlds is called a *big world* and the

worlds beneath it are *small worlds*. Big and small are relative terms, meaning that a small world can be big for some other worlds. The hierarchy creates a tree connected by gates. A gate always connects one big world with all small worlds directly beneath it. For the purpose of topology traversal, it is sufficient to keep only the q-face incident to the gate as an entry to a world.

4.5 Topology Representation

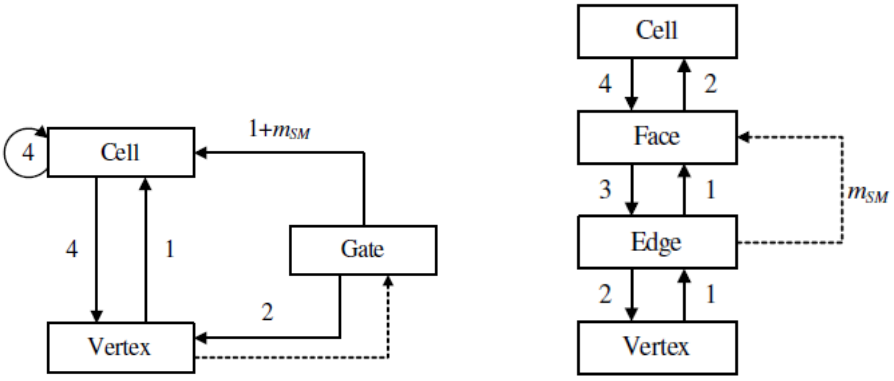
At first, VD used to be stored in a simplified *radial edge data structure* or simply *REDS* [10, 54, 38]. This structure has been originally developed for storing non-manifold models. But VD are not so general and even the authors from [10] claim later in [31] that using REDS was not memory efficient in this case. The approach based on REDS represents all elements in a VD, i.e., vertices, edges, faces and cells, and adds two supplementary entities - a *loop* for handling topological holes in faces and an *edge loop* for dealing with the adjacency of faces to an edge. Representing all VD elements was not necessary for their first algorithm [29, 30] for the construction of VD but it was essential for their second algorithm [24, 25]. In the work regarding topology representation of VD based on REDS [10], the authors have mentioned a more compact data structure such as the Delaunay triangulation as their future work. Not even a year later, in 2006, they introduced a quasi triangulation and *interworld data structure*, abbreviated *IWDS*, and its extended variant *eIWDS*. A similar approach based on Delaunay triangulation has been independently published by Medvedev et el. [44], but this was restricted to a single face-connected component. More work on quasi triangulation have been done since 2006. An exhaustive article [31] from 2010 provides additional properties of the triangulation (including proofs), elaborates the connection among worlds into more detail and describes so called *quasi operators* for solving topological queries, such as obtaining all q-faces incident to a q-vertex in a single world or among all possible worlds.

Data structures for the topology representation of a QT are shown in Figure 4.2.

The first schema in Figure 4.2(a) describes the IWDS data structure. Each q-cell references its four constituting q-vertices and four neighboring q-cells which share a common q-face with the q-cell. Degeneracy anomalies can be handled as a special case of a q-cell. Each q-vertex references one of its incident q-cell, remaining incident q-cells can be obtained by a topology traversal algorithm. Remaining elements, i.e. q-faces and q-edges, are not represented explicitly in this schema and hence gates have to be represented explicitly. A gate references $1 + m_{SM}$ q-cells: one for a big world and m_{SM} for all the small worlds belonging to the gate (these small worlds are children of the big world in the hierarchy tree). A gate also references two vertices constituting the corresponding edge[1].

IWDS can be useful for storing the quasi triangulation or as a data structure for a construction algorithm. The implicit representation of q-edges and q-faces saves the memory and makes the structure somehow simple. On the other hand, further processing may require the explicit representation of q-edges and q-faces. When this is required, then a gate can be represented implicitly. This is the case of eIWDS and its schema is shown in Figure 4.2(b).

[1] This can be problematic when multiple q-edges exist between these q-vertices.



(a) Interworld data structure (IWDS)

(b) Extended interworld data structure (eIWDS)

Figure 4.2: Data structures for topology representation of a QT. Pictures taken from [31].

Chapter 5

Algorithmic Aspects

The construction of a VD for a set of spheres, especially in dimensions $d \geq 3$, is not as easy as in the case of ordinary VD for a set of points. This section provides a brief overview of existing algorithms.

5.1 Will's Approach

A "practical algorithm for the computation of a single additively weighted Voronoi cell" was introduced by Will in 1999 [55]. It is based on the property from Section 3.3, stating that a cell can be represented as a subdivision of a sphere. The subdivision is kept in a winged edge data structure with additional helper edges. These helper edges overcome the problem with topological holes in faces and ensure the edge-connectedness of the graph.

The algorithm constructs the cell of a sphere among n other spheres iteratively. The algorithm is randomized - the ordering of the spheres is a random permutation. At first, an initial subdivision using helper edges is created and initial conflicts (will be explained later) with remaining spheres are computed. Then, the algorithm iteratively adds one sphere after another and maintains the subdivision. At each step, the subdivision represents a valid cell among the spheres added so far. Adding a sphere introduces a bisector that may cut off one or more parts of the cell. The situation when the bisector separates a Voronoi vertex, intersects an edge or a face, is called a *conflict*. These conflicts are associated with elements of the subdivision as well as with the remaining spheres. When a sphere is added, its associated edge conflicts are processed by modifying the subdivision appropriately. After that, redundant old edges are removed. The problem of face conflicts is transformed to the problem of edge conflicts by subdividing the affected faces using helper edges. Then, conflict information is updated and the next sphere can be added.

In his thesis, Will also introduced a theoretical algorithm for the computation of an additively weighted Voronoi cell, working in expected time complexity $O(n^2 logn)$ for general data, where the combinatorial complexity of a single cell can be as high as $O(n^2)$, and in expected time complexity $O(nlog^2 n)$ for data with cell complexity bounded by $O(n)$. The practical algorithm has been designed as a simplified version of the theoretical one, exploiting the moderate combinatorial complexity of cells in molecular systems.

For tasks where more cells have to be constructed, it can be useful to use a pre-processing. In order to decrease the number of spheres n for the computation of a cell, Will proposed to construct a 4D power diagram[1] as discussed in Section 3.5 and then use neighbors of a power cell as the input set of spheres for the computation of an additively weighted Voronoi cell.

[1] In fact, he goes even further - to 5D polyhedra representing a 4D power diagram.

5.2 Region Expansion

The idea of the region expansion algorithm introduced by Kim et al. [24, 25] is to start with an ordinary Voronoi diagram of points (centers of the spheres from the input set). Recall that this is also a valid Voronoi diagram of spheres with equal radii and can be a good approximation of the final diagram. The algorithm then expands one sphere after another to its final size, keeping the diagram to be a valid Voronoi diagram of spheres. A two-dimensional analogy is shown in Figure 5.1. When all spheres are expanded, the algorithm finishes and outputs the final diagram. Expanding a single sphere to its final size means to identify the events which might change the topology of the diagram and to handle them properly. Events occur only at the edges along the boundary of a cell and its so-called radiating faces. They can cause existing edges to vanish, new edges to be born, etc. Each event occurs at some specific value of the expanding radius (the time of the event). The events are sorted in the ascending order and the algorithm takes one event after another in this ordering, updates the topology accordingly and schedules possible new events for processing. The idea is nice and the great advantage is that it constructs the entire diagram, not only a single component, but it has also several drawbacks. First of all, this approach requires a data structure capable of dynamic changes and providing quick access to the edges bounding the faces of the expanding cell as well as to the edges bounding faces radiating from this cell, including the boundary of topological holes in these faces. In other words, it requires something like REDS [10] mentioned in Section 4.5. Furthermore, it may happen that the topology constructed by the expansion of a single sphere is undone by the expansion of its neighboring spheres. Another issue are the analytical requirements, such as finding the roots of a quartic polynomial with non-trivial coefficients.

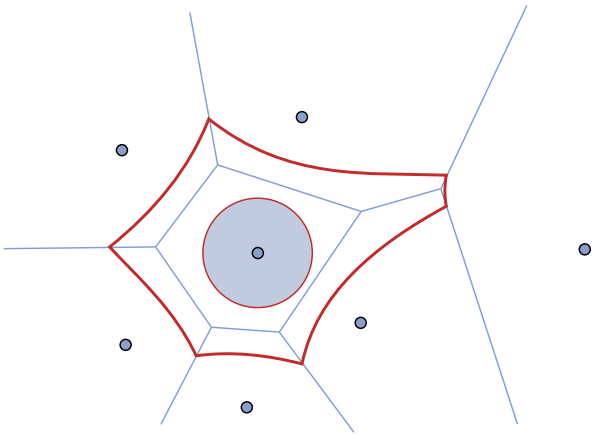


Figure 5.1: 2D analogy of an expanding sphere with a topological change.

5.3 Edge Tracing and Similar Algorithms

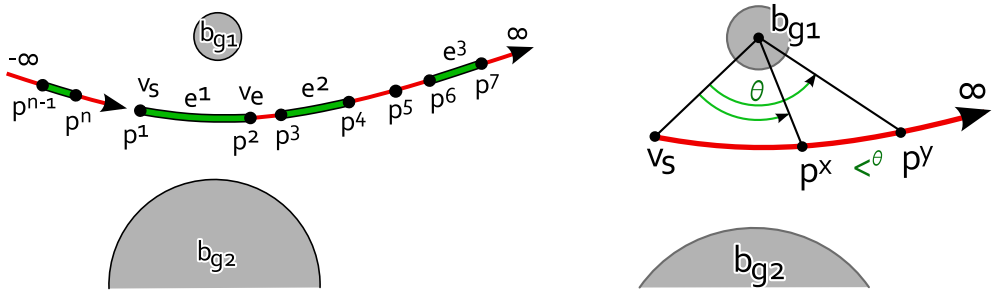
There is a simple idea common to the edge tracing and similar algorithms: Discovering unknown Voronoi vertices from known vertices along edges. In this general form, the idea has been used for several years before the algorithms for the construction of Voronoi diagram of spheres were published. Let us give two examples. In 1999, Luchnikov et al. [39] introduced a numerical algorithm for the construction of Voronoi network of convex objects (including spheres). Their algorithm is based on the computation of a trajectory of an empty sphere, variable in size, moving inside the system. In

2002, McConkey, Sobolev and Edelman [42] published an algorithm for the construction of a power diagram via tracing its edges.

