Diss. ETH No. 13188

Computation of Additively Weighted Voronoi Cells for Applications in Molecular Biology

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZURICH

for the degree of
Dr. sc. techn.

presented by
HANS-MARTIN WILL
Dipl. Math. University of Bonn
born January 6th, 1970
citizen of the Federal Republic of Germany

accepted on the recommendation of
Prof. Dr. E. Welzl, examiner
Prof. Dr. H. Edelsbrunner, co-examiner
Prof. Dr. S. Wodak, co-examiner

1999

Abstract

This thesis is concerned with the design and implementation of an efficient algorithm for the computation of additively weighted Voronoi (AWV) cells for applications in molecular biology, namely volume and density calculations of atoms and amino acid residues. An AWV cell of a sphere σ out of a collection S of spheres describes the nearest neighborhood of σ with respect to all the other spheres in S. To our knowledge, this is the first implementation of an algorithm computing these cells that is suited for practical application.

We begin by studying the geometric and combinatorial properties of AWV cells. We show that an AWV cell can be conveniently described using a spherical subdivision data structure, where each edge corresponds to a circular arc. It is also shown that the best previously known upper bound on the worst-case complexity of one such cell defined by n other spheres, which is $\Theta(n^2)$, is tight. Based on these insights, we present a new randomized incremental algorithm computing one such cell amidst n other spheres in expected time $O(n^2 \log n)$, which is optimal up to a logarithmic factor. However, the experimentally observed behavior of the complexity of those cells arising in the intended domain of application is linear in n. In this case our algorithm performs the task in expected time $O(n \log^2 n)$.

We implemented a variant of this algorithm and took care to provide a robust implementation. Robustness is ensured by application of methods for dynamic error analysis at runtime that trigger numerical perturbations. The empirical behavior of this implementation on real data sets is studied, both from the point of view of robustness and computational resources required.

Finally, we demonstrate the benefits of using AWV cells for volume computations in molecules compared to methods based on convex polyhedra that have been proposed previously.

Zusammenfassung

In dieser Arbeit beschreiben wir den Entwurf und die Implementation eines effizienten Algorithmus für die Berechnung additiv gewichteter Voronoizellen (AWV-Zellen). Diese Implementation zielt auf molekularbiologische Anwendungen ab, namentlich die Berechnung von Volumina und Dichten einzelner Atome und Aminosäurereste. Die AWV-Zelle einer Kugel σ aus einer Menge S von Kugeln beschreibt die nächste Nachbarschaft von σ bezüglich S. Nach unserem Wissen ist dies die erste Implementierung eines solchen Algorithmus, die für den praktischen Einsatz tauglich ist.

Wir beginnen damit, die geometrischen und kombinatorischen Eigenschaften von AWV-Zellen zu studieren. Wir zeigen, dass eine AWV-Zelle sich durch eine Unterteilung einer Kugeloberfläche darstellen lässt, wobei jede Kante in dieser Unterteilung einem Kreisabschnitt entspricht. Weiterhin zeigen wir, dass die bisher beste bekannte obere Schranke auf die schlimmstmögliche Komplexität einer durch n Kugeln definierten Zelle, die $\Theta(n^2)$ ist, scharf ist. Aufbauend auf diesen Erkenntnissen stellen wir einen randomisierten Algorithmus vor, der eine solche Zelle, die durch n Kugeln definiert ist, in $O(n^2 \log n)$ erwarteter Zeit berechnet,was bis auf einen logarithmischen Faktor optimal ist. Jedoch ist das experimentell beobachtete Verhalten der Komplexität dieser Zellen im Bereich der geplanten Anwendung linear in n. In diesem Fall berechnet der neue Algorithmus eine solche Zelle in $O(n \log^2 n)$ erwarteter Zeit.

Wir implementierten eine Variante dieses Algorithmus und legten dabei Wert darauf, eine robuste Implementierung zur Verfügung zu stellen. Um Robustheit zu gewährleisten, benutzen wir Techniken zur dynamischen Fehlerkontrolle zur Laufzeit, die geeignete numerische Perturbationen auslösen. Das empirische Verhalten dieser Implementierung wurde anhand realer Daten studiert, sowohl unter dem Gesichtspunkt der Robustheit als auch der benötigten Laufzeit.

Schliesslich demonstrieren wir die Vorteile von AWV-Zellen zur Volumenberechnung in Molekülen und vergleichen dies mit früheren Methoden, die auf konvexen Polyedern basieren.

Acknowledgements

This thesis would not exist without the support and contributions by my teachers, colleagues and friends. The idea of writing a thesis on the application of Voronoi diagrams to geometric problems arising in molecular biology originates from discussions with B. Korte at the Research Institute of Discrete Mathematics in Bonn. The research group of E. Welzl at ETH Zurich provided the inspiring environment to concentrate on this topic. Emo's influence on this thesis is ubiquitous, and his aim for structural simplicity served as a driving force to carve out the essential. Besides, I am especially grateful to my colleagues L. Kettner and J. Giesen for both friendship and many valuable discussions.

An invitation by the research group of C. Frömmel at the Humboldt University Berlin helped to select the right biological applications, and the discussions with J. Fauck, R. Preißner and A. Goede initiated the idea of calculating atomic volumes using AWV cells. These studies, in turn, would not have been possible without the kind support of the research group of S. Wodak at the Free University of Brussels and the EBI in Cambridge, where I. Coutinho and L. Wernisch helped me in setting up the experiments. Shoshana's guidance and experience kept me from getting lost in all those many numbers when preparing the results of chapter 6. Finally, in the end game, H. Edelsbrunner accepted to examine and report on this thesis even though he was literally on the move, and the final schedule turned out to be rather tight.

Last but not least, I would like to thank my parents. Without their constant encouragement and support this thesis would not have been possible.

Brussels
October 1999

Contents

Chapter 1

Introduction

Proteins comprise more than 50% of the dry weight of most biological species, and virtually any property that characterizes a living organism is controlled or affected by its constituent proteins. Built essentially out of a set of 20 different building blocks, the amino acids, proteins catalyze chemical reactions as enzymes, fight against intruders in the form of antibodies, or they show up as purely structural material as can be found in our nails, hair, skin, or bones. This tremendous diversity is possible because proteins form large and complicated macromolecules built of up to several thousands of atoms. From this perspective, it should come as no surprise that the subject of modern biochemistry is essentially the study of the roles and interactions of proteins in living systems. This core knowledge, in turn, forms the foundation of several applied disciplines collectively known as *molecular life sciences*, such as molecular medicine and rational drug design, to name only two prominent representatives.

The specific role and function of a protein within an organism is governed by its unique stereo-chemical properties[1]. These properties not only comprise purely quantum-chemical features such as polarity or valences, but also include the geometric configuration of the individual atoms in space. Therefore, an important step in the investigation of a protein is the determination of its three dimensional structure. This is typically done using X-ray diffraction or NMR spectroscopy techniques. The result of these studies and the subsequent refinement steps is a set of coordinates for all constituent atoms of the molecule.

To fully understand the reaction of, say, an enzyme and its substrate molecule

[1] Creighton (1993) or Kyte (1995) are general references for this chapter.

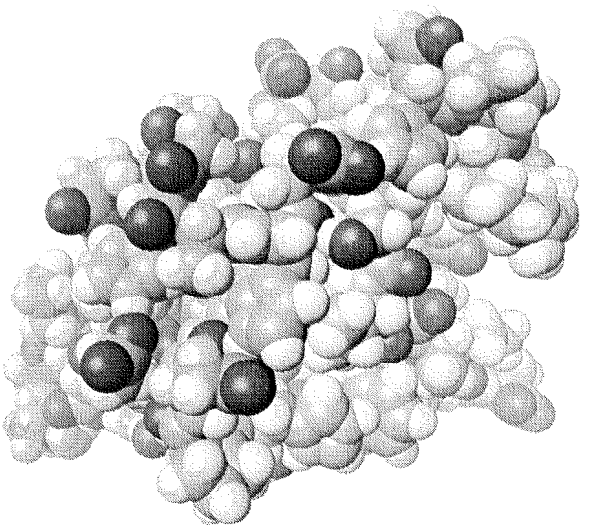


Figure 1.1: The *hard sphere model* of a small molecule involved in electron transfer (iron-sulfur protein, PDB entry 1pij).

at its most profound level would require a detailed quantum-chemical model of the two molecules involved. Though the required theory is well understood[2], the simulation of these detailed models on a computer is by far out of reach for the next decades to come[3]. Hence, to obtain computationally feasible models of the biochemical processes of interest, these models have to be simplified significantly.

A very popular model is the *van der Waals* (VDW) or *hard-sphere* model. In this model, each atom is represented as a sphere in space whose radius is chosen depending on the element or hybridization type of the atom. Since spheres are fundamental and well-understood geometric objects, the van der Waals model has been applied extensively for geometric considerations on macromolecular structures. See figure 1.1 for an illustration. In this thesis, we restrict ourselves to purely geometric problems, and the van der Waals model is chosen as foundation.

This choice is not without problems: Van der Waals radii are not available for all chemical elements, and, even worse, there exist several different assignments of radii to atomic types. To complicate things, sometimes a *united-atom approach* is used. In this approach, certain atoms are assigned a larger radius with the inten-

[2]Cf. Atkins and Friedman (1997) or Szabo and Ostlund (1996)

[3]Cf. Doucet and Weber (1996)

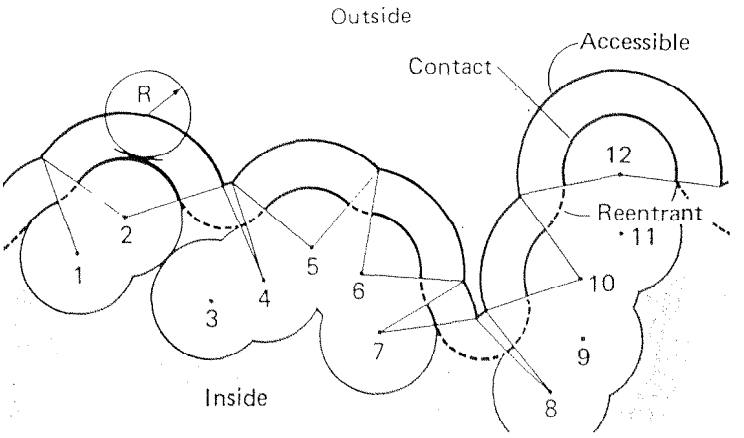


Figure 1.2: The *molecular surface* is obtained by rolling a probe sphere of a specified radius R over the atomic spheres. The molecular surface comprises the contact surface and the reentrant surface. The contact surface is the subset of the union of the surfaces of the spheres that can be touched by the probe sphere. The reentrant surface is a part of the probe sphere surface that fills canyons unreachable to the probe sphere. The *solvent accessible surface* is the surface ruled out by the center of the probe sphere in this process. Picture taken from Creighton (1993).

tion to represent an atomic group, such as the carbon atom representing a methyl group. This variant is commonly used in conjunction with atomic coordinate sets gained from X-ray diffraction patterns, which do not yield reliable information about the location of hydrogen atoms. Hence, it is always necessary to tabulate the specific set of radii used when reporting any kind of experimental result or conclusion.

A concept tightly related to the hard-sphere model is the *molecular surface*, which is also known as Richards (1977) or Connolly (1983) surface. For an illustration of this concept see figure 1.2. In mathematical terms, this surface is defined as follows: Let V denote the union of the volumes of the atomic spheres comprising the molecule. Let

$$A_r = \{x \in \mathbf{R}^3 : B_r(x) \cap V = \emptyset\},$$

where $B_r(x)$ denotes a probe ball of radius r centered at x. Then the *molecular surface* with respect to a probe sphere of radius r is given as the boundary set

$$\partial \bigcup_{x \in A_r} B_r(x).$$

The set ∂A_r is also known as the *solvent accessible surface* of the molecule. There exist many different implementations of algorithms for computing these surfaces by Connolly (1981, 1983), Alard and Wodak (1991), Varshney et al. (1994), Sanner et al. (1995), Akkiraju (1996), to name only a few. The computation of these surfaces is closely related to the power diagram and to the concept of weighted α-shapes, see Edelsbrunner (1992).

1.1 Tessellations of space

When the first crystal structures of proteins became available, it was a surprising to what extent the interior of globular proteins is packed. According to the calculations by Richards (1974), about 75% of the interior volume is filled with atoms, as defined by their van der Waals radii. This value has to be compared with the corresponding value obtained for an optimal packing of identical spheres, which is approximately 74%. Since proteins are synthesized as polymeric chain molecules, a complicated folding process is necessary in order to arrange all atoms into the right configuration. This folding process is driven by the laws of statistical mechanics, and it is the common belief that the primary structure, i.e. the specific sequence of amino acids along the polymeric chain, determines the three dimensional structure of the molecule[4]. Subsequent studies revealed that the aforementioned tight packing is a key factor in the folding process of globular proteins. Since we are at the atomic level, quantifying packing densities essentially means to partition the space occupied by a molecule among its constituent atoms. The reader is referred Gerstein and Chothia (1996) for further motivations to perform these calculations. Most notably, Pontius et al. (1996) propose computing these volumes as a method of quality assessment of experimentally gained crystal structures.

Atomic volumes in proteins were first modeled using Voronoi (1908) tessellations by Richards (1974, 1977) and Finney (1975). For $x, y \in \mathbf{R}^d$, let $d(x,y) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$ denote the Euclidean distance between x and y. Then the Voronoi tessellation of a set of points $P = \{p_i,\ 1 \leq i \leq n\}$ can be defined as follows:

Definition 1 (Voronoi tessellation) *Let $n \in \mathbf{N}$, $P = \{p_i,\ 1 \leq i \leq n\}$, $p_i \in \mathbf{R}^d$. Then*

[4] A few proteins, collectively known as *chaperones*, have been identified that might help other proteins in finding the correct folding. However, since chaperones are proteins themselves, this does not really provide any new aspect to the problem.

the collection $\mathbf{V} = \mathbf{V}(P)$ *of all non-empty sets*

$$V_i = V(p_i) = \left\{ x \in \mathbf{R}^d : \; d(x,p_i) < d(x,p_j) \; \forall 1 \leq j \leq n, j \neq i \right\}$$

is called the Voronoi tessellation *induced by P. For each* $1 \leq i \leq n$*, the set* V_i *is called the (unweighted)* Voronoi cell *of the point* p_i *with respect to P.*

We will also refer to the cell V_i as the cell *defined by* p_i. In the original Voronoi method, which Richards also called *method A*, the Voronoi cells of the centers of the atomic spheres are computed, and the volume assigned to each atom is the volume of its Voronoi cell. As it is well known, Voronoi cells are convex polyhedra, and the cell V_i, $1 \leq i \leq n$, can be written as the common intersection of a set H_i of open *halfspaces*

$$\begin{aligned} V_i &= \bigcap_{h \in H_i} h, \text{ where} \\ H_i &= \{h_{ij}, 1 \leq j \leq n, j \neq i\}, \end{aligned}$$

and for each $1 \leq j \leq n, j \neq i$ the halfspace h_{ij} is defined by

$$h_{ij} = \left\{ x \in \mathbf{R}^d : \; 2\langle p_j - p_i, x \rangle < \|p_j - p_i\|^2 \right\}.$$

Here, $\langle x,y \rangle = \sum_{i=1}^d x_i y_i$ denotes the scalar product, and $\|x\|^2 = \langle x,x \rangle$. We will also use the short-hand $x^2 = \|x\|^2$. However, treating all atoms equally tends to chemically misallocate volume between atoms of *a priori* different size (e.g. carbon and oxygen). Two previous approaches try to consider the different atomic radii by using different rules for the placement of the bisecting planes ∂h_{ij} between the cells.

Already in his initial study, Richards (1974) proposed a *method B* that places the bisecting planes based on chemical reasoning[5]. Again, let $P = \{p_i \in \mathbf{R}^3, \; 1 \leq i \leq n\}$ denote a set of atomic coordinates, and let $\{r_i : \; r_i \geq 0, 1 \leq i \leq n\}$ denote the corresponding atomic radii. As before, we assign to each atom p_i a polyhedron that is given as the common intersection of a set of open halfspaces $H_i = \{h_{ij}, 1 \leq j \leq n, j \neq i\}$. Depending on whether c_i and c_j are covalently bonded or not, the halfspace h_{ij} is defined as follows:

[5]In fact, Richards originally proposed to represent aromatic rings by ellipsoids. However, this variant is very seldomly used.

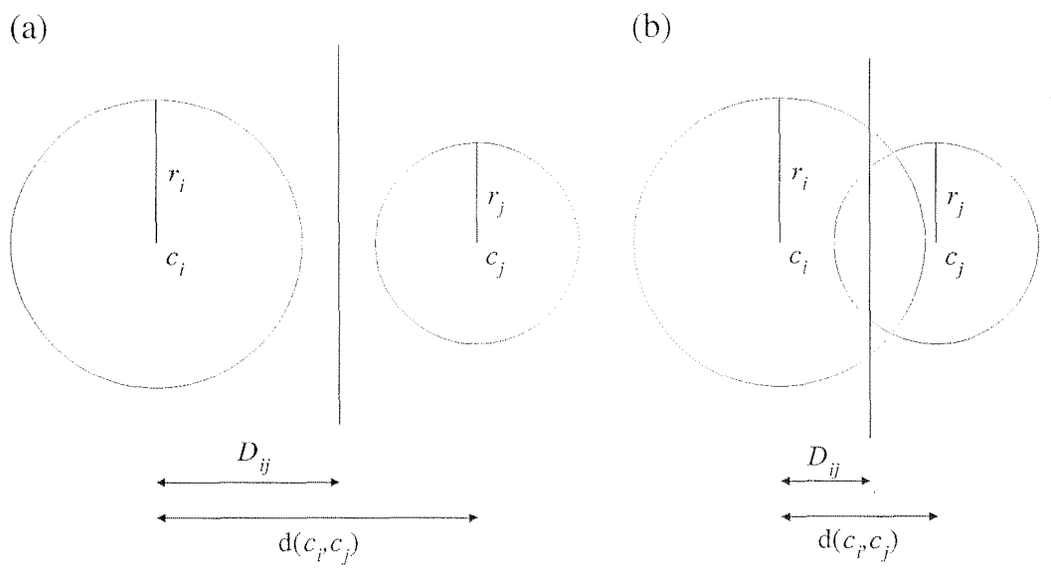


Figure 1.3: Richards' rules for placing bisector surfaces between atoms. The left picture shows the rule for non-bonded atoms, the right picture the rule for bonded atoms.

1. If atoms c_i and c_j are not covalently bonded, then we set

$$h_{ij} = \left\{ x \in \mathbf{R}^3 : \; 2\langle p_j - p_i, x \rangle < (d(c_i,c_j) - r_j + r_i) \cdot d(c_i,c_j) \right\}.$$

   See also the left picture of figure 1.3.

2. If the atoms c_i and c_j are bonded, then we set

$$h_{ij} = \left\{ x \in \mathbf{R}^3 : \; \langle p_j - p_i, x \rangle < \frac{r_i}{r_i + r_j} \|c_j - c_i\|^2 \right\}.$$

   See also the right picture of figure 1.3.

However, as exemplified in figure 1.4, these rules do not assign all intramolecular space to atoms. Richards reports that this "vertex error" is usually very small, reaching up to 3% and occasionally up to 10% of the molecular volume. In Gellatly and Finney's (1982) study of ribonuclease S, this error summed up to 4.1% and 3.7% of the total volume depending on slight variations of the method. On the other hand, Gerstein et al. (1995) measured a total error of only 0.002% with respect to the total volume of pancreatic trypsin inhibitor. Nonetheless, the latter authors discuss the possible use of *curved* instead of planar bisector surfaces.

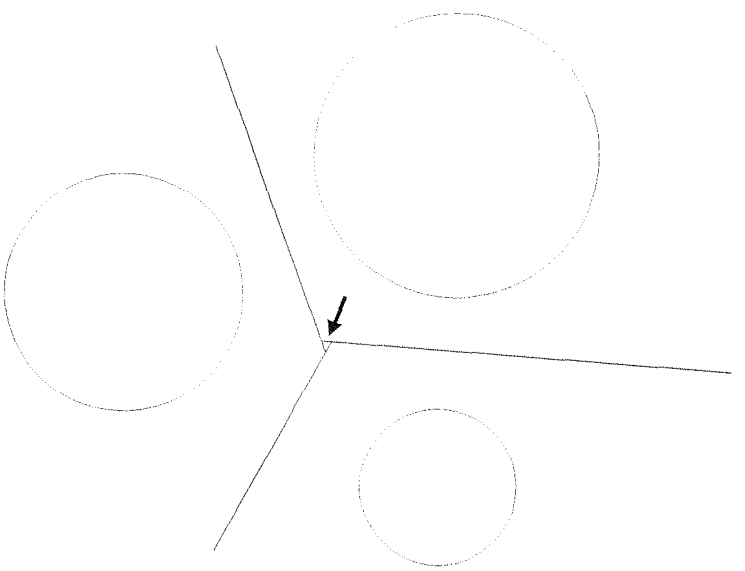


Figure 1.4: The vertex error that can occur with Richards' partitioning scheme. Small tetrahedra may be left unassigned.

Gellatly and Finney (1982) propose the *radical plane method*. Instead of computing ordinary Voronoi cells, the power cells of the atomic spheres are computed. If all radii are equal then this tessellation is equal to the Voronoi tessellation. Power cells are defined via the Laguerre metric, also known as the *power* of a point p with respect to a sphere $\sigma = (c, r)$ with center c and radius r.

Definition 2 (Laguerre metric) *The* power *or* Laguerre metric *of a point* $x \in \mathbf{R}^d$ *with respect to a sphere* $\sigma = (c, r)$ *is given by*

$$p(x, \sigma) = d(x, c)^2 - r^2.$$

We can define the power tessellation defined by a set of spheres $S = \{\sigma_i, 1 \leq i \leq n\}$ in $\mathbf{R}^d$ as follows:

Definition 3 (Power tessellation) *Let* $n \in \mathbf{N}$, $S = \{\sigma_i, 1 \leq i \leq n\}$, $\sigma_i = (c_i, r_i)$, $c_i \in \mathbf{R}^d$, $r_i \in \mathbf{R}$. *Then the collection* $\mathbf{P} = \mathbf{P}(S)$ *of all non-empty sets*

$$P_i = P(\sigma_i) = \left\{ x \in \mathbf{R}^d : \; p(x, \sigma_i) < p(x, \sigma_j) \; \forall 1 \leq j \leq n, j \neq i \right\}$$

is called the power tessellation *induced by* S. *For each* $1 \leq i \leq n$, *the set* P_i *is called the* power cell *of the sphere* σ_i *with respect to* S.

If S is a set of atomic van der Waals spheres, it may happen that the center c_i of an atomic sphere σ_i is not located within its own cell V_i even though the cell is non-empty. This is considered chemically counter-intuitive by later authors, see e.g. Gerstein et al. (1995).

For geometric properties of this tessellation the reader is referred to the papers by Aurenhammer(1987,1991). From the computational point of view, the most important aspect proved in these papers is that a power tessellation in $\mathbf{R}^d$ is in one-to-one correspondence to an upper convex polyhedron in $\mathbf{R}^{d+1}$. Hence, algorithms for computing intersections of halfspaces can be used to compute these tessellations, see e.g. those given by Seidel (1981), Clarkson and Shor (1989), Chazelle (1993).

In a recent study, Goede et al. (1997) compared several methods to partition space among atoms. Based on chemical and experimental reasoning, they propose to assign to each atom the volume of its *additively weighted Voronoi* (AWV) cell. An AWV tessellation divides space based on the distance to the surfaces of the spheres. We will introduce this tessellation in the next section in greater detail. A major problem Goede et al. faced was the fact that they could not devise an analytic and efficient algorithm for the computation of these cells.

1.2 The additively weighted Voronoi tessellation

Again, consider a set of spheres $S = \{\sigma_i, 1 \leq i \leq n\}$ in $\mathbf{R}^d$, each sphere $\sigma_i = (c_i, r_i)$ defined by its center $c_i \in \mathbf{R}^d$ and radius $r_i \in \mathbf{R}$. We can introduce a distance function $d(x, \sigma_i) = d(x, c_i) - r_i$. For a point x outside σ_i this function measures the distance to the surface of the sphere. We assign to each of the spheres $\sigma_i \in S$ the set of all points "nearer" to σ_i than to all other spheres by defining:

Definition 4 (Additively weighted Voronoi tessellation) *Let $n \in \mathbf{N}$, $S = \{\sigma_i, 1 \leq i \leq n\}$, $\sigma_i = (c_i, r_i)$, $c_i \in \mathbf{R}^d$, $r_i \in \mathbf{R}$. Then the collection $\mathbf{V} = \mathbf{V}(S)$ of all non-empty sets*

$$V_i = V(\sigma_i) = \left\{ x \in \mathbf{R}^d : d(x, \sigma_i) < d(x, \sigma_j) \; \forall 1 \leq j \leq n, j \neq i \right\}$$

is called the additively weighted Voronoi (AWV) tessellation *induced by S. For each $1 \leq i \leq n$, the set V_i is called the* additively weighted Voronoi cell *of the sphere σ_i with respect to S.*

Figure 1.5 shows an additively weighted Voronoi cell defined by a small collection of spheres. We distinguish a *tessellation*[6] as a set theoretic object from a *diagram* as a combinatorial object. The combinatorial structure induced by AWV tessellations will be discussed in the next chapter. Another name for the AWV tessellation is *Johnson-Mehl tessellation*, since Johnson and Mehl (1939) introduced this structure as a model for crystal growth processes. According to Stoyan et al. (1995), this specific model for crystal growth processes was already considered by Kolmogoroff (1937).

[6]Formally, slightly generalizing Stoyan et al. (1995), a tessellation of space can be defined as follows:

Definition 5 (Tessellation) *Let d denote an arbitrary but fixed dimension, and let $\mathcal{A}$ denote a collection of open star-shaped subsets of $\mathbf{R}^d$. Then $\mathcal{A}$ is a* tessellation *if and only if*

1. $\forall A_1, A_2 \in \mathcal{A}, A_1 \neq A_2 \Rightarrow A_1 \cap A_2 = \emptyset$,
2. $\bigcup_{A \in \mathcal{A}} \mathrm{cl}(A) = \mathbf{R}^d$, *and*
3. $\forall x \in \mathbf{R}^d, r > 0$ *we have* $\mathrm{card}(\{A \in \mathcal{A} : A \cap B_r(x) \neq \emptyset\}) \in \mathbf{N}$.

Here, $\mathrm{card}(A)$ denotes the cardinality of a set A, and $\mathrm{cl}(A)$ denotes the closure of A. For $x \in \mathbf{R}^d, r > 0$, $B_r(x)$ is the open ball of radius r around x. We will not use this general definition of tessellation. Stoyan et al. (1995) discuss Johnson-Mehl tessellations as special case of general tessellations. However, they define a tessellation to comprise a collection of *convex* polyhedral sets. As we will see, in general this is *not* the case for AWV tessellations.

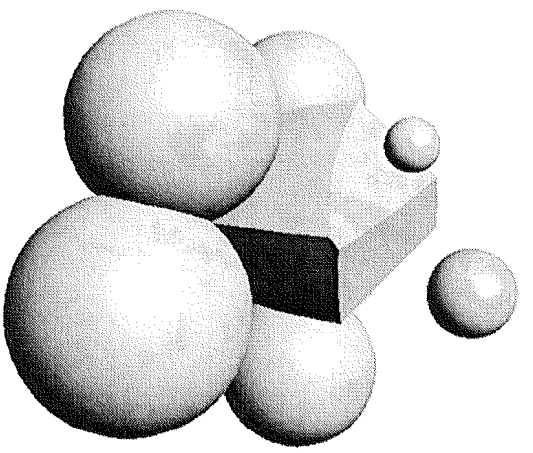


Figure 1.5: An additively weighted Voronoi cell defined by a small collection of spheres in space.

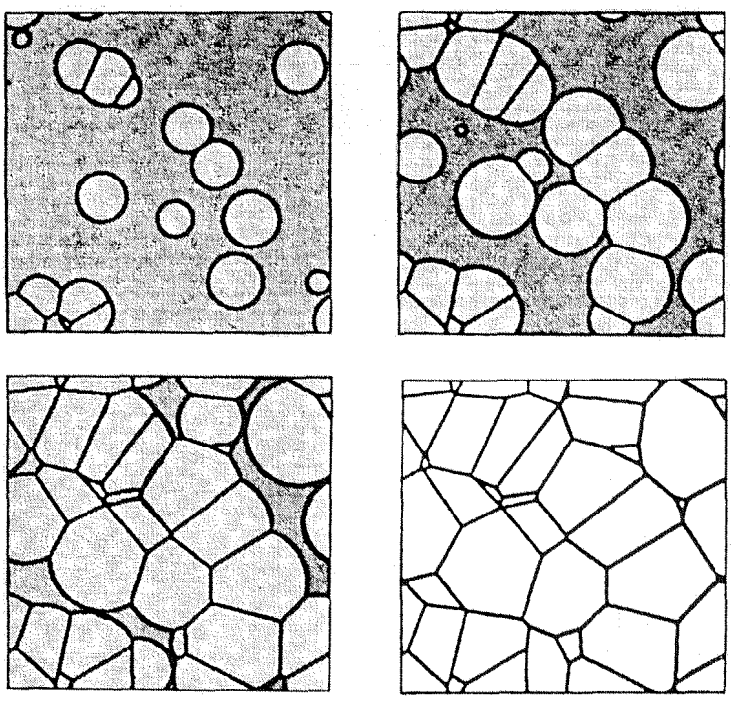


Figure 1.6: A planar AWV tessellation obtained as a model of crystal growth. Crystals start growing radially from seed points, which arrive randomly distributed on both space and time. Whenever two crystals meet, they stop growing at that point. Points arriving at a location already occupied by a crystal are filtered out. This illustration is taken from Mahin et al. (1980). These authors also compare the Johnson-Mehl model to real crystal growth processes and find an astonishing correspondence between the theoretic model and reality.

In the Johnson-Mehl model of crystal growth, crystals start growing radially from a collection of seed points. These seeds, however, are not given at the beginning of the growth process but rather arrive in time and space distributed according to a Poisson distribution, see also figure 1.6. The probabilistic properties of this specific stochastic process have been studied by Kolmogoroff (1937), Johnson and Mehl (1939), Avrami (1939), Meijering (1953), and Møller (1992). For an overview of probabilistic results on AWV tessellations, we refer the reader to the books by Okabe et al. (1992) and by Stoyan et al. (1995). Analytic algorithms for the computation of the tessellation induced by a set S of spheres were given by Sharir (1985) for the planar case and by Aurenhammer (1987) for the general case. Aurenhammer (1987) relates AWV tessellations to the power tessellation defined using the *Laguerre metric* and reduces the computation of a d-

dimensional AWV tessellation to the computation of a $d+2$-dimensional intersection of halfspaces. However, to our best knowledge, no practical implementation of this approach for the non-planar case has been provided so far[7]. Møller (1995) describes a simple algorithm based on discretization to simulate various stochastic processes of AWV tessellations obtained from different point distributions.

1.3 Structure and contributions of this thesis

In this thesis, we will derive and implement a new and efficient algorithm for the computation of AWV cells. To our knowledge, this is the first implementation suitable for practical application. We will analyze the algorithm both theoretically and empirically, and we will demonstrate the benefits of using AWV cells for volume computations in molecules. Specifically, the structure of this thesis is as follows:

In the next chapter, we will discuss the geometric properties of AVW cells. We will show that an AWV cell can be conveniently described using a spherical subdivision data structure, where each edge corresponds to a circular arc. We will also show that the best previously known upper bound on the worst-case complexity of one such cell defined by n other spheres, which is $\Theta(n^2)$, is tight. The following chapter will discuss different methods to compute AWV cells from a theoretical point of view. Based on the new insights gained into the geometry of AWV cells, we will introduce a new randomized incremental algorithm for computing single AWV cells. In chapter 4, we will describe the implementation of a variant of this algorithm. This implementation also involves several pre- and postprocessing steps to make it usable for practical purposes. The practical behavior of this implementation with respect to computing resources and numerical round-off is the subject of chapter 5. Finally, in chapter 6, we use this implementation to compute volumes of atoms and amino acid residues in proteins. Our results demonstrate clearly the superiority of the AWV method over previous methods using planar bisector surfaces.

[7] F. Aurenhammer passed the following note: "Mein Ansatz ist vor einiger Zeit von Leuten aus Japan implementiert worden. Leider habe ich keinen Zugriff auf die Details mehr, das Ergebnis war im Wesentlichen, daß das Verfahren relativ aufwendig ist."

Chapter 2

Geometric properties of additively weighted Voronoi cells

2.1 Introduction

In this chapter, we will discuss the geometric properties of AWV cells in $\mathbf{R}^3$. We will use these properties to design efficient algorithms for their computation in the chapters to follow.

After a short review of previously published properties of AWV cells, we will concentrate on two major issues: First, we will give a detailed account on the geometry of the edges of these cells. The geometric formulas derived in this section will become the very foundation of our new algorithms in the next two chapters. In addition, we will prove a new and tight lower bound of $\Theta(n^2)$ on the combinatorial worst-case complexity of a single AWV cell defined by n spheres in 3 dimensions.

2.2 Previous work

When searching the literature for previous treatises on the geometry of AWV cells, we found only a handful of references. Often, the results are stated without proof, or the proof is formulated only for the planar case.

General references on Voronoi tessellations are the survey paper by Aurenhammer (1991) and the book by Okabe et al. (1992). However, neither of them pro-

vides very much information about the geometry of spatial AWV cells. Specifically, Okabe et al. (1992) discuss only planar AWV tessellations in more detail. Aurenhammer (1991), on the other hand, concentrates on his observation that a cell of the additively weighted Voronoi tessellation in R^d can be represented as the projection of the intersection of a cell of a suitably defined power tessellation in R^{d+1} with a $d+1$-dimensional cone. This construction also provides an upper bound of $O(n^{\lceil d/2 \rceil})$ on the worst case complexity of a single cell defined by n spheres in d dimensions. However, Aurenhammer could discuss optimality of this result only for the planar case.

A rather self-contained and comprehensive description of the geometric properties of planar AWV tessellations was given by Sharir (1985). In his paper, he describes a sweep line algorithm for their efficient computation.

Very useful references on the geometry of AWV-cells turned out to be the two papers by Møller (1992, 1995) about the probabilistic properties of higher-dimensional AWV-cells generated by Poisson point processes. Among other observations, these papers contain general parameterizations of the k-faces, $0 \leq k \leq d$, of these cells in arbitrary dimension d.

Elementary properties. Let $n \in \mathbf{N}$, $S = \{\sigma_i, 1 \leq i \leq n\}$ in $\mathbf{R}^d$, $\sigma_i = (c_i, r_i)$, $c_i \in \mathbf{R}^d$, $r_i \in \mathbf{R}$, and let $\mathbf{V} = \mathbf{V}(S) = \{V_i, 1 \leq i \leq n\}$ be the AWV tessellation induced by S. The reader may observe that translating all $\sigma \in S$ by a vector $\vec{v}$ will simply map each cell V_i to its translate $V_i + \vec{v}$, $1 \leq i \leq n$. Moreover, we are free to add or subtract a common constant Δr to or from all radii without changing the shape of the individual cells. From the latter we can always derive the assumption that either all radii are non-negative, or that a specific radius is equal to zero.

Again, let $B_r(c)$ denote the closed ball of radius r centered at c, and let $\mathrm{int}(A)$ and $\mathrm{cl}(A)$ denote the topological interior and closure of a set $A \subset \mathbf{R}^d$, respectively.

Proposition 1 *The additively weighted Voronoi cell of a sphere σ among n other spheres $\{\sigma_i, 1 \leq i \leq n\}$ is empty, if and only if there exists $1 \leq i \leq n$, such that $\sigma \subset \mathrm{int}(B_{r_i}(c_i))$.*

Proof: The simple proof given in Sharir (1985) is actually independent of the dimension: Observe, that $\sigma \subset \mathrm{int}(B_{r_i}(c_i))$ is equivalent to $d(c, c_i) < r_i - r$. Then for all $x \in \mathbf{R}^d$ we have

$$d(x, \sigma_i) = d(x, c_i) - r_i \quad \leq \quad d(x, c) + d(c, c_i) - r_i$$

$$< \quad d(x,c)+(r_i-r)-r_i = d(x,c)-r = d(x,\sigma)$$

On the other hand, if $V(\sigma)=\emptyset$ then $\forall x \in \mathbf{R}^d : \exists 1 \leq i \leq n : \ d(x,\sigma_i) < d(x,\sigma)$. Setting $x=c$ we obtain

$$\begin{aligned} \exists 1 \leq i \leq n : \ d(c,\sigma_i) = d(c,c_i)-r_i \quad &< \quad d(c,\sigma) = d(c,c)-r \\ \Leftrightarrow \exists 1 \leq i \leq n : \ d(c,c_i) \quad &< \quad r_i - r, \end{aligned}$$

which is equivalent to $\sigma \subset \text{int}(B_{r_i}(c_i))$. □

Definition 6 (star-shaped) *Let $A \subset \mathbf{R}^d$. We say A is a* star-shaped set with kernel *k, if $k \in \text{int}(A)$ and for each $x \in \partial A$ we have*

$$\partial A \cap \{\lambda k + (1-\lambda)x, 0 < \lambda < 1\} = \emptyset.$$

We call a set A star-shaped if there exists a kernel $k \in \text{int}(A)$ such that A is a star-shaped set with kernel k.

Proposition 2 *An AWV cell $V(\sigma) \neq \emptyset$ in $\mathbf{R}^d$ is star-shaped.*

Proof: See also Sharir (1985). Let $x \in V(\sigma)$ and $z \subset (c,x)$ a point in the interior of the line segment connecting c and x. Suppose there exists an $1 \leq i \leq n$ such that $d(z,\sigma_i) < d(z,\sigma)$. Then we have

$$\begin{aligned} d(x,\sigma) \quad &= \quad d(c,x)-r = d(c,z)+d(z,x)-r = d(z,\sigma)+d(z,x) \\ &> \quad d(z,\sigma_i)+d(z,x) = d(z,c_i)+d(z,x)-r_i \\ &\geq \quad d(x,c_i)-r_i = d(x,\sigma_i), \end{aligned}$$

contradicting the assumption that $d(x,\sigma) \leq d(x,\sigma_i)$. □

If the spheres defining two adjacent cells have different radii, then the bisector surface separating these cells bends around the smaller sphere. In fact, the bisector is one branch of a rotational hyperboloid whose foci are the centers of the two defining spheres. Therefore, a cell can be bounded without having a single vertex, as depicted in the left image of figure 2.1. More specifically, the following holds:

Proposition 3 *Let σ_1 and σ_2 be two spheres in $\mathbf{R}^d$. Then the bisector*

$$G_{d-1} = G(\sigma_1,\sigma_2) = \left\{ x \in \mathbf{R}^d : \ d(x,\sigma_1) = d(x,\sigma_2) \right\} \ \textit{is}$$

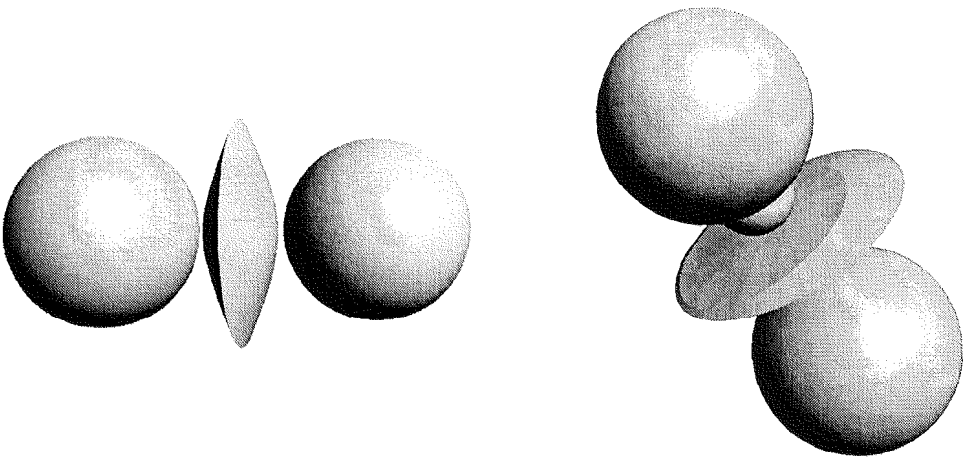


Figure 2.1: Two special cases of additively weighted Voronoi cells. The left picture shows the cell of a small sphere between two larger ones having only a single elliptic edge. In the right picture, we added another small sphere to obtain a disconnected edge skeleton.

1. *the empty set, if one of the spheres is contained in the open interior of the other.*

2. *a half line, if the strictly smaller sphere is contained in the closed, but not the open interior of the other sphere.*

3. *a hyperplane, if both spheres have the same radius* $r_1 = r_2$*, but have distinct centers* $c_1 \neq c_2$*.*

4. *a rotational hyperboloid with* c_1 *and* c_2 *being the foci, which bends around the smaller of the two spheres.*

5. *the whole space, if both spheres coincide.*

The proof in Okabe et al. (1992) for the planar case easily generalizes to higher dimensions.

***k*-faces.** Møller (1992) gave by far the most general description of bisectors in the appendix of his paper on the probabilistic properties of AWV cells generated by Poisson point processes. It is important to note that Møller can assume the most general restrictions concerning degeneracy, since these hold almost surely in the stochastic setting considered.