Kim et al. [29, 30] introduced an *edge tracing algorithm* for the construction of Voronoi diagram of 3D spheres. This work focuses on the properties of the diagram including the computation of the geometry and the basic variant of the algorithm, but disconnectedness and filtering techniques for making the algorithm fast are only marginally mentioned. Medvedev et al. [44] have published an *algorithm for the construction of Voronoi S-network*, which does the same thing but works with a dual representation. Then there is also a *face tracing algorithm* by Kim et al. [34], which is basically the edge tracing formulated for a quasi triangulation (it traces q-faces), but it uses the IWDS and also deals with disconnectedness in the quasi triangulation. We will briefly describe the edge tracing algorithm in this section as it is usually easier to understand this formulation than the one in the dual space. The following description of the edge tracing algorithm comes from our paper [41].

The edge tracing computes the diagram of a set of spherical balls as a graph of Voronoi vertices connected by Voronoi edges. Each vertex is defined by four balls, so exactly four edges are incident to each vertex. As it was mentioned before, the key idea of the algorithm is to discover unknown vertices from known vertices by tracing incident edges. Tracing an edge requires ordering of its points, so first we will look at an edge from the context of a vertex and define some terms before the review of the algorithm.

For a known vertex we have its four defining balls and its position. Three balls from this quadruple are called *gate-balls* and they define an incident edge. The remaining ball is *start-ball* and the vertex in this context is *start-vertex* v_s. The edge can be oriented from v_s. Similarly, the opposite vertex of the edge is called *end-vertex* v_e, it is defined by the same gate-balls and one ball called *end-ball*. In the context of a vertex we know where an edge starts, its orientation and shape but we do not know where it ends, i.e. v_e. We will call such partially unknown edge a *trajectory*. See a 2D analogy of a trajectory in Figure 5.2(a). Gate balls are denoted as b_{g1} and b_{g2}. Note that there can be more than one edge on the trajectory $(e^1, e^2, \dots)$, but we are usually interested in e^1 and the end-vertex v_e. There are also some points $p^1 \dots p^n$ on the trajectory. Some of them can be valid Voronoi vertices, but some of them do not need to be, such as p^5. The ordering between any points p^x and p^y is given by the orientation from v_s and can be realized by using an *angular distance* [30] denoted as $\theta \in [0, 2\pi)$. We will denote the ordering as $<^\theta$. An example is shown in Figure 5.2(b), where p^x is closer to v_s than p^y, so $p^x <^\theta p^y$. Another distance function can be used as well [44]. Depending on the configuration of gate-balls, the shape of a trajectory can be linear, hyperbolic or elliptic. We will focus more on non-elliptic edges as their occurrence in real data is much greater than the occurrence of elliptic edges.



(a) 2D analogy of a trajectory

(b) Angular distance θ

Figure 5.2: Fundamental parts of the edge tracing algorithm.

The algorithm expects that an initial vertex is given. Four trajectories are emanating from this vertex. The algorithm traces each trajectory to find the next vertex on it – the end-vertex – by examining balls from the input set and finding the end-ball. When such a vertex is found, the trajectory is finished, the vertex becomes end-vertex and hence an edge can be created. If it is a new vertex, it produces three new trajectories for tracing. Otherwise the opposite trajectory emanating from the known end-vertex becomes finished too. The tracing step repeats until there are no trajectories left.

The end-vertex for a trajectory is defined by three known gate-balls and the unknown end-ball. A brute force approach examines all the input set except the gate-balls in order to find the end-ball. This process produces candidates for the end-vertex, i.e., points on the trajectory (such as $p^2 \dots p^n$ in Figure 5.2(a)). From all the possible candidates, the one closest to the start-vertex and not beyond ∞ is chosen to be the end-vertex. This search takes $O(n)$. Both the expected and the worst case time complexity of edge tracing using the brute force end-vertex search is $O(mn)$.

An obvious problem is how to find the initial vertex. In [30] they suggest to add other four balls to the input set for which the vertex is known and to trace edges to find the initial vertex that is defined only by balls from the original input set. Another strategy is presented in [44] – they start with some original ball and find the three missing balls one after another by minimizing some distance functions. Another problem is how to determine whether the end-vertex already exists in the graph. The strategy suggested in [30], and more or less used in [44], is to use a hash-table for the vertices.

Chapter 6

Beta-shapes and Beta-complexes

There are applications where having just a Voronoi diagram or a quasi-triangulation for the given set of spheres is not yet enough. Consider a spherical empty probe, i.e., an open sphere of some fixed radius β which shall not intersect any of the given spheres. There are places in space where this probe can be and there are places where it cannot be without violating its emptiness. For example, the probe can represent a bounding sphere of another molecule (e.g. a ligand or a solvent molecule, such as water) interacting with a protein molecule. The accessible and inaccessible space are separated by a surface and hence it could be useful to describe this surface topologically in terms of the spheres from the given set. The (finite) inaccessible space is also important, e.g. for volume computation. Therefore, the concept of beta-shapes has been introduced by Kim et al. [36] in 2006. This concept is inspired by the similar concept of alpha-shapes and weighted alpha-shapes based on Delaunay and regular triangulations, respectively. Alpha-shapes were introduced by Edelsbrunner and Mücke in 1994 [14]. This chapter gives an overview of the concept of beta-shapes.

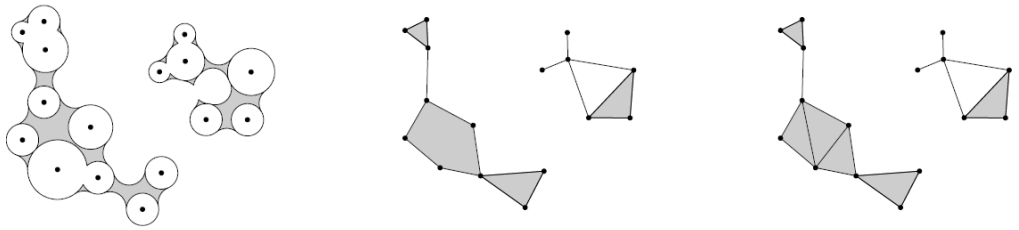
Historically, the construction of beta-shapes was based on the Voronoi diagram at first [36]. The authors gave the reason that a quasi-triangulation is not a valid tessellation of space in general. However, it turned out that it is better to use the quasi-triangulation for the definition and construction of beta-shapes and also the newly defined beta-complexes [49]. The whole theory of beta-shapes and beta-complexes together with their construction from a quasi-triangulation can be found in the work of Kim et al. [32] from 2010.

6.1 Definition

Figure 6.1 shows a 2D analogy of the concept for a set of 13 discs and some particular value of β. The beta-hull is shown in Figure 6.1(a). It is the shape of the space inaccessible by an empty probe of the radius β. There can be holes in the hull such as in the upper right triplet of discs in Figure 6.1(a) since the probe can be placed there. Figure 6.1(b) shows the corresponding beta-shape. The beta-complex is shown in Figure 6.1(c). It may consist of lower-dimensional simplices. The boundary of a beta-shape consists of lower-dimensional simplices from a beta-complex and the entire beta-complex consists of simplices from a quasi-triangulation.

Let us define a three-dimensional beta-shape more formally by Definition 6.1.1 and beta-complex by Definition 6.1.2. These definitions come from [32].

Definition 6.1.1. (beta-shape) Let A be a set of three-dimensional spheres, B its subset $B \subseteq A$ of size $1 \leq |B| \leq 3$ and let $\chi(B)$ be the set of centers of all spheres in B. Suppose that there is an empty probe of radius β touching all spheres in B. Then the simplex σ_B defined by the points $\chi(B)$ is called a *bounding simplex*. The *beta-shape* $\mathcal{S}_\beta$ for A, corresponding to the value of β, is defined by the object bounded by all bounding simplices and occupies a finite region in $\mathbb{R}^3$.



(a) Beta-hull (b) Beta-shape (c) Beta-complex

Figure 6.1: 2D analogy of a beta-hull, beta-shape and beta-complex. Pictures taken from [32].

Definition 6.1.2. (beta-complex) Let A be a set of three-dimensional spheres, B its subset $B \subseteq A$ and let $\chi(B)$ be the set of centers of all spheres in B. Let b be an empty probe of radius ρ touching all spheres in B and σ_B be the simplex defined by $\chi(B)$. Then the beta-complex C_β, corresponding to the given value of β, is defined as the set $C_\beta = \{\sigma_B, \sigma_{B'} \mid B' \subset B, 1 \leq |B| \leq 4$, for all possible B and for any probe b of radius $-r_{max} \leq \rho \leq \beta\}$, where r_{max} is the maximal radius among spheres in A.

6.2 Computation from a Quasi-triangulation

This section gives a very short overview of the computation of the beta-complex C_β and the boundary $\partial\mathcal{S}_\beta$ of a beta-shape $\mathcal{S}_\beta$ for the given value β from the simplices of a quasi-triangulation.

Let us consider β as a variable in the range $[-r_{max}, \infty)$. For the given β, a simplex σ can be a part of the corresponding beta-complex C_β and it can further belong to the boundary $\partial\mathcal{S}_\beta$ of the corresponding beta-shape $\mathcal{S}_\beta$. In order to classify this somehow, a simplex $\sigma \in C_\beta$ has a bounding state, making the simplex σ either

- *singular* if $\sigma \in \partial\mathcal{S}_\beta$ but it does not bound any higher-dimensional simplex in C_β,
- *regular* if $\sigma \in \partial\mathcal{S}_\beta$ and it bounds some higher-dimensional simplex in C_β or
- *interior* if $\sigma \notin \partial\mathcal{S}_\beta$

Note that cell simplices can be only interior.

Let σ be a simplex from a quasi-triangulation. A *beta-interval* is the interval of β values corresponding to a bounding state of σ, so there can be a singular, a regular and an interior beta-interval for the given σ. A *beta-span* is the union of beta-intervals.

Finding a beta-complex C_β for the given value of β means finding simplices in the quasi-triangulation which have a beta-interval containing β. Vertex, edge, face and cell simplices are handled separately. At first, beta-intervals of all simplices must be computed. This is done for all cell simplices first, then for all face, edge and vertex simplices in this order, because the information for computing beta-intervals obtained in one stage is useful in the following stage. When this is done, simplices are sorted using the minimum value of a beta-span as a key. Finally, C_β is searched among these sorted simplices - a binary search is performed in order to find a simplex belonging to C_β, then a sequential search can be performed, finding the interval of all simplices belonging to C_β.

We refer the reader to [32] for the rules of computation of beta-intervals.

Finding a beta-shape can be done in two ways. The first way is to compute the beta-complex and then leave the interior simplices out. The second way is to search singular and regular simplices directly from the quasi-triangulation using an approach very similar to the computation of a beta-complex.

The suggested data structure for storing the quasi-triangulation is eIWDS in this case, representing lower-dimensional simplices explicitly.