He distinguishes "mathematical" faces of the tessellation, which give the functional description of a face, from combinatorial faces, which are the bounded and trimmed components of a mathematical face actually realized in the tessellation:

Definition 7 (Mathematical k-face) *Let $\sigma_0, \ldots, \sigma_j$, $1 \leq j \leq d$ be distinct spheres in $\mathbf{R}^d$. Then*

$$G_k = G(\sigma_0, \ldots, \sigma_j) = \left\{ x \in \mathbf{R}^d : \; d(x, \sigma_0) = \ldots = d(x, \sigma_j) \right\},$$

where $k = d - j$, is called the mathematical k-face *defined by $\sigma_0, \ldots, \sigma_j$.*

For a discrete set A and $r \in \mathbf{N}$, let $\binom{A}{r}$ denote the set of all subsets of A with cardinality r.

Definition 8 (Combinatorial k-face) *Let $S = \{\sigma_i, 0 \leq i < n\}$ be distinct spheres in $\mathbf{R}^d$, $\sigma_0, \ldots, \sigma_j \in S$, $1 \leq j \leq d$. Then each connected component of the set*

$$\begin{aligned} F_0^{(k)} &= F(\sigma_0, \ldots, \sigma_j \mid S) \\ &= \left\{ x \in \mathbf{R}^d : \; d(x, \sigma_0) = \ldots = d(x, \sigma_j) > d(x, \tau) \; \forall \, \tau \in S \setminus \{\sigma_0, \ldots, \sigma_j\} \right\} \end{aligned}$$

where $k = d - j$, is called a (combinatorial) k-face *defined by $\sigma_0, \ldots, \sigma_j$. Further, we denote by*

$$\begin{aligned} F_k &= F_k(S) \\ &= \bigcup_{\{\sigma_0, \ldots, \sigma_j\} \in \binom{S}{j+1}} \left\{ F : \; F \text{ is connected component in } F(\sigma_0, \ldots, \sigma_j \mid S) \right\} \end{aligned}$$

the set of all combinatorial k-faces. We define $F_{-1} = \{\emptyset\}$.

If the spheres $\sigma_i, 1 \leq i \leq n$ are in general position then non-empty mathematical k-faces have Hausdorff dimension k, and combinatorial k-faces are k-dimensional semi-algebraic sets. In fact, this can be used as defining property for what we mean by saying that the spheres are in general position.

Møller (1992) also provides parameterizations for mathematical k-faces in d dimensions. Since the description of these parameterizations is rather lengthy, and we will not use them for our algorithms, the reader is referred to the original paper.

Definition 9 (Additively weighted Voronoi diagram) *Let* $n \in \mathbf{N}$, $S = \{\sigma_i, 1 \leq i \leq n\}$ *in* $\mathbf{R}^d$, $\sigma_i = (c_i, r_i)$, $c_i \in \mathbf{R}^d$, $r_i \in \mathbf{R}$, *and let* $\mathbf{V} = \mathbf{V}(S) = \{V_i, 1 \leq i \leq n\}$ *be the AWV tessellation induced by S. Then the* additively weighted Voronoi diagram *induced by S is the graph* $G = (V, E)$, *where*

$$\begin{aligned} V &= \bigcup_{k=0}^{d} F_k(S), \\ E &= \{\{f_{k-1}, f_k\} : f_{k-1} \in F_{k-1}(S), f_k \in F_k(S), f_{k-1} \subset \mathrm{cl}(f_k), 0 \leq k \leq d\}. \end{aligned}$$

Parameterization over a sphere. Since an AWV cell is a star-shaped, it is natural to look at the projection of the boundary of the cell onto a unit sphere S^{d-1} around the center of its defining sphere.

Proposition 4 *Let* $\sigma_1 = (\mathbf{0}, 0)$, $\sigma_2 = (c_2, r_2)$ *be two spheres with non-empty bisector* $G_{d-1} = G(\sigma_1, \sigma_2)$. *Then the projection of* G_{d-1} *onto a unit sphere* S^{d-1} *around* $c_1 = \mathbf{0}$ *is given by*

$$\left\{x \in S^{d-1} : \langle x, c_2 \rangle \geq -r_2\right\}$$

Proof: See also Møller (1995). Let $x \in G_{d-1}$. Then the inverse of the projection of x onto $p \in S^{d-1}$, i.e. the lifting map ϕ such that $\phi(p) = x$, is given by

$$\phi : p \mapsto \frac{c_2^2 - r_2^2}{2(r_2 + \langle p, c_2 \rangle)} \cdot p.$$

We obtain this mapping by plugging the equations $x = d \cdot p$ and $\|p\| = 1$ into the equation of the graph of the distance function

$$(d + r_2)^2 = (x - c_2)^2,$$

where $d = d(x, \sigma_2)$. We see that the image of the projection of G_{d-1} onto S^{d-1} is the set of points $p \in S^{d-1}$ for which the denominator of $\phi(p)$ is non-negative. □

The relation between AWV diagrams and power diagrams. Aurenhammer (1987) showed the following relation between AWV diagrams and power diagrams, which is very useful from the computational point of view: Let $S = \{\sigma_i, 1 \leq i \leq n\}$ be a set of spheres in $\mathbf{R}^d$. To each $\sigma_i \in S$ we assign its *power cell*

$$P_i = P(\sigma_i) = \left\{x \in \mathbf{R}^d : p(x, \sigma_i) < p(x, \sigma_j) \forall 1 \leq j \leq n, j \neq i\right\},$$

where $p(x,\sigma_i) = d(x,c_i)^2 - r_i^2$. For each $\sigma_i \in S$ let κ_i denote the cone which is the embedding of the graph of the distance function $d(\cdot,\sigma_i)$ in $\mathbf{R}^{d+1}$, i.e.

$$\kappa_i = \left\{ x \in \mathbf{R}^{d+1} : x_{d+1} \geq -r_i, \sum_{j=1}^{d} (x_j - c_{i,j})^2 = (x_{d+1} + r_i)^2 \right\},$$

and let Σ_i denote the lifted sphere

$$\Sigma_i = \left((c_{i,1}, \ldots, c_{i,d}, r_i), \sqrt{2}\, r_i \right).$$

Finally, let $\mathrm{proj}_d : \mathbf{R}^{d+1} \to \mathbf{R}^d$, $(x_1, \ldots, x_{d+1}) \mapsto (x_1, \ldots, x_d)$ denote the natural projection from $\mathbf{R}^{d+1}$ to $\mathbf{R}^d$.

Lemma 1 (Aurenhammer (1987)) *Let $S = \{\sigma_i, 1 \leq i \leq n\}$ be a set of spheres in $\mathbf{R}^d$ and κ_i and Σ_i be defined as above, $1 \leq i \leq n$. Then*

$$V_i = \mathrm{proj}_d (\kappa_i \cap P_i),$$

where P_i is the power cell of Σ_i within the power tessellation defined by the set of lifted spheres $\{\Sigma_i, 1 \leq i \leq n\}$.

This lemma has the following implication: Consider two AWV cells V_i and V_j, $i \neq j$. If V_i and V_j have a face of any dimension in common, then so have the corresponding power cells P_i and P_j, i.e. $\mathrm{cl}(V_i) \cap \mathrm{cl}(V_j) \neq \emptyset$.

A nice presentation of this relation between AWV and power cells can also be found in the book by Boissonnat and Yvinec (1998).

2.3 The edges of a 3-dimensional cell

A mathematical 1-face G_1 defined by three spheres $\sigma_1, \sigma_2, \sigma_3$ in $\mathbf{R}^3$ is symmetric with respect to the plane A through the centers of these spheres. Hence G_1 passes through the common vertex $G_0' = G'(\sigma_1 \cap A, \sigma_2 \cap A, \sigma_3 \cap A)$ in the restricted diagram within the plane of symmetry A, see figure 2.2.

We say a set of spheres S is in *convex position* if each $\sigma \in S$ is located on the convex hull of $\bigcup S$.

Lemma 2 *Let σ_1, σ_2, σ_3 be three spheres in $\mathbf{R}^3$ and assume that the common bisector*

$$G_1 = G(\sigma_1, \sigma_2, \sigma_3) = \{ x \in \mathbf{R}^3 : d(x,\sigma_1) = d(x,\sigma_2) = d(x,\sigma_3) \}$$

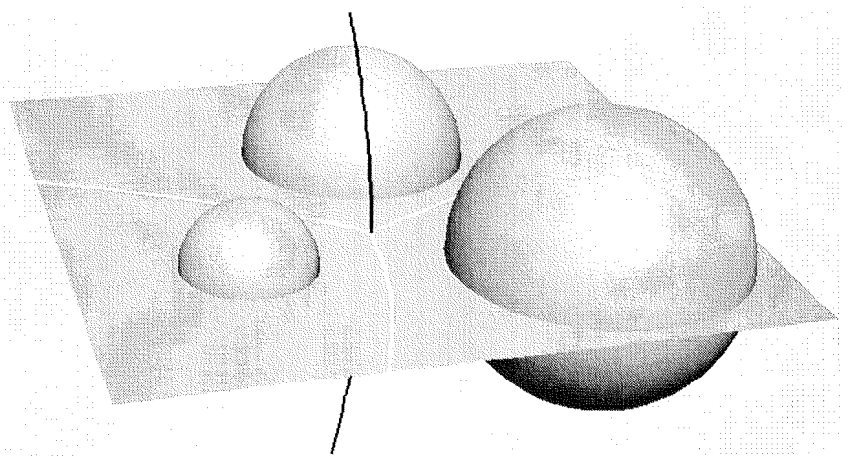


Figure 2.2: Three spheres $\sigma_1, \sigma_2, \sigma_3$ and their plane of symmetry A. In the plane, we see the lower dimensional AWV tessellation. The white edges are the intersection of A with the pairwise bisectors of the three spheres. A mathematical edge — shown in black — defined by these spheres passes through a vertex of the lower dimensional diagram.

has Hausdorff dimension 1. *Then* G_1 *is*

1. *a line if* $r_1 = r_2 = r_3$.
2. *a circle if* c_1, c_2 *and* c_3 *are collinear and the spheres are in non-convex position.*
3. *a branch of a hyperbola, if* σ_1, σ_2 *and* σ_3 *are in convex position.*
4. *a branch of a parabola, if one of the spheres is contained in the convex hull of the other two spheres, and if this sphere touches the convex hull in a single point.*
5. *an ellipse, otherwise.*

Proof: If all radii are equal, than the bisector is simply a line as in the ordinary Voronoi diagram. Hence, we assume $r_1 \neq r_2$, $r_1 \neq r_3$. We also assume $c_1 \neq c_2 \neq c_3$ due to proposition 1.

We parameterize G_1 depending on whether c_1, c_2 and c_3 are collinear or not:

1. Case c_1, c_2 and c_3 are not collinear: According to lemma 1, let P_1 denote the 4-dimensional power cell corresponding to σ_1, and let f be the 2-face of P_1

such that $G_1 = \mathrm{proj}_3(f \cap \kappa_1)$. Furthermore, let $A_3 := \mathrm{aff}(c_{\sigma_1}, c_{\sigma_2}, c_{\sigma_3}) \subset \mathbf{R}^3$ be the affine hyperplane containing the centers of the spheres, $A_4 := A_3 \oplus \vec{e}_4$, $\vec{e}_k$ denoting the kth unit vector.

To parameterize G_1, we choose $\vec{v} \in \mathbf{R}^3$ such that $\vec{v} = (v_1, v_2, v_3) \perp A_3$, i.e. $\vec{v} = (c_1 - c_2) \times (c_1 - c_3)$, and $\vec{w} = (w_1, w_2, w_3, w_4) \in \mathbf{R}^4$ such that $\vec{w} \perp (v_1, v_2, v_3, 0), \vec{w} \perp A_4$, i.e.

$$\vec{w} = \begin{vmatrix} \vec{e}_1 & v_1 & c_{1,1} - c_{2,1} & c_{1,1} - c_{3,1} \\ \vec{e}_2 & v_2 & c_{1,2} - c_{2,2} & c_{1,2} - c_{3,2} \\ \vec{e}_3 & v_3 & c_{1,3} - c_{2,3} & c_{1,3} - c_{3,3} \\ \vec{e}_4 & 0 & r_1 - r_2 & r_1 - r_3 \end{vmatrix}.$$

Further, we determine P as the minimum point of $A_4 \cap \kappa_1$ with respect to the 4th coordinate. Each point $x \in G_1 = f \cap \kappa_1$ has a unique representation $x = P + s\vec{v} + t\vec{w}$. We have to consider three cases depending on the "steepness" of the intersection plane with respect to the cone, i.e. the sign of

$$\Delta = w_4^2 - w_1^2 - w_2^2 - w_3^2. \tag{2.1}$$

(a) If $\Delta > 0$, then the bisector is hyperbolic.

(b) If $\Delta = 0$, then the bisector is parabolic.

(c) If $\Delta < 0$, then the bisector is elliptic.

Without loss of generality, we may assume that we have $c_1 = \mathbf{0}$, $c_2 = (s, 0, 0)$, $c_3 = (u, v, 0)$, $r_1 = 0$, $r_2, r_3 \neq 0$. Then

$$w = \left(sr_2, v^2, s^2 v r_3 - suvr_2, 0, s^2 v^2\right),$$

i.e. the previous sign condition 2.1 is equivalent to

$$s^4 v^4 - s^2 v^2 \left(r_2^2 v^2 + s^2 r_3^2 + u^2 r_2^2 - 2sr_3 u r_2\right) \left\{ \begin{array}{c} < \\ = \\ > \end{array} \right\} 0 \tag{2.2}$$

Let us assume that σ_1, σ_2 and σ_3 are in convex position. This, in turn, is equivalent to the statement that there exists a supporting plane A, i.e. a common tangent plane to all three spheres. An oriented plane $\langle a, x \rangle = b$ is tangent to a sphere $\sigma = (m, r)$ in such a way that a points to the outside of σ, if and only if[1]

$$\langle m, a \rangle - b + r\|a\|_2 = 0.$$

[1] Cf. Benz (1992)

We obtain the following system of equations

$$\begin{aligned} \|a\|_2 &= 1 \\ \langle c_i, a\rangle - b + r_i &= 0, \quad \text{for } i = 1,2,3, \end{aligned}$$

which, in our cases reduces to

$$\begin{aligned} \|a\|_2 &= 1 \\ s a_1 + r_2 &= 0 \\ u a_1 + v a_2 + r_3 &= 0. \end{aligned} \tag{2.3}$$

Since we assumed that G_1 is 1-dimensional and that the centers of the spheres are not collinear, we have $s \neq 0$, $v \neq 0$. Using standard transformations, we obtain the solution

$$\begin{aligned} a_1 &= -\frac{r_2}{s} \\ a_2 &= \frac{u r_2 - s r_3}{v s} \\ a_3 &= \sqrt{D} = \sqrt{1 - a_1^2 - a_2^2} \\ &= \sqrt{1 - \frac{r_2^2}{s^2} - \frac{(u r_2 - s r_3)^2}{s^2 v^2}}. \end{aligned}$$

Multiplying out we see that the discriminant D is just the left hand side Δ of inequality (2.2) divided by the positive term $s^4 v^4$. Observe, that we have a single solution for a common tangent plane if D is zero, which implies geometrically that one of the spheres touches the convex hull of the other two spheres from the interior in a single point.

2. Case c_1, c_2 and c_3 are collinear: We may assume that we have $c_1 = \mathbf{0}$, $c_2 = (s,0,0)$, $c_3 = (u,0,0)$, $r_1 = 0$, $r_2, r_3 \neq 0$. Then the system of equations

$$d = d(x,\sigma_1) = d(x,\sigma_2) = d(x,\sigma_3)$$

solves to

$$\begin{aligned} d &= \frac{1}{2}\frac{u s^2 + r_3^2 s - u r_2^2 - u^2 s}{u r_2 - r_3 s} \\ x_1 &= \frac{1}{2}\frac{s^2 r_3 + r_2 r_3^2 - r_2 u^2 - r_2^2 r_3}{r_3 s - u r_2} \\ x_2 &= \bar{y}\cos\alpha \\ x_3 &= \bar{y}\sin\alpha, \qquad \text{where} \\ \bar{y} &= \sqrt{d^2 - x_1^2}, \end{aligned} \tag{2.4}$$

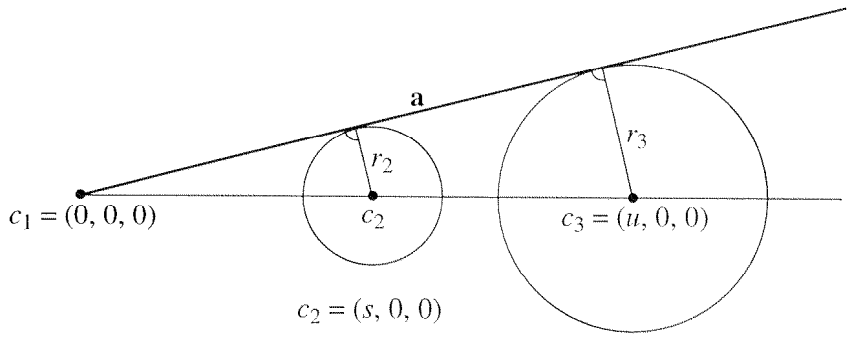


Figure 2.3: Case $v = 0$.

for any $\alpha \in [0 \ldots 2\pi]$. This is either the description of a circle if $\bar{y} \in \mathbf{R}$, or the description of the empty set if $\bar{y} \notin \mathbf{R}$.

To show that $G_1 \neq \emptyset$ implies that the spheres are in non-convex position, we try again to find a common tangent plane starting from the system of equations (2.3). If $v = 0$, then this system of equations yields the requirement

$$a_1 = -\frac{r_2}{s} = -\frac{r_3}{u},$$

as depicted in figure 2.3. Observe, that this implies that the denominator of the equations 2.4 vanishes. Hence, these equations do not have a common solution if there exists a common tangent plane to the three spheres.

□

The following lemma will be the heart of the new algorithms presented in chapters 3 and 4:

Lemma 3 *The projection of a 1-dimensional mathematical edge $G_1 = G(\sigma, \sigma_i, \sigma_j)$ of an AWV cell $V(\sigma)$ onto a unit sphere around the center c of σ is a circular arc.*

Proof: Equating the distance equations

$$d = \frac{c_i^2 - r_i^2}{2\left(r_i + \langle p, c_i \rangle\right)} = \frac{c_j^2 - r_j^2}{2\left(r_j + \langle p, c_j \rangle\right)}$$

for two neighboring spheres $\sigma_i = (c_i, r_i)$ and $\sigma_j = (c_j, r_j)$, $1 \leq i, j \leq n$, $i \neq j$ yields the equation describing the projection of $G(\sigma, \sigma_i) \cap G(\sigma, \sigma_j)$:

$$\langle a_{i,j}, p \rangle = b_{i,j}, \text{ where}$$

$$
\begin{aligned}
a_{i,j} &= c_i \cdot \left(c_j^2 - r_j^2\right) - c_j \cdot \left(c_i^2 - r_i^2\right) \\
b_{i,j} &= r_i \cdot \left(r_j^2 - c_j^2\right) - r_j \cdot \left(r_i^2 - c_i^2\right)
\end{aligned}
$$

□

We denote the halfspaces defined in this way by $h_{i,j} = \{x \in R^3 : \langle a_{i,j}, x \rangle \geq b_{i,j}\}$. For an illustration of this representation, see figure 2.4.

2.4 A tight lower bound on the worst case complexity of an additively weighted Voronoi cell in 3 dimensions

According to lemma 1, a single AWV cell $V = V(\sigma)$ in $\mathbf{R}^3$ can be described as projection of the intersection of a 4-dimensional cone κ and a 4-dimensional convex polytope P to $\mathbf{R}^3$. If σ has n neighboring spheres $S = \{\sigma_i, 1 \leq i \leq n\}$, then P can be defined as the intersection of n halfspaces. By the upper bound theorem[2], the combinatorial complexity of both P and V is $O(n^2)$. To put it boldly, the total complexity of the diagram can be concentrated on a single cell. In this section, we specify a family of configurations of n spheres realizing single AWV cells of combinatorial complexity $\Theta(n^2)$.

We obtain the construction by applying a specific perturbation to a highly degenerate configuration of spheres. We will show that this perturbation leads to the realization of a large number of vertices.

[2]Cf. McMullen (1970)

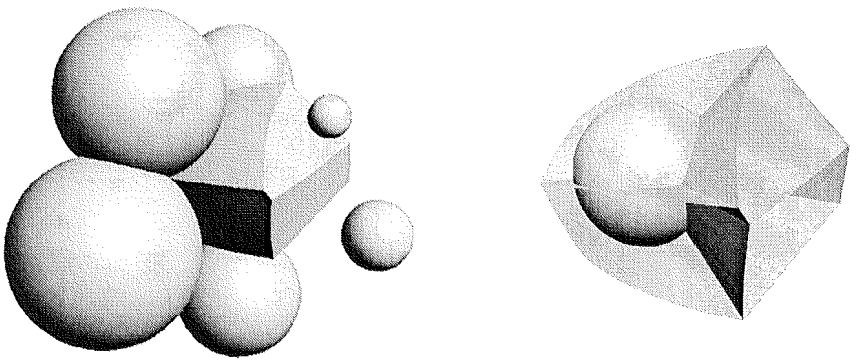


Figure 2.4: The edges of an additively weighted Voronoi cell project as circular arcs.