6.3 Complexity

The worst case time complexity of searching a beta-complex for the given value of β from an already computed quasi-triangulation is the following. Let m be the total number of simplices in the quasi-triangulation and k be the total number of simplices in the beta-complex. The computation of all beta-intervals takes $O(m)$ time. Sorting simplices can be done in $O(m \log m)$. Finding the beta-complex among the sorted simplices takes $O(k + \log m)$. Note that it is sufficient to compute the beta-intervals and to sort the simplices only once, i.e., to do a pre-processing in $O(m \log m)$, beta-complexes for different values of β can be then easily computed in $O(k_\beta + \log m)$.

Chapter 7

Applications to Proteins

This chapter is dedicated to some important applications of the concepts described in this work to the field of bioinformatics. The chapter is organized as follows. Section 7.1 gives a short overview of protein structures. Section 7.2 provides common definitions of surfaces related to molecular systems. Sections 7.3 describes, how such a surface can be computed from a beta-shape. The concept of interaction interface is briefly mentioned in Section 7.4. Computation of pockets from a beta-shape is reviewed in Section 7.5.

7.1 About Proteins

The atoms of a protein molecule are organized into chains made from 20 kinds of amino acids and connected by peptide bonds. A schematic view to such a chain is shown in Figure 7.1. When a protein is being created, these chains fold into a compact form, as it was shown, e.g, in Figure 1.1(c). More than one chain can constitute a protein, e.g., a dimer consists of two chains, a trimer of three chains and so on.

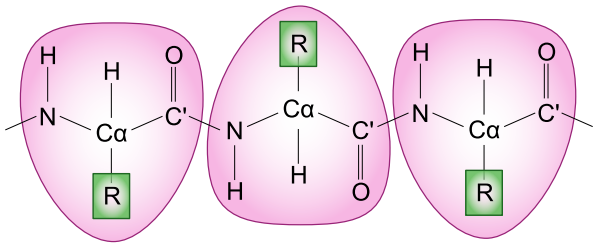


Figure 7.1: A polypeptide chain. The main chain is along $-N - C_\alpha - C'-$, peptide bonds are $C' - N$ and R denotes a residuum from an amino acid forming a side chain.

In the Van der Waals model, each atom is modeled as a small sphere with the corresponding radius [8]. For example, hydrogen has 1.2, carbon 1.7, nitrogen 1.55, oxygen 1.52, phosphorus 1.8 and sulfur also 1.8 angstroms (1Å=10^{-10}m). In this model, an atom often intersects with one or more neighboring atoms.

7.2 Surfaces

The reactions corresponding to the function of a protein occur in special places called active sites. Another smaller molecule travels to this active site, docks there and the function is performed. Since a protein is folded, many atoms are buried inside the protein and hence inaccessible to the molecule.

On the other hand, atoms on the surface may be accessible. But what is the surface of a molecule?

The fundamental surface is the *Van der Waals surface*, which is the boundary of the union of all atoms in a molecule.

But proteins usually exist in solvent, such as water, and this definition does not take this aspect into account. Therefore, other definitions appeared. They use the concept of an empty sphere rolling around the protein, called a *probe*, which represents a bounding sphere of a smaller molecule, such as water (1.4 Å) or a molecule traveling to the active site.

A *solvent accessible surface* (SAS) was proposed by Lee and Richards in 1971 [37]. Given a protein molecule and a probe radius, its SAS with respect to the probe is the locus of the center of the probe tangentially rolling around the protein.

A *molecular surface* (MS), often called a Connolly surface, was proposed by Connolly in 1983 [12]. Consider the complement of the union of all possible empty probes. The boundary of this set is the MS. This surface is further divided into a *contact surface* (CS) and a *reentrant surface* (RS). The CS corresponds to the points on the boundary of atoms and the RS to the remaining points.

A two-dimensional analogy of a SAS and a MS is shown in Figure 7.2.

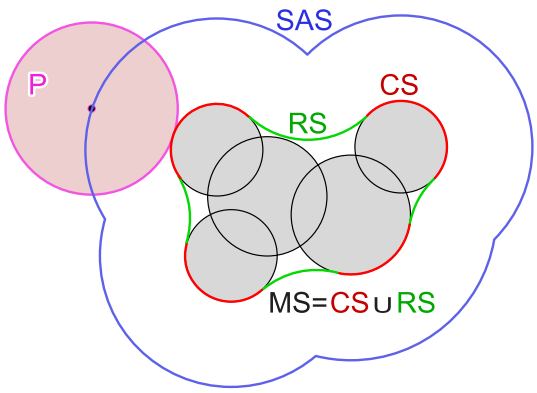


Figure 7.2: A 2D analogy of the solvent accessible surface SAS and the molecular surface MS for the given probe P. The MS consists of the reentrant surface RS and the contact surface CS.

7.3 Computation of Molecular Surface via Beta Shape

It turns out that the molecular surface of a molecule (MS) can be effectively computed when a beta-shape for the balls representing atoms is known. This approach was described by Ryu, Park and Kim in 2007 [48] and we will review it in this section.

An example of a MS for one specific value of β is shown in Figure 7.3. The surface is divided into several patches. These patches are shown in more detail in Figure 7.4. They are classified as *contact patches*, *rolling patches* and *link patches*.

- A contact patch corresponds to the part of an atom surface exposed to a probe.
- A rolling patch is created when a probe can roll around two atoms, such as in Figure 7.4(a). This patch is called *complete*, when the rotation trajectory is a full circle, such as in Figures 7.4(a)

and 7.4(b). Otherwise, the rotation trajectory is a circular arc and the rolling patch is called *partial*, such as in the case of Figure 7.4(c). Furthermore, a rolling patch is *self-intersecting*, when the probe intersects the axis of rotation, such as in Figure 7.4(b). Otherwise, it is *non-self-intersecting*.

- A link patch is the spherical patch formed by a probe touching three atoms, such as in the case of Figure 7.4(c). Note that a link patch may have a hole inside and it can be bounded by rolling patches as well as by other link patches.

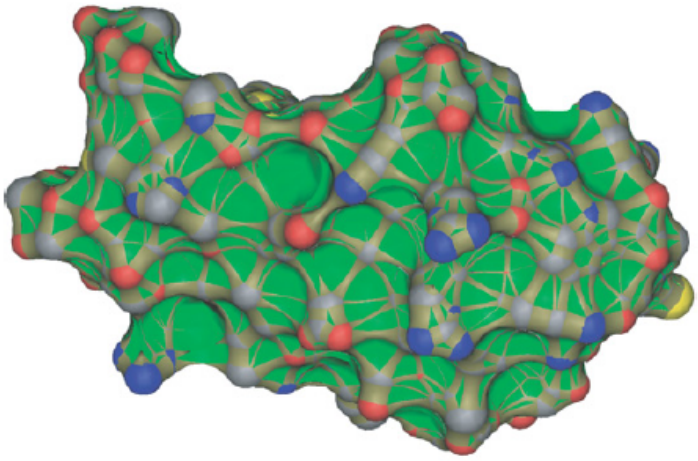
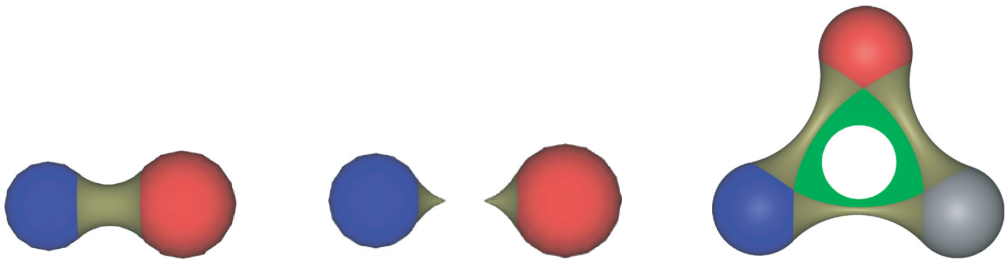


Figure 7.3: A molecular surface. Picture taken from [48].



(a) A complete non-self-intersecting rolling patch.

(b) A complete self-intersecting rolling patch.

(c) A link patch with a hole. The link patch is bounded by three partial rolling patches.

Figure 7.4: Classification of patches constituting a molecular surface. Pictures taken from [48].

Now we will summarize their approach of computing these patches.

If the quasi-triangulation for the set of atoms in a molecule is available, the beta-shape $\mathcal{S}_\beta$ can be effectively computed for any value of β, in particular for β describing the radius of a probe. This $\mathcal{S}_\beta$ then describes topological relations among the atoms constituting the MS. To be more specific, there is a one-to-one correspondence between a rolling patch in the MS and a q-edge in the $\mathcal{S}_\beta$ and also between a link patch in the MS and a q-face in $\mathcal{S}_\beta$.

All rolling patches are computed from q-edges. For a q-edge e, it is first determined if the corresponding rolling patch is complete or partial. When e is dangling in $\mathcal{S}_\beta$, it is complete, otherwise, it

is partial. Then it must be determined if the rolling patch is self-intersecting or not. This is done by a simple comparison based on the two balls representing the q-vertices of e and the value of β. After that, the rolling surface is computed as a surface of revolution generated by a single arc or two arcs rotated along a circle path or a circular arc path.

All link patches are computed from q-faces. A link patch has the shape of a spherical triangle unless it does not intersect another patch. The computation is done in two steps. In the first step, the spherical triangle of every link patch corresponding to every q-face is computed without considering any of the possible intersections. These are called the initial (untrimmed) patches. In the second step, every initial patch is trimmed by the other initial patches in the neighborhood. The intersection can occur at the boundary as well as the interior of the initial patch. In the later case, a topological hole can be created in the initial patch.

The neighborhood can be more than just the adjacent link patches. This is solved by a nice trick including an ordinary Voronoi diagram of points. Recall that an initial link patch is a spherical triangle on a probe sphere. The trick is to associate the center of this probe with the corresponding link patch and compute 3D Voronoi diagram of these centers. The authors have shown that the set of candidates for trimming is the set of neighbors of the corresponding center.

The contact patches are computed from q-vertices of $\mathcal{S}_\beta$. Each q-vertex corresponds to a sphere representing an atom. This sphere is trimmed by the rolling patches corresponding to adjacent q-edges in order to obtain a spherical polygon representing the contact patch.

We refer the reader to [48] for more details and also for the history of another approaches.

7.4 Interaction Interface

Recall that a protein can consist of more than one chain of atoms. In order to study the interaction between groups of atoms (chains or another molecule interacting with a protein), the concept of *interaction interface* separating these groups was proposed. For a long time, this interface used to be constructed using a power diagram by a method proposed by Varshney et al. in 1995 [51]. But it turns out that it can be more precisely defined in terms of the Voronoi diagram of balls as the set of Voronoi faces separating these groups [22]. Furthermore, in this case, the minimal distance from each point of the interface to the group is immediately available. This distance information is used to cut off distant parts of the interface as only the close parts are important in the interaction.

7.5 Extraction of Pockets

Pockets of a protein are the depressed regions on the surface of the molecule. They are also called *docking sites* and play an important role in the interaction between a protein and a ligand, i.e., another molecule that docks in the pocket and, e.g., forces the protein to do its job. Automatic recognition of docking sites on a protein is therefore of a great interest.

Beta-shapes are also used as an effective geometric construct for automatic recognition and extraction of pockets. This section reviews the approach of Kim et al. from 2006 [28]. The main idea was discovered before the concept of beta-shapes. In their previous work, Kim et al. used a Voronoi diagram of balls, but the main idea is the same. We refer the reader to [28] for the history of other

approaches and also to [23], where a method for automatic docking of a ligand in the context of a pocket is described. The method uses a genetic algorithm to find the optimal position of the ligand in the pocket, which minimizes free energy.

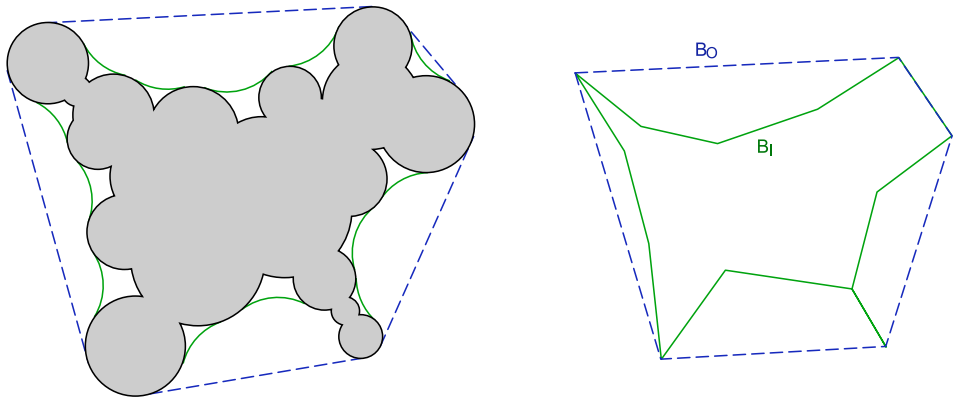
The rest of this section describes the method [28] for the extraction of pockets.

Given a protein molecule, two beta shapes are computed - one for a large β value (e.g., $\beta = \infty$), which is called an *outer beta-shape* B_O, and the other for a small β value (this is a parameter, e.g., $\beta = 8$ Å), called an *inner beta-shape* B_I. The pockets are then recognized from the regions between B_O and B_I.

A two-dimensional example is shown in Figure 7.5. Note that $B_O \subseteq B_I$ topologically, i.e., the elements of B_O are contained in B_I. This can be seen in Figure 7.5(b). Also note that under each edge of B_O, there is a chain of edges in B_I. It is not hard to see that the extraction of pockets in 2D can be done by reporting the inner edge chains corresponding to outer edges.

The situation is more complicated in 3D. There are triangular faces of B_O. All three vertices of an outer face belong also to B_I. For each outer face, there can be a depressed region of inner faces, but this can be just a part of a pocket as there can be two outer faces sharing an outer edge, together constituting a larger depressed region. Therefore the following method was proposed. For each pair of vertices constituting an outer edge, the shortest path of inner edges (in terms of their Euclidean length) is found between these vertices. As each face has three edges, three paths can be found for the face. These paths define a boundary on inner faces. These inner faces are called a *pocket primitive* and associated to the outer face.

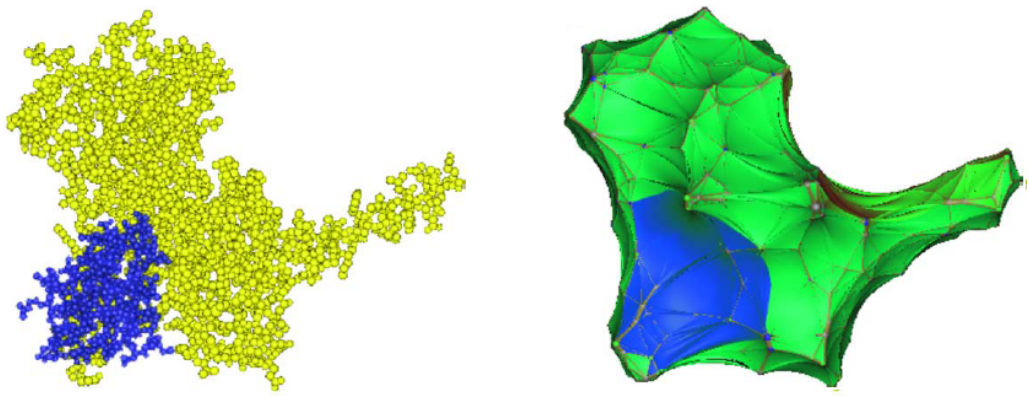
In the last step, pocket primitives are merged together according to some heuristics in order to define pockets. The heuristic is the following. When two outer faces share an outer edge, the average distance between the outer edge and the vertices of the corresponding path is determined. The pocket primitives of these faces are merged when the distance is less than a threshold. The suggested threshold is the global average of these distances.



(a) The outer and inner molecular surface (b) The outer (B_O) and inner (B_I) beta-shape

Figure 7.5: Pockets in 2D - between the outer and the inner beta-shape.

An example of the final result achieved by this method is shown in Figure 7.6. A protein and a ligand is shown in Figure 7.6(a). Two beta shapes were computed, pocket primitives extracted and merged. The final pocket is shown in Figure 7.6(b).



(a) A protein and a ligand

(b) The molecular surface of the protein and a pocket

Figure 7.6: A pocket extracted using the approach based on beta-shapes. Pictures taken from [28].

Chapter 8

Contribution

This chapter presents our contribution. The curiosity about the geometry of elements in a Voronoi diagram balls have lead us to the question "What would happen with the diagram if we switched the meaning of being inside and outside for some of the defining balls?" We came up with the concept of an inverted ball and used it to define the boundary of the diagram. This concept is further presented in Section 8.1. Then we focused on algorithmic aspects of the construction of these diagrams. The edge-tracing algorithm [30] used for this purpose has a very serious drawback - its expected time complexity is $O(n^2)$ for n balls such as atoms in a molecule. Instead of using a spatial grid for this purpose, we extended the idea of geometric filters from [11] and introduced a new approach based on three-dimensional Delaunay triangulation, which can reduce the expected time complexity significantly. This approach is presented in Section 8.2.

8.1 Inverted Ball

Let us start with a short motivation. The Voronoi diagram of a set of spherical balls partitions the space into cells. The interior of each cell represents the points which are closer to the defining ball than to any other ball. The cells corresponding to the balls constituting the convex hull of the input set are unbounded. To be more specific, they are not completely bounded - the set of points has an infinite diameter. Both bounded and unbounded cells may constitute elements of the diagram which are far away from the input set. The question is: Do we really need to describe such distant relations in practice? Or, in another way, how much is it important for an observer to know which planet of a small system is closer if the system is many lightyears far away? Another question is how accurately the coordinates of distant Voronoi vertices can be computed. This is important when the diagram is constructed by an algorithm. When a topological mistake is made, it can propagate. Using a boundary constraint can address these issues.

A common technique is to add an extra site to the input set that represents infinity, but this allows distant elements. Another solution might be extending the input set by adding several extra sites, e.g., $d + 1$ sites forming an outer $(d + 1)$-simplex are sufficient for d-dimensional diagrams. Another technique might be using the input set periodically or introducing some application-dependent constraint such as a cylinder.

Because diagrams in our case are defined by a set of spherical balls, we came up with an idea to use a spherical constraint which would be a part of the input set. We call it an *inverted ball* and it is nothing more than just the complement of the interior of an ordinary ball. It does not matter how many dimensions we are dealing with, only one such a ball is sufficient for bounding all cells like in the case of the infinite site. Furthermore, it can avoid distant elements.

We introduced the concept of an inverted ball in [40]. However it turned out later that this idea was

not new. A two-dimensional approach similar to the inverted ball has been used already by Kim, Kim and Sugihara in [26] for the construction of a Voronoi diagram of circles in a circle and the three-dimensional concept was mentioned also in the work of Kim et. al. [35] where they introduced a multiresolution protein model.

A two-dimensional analogy of an inverted ball is shown in Figure 8.1. There is a diagram for a set of two-dimensional balls, one of them is inverted. Note cells that would otherwise have a great diameter such as unbounded cells are now constrained by the inverted ball.

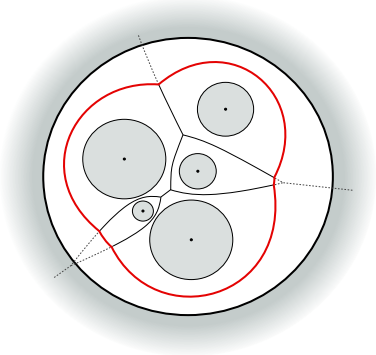


Figure 8.1: A 2D analogy for the inverted ball

In Section 8.1.1, the distance function from Definition 3.1.2 is modified to work with ordinary balls as well as with inverted balls and it is extended to measure the distance between two balls. Section 8.1.2 is dedicated to the the new type of a bisector - an ellipsoid - caused by the inverted ball. Examples of diagrams using the inverted ball are shown in Section 8.1.3.

8.1.1 Distance Function

The distance function has to be modified in order to have the inverted ball in the input set. Without a loss of generality, we require all balls from the original input set to have non-negative radii, since adding a positive constant to all original radii does not change the diagram. Then we can distinguish the inverted ball from them by a negative radius.

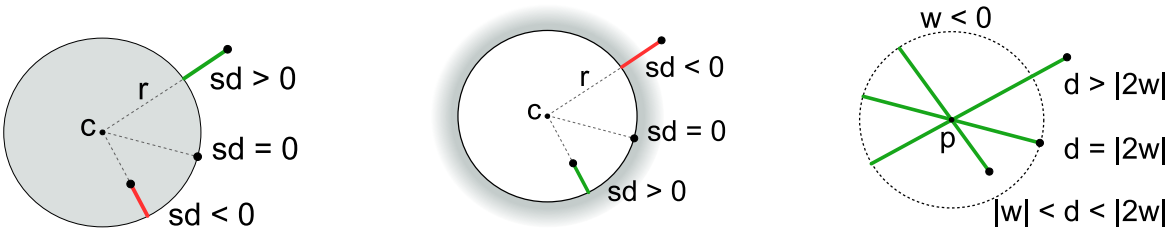
Definition 8.1.1. For a ball $b = (c, r)$ and a point x let the *extended signed distance function* be defined as follows

$$sd(b, x) = sgn_{0+}(r)(\|c - x\| - |r|)$$

where $sgn_{0+}(r)$ is $+1$ for $r \geq 0$ and -1 for $r < 0$, $c \in R^3$ is the center and $r \in R$ is the signed radius of the ball.