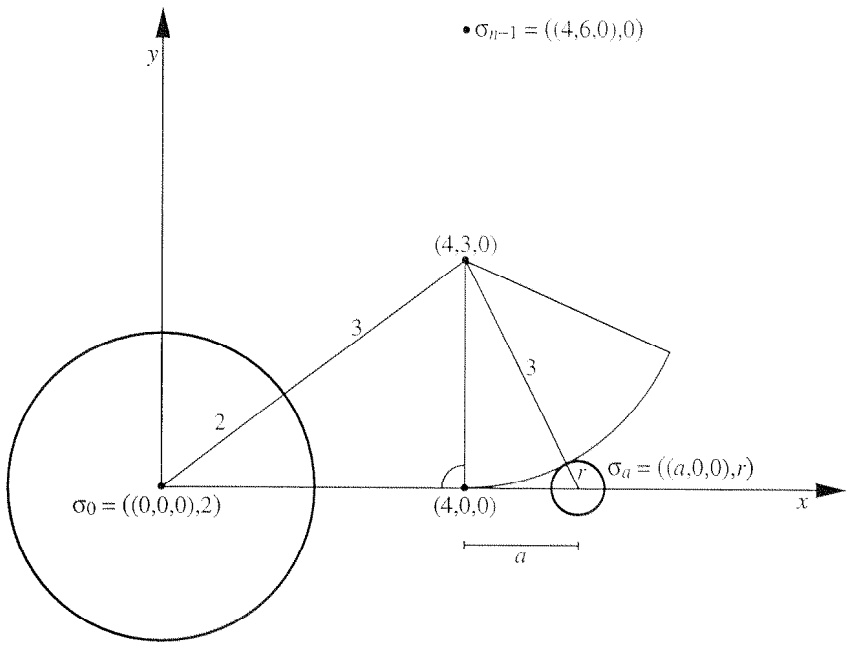


Figure 2.5: Cross section $z = 0$ of our worst case construction. For $a = \frac{i}{n}$, $1 < i \leq \lceil \frac{n}{2} \rceil$, $\sigma_i = \sigma_a$ is the sphere centered at $(4+(1+\varepsilon)a, 0, 0)$ with radius $r = \sqrt{9+a^2} - 3$. If $\varepsilon = 0$, σ_a is tangent to a sphere centered at $(4,3,0)$ with radius 3.

Theorem 1 *An additively weighted Voronoi cell in* $\mathbf{R}^3$ *defined by n spheres can realize* $\Theta(n^2)$ *vertices.*

Proof: Consider the following set of spheres depending on n and ε:

$$\begin{aligned}
\sigma_0 &= ((0,0,0),2) \\
\sigma_i &= \left(\left(4+(1+\varepsilon)\tfrac{i}{n}, 0, 0\right), \sqrt{9+\tfrac{i^2}{n^2}} - 3 \right), \quad \text{for } 1 \leq i < \lceil \tfrac{n}{2} \rceil \\
\sigma_i &= \left(\left(4, 6\cos\tfrac{4\pi(i+1)}{n}, 6\sin\tfrac{4\pi(i+1)}{n}\right), 0 \right), \quad \text{for } \lceil \tfrac{n}{2} \rceil \leq i < n
\end{aligned}$$

Figure 2.5 shows a cross section of this configuration. We focus our interest on the combinatorial complexity of $V(\sigma_0)$. If $\varepsilon = 0$ then $V(\sigma_0)$ has a single circular edge, namely the circle of radius 3 around the center $(4,0,0)$ parallel to the yz-plane of the coordinate system.

We will show the following behavior of this configuration depending on $\varepsilon > 0$:

1. For small $\varepsilon > 0$, each triple $(\sigma_0, \sigma_i, \sigma_{i+1})$ of spheres, $1 \leq i < \lceil \frac{n}{2} \rceil - 1$, generates a circular edge e_i of $V(\sigma_0)$.

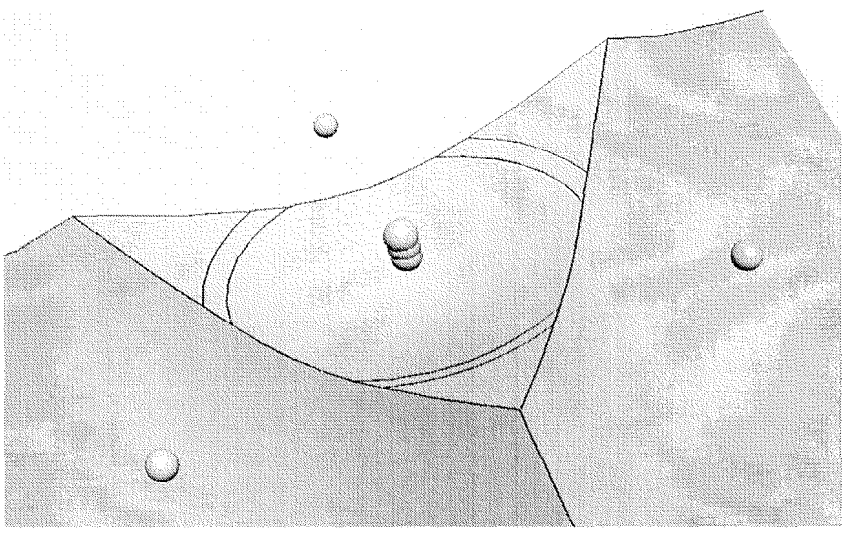


Figure 2.6: The worst case construction for $n = 7$ seen from the positive x direction. The spheres have been re-scaled a little to make the effect more visible.

2. For sufficiently small $\varepsilon > 0$, each σ_j, $\lceil \frac{n}{2} \rceil \leq j < n$, generates two vertices with each edge e_i, $1 \leq i < \lceil \frac{n}{2} \rceil - 1$.

The case $n = 7$ is depicted in figure 2.6 □

Lemma 4 *Using the notation introduced above, the following statement holds: For small $\varepsilon > 0$, each triple $(\sigma_0, \sigma_i, \sigma_{i+1})$, $1 \leq i < \lceil \frac{n}{2} \rceil - 1$, generates a circular edge e_i of $V(\sigma_0)$.*

Proof: We restrict our argumentation to the cross section $z = 0$ as shown in figure 2.5. Then we have to show the following: For n and ε let

$$\begin{aligned} C_0 &= ((0,0),2) \\ C_i &= \left(\left(4 + (1+\varepsilon)\tfrac{i}{n}, 0\right), \sqrt{9 + \tfrac{i^2}{n^2}} - 3 \right), \quad \text{for } 1 \leq i < \lceil \tfrac{n}{2} \rceil . \end{aligned}$$

Then each triple (C_0, C_i, C_{i+1}) of circles, $1 \leq i < \lceil \frac{n}{2} \rceil - 1$, generates two vertices $z_i, \bar{z}_i$ of $V(C_0)$. Rotating these circles and vertices around the the x-axis will bring us back to the 3-dimensional case.

Consider three circles $((0,0),2)$, $((x_1,0),r_1)$, and $(x_2,0),r_2)$. An AWV-vertex (x,y) at distance d generated by these circles is a solution to the following system

of equations:

$$\begin{aligned} x^2+y^2-(d+2)^2 &= 0 \\ (x-x_1)^2+y^2-(d+r_1)^2 &= 0 \\ (x-x_2)^2+y^2-(d+r_2)^2 &= 0 \end{aligned}$$

Standard transformations yield the solution

$$\begin{aligned} x &= \frac{(r_1^2-x_1^2-4)(4-2r_2^2)-(r_2^2-x_2^2-4)(4-2r_1^2)}{2x_1(2r_2-4)-2x_2(2r_1-4)} \\ d &= \frac{2(r_1^2-x_1^2-4)x_2-2(r_2^2-x_2^2-4)x_1}{2x_1(2r_2-4)-2x_2(2r_1-4)} \\ y &= \pm\sqrt{d^2-x^2}. \end{aligned}$$

According to our construction, we substitute

$$\begin{aligned} x_1 &\longrightarrow 4+(1+\varepsilon)a_1, \quad r_1 \longrightarrow \sqrt{9+a_1^2}-3 \\ x_2 &\longrightarrow 4+(1+\varepsilon)a_2, \quad x_2 \longrightarrow \sqrt{9+a_2^2}-3 \end{aligned}$$

to obtain functions $x(a_1,a_2;\varepsilon)$, $y(a_1,a_2;\varepsilon)$, and $d(a_1,a_2;\varepsilon)$. For an additional circle $C_a = ((x_a,0),r_a)$ with parameter a, i.e. $x_a = 4+(1+\varepsilon)a$, $r_a = \sqrt{9+a^2}-3$ let $\mathrm{dist}(a,a_1,a_2;\varepsilon)$ denote the function measuring the distance of $(x(a_1,a_2;\varepsilon),y(a_1,a_2;\varepsilon))$ to C_a:

$$\mathrm{dist}(a;a_1,a_2;\varepsilon) = \sqrt{(x(a_1,a_2;\varepsilon)-x_a)^2+y(a_1,a_2;\varepsilon)^2}-r_a$$

We have to show that for fixed $0<a_1,a_2<1$ and $\varepsilon>0$ and for all $a \in (0,1)\setminus[a_1,a_2]$

$$\mathrm{dist}(a;a_1,a_2;\varepsilon) > d(a_1,a_2;\varepsilon),$$

which is equivalent to showing

$$f_{a_1,a_2;\varepsilon}(a) := (x(a_1,a_2;\varepsilon)-x_a)^2+y(a_1,a_2;\varepsilon)^2-(r_a+d(a_1,a_2;\varepsilon))^2 > 0.$$

Since $f_{a_1,a_2,0}(a)=0$ and $f_{a_1,a_2;\varepsilon}(a)$ is continuous in a_1, a_2, a, and ε, it is sufficient to show that for any fixed ε,a_1,a_2 the function $f_{a_1,a_2;\varepsilon}(a)$ is convex. By continuity, it suffices to show

$$\frac{d}{da^2}f_{a_1,a_2;\varepsilon}(a) > 0$$

We calculate $\frac{d}{da^2} f_{a_1,a_2;\varepsilon}$ as

$$\frac{d}{da^2} f_{a_1,a_2;\varepsilon}(a) = 2\left(1 - \frac{a^2}{9+a^2}\right)\frac{d(a_1,a_2;\varepsilon)-3}{\sqrt{9+a^2}} + 2\varepsilon(1+\varepsilon).$$

This expression is positive if we can show that $d(a_1,a_2;\varepsilon)-3>0$. We calculate the latter expression as

$$\begin{aligned} d(a_1,a_2;\varepsilon)-3 &= \frac{\varepsilon}{2}\frac{A}{B}, \text{ where} \\ A &= 4(2+\varepsilon)(a_2^2-a_1^2) + (2+3\varepsilon)\,a_1\,a_2\,(a_2-a_1) \\ B &= 4\left(\sqrt{9+a_2^2}-\sqrt{9+a_1^2}\right) \\ &\quad + (1+\varepsilon)\left(a_1\sqrt{9+a_2^2} - a_2\sqrt{9+a_1^2} + 5\,(a_2-a_1)\right). \end{aligned}$$

In the case $a_2 > a_1$ we see that $A>0$. $B>0$ follows if we can show $a_1\sqrt{9+a_2^2} - a_2\sqrt{9+a_1^2} > 0$. First, observe that $a_1\sqrt{9+a_2^2} = a_2\sqrt{9+a_1^2}$ has only one unique solution satisfying $a_1, a_2 \geq 0$, namely $a_1 = a_2$. Let $\phi[x/y]$ denote the substitution of all free occurrences of x in expression ϕ with y. Then, since

$$\left(\frac{\partial}{\partial a_2}\left(a_1\sqrt{9+a_2^2} - a_2\sqrt{9+a_1^2}\right)\right)[a_2/a_1] = \frac{a_1^2}{\sqrt{9+a_1^2}} > 0,$$

we have $B>0$ for $a_2 > a_1$. The case $a_1 < a_2$ is symmetric.

□

Lemma 5 *Using the notation introduced above, the following statement holds: For sufficiently small $\varepsilon > 0$, each σ_j, $\lceil \frac{n}{2} \rceil \leq j < n$, generates two vertices with each edge e_i, $1 \leq i < \lceil \frac{n}{2} \rceil - 1$.*

Proof: Let e be the edge generated by two spheres

$$\begin{aligned} \sigma_{a_1} &= ((x_1,0,0),r_1) = \left((4+(1+\varepsilon)\,a_1,0,0), \sqrt{9-a_1^2}-3\right) \\ \sigma_{a_2} &= ((x_2,0,0),r_2) = \left((4+(1+\varepsilon)\,a_2,0,0), \sqrt{9-a_2^2}-3\right) \end{aligned}$$

together with σ_0. We will show that for small $\varepsilon > 0$ the sphere $\sigma_{n-1} = ((4,6,0),0)$ generates two vertices on e. The lemma follows from the fact that our construction is rotationally symmetric with respect to the x-axis.

An AWV-vertex (x,y,z) at distance d generated by these four spheres is a common solution to the system of equations

$$\begin{aligned} x^2+y^2+z^2-(d+2)^2 &= 0 \\ (x-x_1)^2+y^2+z^2-(d+r_1)^2 &= 0 \\ (x-x_2)^2+y^2+z^2-(d+r_2)^2 &= 0 \\ (x-4)^2+(y-6)^2+z^2-d^2 &= 0. \end{aligned}$$

Using standard transformations, we obtain the solutions

$$\begin{aligned} x &= \frac{(r_1^2-x_1^2-4)(4-2r_2^2)-(r_2^2-x_2^2-4)(4-2r_1^2)}{2x_1(2r_2-4)-2x_2(2r_1-4)} \\ d &= \frac{2(r_1^2-x_1^2-4)x_2-2(r_2^2-x_2^2-4)x_1}{2x_1(2r_2-4)-2x_2(2r_1-4)} \\ y &= \frac{-8x+4d+56}{12} \\ z &= \pm\frac{1}{3}\sqrt{8d^2+8d-13x^2+56x+4xd-160}. \end{aligned}$$

As in the previous lemma, we substitute

$$\begin{aligned} x_1 &\longrightarrow 4+(1+\varepsilon)a_1, \quad r_1 \longrightarrow \sqrt{9+a_1^2}-3 \\ x_2 &\longrightarrow 4+(1+\varepsilon)a_2, \quad x_2 \longrightarrow \sqrt{9+a_2^2}-3 \end{aligned}$$

in the discriminant $\Delta = 8d^2+8d-13x^2+56x+4xd-160$. We have to show, that for any valid parameterization $a_1 \neq a_2 \in (0,1)$ we have $\Delta_{a_1,a_2}(\varepsilon) > 0$ for sufficiently small $\varepsilon > 0$.

Observe, that the common denominator of Δ is the quadratic term

$$(2x_1(2r_2-4)-2x_2(2r_1-4))^2,$$

which is always greater than zero if $a_1 \neq a_2$. Hence, it suffices to show the positivity of the numerator

$$\begin{aligned} f_{a_1,a_2}(\varepsilon) &= 32(\theta_4\theta_2-\theta_3\theta_1)^2+16(\theta_4\theta_2-\theta_3\theta_1)\theta_6-160{\theta_6}^2-13{\theta_5}^2 \\ &\quad +56\theta_5\theta_6+8\theta_5(\theta_4\theta_2-\theta_3\theta_1), \text{ where} \\ &\theta_1 := 4+(1+\varepsilon)a_1 \\ &\theta_2 := 4+(1+\varepsilon)a_2 \end{aligned}$$

$$\theta_3 := \left(\sqrt{9+{a_2}^2}-3\right)^2-{\theta_2}^2-4$$
$$\theta_4 := \left(\sqrt{9+{a_1}^2}-3\right)^2-{\theta_1}^2-4$$
$$\theta_5 := \theta_4\left(10-2\sqrt{9+{a_2}^2}\right)-\theta_3\left(10-2\sqrt{9+{a_1}^2}\right)$$
$$\theta_6 := 2\theta_1\left(2\sqrt{9+{a_2}^2}-10\right)-2\theta_2\left(2\sqrt{9+{a_1}^2}-10\right).$$

Since, by construction, $\Delta_{a_1,a_2}(0)=0$, it suffices to show $\left(\frac{d}{d\varepsilon}f_{a_1,a_2}(\varepsilon)\right)(0)>0$. Using MAPLE we calculate

$$\begin{aligned}
g := \left(\frac{d}{d\varepsilon}f_{a_1,a_2}(\varepsilon)\right)(0) = \\
&-20736\left({a_1}^2+{a_2}^2\right)+6336a_1a_2\left(\sqrt{9+{a_1}^2}a_1+\sqrt{9+{a_2}^2}a_2\right)\\
&-7488a_1a_2\left(\sqrt{9+{a_1}^2}a_2+\sqrt{9+{a_2}^2}a_1\right)\\
&+1152\left(a_1^3\sqrt{9+a_2^2}+a_2^2a_1^2\sqrt{9+a_1^2}+a_2^3\sqrt{9+a_1^2}+a_1^2a_2^2\sqrt{9+a_2^2}\right)\\
&-1152\left(a_1a_2^3\sqrt{9+a_1^2}+a_1^3a_2\sqrt{9+a_2^2}\right)\\
&+2304\left(a_1^2+a_2^2\right)a_2^2\sqrt{9+a_2^2}\sqrt{9+a_1^2}+5760\left(a_1a_2^3+a_1^3a_2\right)\\
&+6912\left(\sqrt{9+a_1^2}a_1^2+\sqrt{9+a_2^2}+a_2^2-\sqrt{9+a_2^2}a_1^2-\sqrt{9+a_1^2}+a_2^2\right)\\
&+576\left((a_1+a_2)a_1a_2\sqrt{9+a_1^2}\sqrt{9+a_2^2}-a_1^2a_2^2\left(a_1+a_2\right)\right)\\
&-8640\left(a_1+a_2\right)a_1a_2+3456\left(a_1^3+a_2^3\right)-16128a_1^2a_2^2
\end{aligned}$$

Figure 2.7 shows a plot of this expression. Such a linear combination of radicals can vanish only if either the coefficients of the different radicals sum up two zero or if the radicals are linear dependent over the rational numbers. The first case can be excluded by substituting values for a_1 and a_2 into this expression. Therefore the expression g vanishes only if $a_1=a_2$. We have

$$\left(\frac{\partial}{\partial a_1}g\right)[a_1/a_2]\equiv 0,$$

this can also be seen in figure 2.7. So, similar to the proof of the previous lemma, we are done if we can show that g is convex for $(a_1,a_2)\in(0,1)\times(0,1)$, i.e. we

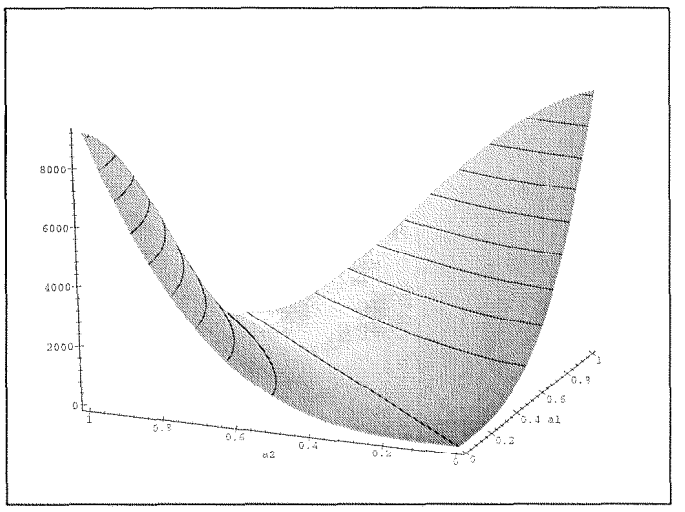


Figure 2.7: A plot of the expression $\left(\frac{d}{d\varepsilon} f_{a_1,a_2}(\varepsilon)\right)(0)$. As it is easily seen, the function vanishes for $a_1 = a_2$ and is positive elsewhere.

have to show that

$$h(a_2) := \left(\frac{\partial}{\partial a_1^2} g\right)[a_1/a_2] > 0 \text{ for all } a_2 \in (0,1).$$

We calculate

$$\begin{aligned} h(a_2) &= 1152 a_2 \Big(108 + 162 a_2 + 14 {a_2}^3 + 6 a_2 \sqrt{9+{a_2}^2} + 36 \sqrt{9+{a_2}^2} \\ &\quad + 13 \sqrt{9+{a_2}^2}\, {a_2}^2 + 21 {a_2}^2\Big) \Big/ \left(9+{a_2}^2\right) \end{aligned}$$

Again using MAPLE, we calculate that this expression has four roots, of which only 0 is contained in the interval $[0,1]$. We evaluate $h(1)$ to verify that indeed $h(a_2) > 0$ for all $a_2 \in (0,1)$. □

Conclusion

In this chapter, we gave a detailed account on the geometry of the edges of AWV cells. We proved a new and tight lower bound of $\Theta(n^2)$ on the combinatorial worst-case complexity of a single AWV cell defined by n spheres in 3 dimensions.