Definition 8.1.1 extends the signed distance function to work with ordinary balls with non-negative radii as well as with inverted balls which are represented by negative radii. It can be interpreted as the shortest distance between a point and the boundary of a ball.

Figures 8.2(a) and 8.2(b) show the extended signed distance for an ordinary ball and for an inverted ball, respectively. The function is negative for points inside the ball, positive outside and zero for points on the boundary. Figure 8.2(c) is the case of negative weight in additively weighted Voronoi diagrams. The meaning is different than the meaning of the signed distance.



(a) Ordinary ball (b) Inverted ball (c) Negative weight

Figure 8.2: Extended signed distance of a point and a ball.

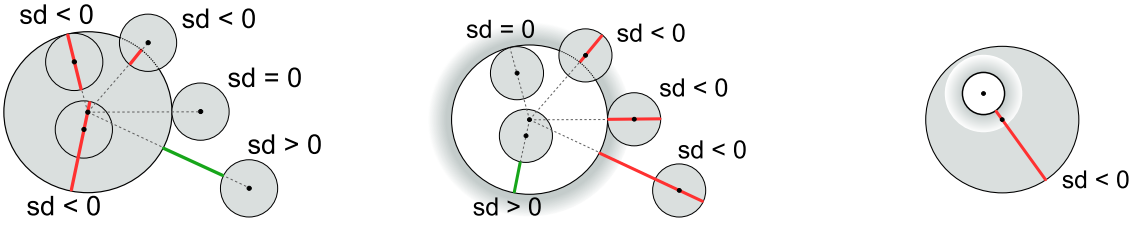
Definition 8.1.1 provides the distance between two balls when one of them may be inverted. This is important to give a meaning to the nearest neighbor and for determining if one ball is fully contained in another ball. It is done by reducing one of them to a point. In fact, we will select the ball with the radius closer to zero, subtract the radius from the ball and add it to the radius of the other. Reduction of an inverted ball implies a sign change since all points (except the center) are considered inside of such a reduced inverted ball.

Definition 8.1.2. For two balls $b_1 = (c_1, r_1)$ and $b_2 = (c_2, r_2)$ where at least one of them is not inverted, let the signed distance between them be defined as

$$sd(b_1, b_2) = sgn_{0+}(r_i)sd(b, c_2)$$

where $b = (c_1, r_1 + r_2)$ is a modified ball, $i = 1$ if $|r_1| < |r_2|$ and $i = 2$ if $|r_1| \geq |r_2|$.

Several cases of signed distance between two balls according to Definition 8.1.2 are shown in Figure 8.3. The distance is measured between a big ball and a small ball. Figure 8.3(a) shows several configurations when only ordinary balls are involved. When a ball intersects another ball, the distance gets the negative sign and its magnitude represents the amount of penetration of one ball to another. When two balls just touch each other externally, the distance is zero. When they do not intersect, the distance is positive. Things get more complicated in Figure 8.3(b) since there is an inverted ball involved. This is almost the same configuration as in Figure 8.3(a) but the meaning of being inside is changed. In Figure 8.3(c), there is an interesting configuration when the inverted ball has a smaller magnitude than the ordinary ball. In this case, the distance is always negative since interiors of these balls always intersect.



(a) Ordinary balls (b) Inverted ball (c) Small inverted ball

Figure 8.3: Extended signed distance among balls

8.1.2 The Geometry of Voronoi Faces and Edges

A face in a Voronoi diagram of ordinary balls is a part of a hyperboloid of two sheets or a plane (we do not consider degenerate cases when a face would have the dimensionality other than two). But what if an inverted ball is involved? Then Lemma 8.1.1 says that a face may be a part of an ellipsoid.

Lemma 8.1.1. *Suppose we are given one inverted and one ordinary ball. If any of them is not fully contained in another ball, the set of points equidistant to these balls is an ellipsoid. At most one of its semi-axes has different length than the other.*

Proof. Denote the inverted ball as $b_1 = (c_1, r_1)$ and the ordinary ball as $b_2 = (c_2, r_2)$. Assume without loss of generality that $r_2 = 0$ (subtracting r_2 from radii of these balls does not change the set of equidistant points). Hence b_2 can be considered to be a point. Furthermore, it is sufficient to prove only the two-dimensional case in an arbitrary plane that contains both c_1 and c_2 as it is shown in Figure 8.4, i.e., that the set of equidistant points is an ellipse – denoted as f. This assumption is satisfied thanks to the rotation-symmetry with the axis defined by c_1 and c_2. The rotation-symmetry implies the shape of the ellipsoid, i.e., that at most one of its semi-axes has different length than the others – it is the rotation axis. It is easy to see that f is an ellipse: Consider an arbitrary point equidistant to both b_1 and b_2 with respect to the signed distance from Definition 8.1.1. It is the center of an empty sphere tangent to b_1 and b_2, denote it as $s = (c, r)$. Euclidean distance from c to c_1 equals to $r_1 - r$ and from c to c_2 equals to r. Their sum equals to r_1 which is constant for all points of f, hence f is an ellipse with foci c_1 and c_2. □

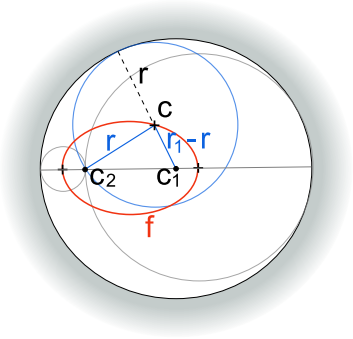
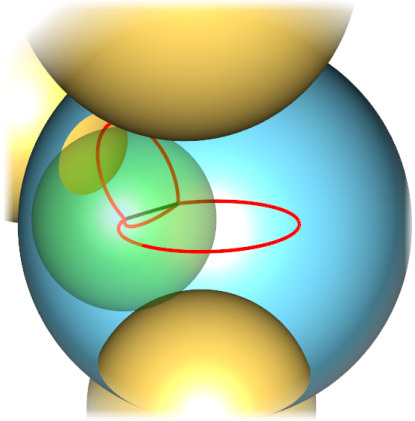


Figure 8.4: 2D analogy of an elliptic bisector f between an inverted ball (c_1, r_1) and a ball $(c_2, 0)$ with zero radius.

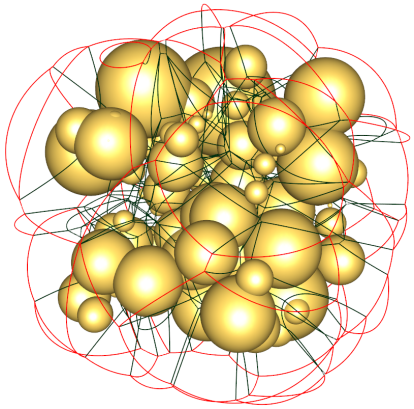
Conjecture 8.1.1. Suppose we are given one inverted and two ordinary balls. If any of them is not fully contained in another ball, the set of points equidistant to these balls is an ellipse.

8.1.3 Results

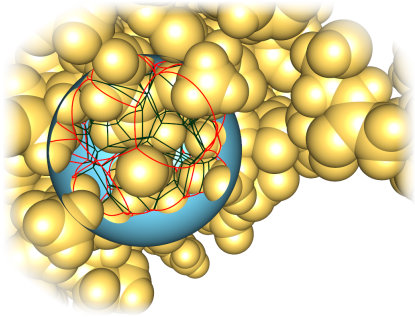
Figure 8.5 shows results from our implementation of the edge-tracing algorithm that has been modified to work with an inverted ball. A simple case of four balls with one inverted is shown in Figure 8.5(a). The empty sphere corresponding to a Voronoi vertex is shown as well. Figure 8.5(b) gives a more complex example for a random set of balls. The inverted ball is hidden but the Voronoi edges defined by the ball are highlighted in red. Figure 8.5(c) shows a protein molecule with one inverted ball. Figure 8.5(d) is the model of a cow bounded by the inverted ball. Voronoi edges are hidden in this case.



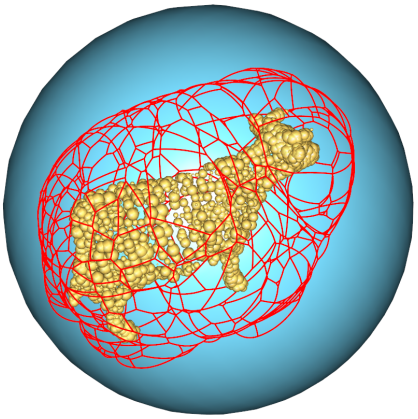
(a) A simple example of four balls, one of them is inverted.



(b) More ordinary balls. The inverted ball is not shown.



(c) An inverted ball and a protein molecule.



(d) A cow "outside" an inverted ball. Remaining Voronoi edges are hidden.

Figure 8.5: 3D Voronoi diagrams using inverted balls. Ordinary balls are yellow, inverted balls are blue. Voronoi edges defined by an inverted ball are red.

8.2 Fast Discovery of Voronoi Vertices using Delaunay Triangulation

This section gives a short overview of our paper [41], describing the idea how to make the edge-tracing algorithm fast. We will focus on the problem of finding the end-vertex for a given trajectory.

Recall Section 5.3 for the description of the edge-tracing algorithm and Figure 5.2 for the meaning of a trajectory and comparing the distance between its points.

The brute force end-vertex search makes the edge-tracing impractical due to the overall time complexity. For a trajectory, almost the entire input set of balls is investigated to find the fourth ball to define the end-vertex. But the search space can be reduced already from the knowledge of balls defining the trajectory by using some geometric filters called *feasible regions*. They were introduced by Cho et al. [11] for this purpose and enable the algorithm to compute some edges faster than $O(n)$. The method for searching the end-vertex used by [11] is the following. They create an outer approximation[1] of their feasible region. This approximation consists of a sphere and a halfspace. They search this space for the end-ball required to define the end-vertex, starting in the sphere and continue in the halfspace if necessary. They use a spatial grid, where balls are associated with grid cells. According to their experiments, the end-vertex is found in a constant time on average for proteins. Medvedev et al. [44] use a similar approach. The main problems with these grid-based approaches are the proper choice of grid resolution and the efficiency dependence on the uniformity of data. Therefore we came up with a different approach.

Our method uses the idea of feasible regions together with the idea of solving spherical range queries with 3D Delaunay triangulation. The first one is modified to work with the second, the second is simplified to our problem and we connect them together by using a simple breadth-first graph search running through edges of the triangulation.

The rest of this section is organized as follows. Feasible regions are generalized in Section 8.2.1. The problem is simplified in Section 8.2.2. A lemma fundamental to our algorithm is given in Section 8.2.3 and Section 8.2.4 is dedicated to the algorithm.

8.2.1 Feasible Region Relative to a Point

In our approach, we defined a *feasible region relative to a point* p, denoted as $R(p)$. Its two-dimensional analogy is shown in Figure 8.6. It is the union of spheres externally tangent to all gate-balls b_{gi}, with centers on the trajectory from the start-vertex v_s to the point p. The space of the sphere $s(v_s)$ centered at v_s is excluded because the sphere is always empty.

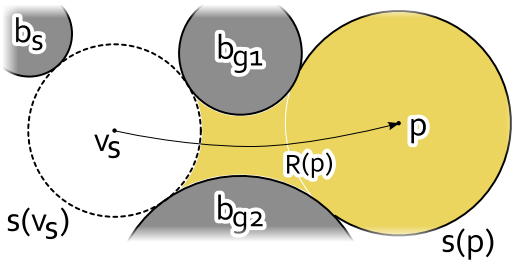


Figure 8.6: 2D analogy of the feasible region relative to a point p

The relation of $R(p)$ to the end-vertex search is the following. Suppose that p is a candidate to the end-vertex. If there is a ball intersecting the interior of $R(p)$, then the ball together with gate-balls can define a better end-vertex candidate, which is closer to v_s. Otherwise, the candidate is the real end-vertex.

[1] this is the same as our $\overline{R}(p_\infty)$ explained later in this chapter

8.2.2 Making Things Easier

In this section, the task of finding balls intersecting a relative feasible region $R(p)$ is simplified to finding points inside a sphere or a half-space. This is achieved by doing some outer approximations.

The simplification is illustrated in Figure 8.7. This is the case of a non-elliptic trajectory. The region $R(p)$ is approximated by the *first approximation*, denoted as $\overline{R}(p)$. It is the union of two spheres - one of them is defined only by the gate balls and the second one corresponds to the sphere at the point p, such as the $s(p)$ in Figure 8.6. When $p = p_\infty$, the second sphere is a halfspace. This is not possible for elliptic trajectories - in that case we approximate $R(p)$ for an arbitrary p by the minimum bounding sphere and denote it $\overline{R}(p)$, too. Furthermore, each ball b_i is approximated by a concentric ball $\overline{b}_i$ with the radius r_{max}, which is the radius of the maximal ball. Now we can collapse all $\overline{b}_i$ to their centers and expand $\overline{R}(p)$ by $(1 + \epsilon)r_{max}$. We denote the expanded approximations as $\overline{\overline{R}}(p)$ and call them the *second approximation*. The small $\epsilon > 0$ adjustment of r_{max} is for a robust inclusion of gate-balls.

Instead of inspecting balls intersecting $R(p)$, now it is sufficient to inspect balls whose centers are inside $\overline{\overline{R}}(p)$, i.e. to solve spherical (or half-space) queries against a set of points. The set of inspected balls will be a superset of the set we actually wanted, since we used the outer approximation.

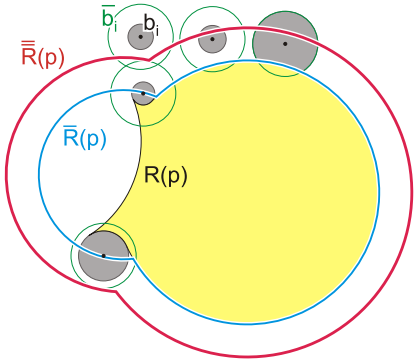


Figure 8.7: Balls b_i of different radii and their approximations $\overline{b}_i$; A relative feasible region $R(p)$ and its first approximation $\overline{R}(p)$ and second approximation $\overline{\overline{R}}_r(p)$. Any ball intersecting $R(p)$ has its center inside $\overline{\overline{R}}_r(p)$.

8.2.3 There is Always a Path

This section provides a property of Delaunay triangulation important for our approach. It is Lemma 8.2.1 which for a given sphere guarantees the existence of a path between any two Delaunay vertices inside the sphere using only the edges inside the sphere. Corollary 8.2.2 extends this guarantee for two spheres sharing a common Delaunay vertex.

By the term *open sphere* S we mean a sphere without its boundary δS. By the term $G[V']$ we mean a subgraph of $G = (V, E)$ induced by $V' \subseteq V$, i.e. a graph on nodes V' with edges $\{v_i, v_j\} \in E$ such that $v_i, v_j \in V'$. Nodes $V \cap S$ are the nodes from V intersecting S.

Lemma 8.2.1. *(Optimistic) Suppose that the $d-$dimensional Delaunay triangulation of a set of points V is given together with an arbitrary $d-$dimensional open sphere S. Let $G = (V, E)$ be a graph where nodes V are the Delaunay vertices and edges E are the edges of Delaunay simplexes. Then for each pair of nodes $u, v \in V \cap S$ there exists a path between u and v in $G[V \cap S]$.*

Proof. Omitted but can be found in our paper [41]. □

Note that if we allowed S in Lemma 8.2.1 to be (for example) a general ellipsoid instead of a sphere, we could not guarantee the existence of such a path.

An immediate corollary for two spheres S_1 and S_2 follows from Lemma 8.2.1 by using a transitivity of paths sharing a common node (i.e. a point inside $S_1 \cap S_2$).

Corollary 8.2.2. *If two open spheres S_1 and S_2 are given instead of S and if there is a node $w \in V \cap S_1 \cap S_2$ then for each pair of nodes $u, v \in V \cap (S_1 \cup S_2)$ there exists a path between u and v in $G[V \cap (S_1 \cup S_2)]$.*

Since a halfspace can be considered as a limiting case of a sphere, we can combine a sphere and a halfspace instead of two spheres and Corollary 8.2.2 still holds.

8.2.4 Finding the End-vertex

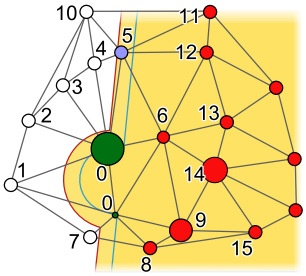
Here we briefly summarize our algorithm for finding the end-vertex for a trajectory. It requires Delaunay triangulation DT of ball centers. DT is a graph of Delaunay vertices connected by Delaunay edges. Each vertex corresponds to a ball. The graph is searched in a breadth-first manner (BFS) and the search is kept in the space bounded by an approximation of feasible region. This space is being reduced as the search progresses.

Algorithm *FindEndVertex*. It requires DT of ball centers and returns the end-vertex.

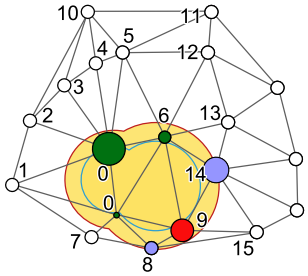
---

1. Initialize $p \leftarrow p_\infty$ for a non-elliptic trajectory or set $p \leftarrow$ the second vertex defined by the elliptic trajectory.
2. Compute initial $\overline{\overline{R}}(p)$.
3. BFS queue is empty. Enqueue the gate balls.
4. Start BFS. For each visited vertex:
   (a) (early-accept)
   if the ball is a gate ball, enqueue all neighbors and continue BFS.
   (b) (early-reject)
   if the center of the ball is outside $\overline{\overline{R}}(p)$, do **not** try to enqueue neighbors and continue BFS.
   (c) (early-accept)
   if the ball does not intersect $\overline{R}(p)$, enqueue all neighbors and continue BFS.
   (d) (end-vertex computation)
   otherwise compute a new end-vertex candidate p'. If p' is closer to the start vertex, set $p \leftarrow p'$, recompute $\overline{R}(p)$ and $\overline{\overline{R}}(p)$, enqueue neighbors and continue BFS.
5. return p as the end-vertex

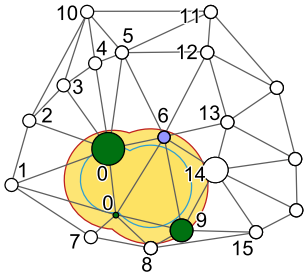
Figure 8.8 shows a 2D example of finding the end-ball for the end-vertex. There are balls from the input set with numbers meaning positions in the BFS queue. Balls numbered 0 are the gate-balls. The approximation of feasible region corresponding to these gate-balls is shown as well and used as a spatial filter. Figure 8.8(a) shows the initial situation, where the filter consists of a sphere and a half-space. All unvisited neighbors of the gate-balls are scheduled for visiting, i.e. enqueued at positions $1-9$, and the search starts. Because the balls $1-4$ have their centers outside the filter, they are early-rejected and hence their neighbors will not be scheduled for visiting. Corollary 8.2.2 guarantees that all balls with centers inside the filter will be still reachable by a path with nodes inside the filter - no candidate will be lost by the early reject. The ball 5 has its center inside the filter, but it cannot constitute any better end-vertex candidate. This ball is early-accepted, so its unvisited neighbors are scheduled as $10-12$. The ball 6 constitutes a candidate to the end-vertex, so the filter is recomputed and neighbors are scheduled as 13 and 14. Figure 8.8(b) shows the new situation where the ball 6 is a candidate to the end-ball. The ball 7 is early-rejected, the ball 8 is early-accepted and hence its remaining neighbor is scheduled as 15. The ball 9 constitutes a better end-vertex candidate. The filter is recomputed and the new situation is shown in Figure 8.8(c). All remaining balls $10-15$ are early-rejected and the queue is empty. The search is finished with the ball 9 as the end-ball constituting the end-vertex.



(a) The neighbors of gate balls are scheduled for visiting and get numbers $1-9$. The balls $1-4$ have centers outside the filter, so they are early-rejected. The ball 5 is early-accepted and hence its neighbors are scheduled as $10-12$. The ball 6 will be the first end-vertex candidate, its neighbors are scheduled as 13 and 14.



(b) A new filter is created from the ball 6. The ball 7 is early-rejected, 8 is early-accepted and hence its remaining neighbor is scheduled as 15. The ball 9 will be the second end-vertex candidate.



(c) A new filter is created from the ball 9. The balls $10-15$ are skipped by the early-rejection test, the queue becomes empty and the ball 9 becomes the final end-ball.

Figure 8.8: A 2D example of finding the end-ball to define the end-vertex. Delaunay triangulation is built over the ball centers. Numbers denote the scheduled positions of balls in a BFS queue during the graph search. White balls are the balls for early-reject, blue are for early-accept, red are for vertex candidates computation, green are the candidates for the final result.