To our best knowledge, the exact worst-case complexity of single AWV cells in odd dimensions $d \geq 5$ and of the complete AWV *diagram* for even dimensions

$d \geq 4$ is still open. The lower bound construction we gave in this chapter might suggest that the AWV diagram can achieve an intrinsically higher complexity in even dimensions $d > 2$ than it is possible for the unweighted diagram. Providing a tight bound on the worst-case complexity of AWV cells and diagrams in higher dimensions might be a challenging problem for future research.

Chapter 3

Computing additively weighted Voronoi cells

3.1 Introduction

From the theoretical point of view, lemma 1 solves the problem to compute the cells of an AWV diagram in $\mathbf{R}^d$ by giving an implicit representation of these cells: According to this lemma, all we have to do is to compute the corresponding power diagram in $\mathbf{R}^{d+1}$, and then to intersect each of the resulting power cells with a $d+1$-dimensional cone.

In the 3-dimensional case, which is the case relevant to our intended applications, the power diagram can be computed in time $O(n^2)$, where n is the total number of spheres, using, for example, the algorithms given by Seidel (1981), Clarkson and Shor (1989), or Chazelle (1993). The second step, i.e. the extraction of an explicit representation of the AWV cells, is the major topic of the present chapter. We will restrict ourselves to the 3-dimensional case.

In section 3.2, we will discuss how we can extract the geometry of an AWV cell directly from the power diagram. This approach makes use of the information encoded in the 2 and 3 dimensional faces of the power diagram. However, the resulting representation is rather inconvenient for visualization or volume computations. Moreover, as we will see in chapter 5, the approach suffers from numerical problems when implemented using floating point arithmetic.

To remedy this, we will discuss in section 3.3 approaches based on spherical parameterizations of the resulting cells. These are much better suited for the appli-

cations we have in mind, but effectively use only the information encoded in the 3-dimensional faces of the power diagram. Because the spheres given by molecular models are nicely distributed in space, the numerical behavior of this approach is very satisfying.

3.2 Direct extraction

3.2.1 Regular patches

Let the letters (x,y,z) denote the coordinates of points in $\mathbf{R}^3$ and let (x,y,z,d) denote the coordinates of points in $\mathbf{R}^4$. Let $S = \{\sigma_i, 1 \leq i \leq n\}$ be a set of spheres in $\mathbf{R}^3$ and let $\Sigma_i = \left((c_{i,1}, c_{i,2}, c_{i,3}, r_i), \sqrt{2}\, r_i\right)$ denote the corresponding lifting of sphere σ_i into $\mathbf{R}^4$ for $1 \leq i \leq n$. Finally, let P_i denote the 4-dimensional power cell of Σ_i, and let V_i denote the AWV cell of σ_i for $1 \leq i \leq n$. Lemma 1 tells us that

$$V_i = \mathrm{proj}_d\left(\kappa_i \cap P_i\right). \tag{3.1}$$

However, in spite of an extensive survey of the computational geometry literature, we could not find a reference to a previous implementation of an algorithm based on this lemma.

Therefore, we have to answer the following two questions:

1. Which data structure is the most suitable to represent the left hand side of equation 3.1?
2. How can we evaluate the right hand side of equation 3.1 efficiently to actually obtain this representation?

The right hand side of equation 3.1 describes the faces of the cell as the projection of the intersections of linear subspaces with a cone. These intersections are most naturally computed using successive elimination of the variables involved. Hence, it seems to be natural to choose a data structure supporting this computation. We represent the boundary of each cell as a set of patches described by a triangular description. Our chosen representation is similar to a *delineation* as defined by Collins (1975). Of course, we do not have to compute a full cylindrical algebraic decomposition of the cell, but rather it is sufficient to compute a thinned-out version for each face individually, very much in the spirit of the stratification scheme for semi-algebraic sets proposed by Chazelle et al. (1991).

Definition 10 (regular patch) *A* regular patch *is defined as a quintuple of values and functions*

$$\begin{array}{lll} x_{\min}, & x_{\max}, & x_{\min}, x_{\max} \in \mathbf{R} \\ y_{\min}(x), & y_{\max}(x), & x \in (x_{\min}, x_{\max}) \\ z(x,y) & & y \in (y_{\min}(x), y_{\max}(x)), \end{array}$$

such that $x_{\min} \leq x_{\max}$, $y_{\min}(x) \to \mathbf{R}$ *and* $y_{\max}(x) \to \mathbf{R}$ *are continuous and* $y_{\min}(x) < y_{\max}(x)$ *for* $x \in (x_{\min}, x_{\max})$, *and* $z(x,y) \to \mathbf{R}$ *is continuous in each point* (x,y) *such that* $x \in (x_{\min}, x_{\max})$ *and* $y \in (y_{\min}(x), y_{\max}(x))$.

For each $1 \leq i \leq n$ we will represent the boundary ∂V_i as a disjoint collection of regular patches $\mathcal{R}_i$ such that $\partial V_i = \bigcup_{R \in \mathcal{R}_i} \mathrm{cl}(R)$. Computing such a representation is not trivial because a mathematical face of an AWV cell can generate several combinatorial faces, each of them possibly containing holes.

3.2.2 Extraction algorithm

The algorithm we propose to compute an explicit representation of an AWV cell from the corresponding 4-dimensional power cell P is a specialization of the algorithm given by Chazelle et al. (1991). The algorithm transforms the defining equations and inequalities describing the faces, edges and vertices of the AVW cell into a triangular form by successive elimination of the variables d, z, and y. Then, in a second step, it selects the relevant patches based on successive substitution and then checking the necessary sign conditions.

To describe the algorithm, we introduce the following notation: We assume that we wish to compute the AWV cell of a sphere $\sigma \in S$ of radius 0 around the origin. We denote by P the 4-dimensional power cell of σ within all the other spheres in S.

Let $f_1, f_2, \ldots, f_{j+1}$ be polynomials in the variables $v_1, \ldots, v_k$, $j < k$. Then we denote by $\mathrm{solve}_{v_1, v_2, \ldots, v_j}(f_1, f_2, \ldots, f_{j+1})$ the function that computes the polynomial g obtained by eliminating the variables $v_1, v_2, \ldots, v_j$ from $f_1, f_2, \ldots, f_{j+1}$. If the input polynomials $f_1, f_2, \ldots, f_{j+1}$ do not describe a complete intersection, then $g = 0$ or $g = 1$, depending on whether there exists a common (complex) solution to these polynomials at all or not. $\mathrm{solve}_{v_1, v_2, \ldots, v_j}(f_1, f_2, \ldots, f_{j+1})$ can be computed using Kronecker's elimination procedure based on resultants, see van der Waerden (1955).

Given a quadratic polynomial $p = av^2 + bv + c$ in a variable v, we denote the discriminant by $\mathrm{disc}_v(p) = b^2 - 4ac$. If $a = 0$ then we set $\mathrm{disc}_v(p) = 1$. A quadratic polynomial $p = av^2 + bv + c$ in a variable v together with a sign $\eta \in \{+,-\}$ can be used to identify its roots. Assume that a, b, c are dependent on variables $v_1, \ldots, v_d$, $v \neq v_i$, for $1 \leq i \leq d$. Then we define $p_v(\alpha_1, \ldots, \alpha_d, \eta)$ as follows: If $a(\alpha_1, \ldots, \alpha_d) \neq 0$ then

$$p_v(\alpha_1, \ldots, \alpha_d, \eta) = \frac{-b(\alpha_1, \ldots, \alpha_d)\eta\sqrt{b(\alpha_1, \ldots, \alpha_d)^2 - 4a(\alpha_1, \ldots, \alpha_d)c(\alpha_1, \ldots, \alpha_d)}}{2a(\alpha_1, \ldots, \alpha_d)},$$

if this number is real. Otherwise we say that $p_v(\alpha_1, \ldots, \alpha_d, \eta)$ does not exist. If $a(\alpha_1, \ldots, \alpha_d) = 0$ then $p_v(\alpha_1, \ldots, \alpha_d, +)$ is the unique solution of the linear equation $c(\alpha_1, \ldots, \alpha_d) - b(\alpha_1, \ldots, \alpha_d)v = 0$ if such a solution exists, and $p_v(\alpha_1, \ldots, \alpha_d, -)$ does not exist at all.

For each j-face f of P let $\mathrm{def}_f(i)$, $0 \leq i < 4 - j$ denote the defining hyperplanes of the support of f. Let κ be the polynomial $x^2 + y^2 + z^2 - d^2$.

Having set up all the required notation, the extraction algorithm can be formulated as follows:

1. Compute a triangulation $\mathcal{T}$ of P.
2. For each $\Delta \in \mathcal{T}$ do:
   (a) For each facet $f \subset \Delta$, $f \subset \partial P$, compute the polynomial equation in the variables x, y, z describing a hyperbolic surface in 3 dimensions
   $$\mathrm{poly}_f = \mathrm{solve}_d(\kappa, \mathrm{def}_f(0)).$$
   (b) Calculate the bivariate quadratic polynomial $\mathrm{disc}_f = \mathrm{disc}_z\left(\mathrm{poly}_f\right)$.
   (c) Calculate the set of boundary curve polynomials in x and y
   $$C = \left\{\mathrm{solve}_{z,d}(\kappa, \mathrm{def}_r(0), \mathrm{def}_r(1)), r \text{ ridge of } \Delta \text{ on boundary of } f\right\}.$$
   (d) Calculate the set of event points
   $$E_C = \left\{p,\ p \text{ is a real solution to } \mathrm{disc}_y(c), c \in C\right\}.$$
   (e) For each pair of polynomials $(p_1, p_2) \in (C \cup \{\mathrm{disc}_f\})^2$ and each pair of signs $(\eta_1, \eta_2) \in \{+,-\}^2$ do:

i. Calculate the set of intersection polynomials

$$I = \left\{ \mathrm{solve}_y(c_1, c_2) : \; c_1 \in C \cup \{\mathrm{disc}_f\}, c_2 \in \{p_1, p_2\} \right\}.$$

ii. Calculate the set of event points

$$E_I = \{x : \; x \text{ is a real solution to some } p \in I\}$$

iii. Let $E = \{\pm\infty\} \cup E_I \cup E_C$ and let $L = \{l_0, \ldots, l_m\}$ be the sequence of all event points in E sorted in increasing order.

iv. Scanning L in increasing order, identify maximal subintervals (l_j, l_k), $0 \leq j < k \leq m$ such that the following conditions are satisfied:

   A. Both $p_1(x, \eta_1)$ and $p_2(x, \eta_2)$ exist for each $x \in (l_j, l_k)$.

   B. $p_1(x, \eta_1) < p_2(x, \eta_2)$ for each $x \in (l_j, l_k)$.

   C. For all $p \in (C \cup \{\mathrm{disc}_f\}) \setminus \{p_1, p_2\}$ and $\eta \in \{+, -\}$ either $p(x, \eta)$ does not exist or $p(x, \eta) \notin (p_1(x, \eta_1), p_2(x, \eta_2))$.

v. For each of the retained subintervals (l_j, l_k) from L and for $\eta \in \{+, -\}$ create a patch

$$\begin{array}{cc} l_j & l_k \\ p_1(\cdot, \eta_1) & p_2(\cdot, \eta_2) \\ \mathrm{poly}_f(\cdot, \cdot, \eta), & \end{array}$$

   if $\mathrm{poly}_f(x, y, \eta)$ does exist as boundary of the AWV cell for all $x \in (l_j, l_k)$, and $y \in (p_1(x, \eta_1), p_2(x, \eta_2))$.

The running time of the preceding algorithm is trivially proportional to the structural complexity of P. Hence, this approach allows us to compute an explicit representation of all AWV cells defined by a set $S = \{\sigma_i, 1 \leq i \leq n\}$ of n spheres in time $O(n^2)$.

However, with regard to our intended applications, this algorithm suffers from two problems: First of all, regular patches are not a very suitable starting point for computing the volume of individual AWV cells. Second, as we we will demonstrate in chapter 5, the algorithm is subject to large numerical errors when implemented using floating point arithmetic.

3.3 Lower envelope algorithms

3.3.1 Subdivisions of the sphere

Since a non-empty cell is a star-shaped region, it is natural to represent its surface using a spherical parameterization, i.e. we parameterize a cell V by a unit sphere around the center c of its defining sphere σ. We denote this unit sphere by S^2. In the following, we assume that c is the origin and that $r = 0$.

Let $\pi : p \to \frac{p}{\|p\|}$ denote the map projecting a point $p \neq c_i$, onto the parameter range S^2. The collection

$$P := \{\pi(f) : f \text{ is a combinatorial 2-face of } V\}$$

is a subdivision of S^2. For any element $x \in P$ let $\phi(x)$ denote its lifting back to the current boundary of the cell. As it is common parlance, we identify a map with its image. According to proposition 4 and lemma 3, the boundaries of the elements of P can be represented as circular arcs on S^2.

Hence, the problem of computing an additively weighted Voronoi cell can be stated as follows: Given a set of spheres $S = \{\sigma_i, 1 \leq i \leq n\}$ we seek for the lower envelope $\min_{1 \leq i \leq n} d_i$ of the functions

$$d_i(p) = \frac{c_i^2 - r_i^2}{2(r_i + \langle p, c_i \rangle)}$$

and the subdivision P it induces on the parameter space S^2. This subdivision can be described using a set of planes intersecting S^2. Figure 3.1 shows an example of this representation. Of course, if we have a description of the 4-dimensional power cell $P(\Sigma_i)$ available that corresponds to V_i when applying Aurenhammer's lifting procedure, then we can restrict the set of spheres to consider in computing V_i to the subset of spheres

$$D_i = \{\sigma_j : \Sigma_j \text{ is facet-defining for } P(\Sigma_j),\ 1 \leq j \leq n\}.$$

3.3.2 Random incremental construction using vertical decomposition

Randomized incremental (RIC) algorithms computing the lower envelope of algebraic surface patches in 3 dimensions were proposed by Mulmuley (1989, 1994)

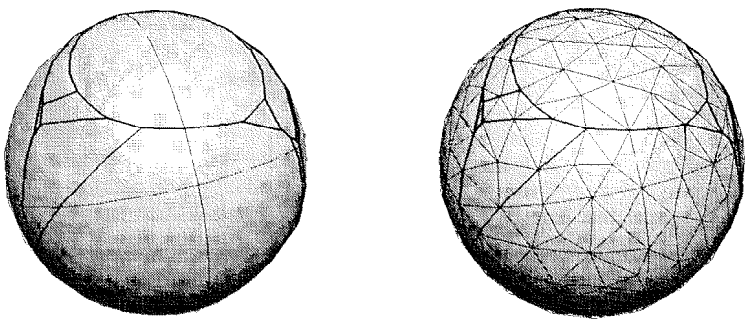


Figure 3.1: A spherical subdivision representing an additively weighted Voronoi cell. The left picture shows the unrefined map, the right picture displays a refined constrained Delaunay triangulation of this map used to produce the renderings of cells throughout this thesis.

and Boissonnat and Dobrindt (1992, 1996). Mulmuley's algorithm is a static RIC algorithm using conflict lists. The algorithm by Boissonnat and Dobrindt, on the other hand, is semi-dynamic and utilizes a history data structure based on trees. The common outline of these two algorithms is as follows:

- The input to these algorithms is a set $S := \{p_1, \ldots, p_n\}$ of bounded algebraic surface patches of constant description complexity. This means, that each patch $p \in S$ is specified as a partially defined function $f_p(x,y)$ for $(x,y) \in D_p$, the domain of p. The algorithms assume f_p to be monotone in x and y. Note, that any algebraic surface patch of fixed maximum degree d can be decomposed into a constant number of monotone patches, where the constant only depends on d. However, the authors only report implementations for triangles, i.e. patches defined by linear functions.

- The algorithms represent the lower envelope

$$\left(\min_{1\le i\le n} p_i\right)(x,y) = \min_{1\le i\le n} p_i(x,y)$$

  using a trapezoidal decomposition T of the x,y plane, such that for each trapezoid $t \in T$ there is a unique $p_t \in S$ satisfying

$$\forall (x,y) \in t: \ \min_{1\le i\le n} p_i(x,y) = p_t(x,y).$$

  Each trapezoid is bounded to the left and to the right by line segments parallel to the y-axis, and is bounded to the top and to the bottom by an x-monotone algebraic curve of bounded degree. In the case of computing the lower envelope of triangles, these curves are line segments.

Two trapezoids t_1 and t_2 are considered to be neighbors in T if they share a common vertical slab and are bounded by the same curve either at the top or at the bottom. Incidence information is stored only between trapezoids being neighbors in this strict sense.

In the context of spherical subdivisions as needed for the computation of a single AWV cell, we use the following definition of a trapezoid on a sphere as given by Halperin and Shelton (1997, 1998): Fix a pair of antipodal points on the sphere as *poles*. We call the great circles through the poles *polar circles* and arcs of polar circles *polar arcs*. For an arbitrary circle c on the sphere we call each of the two points of c that are tangent to a polar circle a *polar tangency*. A trapezoid is bounded to the left and to the right by polar circles, and the top and the bottom are circular arcs, which may degenerate to one of the poles.

- The algorithms start with an empty decomposition T_0 and add one patch $p \in S$ after another in random order π. In this manner, they compute a sequence of trapezoidal decompositions $T_0, T_1, \ldots, T_n$, such that T_i is the trapezoidal decomposition representing the lower envelope defined by $p_{\pi(1)}, \ldots, p_{\pi(i)}$, $1 \leq i \leq n$. They check for the following conflict types to determine which trapezoids of T_{i-1} have to be updated when adding the i-th patch $p := p_{\pi(i)}$:

  1. *Vertex conflicts*: Let v be a vertex of a trapezoid $t \in T_{i-1}$. v conflicts p if $p_t(v) > p(v)$.
  2. *Edge conflicts*: Let e be the top or the bottom boundary of a trapezoid $t \in T_{i-1}$. e conflicts p if there are points $(x_1, y_1), (x_2, y_2) \in e$ such that $p_t(x_1, y_1) > p(x_1, y_1)$ and $p_t(x_2, y_2) < p(x_2, y_2)$.
  3. *Face conflicts*: t conflicts p if $D_p \subset t$ and there exists a point $(x, y) \in t$ such that $p_t(x, y) > p(x, y)$.

Mulmuley's algorithm propagates this conflict information after each step using conflict lists, i.e. it explicitly stores this information between all trapezoids and all patches not yet added. The algorithm by Boissonnat and Dobrindt, on the other hand, traverses the history graph associated with $T_0, \ldots, T_i$ to determine this information right before performing the insertion of the i-th patch. The history graph is obtained by linking a trapezoid t_1 in T_j to a trapezoid t_2 in T_{j+1}, $0 \leq j < i$, when t_2 was created because of a conflict between t_1 and $p_{\pi(j+1)}$.

The authors give the following bounds on the running times of their algorithms: The first and older bound by Mulmuley is based on θ-series, which try to capture the depth structure of the input objects:

$$\theta_a(s) = \sum_l \frac{v_a(l)}{l^s},$$

where $v_a(l)$ is the number of *junctions* of degree a at *level* $l-1$. Using our previous definitions, a junction can be defined as follows: Consider each of the different conflicts which might be generated for some specific order of insertion of the patches. Each conflict c is located at a specific point (x,y) in the plane (for each face conflict we can choose a representative). The collection of all these points for all possible orders of insertion is the set of junctions. The degree of a junction j is the number of input objects needed to define j. The level of a junction j is the number of patches which prevent j from being part of the final output T_n. Using these definitions, Mulmuley proved the following bound:

Theorem 2 (Mulmuley (1994a)) *The total expected time taken by the algorithm is* $O(n\log^2 n + \theta_1(1) + \theta_3(2) + \theta_2(2)\log n + \theta_3(3)\log n)$.

Boissonnat and Dobrindt, on the other hand, give their bound in terms of the expected complexity of the trapezoidal decomposition generated by a random sample of the input objects. They remark that they get the same bound as given by Mulmuley when doing the analysis in terms of θ-series.

Theorem 3 (Boissonnat and Dobrindt (1996)) *The lower envelope of a set of* n *surface patches in* $\mathbf{R}^3$ *satisfying the above conditions can be constructed in* $O(n\log n\sum_{r=1}^{n}\tau(r)/r^2)$ *expected time with* $O(\sum_{r=1}^{n}\tau(r)/r)$ *expected space, where* $\tau(r)$ *is the expected complexity of the lower envelope of* r *surface patches. Furthermore, the insertion of the* n*-th surface patch can be done in* $O(\log n\sum_{r=1}^{n}\tau(r)/r^2)$ *expected time.*

When computing a single AWV cell, a tight worst-case upper bound on $\tau(n)$ is $O(n^2)$, as shown in the previous chapter. In this case, the bound on the expected time for computing an AWV cell defined within n other spheres simplifies to $O(n^2\log n)$.

3.3.3 Non-vertical refinement

Halperin and Shelton (1997,1998) reported numeric problems when using trapezoidal decompositions to represent molecular surfaces. Besides a small number of degeneracies necessarily present due to the input data, these problems result from the introduction of polar arcs through polar tangency points, which are needed to define the left and right boundary arcs of certain trapezoids. They reported that the angles between these polar arcs may be very small, such that sorting these arcs along the equator suffers from large numerical errors when done using floating point arithmetic. Hence, we will describe another data structure that does not rely on the choice of a particular set of poles. In chapter 5, we will examine these issues in an experimental setup.

Let $\tau(r;n)$ denote a function bounding the expected complexity of a single additively weighted Voronoi cell $V(\sigma)$ defined by r out of n input spheres. As a shorthand notation, we leave out the second argument value for the parameter n and simply write $\tau(r)$, if the argument value is implicitly given in the context or if $\tau(r;n)$ does not depend on n. As shown in the previous chapter, a tight worst-case upper bound on $\tau(r)$ is $O(r^2)$. On the other hand, the interior of a protein possesses a packing density comparable to crystal structures[1]. Probabilistic models of crystals and quasi-crystals based on additively weighted Voronoi cells suggest a constant bound on the expected complexity of such a cell, regardless of n and r^2. Hence, for families of restricted data sets, $\tau(r;n)$ might be considerably lower than the worst-case bound.

Our goal is to formulate an algorithm computing $V(\sigma)$ with a running time tightly dependent on $\tau(r) = \tau_f(r) + \tau_e(r) + \tau_v(r)$, $\tau_f(r)$, $\tau_e(r)$ and $\tau_v(r)$ bounding the expected number of faces, edges and vertices of an additively weighted Voronoi cell amidst r out of n other spheres, respectively. To simplify the presentation, we assume that the defining sphere σ is centered at the origin.

Similar to the algorithms discussed in the previous section, our new algorithm works incrementally by adding the spheres σ_i from the input set $S = \{\sigma_i, 1 \leq i \leq n\}$ in their given order. At each step i, the algorithm maintains a subdivision P_i of S^2 that describes the lower envelope $\min_{1 \leq i \leq n} d_i$ of the functions

$$d_i(p) = \frac{c_i^2 - r_i^2}{2(r_i + \langle p, c_i \rangle)}$$

[1] Cf. Kyte (1995)

[2] Cf. Meijering (1953), Møller (1992)

defined by $\sigma_1, \ldots, \sigma_i$. For the analysis, we will turn this algorithm into a randomized one by randomly permuting S in the beginning.

Similar to the algorithm by Mulmuley, our new algorithm maintains a set of conflicts C_i. Again, C_i is a relation between the combinatorial elements of P_i, i.e. the vertices, edges, and faces, and all sites from the set $\bar{S}_i = S \setminus S^{(i)}$, $S^{(i)} = \{\sigma_1, \ldots, \sigma_i\}$.

The basic ideas behind our approach are the following:

- We want to analyze our algorithm in the framework by Clarkson and Shor (1989). This framework requires that each object that our algorithm creates to represent its output is defined in terms of at most a constant number of input sites. Hence, we have to refine the faces of the spherical subdivision representing the boundary of the AWV cell into elements of constant description complexity. The previous algorithms achieved this using trapezoids. We take a different approach:

  Let f be a combinatorial face of the cell $V(\sigma)$. The projection of f onto the parameter space S^2 can be represented as the intersection of a convex polytope H_f with S^2. Instead of using H_f directly, we use a triangulation of H_f. Obviously, the intersection of each simplex Δ of this triangulation with S^2 has a description of constant combinatorial size.

- It may happen that in the course of the algorithm the apex used to triangulate one of these polytopes (or only a facet of them) is cut off. Then the complete polytope (or at least the part next to the respective facet) must be re-triangulated. Therefore, the sizes of these polytopes have to be taken into account in the analysis of the algorithm.

- The polytopes may accumulate redundant parts in the course of the algorithm: They might accumulate edges and facets that do not intersect S^2 at all. Eliminating redundant defining halfspaces from the polytopes after each insertion of a site is too expensive. Therefore, at steps $i = 2^k$, $1 \leq k \leq \lfloor \log n \rfloor$, the algorithm performs a cleanup operation on P_i. This cleanup guarantees a deviation of the combinatorial complexity of the polytopes from the combinatorial complexity of the spherical subdivision by at most a constant factor throughout the course of the algorithm.

We will now discuss our approach in detail.

The subdivision. We describe our algorithm restricted to computing $V(\sigma)$ within the first octant. Eight similar copies will compute $V(\sigma)$.

For each $1 \leq i \leq n$, let $\phi_i(p)$ denote the partially defined function $\phi_i(p) : p \mapsto p \cdot d_i(p)$ describing the bisector surface between σ and σ_i, and let ϕ_i denote the bisector surface itself. In addition, the algorithm uses a symbolic value of ϕ_0 to represent the unbounded part of the cell.

For $1 \leq i \leq n$, the subdivision P_i is represented by a collection of polytopes $H_j(i)$, $1 \leq j \leq i$, such that $\phi_j(H_j(i) \cap S^2) = \{x \in \partial V : \ d(x,\sigma) = d(x,\sigma_j)\}$. $H_0(i)$ represents the unbounded portion of the cell. Set $\Delta_0 = \{(x,y,z) \in R^3 : x \geq 0, y \geq 0, z \geq 0, x+y+z \leq 3\}$. For $1 \leq j \leq i \leq n$ let $I_j(i)$ be the minimum subset of $\{0,\ldots,i\}$ identifying non-redundant halfspaces, i.e.

$$S^2 \cap \Delta_0 \cap \bigcap_{0 \leq k \leq i, k \neq j} h_{j,k} = S^2 \cap \Delta_0 \cap \bigcap_{k \in I_j(i)} h_{j,k}.$$

Define $H_i(i) = \Delta_0 \cap \bigcap_{k \in I_i(i)} h_{i,k}$. Recursively we define for $0 \leq j < i$

$$H_j(i) = \begin{cases} \Delta_0 \cap \bigcap_{k \in I_j(i)} h_{j,k} & \text{if } i \text{ is a power of 2} \\ H_j(i-1) \cap h_{j,i} & \text{if } S^2 \cap H_j(i-1) \cap h_{j,i} \neq S^2 \cap H_j(i-1) \\ H_j(i-1) & \text{otherwise} \end{cases}$$

That is, either $H_j(i)$ is the result of a cleanup operation, or it was changed due to some conflict.

At each step i, the algorithm maintains a canonical triangulation of all $H_j(i)$, $0 \leq j \leq i$, for which $H_j(i) \cap S^2 \neq \emptyset$:

Definition 11 *Let $H \subset R^3$ be a convex polytope and $\vec{a} \in R^3$ an arbitrary but fixed direction. For each facet $f \subset H$, choose the minimum vertex $v_{\min}(f)$ with respect to direction $\vec{a}$ and triangulate f towards $v_{\min}(f)$. The* canonical triangulation[3] *is obtained as the collection of all 3-dimensional simplices being the convex hull of one of these triangles and $v_{\min}(H)$, the minimum vertex of H with respect to $\vec{a}$.*

Let $\mathcal{T}_j(i)$ denote the collection of all those simplices of $H_j(i)$. We store adjacency information between all simplices $\Delta \in \mathcal{T}_j(i)$, but not between simplices from different sets $\mathcal{T}_j(i)$ and $\mathcal{T}_k(i)$, $j \neq k$. The subdivision P_i is obtained as the collection

[3]Apparently, the present algorithm also introduces an additional prescribed direction. However, contrary to the situation when constructing tangent planes to circular arcs as it occurs in vertical decomposition schemes, ties with respect to $\vec{a}$ can be broken very easily using, say, a lexicographical ordering of the coordinates, and we do not have to deal with algebraic numbers (or their approximations).

of all intersections of these simplices with S^2. Let $\Delta \in \mathcal{T}_j(i)$ be a simplex obtained by lifting a triangle t of the triangulation of a facet f of $H_j(i)$ towards the minimal vertex $v_{\min}\left(H_j(i)\right)$. Then we set $v_{\min(f)}(\Delta) = v_{\min}(f)$ and $v_{\min}(\Delta) = v_{\min}\left(H_j(i)\right)$.

Search structures. To introduce new conflicts efficiently into our data structure, we use the following two data structures: On each facet $f \in H_j(i)$, $0 \leq j \leq i$, we maintain a binary tree search structure for point location within f. Using red-black trees[4], this structure can be constructed in time linear in the number of vertices $|f|$, queries can be answered in time $O(\log |f|)$, and once the location of insertion is known, the structure can be updated in amortized constant time.

At certain steps, namely if $v_{\min}(H_j(i)) \neq v_{\min}(H_j(i-1))$, we establish a *static* point location structure for the complete polytope $H_j(i)$ using the following result[5]:

Theorem 4 (Kirkpatrick (1983)) *For any n vertex planar graph, one can build a point location structure of $O(n)$ size in $O(n \log n)$ time, guaranteeing $O(\log n)$ query time.*

So, given a polytope H with n facets and a set M of m points, we can determine for each point $p \in M$ the simplex Δ of the canonical triangulation of H such that $p \in \Delta$, or verify that no such Δ exists in time $O((n+m)\log n)$.

Conflict information. The algorithm maintains three kinds of conflicts that are associated with the vertices, edge fragments and face fragments of all $\Delta \in \bigcup_{j=0}^{i} \mathcal{T}_j(i)$:

1. Vertex conflicts: A vertex $v \in P_i$ conflicts with $\sigma_j \in \bar{S}_i$ if ϕ_j will cut the vertex $\phi(v)$ off the cell.

2. Edge conflicts: An edge $e \in P_i$ conflicts with $\sigma_j \in \bar{S}_i$ if ϕ_j intersects $\phi(e)$. We maintain a distinct conflict for each point of intersection. For a single edge e the set of all conflicts $\{(e, \cdot)\}$ is linearly ordered along e. This allows us to split an edge e in constant time regardless of the number of conflicts allocated to it.

[4]Cf. Guibas and Sedgewick (1978)

[5]There are several other algorithms for planar point location problems, such as Edelsbrunner et al. (1986), Sarnak and Tarjan (1986), or Adamy and Seidel (1998). However, all these methods are based on monotone subdivisions.

3. Face conflicts: A face $f \in P_i, \phi(f) \subset \phi_k$ for some $0 \leq k \leq i$ conflicts with $\sigma_j \in \bar{S}_i$ if $\phi_k \cap \phi_j \neq \emptyset$ and $\phi_k \cap \phi_j \subset \text{int}(\phi(f))$. For each $\Delta \in P_i$, each face conflict is represented by a point contained in the conflicting region $\Delta \cap S^2$.

We make the following assumptions concerning general position: Any edge conflict is defined as the projection of the intersection of exactly three surfaces ϕ_i, ϕ_j, ϕ_k, $0 \leq i < j < k \leq n$, and all points of intersection are not points of tangency.

Initialization. The algorithm begins by constructing an unbounded cell represented by $H_0(0) = \Delta_0$, $\mathcal{T}_0(0) = \{\Delta_0\}$ and setting $P_0 = S^2 \cap \Delta_0$. For each $1 \leq i \leq n$ all conflicts of σ_i with respect to P_0 are calculated. All this can be done in time $O(n \log n)$ and space $O(n)$.

Update step. The i-th update step when adding sphere σ_i to P_{i-1} resulting in P_i is as follows: Let

$$\begin{aligned} C_{i-1,j}(\sigma_i) &= \{\Delta : \Delta \in \mathcal{T}_j(i-1), \sigma_i \text{ conflicts } \Delta\} \\ C_{i-1}(\sigma_i) &= \bigcup_{j=1}^{i-1} C_{i-1,j}(\sigma_i) \\ A_{i-1}(\sigma_i) &= \{j : 1 \leq j < i, \exists \Delta \in \mathcal{T}_j(i-1), \sigma_i \text{ conflicts } \Delta\} \end{aligned}$$

$C_{i-1,j}(\sigma_i)$ is the set of simplices of the j-th polytope that have to be changed due to the addition of σ_i. $C_{i-1}(\sigma_i)$ is the collection of all these simplices, and $A_{i-1}(\sigma_i)$ the the index set of all polytopes that have to be changed. For each $j \in A_{i-1}(\sigma_i)$ we compute the polytope $H_j(i)$ from $H_j(i-1)$ using the conflict information $C_{i-1,j}(\sigma_i)$. We re-triangulate the updated part of $H_j(i)$ and update the search structures on the facets. If no minimal vertex $v_{\min}(f)$ of a facet $f \subset H_j(i-1)$ or even the total minimal vertex $v_{\min}(H_j(i-1))$ is deleted then this update can be performed within $O\left(|C_{i-1,j}(\sigma_i)| \log(n)\right)$ time.

For each conflict c associated with a deleted simplex in $\mathcal{T}_j(i-1)$, we can determine in constant time whether it remains a conflict for a simplex $\Delta \in \mathcal{T}_j(i)$, and, if so, reinsert it as conflict in the updated structure in time $O(\log n)$. Conflicts that are located on a new face that is created for site σ_i are collected into a set C_{new}.

If we delete a minimum vertex $v_{\min}(f)$ of a facet $f \subset H_j(i-1)$, then we re-triangulate f and reallocate all affected conflicts to their new locations. This can

be done in time $O(|f|+m\log n)$, m being the number of conflicts to be reallocated, and requires no additional space.

Similarly, if the vertex $v_{\min}(H_j(i-1))$ happens to be deleted, then we triangulate $H_j(i)$ from scratch. We compute the point location structure described above and use it to assign all conflicts associated with any $\Delta \in \mathcal{T}_j(i-1)$ to their new locations. This amounts to $O(|H_j(i)|\log n + m\log n)$ time and temporarily requires space $O(|H_j(i)|)$.

If $C_{i-1}(\sigma_i) \neq \emptyset$, we compute the polytope $H_i(i)$, its canonical triangulation, the subdivision P_i, and the point location structures. Note, that exactly sites of simplices in $C_{i-1}(\sigma_i)$ have facet defining halfspaces $h_{i,j}$ for $H_i(i)$. This amounts to time requirements $O(|C_{i-1}(\sigma_i)|\log n)$ and $O(|C_{i-1}(\sigma_i)|)$ space. Then for each conflict $c \in C_{\text{new}}$, we find $\Delta \in \mathcal{T}_i(i)$ such that $c \in \Delta$ in $O(n)$ time, and check if its corresponding site σ_k, $k > i$, associated with c still conflicts Δ. If so, we allocate this conflict to Δ and visit recursively all neighboring simplices, as long as they have not been visited yet and provided they also conflict with σ_k. This traversal can be charged onto the number of newly created conflicts times a factor of $O(\log n)$ for the sorted insertion.

Finally, if $i \in \{2^k, 1 \leq k \leq \lfloor \log n \rfloor\}$, we perform a cleanup operation. For each polytope $H_j(i)$, such that $H_j(i) \cap S^2 \neq \emptyset$, we determine the set $I_j(i)$. We compute the $H_j(i)$, $0 \leq j \leq i$, their triangulations, the point location structures, and we reallocate all conflicts to their new locations.

Probabilistic analysis. We want to analyze the expected work performed by the algorithm if the σ_i are inserted in random order. We do this, as it is common practice, in the manner described by Clarkson and Shor (1989) and Seidel (1993).

For $R = \{\sigma_{i_1}, \ldots, \sigma_{i_r}\} \subset S$ let $f(R) = \sum_{i=1}^{r} |\mathcal{T}_{i_j}(r)|$ be the total complexity of the representation of the cell defined by R. Let

$$f_r = \frac{1}{\binom{n}{r}} \sum_{R \subset S, |R|=r} f(R)$$

be the expectation of this value. In a first step, we will analyze the behavior of our algorithm in terms of n and f_r. Then we will bound f_r in terms of $\tau(r)$.

Proposition 5 *Let $S = \{\sigma_i, 1 \leq i \leq n\}$. Then the expected total number of conflicts created or reallocated by the above algorithm when adding the elements from S*

in random order is bounded by

$$11(n-1)f_1 + \frac{11}{n+1}f_{n+1} + 110n\sum_{r=1}^{n}\frac{f_{r+1}}{r(r+1)} - 121\sum_{r=1}^{n}\frac{f_{r+1}}{r+1}.$$

Proof: Observe, that every simplex of our subdivision is defined by at most 11 spheres. Setting the parameter $d = 11$ in the generic analysis due to Clarkson[6], we obtain the claimed bound. □

Proposition 6 *Let $S = \{\sigma_i, 1 \leq i \leq n\}$. Then the expected total number of simplices created due to re-triangulation operations by the above algorithm when adding the elements from S in random order is bounded by*

$$6\sum_{r=1}^{n}\frac{f_r}{r}.$$

Proof: As in the proof of the previous proposition we apply backwards analysis. Let Δ be a simplex that is created during step r of the algorithm because either one facet or a complete polytope requires re-triangulation. Running the algorithm backwards, this is equivalent to the situation that removing σ_r from the cell at step r would delete at least one of the two minimal vertices $v_{\min(f)}(\Delta)$ or $v_{\min}(\Delta)$. Each of these vertices is defined by at most 3 sites from S. Therefore, the expected value T_r of the number of simplices created due to re-triangulation in step r is bounded by

$$T_r \leq \frac{1}{\binom{n}{r}}\sum_{R\subset S,|R|=r}\frac{6}{r}f(R) = \frac{6}{r}f_r.$$

Summing up for $r = 1\ldots n$ we obtain the claimed bound. □

Because the functions $\tau_f(r)$ and $\tau_e(r)$ might not be monotone increasing in r, we set $\hat{\tau}_X(r) = \max_{1\leq i\leq r}\tau_X(i)$ for $X \in \{e, f, v, \cdot\}$.

Proposition 7 *Let $S = \{\sigma_i, 1 \leq i \leq n\}$. Then at each step r of the algorithm the following bound holds:*

$$f_r \leq 2\hat{\tau}_f(r) + 8\hat{\tau}_e(r)$$

Proof: Let f be the number of faces and e the number of edges of an additively weighted Voronoi cell V. Then, by Euler's relation, the number of simplices

[6]Cf. Seidel (1993)

needed to represent V is bounded by $f+4e$. Therefore, if r is a power of 2, the cleanup operation guarantees $f_r \leq \tau_f(r)+4\tau_e(r)$. Otherwise, let $r^* = \lfloor \log r \rfloor$ be the index of the latest cleanup. Any creation of a new polytope is caused by the creation of at least one new face of the cell, and any addition of a halfspace to a polytope is caused by the creation of at least one new edge of the cell. Hence,

$$\begin{aligned} f_r &\leq \tau_f(r^*)+4\tau_e(r^*)+\sum_{i=r^*+1}^{r}\frac{\tau_f(i)+4\tau_e(i)}{i} \\ &\leq \tau_f(r^*)+4\tau_e(r^*)+\sum_{i=r^*+1}^{r}\frac{\hat{\tau}_f(r)+4\hat{\tau}_e(r)}{r^*+1} \\ &= \tau_f(r^*)+4\tau_e(r^*)+\frac{r-r^*}{r^*+1}\left(\hat{\tau}_f(r)+4\hat{\tau}_e(r)\right) \\ &\leq 2\hat{\tau}_f(r)+8\hat{\tau}_e(r) \end{aligned}$$

□

Theorem 5 *Let $S=\{\sigma_i, 1\leq i\leq n\}$. Then the described algorithm computes the additively weighted Voronoi cell of a sphere σ in expected time*

$$O\left(\sum_{r=1}^{n}\frac{n+r}{r^2}\hat{\tau}(r)\log n\right).$$

Observe, that the time requirements are simply the space requirements times a factor of $\log n$.

Proof: According to propositions 5 and 7 the expected total number of conflicts created is $O\left(n\sum_{r=1}^{n}\frac{\hat{\tau}(r)}{r^2}\right)$. Similarly, as shown in proposition 6, the expected total number of simplices created due to re-triangulation is bounded by $O\left(\sum_{i=1}^{r}\frac{\hat{\tau}(r)}{r}\right)$. Hence, the expected total computational effort required by these steps is bounded by

$$O\left(\sum_{r=1}^{n}\frac{n+r}{r^2}\hat{\tau}(r)\log n\right).$$

A cleanup operation at step i, i a power of 2, obviously involves a subset of all operations performed by the algorithm up to step i. Therefore, the expected total computational effort required for cleanup is bounded by the sum

$$O\left(\sum_{k=1}^{\lfloor \log n\rfloor}\sum_{r=1}^{2^k}\frac{n+r}{r^2}\hat{\tau}(r)\log n\right).$$

□

Corollary 1 *The described algorithm computes the additively weighted Voronoi cell of a sphere σ amidst n other spheres in expected time $O(n^2 \log n)$.*

For restricted families $\mathcal{F}$ of input sets, such as all configurations of spheres arising from molecular models of globular proteins, we might be able to give a much better bound for $\tau(r;n)$.