The correctness of the algorithm is guaranteed by Corollary 8.2.2 - all balls intersecting each computed $R(p)$ can be visited by the BFS.

The worst case time complexity is dominated by the BFS. It is $O(m+n)$ for m Delaunay edges and n vertices and corresponds to the case when the spatial filters are not efficient, such as when there is a great ball among many small balls. The filters can be efficient when the maximal difference in the radii of balls is small. Protein molecules have this property and experiments showed that the time complexity is $O(1)$ on average in this case.

8.2.5 Results

This section shows some of our results regarding the expected time complexity of the algorithm.

Figure 8.9 shows how much time it takes to compute the diagram for molecules with different numbers of atoms. Our edge-tracing implementation used a brute force end-vertex search in the first case and the filtering approach using our relative feasible regions searched via 3D Delaunay triangulation in the second case. The second case includes the overhead of computing the triangulation using [57]. Time is measured in CPU seconds of Intel$^{(R)}$ Core$^{(TM)}$2 Quad CPU 2.4 GHz, 4 GB RAM. Figure 8.9 only shows that the improvement to the expected running time is significant.

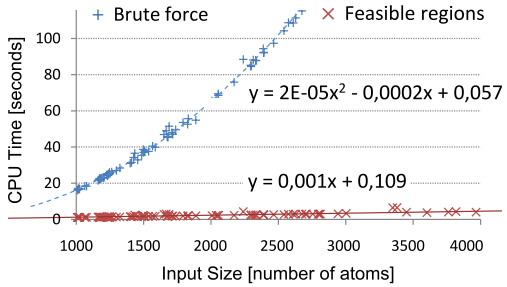
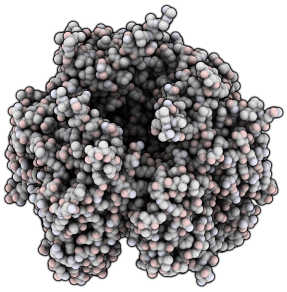
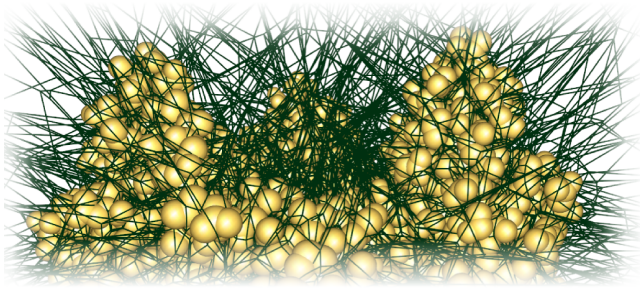


Figure 8.9: Running time of our edge-tracing implementation using two different approaches of searching end-vertices.

Figure 8.10 shows one particular result for a large protein (about 9000 atoms) with PDB ID 1HX6 visualized by QuteMol [50]. From Figure 8.10(a) it is apparent that the distribution of atoms is non-uniform: There is a large hole in the molecule (compared to the size of its atoms) and several clefts are visible as well. A part of the corresponding Voronoi diagram computed and visualized by our program is shown in Figure 8.10(b). Thanks to our speed-up approach, the computation took only about 11.2 CPU seconds and the average work per edge corresponds to the trend measured for (hundreds of) proteins.



(a) The protein 1HX6. Note the distribution of atoms - it is non-uniform. The molecule is hollow and has clefts. Rendered by QuteMol [50].



(b) The diagram of 1HX6 has about 61000 vertices, 122000 edges (only 10 of them are elliptic). The computation took 11.2 CPU seconds. The average number of vertex candidates per one edge was 7.6, only 2.7 of them were improving and the average number of early rejects with early accepts was about 51.

Figure 8.10: The protein 1HX6 with about 9000 atoms of radii ranging from 1.52 to 1.8 A.

Chapter 9

Future Work and Conclusion

This chapter shows the directions of our future research divided into three areas, namely the area of applications, geometric aspects and algorithmic aspects, and concludes this thesis.

9.1 Applications

We want to use the geometric construct of Voronoi diagram of 3D balls in related problems from biochemistry in the context of a common project with the computer graphics group at Masaryk University. This group of researchers also provides us with a connection to the area of biochemistry. One of the great points of their research is the study of empty channels in a molecule, allowing a smaller molecule (ligand) to get to the active site inside a protein (receptor). Protein molecules are dynamic - the atoms can move in time - so the channel is also dynamic, i.e., it can change in time, it can even disappear or switch to another channel.

Our intended applications are:

- Channels - the computation of an ideal channel, at least for the purpose of being a reference to evaluate the approximated solutions used so far.
- Cavities and pockets - their identification in molecules, using parameters such as the radius of a sphere which has to fit in. It could be interesting to explore these things along a channel, i.e., locally. Existing approaches work globally in the context of the whole molecule.
- Level of protein detail - large systems of atoms are computationally expensive. A hierarchical approach based on spheres can save the computational effort.
- Volume and surface computation - for molecules, pockets, cavities and channels. The volume and surface of a channel computed in a time interval can give useful information about its development in this interval.

To the best of our knowledge in 2010, although these approaches are already known or under development, they have not been used in the dynamic setting so far.

9.2 Geometric Aspects

The geometry of elements of a Voronoi diagram of balls is still the area of our interest. How they behave under a spherical inversion? We know that inversions have been used in 2D for the computation of vertices, but we are not aware of any higher-dimensional approach. Another idea is the filling of the

empty space in a similar way to the Apollonian gasket, which is a space filling fractal constructed from several spheres by adding empty tangent spheres. Maybe this will not have any practical application, but it is definitely an interesting area to be explored.

9.3 Algorithmic Aspects

The construction of Voronoi diagram of balls is still more computationally intensive than in the case of points. There are several ways to explore how to make this construction even faster. We see the following options:

- Our method for a fast discovery of Voronoi vertices described in Section 8.2 should be improved to cope with greater difference in radii. The approach of covering a large ball by a number of smaller balls [56] seems to be an attractive option and worth to explore.
- Use two levels of hierarchy. At the low level, a brute force end-vertex search could be used, solved quickly on the hardware, e.g., on CUDA, using the power of massive parallelism. When the original input set of balls would be divided into clusters, these clusters would be contained in larger balls. These larger balls then form the higher level of the hierarchy and our method from Section 8.2 could be used at this level.
- The edge-tracing algorithm maintains a list of trajectories to be traced. At least half of them is unique and it might be interesting to trace them in parallel.
- Utilize some divide-and-conquer approach. When the balls are divided into independent clusters, the diagram in these clusters can be computed in parallel. Each cluster can be bound by our inverted ball from Section 8.1. After the diagram for each cluster is computed, inverted balls can be removed and the remaining trajectories traced. The motivation is that many balls deep inside clusters can be considered finished,i.e., their Voronoi cells are completed, and can be left out from the remaining computation.
- An interesting approach could be to use the four-dimensional power diagram as discussed in Section 3.5. The neighbors of a power cell are a superset of neighbors in the Voronoi diagram of balls. It might pay off to use this information in the edge-tracing algorithm. A trajectory is defined by three gate balls and the end-ball will be in the common intersection of the corresponding three sets of neighbors.

Another challenge is the insertion and removal of a ball for an existing three-dimensional diagram. To the best of our knowledge, only the incremental algorithm of Boissonnat and Karavelas [6] seems to be able to insert or remove a ball. Would it be possible to use the edge-tracing algorithm for this task? This could turn out to be useful in the case when just a small part of a protein is substituted for something else, e.g., an amino-acid can be changed for another amino-acid or for some of its conformal variant in order to open a channel to the active site and keep the function of a protein.

A great challenge is the maintenance of the diagram when the balls are allowed to move in time, such as in the case of a dynamic protein. When the changes in positions are sufficiently small, the topology does not change. But when they are bigger, the topology can change locally or globally. For local changes it might pay off to use the diagram from the previous frame and modify it to get the new diagram. But when the changes are global, it would be reasonable to compute the new diagram from

scratch. We know about only one existing approach by Gavrilova and Rokne [17], but this is for the continuous case and needs to be solved numerically.

9.4 Conclusion

This work gave an insight to the historical background of spatial tessellation and described related concepts used in the modeling of molecules, namely the Voronoi diagram of 3D balls, the quasi-triangulation, beta-shapes and beta-complexes. As of 2010, these concepts are still under active research. This work also summarized algorithms for their construction. Our contribution to this field from the geometrical point of view is the concept of an inverted ball and its use as a boundary constraint, introducing ellipsoidal faces to the diagram, and from the algorithmic point of view, it is the method of fast construction for Voronoi diagrams of protein molecules using Delaunay triangulation of atom centers. We have also shown future directions of our research and pointed out some of our new ideas we would like to explore.

Activities

Reviewed Publications

- M. Maňák and I. Kolingerová. Inverted ball as boundary-constraint for Voronoi diagrams of 3D spheres. In *25th European Workshop on Computational Geometry*, pages 289–292, Brussels, Belgium, March 2009. Universite Libre de Bruxelles.
- M. Maňák and I. Kolingerová. Fast discovery of Voronoi vertices in the construction of Voronoi diagram of 3D balls. Accepted for publication: IEEE Computer Society, International Symposium on Voronoi Diagrams in Science and Engineering, 2010.

Student Publications

- M. Maňák. Voronoi diagrams for spheres in E^3. Master thesis, Charles University in Prague, September 2008

Related Talks

- M. Maňák. Medial Axis. University of West Bohemia in Pilsen, Czech Republic, May 2010
- M. Maňák. Voronoi diagramy pro množinu koulí. University of West Bohemia in Pilsen, Czech Republic, December 2008

Participations on Scientific Projects

- Triangulated Models for Haptic and Virtual Reality. Project leader Ivana Kolingerová.
  Funded by The Czech Science Foundation (GACR), project No. GA 201/09/0097, 2009.
- Analysis and Visualization of Protein Structures. Project leader Jiří Sochor.
  Funded by The Czech Science Foundation (GACR), project No. P202/10/1435, 2010.

Bibliography

[1] O. Aichholzer, F. Aurenhammer, T. Hackl, B. Kornberger, M. Peternell, and H. Pottmann. Approximating boundary-triangulated objects with balls. In *Proc. 23rd European Workshop on Computational Geometry*, pages 130–133. TU Graz, 2007.

[2] F. Aurenhammer. Power diagrams: properties, algorithms and applications. *SIAM J. Comput.*, 16(1):78–96, 1987.

[3] J. D. Bernal. A geometrical approach to the structure of liquids. *Nature*, 183(4655):141–147, January 1959.