Corollary 2 *If the function $\tau(r)$ is at most linear in r, i.e. if $\tau(r) = O(r)$, then the algorithm accomplishes the computation in expected running time $O(n \log^2 n)$.*

Proof: Observe that $\tau(r) = O(r)$ implies $\hat{\tau}(r) = O(r)$. So assume that there exist $r_0 \in \mathbf{N}$ and $a \geq 1$ such that for all $r > r_0$ we have $\hat{\tau}(r) \leq ar$. Then

$$\begin{aligned}
\sum_{r=1}^{n} \frac{n+r}{r^2} \hat{\tau}(r) \log n &\leq C_{a,r_0} + \sum_{r=r_0+1}^{n} \frac{n+r}{r^2} ar \log n \\
&= C_{a,r_0} + a \log n \sum_{r=r_0+1}^{n} \frac{n+r}{r} \\
&= C_{a,r_0} + an(H_n - H_{r_0}) \log n + a(n - r_0) \log n,
\end{aligned}$$

where C_{a,r_0} is a suitably chosen constant, and H_n denotes the n-th harmonic number. □

Chapter 4

A practical algorithm for the computation of a single additively weighted Voronoi cell

4.1 Introduction

To begin, let us recapitulate the algorithm presented in the previous chapter. Our major effort concentrated on guaranteeing the bounded description complexity of the individual fragments of the subdivision, which we needed for the probabilistic analysis. To achieve this, we introduced a triangulation of the polytopes H_i describing the faces defined by the spheres σ_i, $1 \leq i \leq n$. On these polytopes we introduced point location structures to insert new conflicts efficiently. These data structures could be damaged in the course of the algorithm. The necessary rebuild steps of these data structures require an effort depending on the complexity of the polytopes. Hence, we came up with clean-up operations to establish a tight dependency of the running time of the algorithm on the combinatorial complexity of the AWV cell.

As it turns out, the AWV cells occurring within our intended domain of application behave much better, i.e. they exhibit only a rather moderate combinatorial complexity. We will examine this behavior in the discussion of our experimental results in chapter 5.

In the present chapter, we describe an algorithm that exploits this low complexity. Again, the algorithm will be a randomized incremental algorithm. Since we

assume that the polytopes H_i have a moderate complexity, there is no need to triangulate them. In fact, we will go even further: We will not work with polytopes at all, but will rather work directly using a spherical subdivision data structure describing the partition P of the parameter space S^2. As we will see, this representation is suited very well for further processing steps of the computed cell. However, we do not simplify the algorithm in every respect by this design decision: In the algorithm of the previous chapter we did not care whether the intersection of a simplex $\Delta \in \mathcal{T}_i$ with the parameter space S^2 yielded one or more connected components. In the algorithm to be described in this chapter, on the other hand, we will have to distinguish the different components of each $H_{i,j} \cap S^2$.

The outline of this chapter is as follows: First, we will introduce the geometric primitives employed by the algorithm. Then, we will introduce the data structures used by the new algorithm to represent the subdivision P. After that, we will describe the algorithm itself in detail. Finally, we discuss the further processing steps of the computed cells that are necessary for volume computations and visualization.

4.2 Geometric primitives

The algorithm is formulated in terms of geometric primitives based on oriented geometry[1]. During the time the algorithm was implemented, Andrade and Stolfi (1998) presented how to evaluate these predicates exactly with purely rational operations only. Currently, our implementation does not use these exact predicates but rather relies on controlled floating point arithmetic with dynamic error bounds to trigger perturbation operations. Nonetheless, the following operations are specified in the notation used in the paper by Andrade and Stolfi, if only as a convenience to the reader.

Oriented geometry. The algorithm works in 3-dimensional projective space $\mathbf{P}^3$. Each point $p \in \mathbf{P}^3$ can be represented by its four coordinates $[w,x,y,z]$. If $w \neq 0$, this corresponds to the point $(x/w, y/w, z/w) \in \mathbf{R}^3$. If $w = 0$, then p represents the point at infinity in direction (x,y,z).

We represent a halfspace by an oriented plane $\alpha = \langle \alpha_0, \alpha_1, \alpha_2, \alpha_3 \rangle$. Let p be a point with homogenous coordinates $[w,x,y,z]$. We say that p is on the positive

[1] Cf. Stolfi (1991)

side of α if $\alpha_0 w + \alpha_1 x + \alpha_2 y + \alpha_3 z > 0$. An oriented line l can be represented by six Plücker coordinates $\langle l_0, l_1, l_2, l_3, l_4, l_5 \rangle$. A 6-tuple of coordinates, in turn, represents an oriented line if and only if $l_0 l_5 - l_1 l_4 + l_2 l_3 = 0$. We call the point at infinity $\mathrm{dir}(l) = [0, l_5, -l_4, l_2]$ the direction of l. We notate the halfspace oriented opposite to α as $\neg\alpha$, and line oriented opposite to l as $\neg l$.

Let $p = [p_0, p_1, p_2, p_3]$ and $q = [q_0, q_1, q_2, q_3]$ be two points, $\alpha = \langle \alpha_0, \alpha_1, \alpha_2, \alpha_3 \rangle$ and $\beta = \langle \beta_0, \beta_1, \beta_2, \beta_3 \rangle$ be two planes, and $l = \langle l_0, l_1, l_2, l_3, l_4, l_5 \rangle$ a line. Then the basic operations $\wedge$ (meet) and $\vee$ (join) are defined as

$$\begin{aligned}
\alpha \wedge \beta &= \langle \alpha_0\beta_1 - \alpha_1\beta_0, \alpha_0\beta_2 - \alpha_2\beta_0, \alpha_1\beta_2 - \alpha_2\beta_1, \\
&\quad \alpha_0\beta_3 - \alpha_3\beta_0, \alpha_1\beta_3 - \alpha_3\beta_1, \alpha_2\beta_3 - \alpha_3\beta_2 \rangle \\
l \wedge \alpha &= [-l_2\alpha_3 + l_4\alpha_2 - l_5\alpha_1, l_1\alpha_3 - l_3\alpha_2 + l_5\alpha_0, \\
&\quad -l_0\alpha_3 + l_3\alpha_1 - l_4\alpha_0, l_0\alpha_2 - l_1\alpha_1 + l_2\alpha_0] \\
p \vee l &= \langle l_0 p_1 + l_1 p_2 + l_3 p_3, -l_0 p_0 + l_2 p_2 + l_4 p_3, \\
&\quad -l_1 p_0 - l_2 p_1 + l_5 p_3, -l_3 p_0 - l_4 p_1 - l_5 p_2 \rangle \\
p \vee q &= \langle p_2 q_3 - p_3 q_2, p_3 q_1 - p_1 q_3, p_0 q_3 - p_3 q_0, \\
&\quad p_1 q_2 - p_2 q_1, p_2 q_0 - p_0 q_2, p_1 q_1 - p_1 q_0 \rangle
\end{aligned}$$

Given an oriented line $l = \langle l_{01}, l_{02}, l_{12}, l_{03}, l_{13}, l_{23} \rangle$, we define $*l = \langle l_{23}, l_{13}, l_{03}, l_{12}, l_{02}, l_{01} \rangle$.

Oriented geometry on the sphere. As we have shown in lemma 3, each halfedge of the spherical subdivision describing the cell can be represented by an oriented circular arc. We call an oriented circle on S^2 an *S-circle*. Each S-circle c can be represented by an oriented plane α such that $c = S^2 \cap \alpha$ and c is oriented positively with respect to the normal of α. This fact can be notated as $c = \mathrm{scrc}(\alpha)$. α is called the *supporting plane* of c, and this is written as $\alpha = \mathrm{spln}(c)$. Hence, c can be represented by the coefficients of the plane α. If $\alpha \cap S^2 \neq \emptyset$, i.e. $\alpha_0^2 \leq \alpha_1^2 + \alpha_2^2 + \alpha_3^2$, we write $\mathrm{scrc}(\alpha) = (\!(\alpha_0, \alpha_1, \alpha_2, \alpha_3)\!)$. The direction orthogonal to $\mathrm{spln}(c)$ and pointing to its positive side is called the *normal* of c, and is denoted by $\mathrm{snrm}(c)$. We think of $\mathrm{snrm}(c)$ as the point at infinity $[0, \alpha_1, \alpha_2, \alpha_3]$. The *S-center* of c is its center on the sphere, i.e. the point $\left[\sqrt{\alpha_1^2 + \alpha_2^2 + \alpha_3^2}, \alpha_1, \alpha_2, \alpha_3\right]$. See figure 4.1 for an illustration of these concepts.

Let p and q be two points on an S-circle $c = \mathrm{scrc}(\alpha)$. Then p and q divide c into two connected parts called *S-arcs*. We define the *S-arc from p to q on c*, denoted

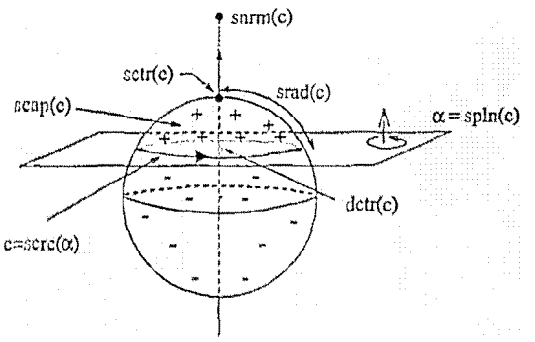


Figure 4.1: The elements of an S-circle. Picture taken from Andrade and Stolfi (1998).

by $\mathrm{sarc}(p,q,c)$, as the set of points encountered on c as we move starting at p in the positive direction along c until we reach q. Given $A = \mathrm{sarc}(p,q,c)$, we write $c = \mathrm{scrc}(A)$, $p = \mathrm{org}(A)$ and $q = \mathrm{dst}(A)$. If $\beta = p \vee q \vee \mathrm{snrm}(c)$, then A is the part of c on the positive side of β.

Let a and b be two S-circles with $\mathrm{spln}(a) = \alpha$ and $\mathrm{spln}(b) = \beta$. In general a and b intersect in either two points, or they have an empty intersection. We define the *(canonical) meeting point* of a and b as the point $p = a \wedge b$ where a crosses b from its positive side into its negative side. If $l = \alpha \wedge \beta$, then p is the point $\mathrm{ext}(l)$ where l leaves S^2. See also figure 4.2.

We denote by $\mathrm{ent}(l)$ the point where l enters S^2, and set $\mathrm{mid}(l) = \frac{\mathrm{ext}(l)+\mathrm{ent}(l)}{2}$. If $l = \langle l_0, l_1, l_2, l_3, l_4, l_5 \rangle$, we have

$$\begin{aligned} \mathrm{mid}(l) &= [\mu, -l_1 l_2 - l_3 l_4, l_0 l_2 - l_3 l_5, l_0 l_4 + l_1 l_5], \\ \mathrm{ext}(l) &= \mathrm{mid}(l) + \sqrt{\delta}\,\mathrm{dir}(l), \\ \mathrm{ent}(l) &= \mathrm{mid}(l) - \sqrt{\delta}\,\mathrm{dir}(l), \end{aligned} \tag{4.1}$$

where $\mu = l_2^2 + l_4^2 + l_5^2$ and $\delta = \mu - (l_0^2 + l_1^2 + l_3^2)$.

All edges of the spherical subdivision will correspond to S-arcs, and all vertices will be defined as canonical meeting points of the oriented planes associated with the incident edges.

The geometric primitive employed by the algorithm to navigate within the spherical subdivision is the following: Let p, q and r be three points on an S-circle c. Then $\otimes_c(p,q,r)$ is true[2], if p, q and r occur in that order along c. If $\alpha =$

[2]Here we deviate a little from the notation given in Andrade and Stolfi (1998)

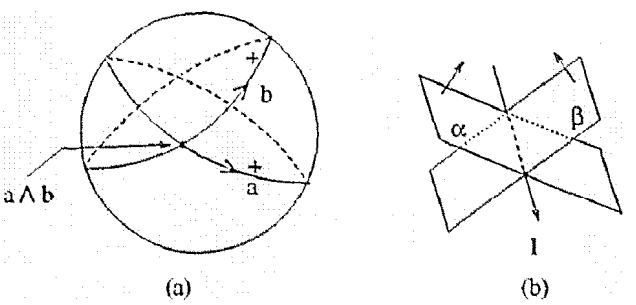


Figure 4.2: Oriented intersection of two S-circles. Picture taken from Andrade and Stolfi (1998).

$\langle \alpha_0, \alpha_1, \alpha_2 \rangle = \mathrm{spln}(c)$, we have

$$\otimes_c(p,q,r) \Longleftrightarrow \begin{vmatrix} p_0 & p_1 & p_2 & p_3 \\ q_0 & q_1 & q_2 & q_3 \\ r_0 & r_1 & r_2 & r_3 \\ 0 & \alpha_1 & \alpha_2 & \alpha_3 \end{vmatrix} > 0 \tag{4.2}$$

In principle, as demonstrated in Andrade and Stolfi (1998), this predicate can be evaluated without computing the roots in (4.1), if p, q and r as given as canonical meeting points.

Implementation note. *The implementation uses distinct classes to represent planes, lines, spheres and canonical meeting points according to the previous specification. However, the plane and line classes are a priori non-oriented. The actual orientation of planes and lines is stored in the low order bit of the pointer variables referring to these non-oriented objects variables. The latter pointer types are, of course, themselves wrapped into classes.*
In addition, a computation cache is maintained for all the computations described in this section. In this manner, no computation has to be performed twice. These classes also keep track of numerical round-off errors as will be described in chapter 5.

4.3 Combinatorial description

The fundamental data structure of our algorithm is a subdivision of the unit sphere. In this section, we introduce the underlying mathematical concepts and describe the data structures used for their representation.

4.3.1 Spherical subdivisions

Most of the following definitions are taken from the books by van Lint and Wilson (1992) and by de Berg et al. (1997).

By a *surface*, we mean a compact 2-manifold. A *Jordan arc* is the image of a continuous one-to-one mapping of the unit interval. An embedding of a graph G on a surface $\mathcal{S}$ is a drawing of G on $\mathcal{S}$, such that no two edges cross:

Definition 12 (Embedding) *Let $G = (V, E)$ be a graph, $\mathcal{S}$ a surface. An embedding $\pi(G)$ of G on $\mathcal{S}$ is a pair of maps $(\pi(V), \pi(E))$ with the following properties:*

1. *$\pi(V) : V \mapsto \mathcal{S}$ is an injection.*
2. *$\pi(E)$ maps each edge $e = \{u, v\} \in E$ to a Jordan arc on $\mathcal{S}$ that connects $\pi(u)$ to $\pi(v)$.*
3. *For each $e \in E$, we have for all $e' \neq e \in E$ that $\text{int}(\pi(e)) \cap \text{int}(\pi(e')) = \emptyset$.*
4. *For each $e \in E$, for each $v \in V$, we have $\pi(v) \notin \text{int}(\pi(e))$.*

Given an embedding $\pi(G)$ of a graph $G = (V, E)$ on a surface $\mathcal{S}$, we can define the faces $F(G)$ as the connected components of $\mathcal{S} \setminus (\pi(V) \cup \bigcup_{e \in E} \pi(e))$.

Definition 13 (Spherical subdivision) *A* spherical subdivision *is the subdivision of the unit sphere S^2 into points, arcs and faces induced by the embedding of a planar connected finite graph G onto S^2.*

Note, that each face f of a spherical subdivision is a 2-cell, i.e. it is a compact set homeomorphic to a disc. In the following, we restrict ourselves to embeddings on the unit sphere, where all arcs are circular, i.e. each arc is the subset of the non-empty intersection of a plane with S^2. Moreover, the underlying graphs will be restricted to be biconnected and to contain no loop edges.

Definition 14 (Subdivision of a graph) *Let $G = (V, E)$ be a graph. A graph H is a subdivision of G, if $V \subset V(H)$ and each edge $e = \{u, w\} \in E$ has been replaced by a path $u, v_1, \ldots, v_{r(e)}, w$, $r(e) \geq 0$, in $E(H)$, such that all inner vertices v_i, $0 \leq i < r(e)$ are incident to no other edges.*

This definition carries over to embeddings:

Definition 15 *An embedding $\pi(H)$ of a graph H onto a surface S is a subdivision of the embedding $\pi(G)$ of a graph G onto S, if H is a subdivision of G, and for each edge $e = \{u,w\} \in E$ that was replaced by a path $u, v_1, \ldots, v_{r(e)}, w$, $r(e) \geq 0$ in $E(H)$, we have $\pi(v_i) \in \pi(e)$ appearing in the specified order along $\pi(e)$.*

The final output of our algorithm will be a polyhedral approximation of the actual additively weighted Voronoi cell.

Definition 16 (Straight approximation) *Let $\pi(G)$ be the embedding of a graph G on the unit sphere S^2. An embedding $\pi(H)$ of a graph H on S^2 is a* straight approximation *of $\pi(G)$, if $\pi(H)$ is a subdivision of $\pi(G)$ and all arcs of $\pi(H)$ are subsets of great circles on S^2.*

We call a spherical subdivision P_H induced by the embedding $\pi(H)$ of a graph H a straight approximation of a spherical subdivision P_G induced by an embedding $\pi(G)$, if $\pi(H)$ is a straight approximation of $\pi(G)$.

4.3.2 Data structures

We represent spherical subdivisions using a halfedge data structure[3]. The data structure comprises five different object types: **halfedge**, **vertex**, **face**, **site** and **data**. **halfedge**, **vertex** and **face** represent the actual subdivision, **site** is a descriptor of the individual sites σ_i, and **data** is additional information associated with a pair of **halfedge** objects. These object types possess certain attribute fields:

1. An object e of type **halfedge** has the following attributes:

   e.twin The twin halfedge connecting the same vertices as e but pointing in the opposite direction. It is always e.twin.twin $= e$.

   e.next The counterclockwise successor halfedge of e along the boundary of the incident face e.face.

   e.face The face incident to and bounded by e.

   e.vertex The vertex at the arrow, i.e. incident to both e and e.next.twin.

   e.data Common data to describe the edge $\{e, e.\text{twin}\}$.

   A *pair of halfedge objects* is a set $\{e, e'\}$ such that e.twin $= e'$ and e'.twin $= e$. We also simply speak of an *edge* when referring to a pair of halfedges.

[3] Cf. Weiler (1985)

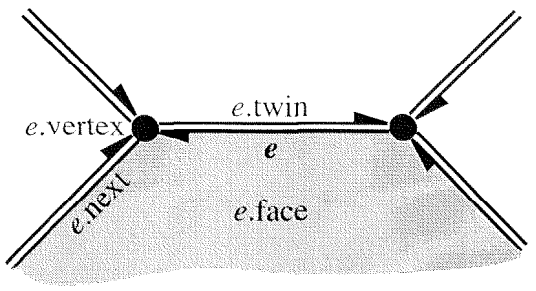


Figure 4.3: The fields associated with a single halfedge e.

2. An object v of type **vertex** has the following attributes:

   v.star An edge e such that $v = e$.vertex.

   v.coordinates The coordinates of v within parameter space S^2.

   v.lifted-coordinates The lifted coordinates $\phi(v)$ on the boundary of the cell.

   v.distance The distance $d(\phi(v), \sigma)$.

   v.conflicts Access to conflict information associated with v.

3. An object f of type **face** has the following attributes:

   f.boundary An incident boundary edge e of f such that $f = e$.face.

   f.neighbor A reference to the descriptor object of type **site** describing the input site whose cell is separated from the current cell by f.

   f.conflicts Access to conflict information associated with f.

4. An object s of type **site** has the following attributes:

   s.sphere The geometric representation of the sphere σ_i represented by s.

   s.boundary The oriented plane $h_{i,0}$ representing the projection of the boundary of the bisector $G(\sigma, \sigma_i)$ onto S^2. See also proposition 4.

   s.conflicts Access to conflict information associated with s.

5. An object d of type **data** has the following attributes:

   d.owners[2] References to the two twin halfedges e and e' described by d. If d.owners[0] $= e$, then the information within e is oriented according to e. For describing e', the orientation of the information has to be reversed.

d.halfspace If we have d.owners[0].face $= f$, d.owners[1].face $= f'$, f.neighbor.sphere $= \sigma_i$, and f'.neighbor.sphere $= \sigma_j$, then d.halfspace is the oriented plane $h_{i,j}$.

d.orientation A partially specialized determinant to order points along the edge represented by d. Observe, that the determinant in equation 4.2 can be written as

$$\begin{vmatrix} p_0 & p_1 & p_2 & p_3 \\ q_0 & q_1 & q_2 & q_3 \\ r_0 & r_1 & r_2 & r_3 \\ \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 \end{vmatrix} = \langle (q \wedge r), *(p \wedge \alpha) \rangle ,$$

where

$$\begin{aligned} p &= d.\text{owners}[1].\text{vertex.coordinates} \\ \alpha &= \text{snrm}(d.\text{halfspace}) . \end{aligned}$$

Hence, we store d.orientation $= p \wedge \alpha$.

s.conflicts Access to conflict information associated with the edge represented by d.