[4] J. D. Bernal and J. L. Finney. Random close-packed hard-sphere model. II. geometry of random packing of hard spheres. *Discussions of the Faraday Society*, 43:62, 1967.

[5] J.-D. Boissonnat and C. Delage. Convex hull and Voronoi diagram of additively weighted points. In *ESA*, pages 367–378, 2005.

[6] J.-D. Boissonnat and M. I. Karavelas. On the combinatorial complexity of Euclidean Voronoi cells and convex hulls of d-dimensional spheres. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 305–312, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[7] J.-D. Boissonnat and M. Yvinec. *Algorithmic geometry*. Cambridge University Press, New York, NY, USA, 1998.

[8] A. Bondi. van der Waals Volumes and Radii. *Journal of Physical Chemistry*, 68(3):441–451, March 1964.

[9] G. Bradshaw and C. O'Sullivan. Adaptive medial-axis approximation for sphere-tree construction. *ACM Trans. Graph.*, 23(1):1–26, 2004.

[10] Y. Cho, D. Kim, and D.-S. Kim. Topology representation for the Voronoi diagram of 3D spheres. *International Journal of CAD/CAM*, 5(1):59–68, 2005.

[11] Y. Cho, D. Kim, H.-C. Lee, J. Y. Park, and D.-S. Kim. Reduction of the search space in the edge-tracing algorithm for the Voronoi diagram of 3D balls. In *ICCSA (1)*, pages 111–120, 2006.

[12] M. L. Connolly. Analytical molecular surface calculation. *Journal of Applied Crystallography*, 16(5):548–558, October 1983.

[13] H. Edelsbrunner, M. Facello, P. Fu, and J. Liang. Measuring proteins and voids in proteins. *Hawaii International Conference on System Sciences*, 0:256, 1995.

[14] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Trans. Graph.*, 13(1):43–72, 1994.

[15] M. Gavrilova. A reliable algorithm for computing the generalized Voronoi diagram for a set of spheres in the Euclidean d-dimensional space.

[16] M. Gavrilova. *Proximity and Applications in General Metrics*. PhD thesis, The University of Calgary, Dept. of Computer Science, Calgary, AB, Canada, May 1998.

[17] M. L. Gavrilova and J. Rokne. Updating the topology of the dynamic Voronoi diagram for spheres in Euclidean d-dimensional space. *Computer-Aided Geometric Design*, 20:231–242, 2003.

[18] B. J. Gellatly and J. L. Finney. Calculation of protein volumes: an alternative to the Voronoi procedure. *J Mol Biol*, 161(2):305–322, October 1982.

[19] D. Gish and J. M. Ribando. Apollonius problem: A study of solutions and their connections. *American Journal of Undergraduate Research*, 3(1), 2004.

[20] A. Goede, R. Preissner, and C. Frömmel. Voronoi cell: New method for allocation of space among atoms: Elimination of avoidable errors in calculation of atomic volume and density. *Journal of Computational Chemistry*, 18(9):1113–1123, 1997.

[21] M. G. Jerry, J. Tsai, and M. Levitt. The volume of atoms on the protein surface: Calculated from simulation, using Voronoi polyhedra, 1995.

[22] C.-M. Kim, C.-I. Won, Y. Cho, D. Kim, S. Lee, J. Bhak, and D.-S. Kim. Interaction interfaces in proteins via the voronoi diagram of atoms. *Comput. Aided Des.*, 38(11):1192–1204, 2006.

[23] C.-M. Kim, C.-I. Won, J. Ryu, J.-K. Kim, J. Bhak, and D.-S. Kim. Protein-ligand docking based on β-shape. *International Symposium on Voronoi Diagrams in Science and Engineering*, 0:245–253, 2009.

[24] D. Kim, Cho, and D.-S. Kim. Region expansion by flipping edges for Euclidean Voronoi diagrams of 3D spheres based on a radial data structure. In *ICCSA (1)*, pages 716–725, 2005.

[25] D. Kim and D.-S. Kim. Region-expansion for the Voronoi diagram of 3D spheres. *Computer-Aided Design*, 38:417–430, 2006.

[26] D. Kim, D.-S. Kim, and K. Sugihara. Euclidean Voronoi diagram for circles in a circle. *Int. J. Comput. Geometry Appl.*, 15(2):209–228, 2005.

[27] D.-S. Kim. A single beta-complex solves all geometry problems in a molecule. *International Symposium on Voronoi Diagrams in Science and Engineering*, 0:254–260, 2009.

[28] D.-S. Kim, C.-H. Cho, D. Kim, and Y. Cho. Recognition of docking sites on a protein using β-shape based on voronoi diagram of atoms. *Comput. Aided Des.*, 38(5):431–443, 2006.

[29] D.-S. Kim, Y. Cho, and D. Kim. Edge-tracing algorithm for Euclidean Voronoi diagram of 3D spheres. In *Proceedings of the 16th Canadian Conference on Computational Geometry (CCCG'04)*, pages 176–179, 2004.

[30] D.-S. Kim, Y. Cho, and D. Kim. Euclidean Voronoi diagram of 3D balls and its computation via tracing edges. *Computer-Aided Design*, 37:1412–1424, 2005.

[31] D.-S. Kim, Y. Cho, and K. Sugihara. Quasi-worlds and quasi-operators on quasi-triangulations. Preprint submitted to Elsevier, January 2010.

[32] D.-S. Kim, Y. Cho, K. Sugihara, J. Ryu, and D. Kim. Three-dimensional beta-shapes and beta-complexes via quasi-triangulation, 2010. Revision under review.

[33] D.-S. Kim, D. Kim, and Y. Cho. Euclidean Voronoi diagrams of 3D spheres: Their construction and related problems from biochemistry. In *IMA Conference on the Mathematics of Surfaces*, pages 255–271, 2005.

[34] D.-S. Kim, D. Kim, Y. Cho, and K. Sugihara. Quasi-triangulation and interworld data structure in three dimensions. *Computer-Aided Design*, 38(7):808–819, 2006.

[35] D.-S. Kim, B. Lee, C. I. Won, D. Kim, J. Ryu, Y. Cho, C.-M. Kim, S. Lee, and J. Bhak. Multi-resolution protein model. In *ICCSA (2)*, pages 639–652, 2007.

[36] D.-S. Kim, J. Seo, D. Kim, J. Ryu, and C.-H. Cho. Three-dimensional beta shapes. *Computer-Aided Design*, 38(11):1179 – 1191, 2006.

[37] B. Lee and F. Richards. The interpretation of protein structures: Estimation of static accessibility. *Journal of Molecular Biology*, 55(3):379 – 400, IN3–IN4, 1971.

[38] S. H. Lee and K. Lee. Partial entity structure: a compact non-manifold boundary representation based on partial topological entities. In *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 159–170, New York, NY, USA, 2001. ACM.

[39] V. A. Luchnikov, N. N. Medvedev, L. Oger, and J.-P. Troadec. Voronoi-Delaunay analysis of voids in systems of nonspherical particles. *Phys. Rev. E*, 59(6):7205–7212, June 1999.

[40] M. Manak and I. Kolingerova. Inverted ball as boundary-constraint for Voronoi diagrams of 3D spheres. In *25th European Workshop on Computational Geometry*, pages 289–292, Brussels, Belgium, March 2009. Universite Libre de Bruxelles.

[41] M. Manak and I. Kolingerova. Fast discovery of Voronoi vertices in the construction of Voronoi diagram of 3D balls. Accepted for publication: IEEE Computer Society, International Symposium on Voronoi Diagrams in Science and Engineering, 2010.

[42] B. McConkey, V. Sobolev, and M. Edelman. Quantification of protein surfaces, volumes and atom-atom contacts using a constrained Voronoi procedure. *Bioinformatics*, 18(10):1365–1373, 2002.

[43] P. Medek, P. Beneš, and J. Sochor. Computation of tunnels in protein molecules using Delaunay triangulation. *Journal of WSCG*, 15(1–3):107–114, January 2007.

[44] N. N. Medvedev, V. P. Voloshin, V. A. Luchnikov, and M. L. Gavrilova. An algorithm for three-dimensional Voronoi s-network. *Journal of Computational Chemistry*, 27(14):1676–1692, 2006.

[45] A. Mirzaian. Minimum weight Euclidean matching and weighted relative neighborhood graphs. In *WADS '93: Proceedings of the Third Workshop on Algorithms and Data Structures*, pages 506–517, London, UK, 1993. Springer-Verlag.

[46] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley series in probability and statistics. Wiley, second edition, 2000.

[47] F. M. Richards. The interpretation of protein structures: Total volume, group volume distributions and packing density. *Journal of Molecular Biology*, 82(1):1 – 14, 1974.

[48] J. Ryu, R. Park, and D.-S. Kim. Molecular surfaces on proteins via beta shapes. *Computer-Aided Design*, 39(12):1042 – 1057, 2007.

[49] J. Seo, Y. Cho, D. Kim, and D.-S. Kim. An efficient algorithm for three-dimensional β-complex and β-shape via a quasi-triangulation. In *SPM '07: Proceedings of the 2007 ACM symposium on Solid and physical modeling*, pages 323–328, New York, NY, USA, 2007. ACM.

[50] M. Tarini, P. Cignoni, and C. Montani. Ambient occlusion and edge cueing for enhancing real time molecular visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1237–1244, 2006. `http://qutemol.sourceforge.net`.

[51] A. Varshney, F. P. Brooks Jr, D. Manocha, W. V. Wright, and D. C. Richardson. Defining, computing, and visualizing molecular interfaces. In *VIS '95: Proceedings of the 6th conference on Visualization '95*, page 36, Washington, DC, USA, 1995. IEEE Computer Society.

[52] G. Voronoi. Nouvelles applications des paramétres continus á la théorie des formes quadratiques. *Journal fur die Reine and Angewandte Mathematic*, 133:97–178, 1907.

[53] R. Wang, K. Zhou, J. Snyder, X. Liu, H. Bao, Q. Peng, and B. Guo. Variational sphere set approximation for solid objects. *The Visual Computer*, 22(9):612–621, 2006.

[54] K. Weiler. The radial edge structure: A topological representation for non-manifold geometric boundary modeling. In *Geometric Modeling for CAD Applications*, pages 3–36, New York, NY, USA, 2001. ACM.

[55] H. M. Will. Computation of additively weighted Voronoi cells for applications in molecular biology, 1999.

[56] E. Yaffe and D. Halperin. Approximating the pathway axis and the persistence diagram of a collection of balls in 3-space. In *SCG '08: Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 260–269, New York, NY, USA, 2008. ACM.

[57] M. Zemek and I. Kolingerova. A library for the construction of a dynamic regular and Delaunay tetrahedronization, 2009. http://www.kiv.zcu.cz/en/research/software/detail.html?id=39.