Implementation note. *The implementation maintains objects of type* **vertex**, **face**, *and* **data** *as garbage collected objects using reference counting. Hence,* e.face, e.vertex *and* e.data *are smart pointers that automatically perform the necessary bookkeeping operations. Only edges are deleted explicitly in the code.*

To provide a concise description of the algorithms, we introduce a number of elementary operations on the halfedge data structure. Let e, e' : **halfedge**, f, f' : **face**, v : **vertex**.

split(e, v) Split edge e at a new vertex v: Let $e' = e$.twin and $v' = e$.vertex before the operation. Then split(e, v) creates a new pair of halfedges $\{e_{\text{new}}, e'_{\text{new}}\}$ such that e.next $= e_{\text{new}}$, e'_{new}.next $= e'$, e.face $= e_{\text{new}}$.face, e'.face $= e'_{\text{new}}$.face, e.vertex $= v$, and e_{new}.vertex $= v'$. The new halfedge e_{new} is returned as result value. See figure 4.4.

join(e) This is the inverse operation of split(e, v).

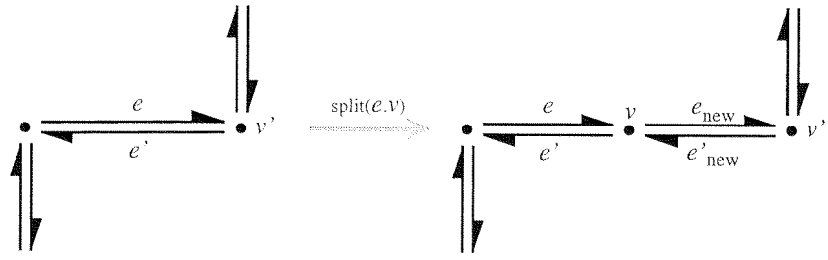


Figure 4.4: The operation split (e, v).

link (e, e', f') Introduce a new face f' by splitting a face f incident to both e and e' by introducing a new edge between them: Let $n = e$.next and $n' = e'$.next before the operation. link (e, e', f') creates a new pair of halfedges $\{e_{\text{new}}, e'_{\text{new}}\}$ such that e.next $= e_{\text{new}}$, e_{new}.next $= n'$, e'.next $= e'_{\text{new}}$, e'_{new}.next $= n$, e_{new}.vertex $= e'$.vertex, and e'_{new}.vertex $= e$.vertex. The new face f' is introduced in such a way that e.face $= e_{\text{new}}$.face $= f$ and e'.face $= e'_{\text{new}}$.face $= f'$. The new halfedge e_{new} is returned as result value. See figure 4.5.

unlink (e) This is the inverse operation of link (e', e'', f'). It removes the pair of halfedges $\{e, e.\text{twin}\}$ and joins the resulting face.

attach (e, e') Precondition to this operation is e.twin $= e$.next. Let $e'' = e'$.next and $v = e$.vertex before the operation. Then attach (e, e') sets e.vertex $= e'$.vertex, e.next $= e''$ and e'.next $= e$.twin. The function returns v. See figure 4.6.

detach (e, v) This is the inverse operation of $v =$ attach (e, e'), if $e' = e$.next and e.next $\neq e$.twin.

The algorithm maintains the following two invariants on the data structure:

1. The skeleton graph, i.e. the graph formed by the **halfedge** and **vertex** objects, is connected.

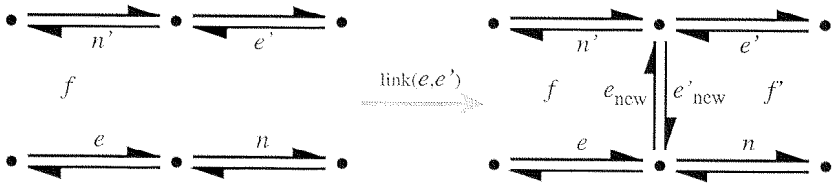


Figure 4.5: The operation link (e, e', f').

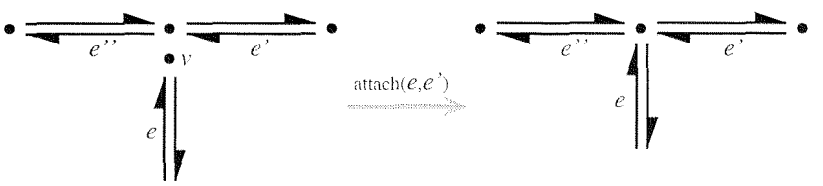


Figure 4.6: The operation attach (e, e').

2. No edge forms a closed loop.

These invariants are maintained by introducing *helper edges* into the spherical subdivision. These helper edges are introduced in two circumstances:

1. The initial subdivision P_0 dividing S^2 into eight equal parts obtained by intersecting S^2 with the coordinate planes is defined by helper edges.

2. Helper edges are introduced when a face f_i would be bounded by a loop edge e from a surrounding face f_o. In this case, the outer face f_o, and hence the loop edge e is split, and the end vertices of the fragments of e are connected to the corresponding vertices on the original boundary of f_o. See also figure 4.9 on page 69, where the details of this operation are discussed.

4.3.3 Conflict information

As with the previous algorithm, we maintain conflicts that are associated with the vertices, edge frements and face fragments of the current subdivision. Again, we denote the subdivision after the insertion of the i-th sphere σ_i by P_i.

1. Vertex conflicts: A vertex $v \in P_i$ conflicts with $\sigma_j \in \bar{S}_i$ if ϕ_j will cut the vertex $\phi(v)$ off the cell. In fact, vertex conflicts are not used for any manipulation of the spherical subdivision. Their only purpose is to trigger the generation of new conflict information after a site has been added.

2. Edge conflicts: An edge $e \in P_i$ conflicts with $\sigma_j \in \bar{S}_i$ if ϕ_j intersects $\phi(e)$. We maintain a distinct conflict for each point of intersection. For a single edge e the set of all conflicts $\{(e, \cdot)\}$ is linearly ordered along e. Note, that this information is shared for each pair of halfedges.

3. Face conflicts: A face $f \in P_i, \phi(f) \subset \phi_k$ for some $0 \leq k \leq i$ conflicts with $\sigma_j \in \bar{S}_i$ if $\phi_k \cap \phi_j \neq \emptyset$ and $\phi_k \cap \phi_j \subset \text{int}(\phi(f))$. Each face conflict c is represented by a unique point contained in the conflicting region: Let $\alpha = (\alpha_0, \alpha_1, \alpha_2, \alpha_3)$ be the s-circle describing $\phi_k \cap \phi_j$, and let $l = \{[1, \lambda\alpha_1, \lambda\alpha_2, \lambda\alpha_3], \lambda \in \mathbf{R}\}$. Then c is represented by $p = f \cap l$. p is uniquely defined since each subdivision P_i, $1 \leq i \leq n$, is a refinement of P_0.

   Face conflicts are used in two flavors:

   (a) Disc conflicts: A disc conflict (f, s) tells the algorithm to cut a hole into the face f. The interior part is labeled with the conflicting site s.

   (b) Ring conflicts: A ring conflict (f, s) tells the algorithm that when site s is added, a hole has to be cut into the *new* face to remain a part of f.

Implementation note. *The implementation uses three classes derived from a common* **conflict** *base class to represent conflicts. Both sites to be added and elements of the spherical subdivision maintain their incident conflicts using doubly linked lists for unit cost removal.*

4.4 The algorithm

In this section, we describe the main algorithm we implemented for computing a single AWV cell. After giving an outline of the algorithm, we will discuss the insertion of new edges into the spherical subdivision and how the conflict information is updated.

4.4.1 Outline

The outline of the algorithm is very similar to the algorithm from the previous chapter: We assume the sites $\sigma_1, \ldots, \sigma_n$ to be given in that order after having applied a random permutation to the input set. Then the algorithm works as follows:

1. *Initialization:* Construct an initial spherical subdivision P_0 by cutting the unit sphere S^2 with the three coordinate planes $x = 0$, $y = 0$ and $z = 0$.

   Compute initial conflict information between each vertex, edge and face of P_0 and each site σ_i, $1 \leq i \leq n$.

2. *Incremental step:* For each $i = 1 \ldots n$ perform the following operations:

   (a) Create new edges due to edge conflicts generated by σ_i.

   (b) Remove redundant old edges that no longer separate different faces. Rejoin chains of edges into single edges.

   (c) Create new edges due to face conflicts:

      i. Process disc conflicts generated by σ_i.

      ii. Process ring conflicts generated by σ_i.

   (d) Update conflict information.

4.4.2 Changing the subdivision

In this subsection, we describe the individual steps that are necessary to update subdivision P_{i-1} to P_i when adding site σ_i. Let B_i denote the set of edges in P_i separating a face defined by σ_i from a face defined by one of the other spheres σ_j, $1 \leq j < i$. Obviously, B_i is the set of new boundary edges to be introduced in step i of the algorithm.

Processing edge conflicts

Let $B_E = B_E^{(i)}$ denote the subset of B_i introduced due to edge conflicts. B_E is easily seen to form a set of cycles in P_i. In fact, in the absence of geometric degeneracies, i.e. if no $e \in B_E$ passes through a vertex $v \in P_{i-1}$, an even stronger property holds: Whenever the new boundary B_E enters a face $f \in F_{i-1}$ through an edge conflict, then it also leaves f through an edge conflict. This implies that for all faces $f \in P_{i-1}$ the sum of the edge conflicts with σ_i on its incident boundary edges is even.

The pseudo code of this part of the algorithm is shown in figure 4.7. The following functions have not been introduced yet:

1. any-edge-conflict(σ_i) simply returns any edge conflict of an edge e with σ_i that has not been visited yet.

2. enter-edge(c) returns that halfedge h located on the edge e conflicting with σ_i at c such that h enters the new face to be created at c. Hence, the algorithm traverses the subdivision outside the new face to be created.

3. find-ccw-conflict() returns the neighboring conflict c_n to the present conflict c_p and the boundary halfedge of the current face it is located on. The neighboring conflict c_n is selected such that the new oriented edge arc to be created connects c_p and c_n in its positive sense of orientation.

Removal and contraction of superfluous edges

After the new boundary edges B_E have been introduced, the interior of the newly created faces is cleaned from edges and edge fragments that have become unnecessary. The identification and removal of these edges is performed by simple BFS traversals rooted at halfedges in B_E.

Figure 4.8 shows the pseudo-code for collecting all removable edges given the set of new boundary edges *new-boundary*. find-deletable (*new-boundary*) returns for each edge e that can be deleted a representative halfedge. In the implementation, the function is called with *new-boundary* $= B_E$.

The following functions have not been introduced yet:

1. mark(*halfedge*) sets a Boolean flag to mark *halfedge* as visited.
2. is-marked(*halfedge*) is `true` if *halfedge* has already been visited.

After the set of deletable edges $D =$ find-deletable (B_E) has been computed, the individual edges $e \in D$ are removed by unlink (e) and detach (e, v) operations. Then, in a second traversal very similar to the code shown for find-deletable, edge chains starting at a half edge e are contracted into single edges by join (e) operations.

Processing face conflicts

The treatment of face conflicts $B_F = B_F^{(i)} = B_i \setminus B_E$ is conceptually very simple. If we would not have to keep the skeleton graph connected, then each conflict $c \in B_F$ would give rise to a single circular edge on S^2 without any further vertex.

Let $c \in B_F$ be a face conflict to be treated in step i of the algorithm. Let f be the face conflicting σ_i at c and assume that f.neighbor.sphere $= \sigma_j$. Then the new circular edge e to be introduced and refined represents the circle $\mathrm{scrc}(h_{i,j})$. Let v denote the normal $\mathrm{snrm}(h_{i,j})$, and let p denote the unique intersection point $f \cap \{\lambda v, \lambda \in \mathbf{R}\}$. We choose two oriented planes α and β defining two great circles a and b, such that

```
process-edge-conflicts(σ_i):
    while conflict := any-edge-conflict(σ_i) do
        start-edge := enter-edge(conflict)
        last-edge := split(start-edge, conflict)
        next-edge := last-edge.next

        loop
            (next-edge, conflict) := find-ccw-conflict()

            exit if next-edge = last-edge

            continuation-edge := split(next-edge, conflict)

            if last-edge = next-edge then
                last-edge := next-edge.next
            end

            link(next-edge, last-edge, new(face))
            next-edge := continuation-edge.next
            last-edge := continuation-edge
        end

        link(start-edge, last-edge, new(face))
    end
```

Figure 4.7: Processing edge conflicts of site σ_i.

```
find-deletable (new-boundary):
    find-deletable (halfedge, deletable):
        if is-marked(halfedge) ∨ halfedge ∈ new-boundary then
            return deletable
        end

        mark(halfedge)
        mark(halfedge.twin)
        deletable := deletable ∪ {halfedge}
        around := halfedge.next

        while halfedge ∉ new-boundary ∧ around ≠ halfedge.twin do
            deletable := find-deletable (around, deletable)
            around := around.twin.next
        end

        return deletable

    deletable := ∅

    for halfedge ∈ new-boundary do
        deletable := find-deletable (halfedge, deletable)
    end
```

Figure 4.8: Identification of deletable edges when adding σ_i.

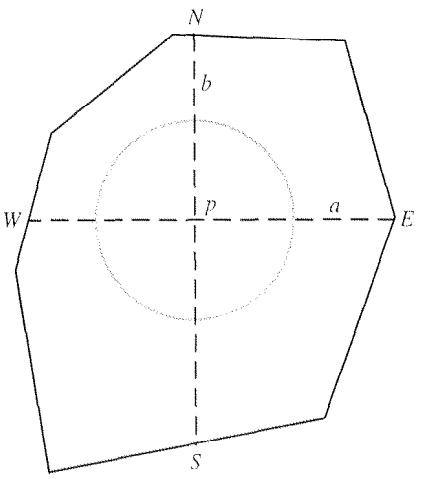


Figure 4.9: Face conflicts are reduced to edge conflicts by splitting the original face by two orthogonal planes.

1. snrm(α) is randomly chosen within $S^2/\{\lambda v\}$,

2. snrm(β) $\perp v$, and snrm(α) $\perp$ snrm(β), and

3. $p \in a$ and $p \in b$.

We cut f into the four pieces obtained by connecting p to the nearest intersections of the boundary of f with the circles a and b in each direction. See figure 4.9 for an illustration. We label these four additional edges connecting p to four new vertices N, S, W, E on the boundary of f as helper edges.

In this way, we have reduced the face conflict c to a special case of four edge conflicts on the edges $\{p,N\}$, $\{p,E\}$,$\{p,S\}$, and $\{p,W\}$ that can be handled as described previously.

4.4.3 Update of conflict information

Besides changing the combinatorial structure of the subdivision, the update operation adding site σ_i has maintain the conflict information associated will all σ_j, $i < j \leq n$. Basically, this update takes place in two steps:

1. During the change of the subdivision:

- Whenever a face f is split by the introduction of a new edge e into faces f_1 and f_2, and there is a face conflict c between f and a site σ_j, $j > i$, there are three possibilities of how this conflict information has to be distributed among f_1 and f_2:

  (a) σ_j still generates a face conflict with f_1 or f_2.

  (b) σ_j generates an edge conflict on e.

  (c) e is a helper edge, and after recoloring f_1 and f_2 with σ_i the site σ_j conflicts neither f_1 nor f_2.

- When a redundant edge e with edge conflicts is removed, then the sites conflicting e are stored into a candidate set C_f of the incident face f. Similarly, when a vertex v with conflicts to sites σ_j, $j > i$, is removed during the merge of edge chains, these conflicting sites are also stored into the candidate sets C_{f_1} and C_{f_2} of the two incident faces f_1 and f_2. These candidate sets are used during step 2.

2. After updating the subdivision:

   - Let f be a face, such that f.neighbor.sphere $= \sigma_i$. For each vertex conflict c between a boundary vertex v of f and a sphere σ_j, $j > i$, the algorithm tests if $d(\phi_j(v.\text{coordinates}), \sigma) < v.\text{distance}$. If so, σ_j is added to the set C_f of conflicting sites for f.

   - For each face f, all candidates $\sigma \in C_f$ are checked against all boundary edges of f. In fact, this calculation is performed only for one of the two halfedges e constituting an edge, namely if and only if e.data.owners$[0] = e$.

   - For each face f and site $\sigma \in C_f$ such that

     (a) σ conflicts all vertices of the boundary of f but has no edge conflict with any edge e on the boundary of f, the algorithm checks for a ring conflict.

     (b) σ conflicts neither any edge nor any vertex of the boundary of f, the algorithm checks for a disc conflict. If f.neighbor.sphere $= \sigma_i$ and $\sigma = \sigma_j$ this amounts to checking if the S-circle $\text{scrc}(h_{j,i})$ exists and, if so, if its S-center is contained in f. The latter operation requires work linear in the number of boundary edges of f.

4.5 Preprocessing

The previous algorithm can compute the AWV cell of a single sphere out of a set $S = \{\sigma_1, \ldots, \sigma_n\}$ of spheres. To compute the AWV cell V_i for each individual sphere σ_i, we would have to call the algorithm n times, each time passing $n-1$ input sites as argument. To restrict the number of spheres that have to be considered in each run, we perform a preprocessing step that identfies for each sphere σ_i, $1 \leq i \leq n$, of the input set S a set of neighbors N_i, such that $\mathrm{cl}(V_i) \cap \mathrm{cl}(V_j) \neq \emptyset$ implies either $i = j$ or else $\sigma_j \in N_i$. By Aurenhammer's lifting procedure, to each AWV cell V_i there exists a corresponding 4-dimensional power cell $P(\Sigma_i)$, such that for all $1 \leq i \neq j \leq n$ the inequality $\mathrm{cl}(V_i) \cap \mathrm{cl}(V_j) \neq \emptyset$ implies $\mathrm{cl}(P(\Sigma_i)) \cap \mathrm{cl}(P(\Sigma_j)) \neq \emptyset$. Hence, we calculate N_i as the set of all σ_j, $1 \leq j \leq n$, $i \neq j$, such that $\mathrm{cl}(P(\Sigma_i)) \cap \mathrm{cl}(P(\Sigma_j)) \neq \emptyset$. Since Aurenhammer (1987) also showed that a 4-dimensional power diagram corresponds to a 5-dimensional lower convex polyhedron, we are left with the problem of computing the intersection

$$H = \bigcap_{i=1}^{n} h_i,$$

where each h_i is a halfspace $h_i = \{x \in \mathbf{R}^5 : \langle a_i, x \rangle \leq b_i\}$. If we write $\sigma_i = ((c_{i,1}, c_{i,2}, c_{i,3}), r_i)$, then we have

$$\begin{aligned} a_i &= (2c_{i,1}, 2c_{i,2}, 2c_{i,3}, 2r_i, -1) \\ b_i &= c_i^2 - r_i^2. \end{aligned}$$

To obtain the sets N_i, we compute the skeleton graph $G = (V, E)$ of H, where V is the set of vertices of V and E the set of edges of the polyhedron. We include a sphere σ_j in the set N_i, if there exists an edge $e \in E$ such that $e \subset \mathrm{cl}(f_i) \cap \mathrm{cl}(f_j)$, where for each $1 \leq k \leq n$ the set f_k is the facet of H with supporting plane $f_k = \{x \in \mathbf{R}^5 : \langle a_k, x \rangle = b_k\}$ or the empty set, if no such facet exists. Again, we only consider the non-degenerate case, and *define* an input S to be non-degenerate, if for each set of indices $1 \leq i_1 < i_2 < i_3 < i_4 < i_5 \leq n$ the intersection $f_{i_1} \cap f_{i_2} \cap f_{i_3} \cap f_{i_4} \cap f_{i_5}$ has Hausdorff-dimension 0, and for each six-tuple of indices the corresponding intersection is empty. For any vertex $v \in V$, let $\mathrm{def}(v)$ denote the quintuple of indices $1 \leq i_1 < i_2 < i_3 < i_4 < i_5 \leq n$, such that $f_{i_1} \cap f_{i_2} \cap f_{i_3} \cap f_{i_4} \cap f_{i_5} = \{v\}$.

To compute the skeleton graph $G = (V, E)$ of the polyhedron H, we implemented the RIC algorithm as described in sections 3.2 and 7.3 of the textbook by Mulmuley (1994b). Due to our non-degeneracy assumptions, G is 5-regular, i.e. the set of

neighbors $\Gamma(v)$ of each vertex $v \in V$ has cardinality 5. Effectively, the algorithm computes the intersection of H with a sufficiently large 5-dimensional hypercube. Hence, w.l.o.g., we can talk of H as a polytope.

Implementation note. *The algorithm was implemented as static algorithm using conflict lists. The numerical predicates and computational primitives are implemented using built-in floating point arithmetic. To increase locality of reference, intermediate data is stored on the machine stack, and dynamically allocated objects are managed in a dedicated memory pool.*

4.5.1 Verification of output

Since we use simple floating point arithmetic for computing the intersection H, the implementation is vulnerable to numerical round-off errors and degeneracies. To remedy this, we implemented a simple verification procedure that checks the validity of the computed polytope after the algorithm has finished. Mehlhorn et al. (1996) propose a procedure for verifying the output of an algorithm computing the convex hull of a set of points in $\mathbf{R}^d$. The output O of their convex hull algorithm is a representation of a simplicial piecewise linear hypersurface without boundary. The verification procedure decides whether this hypersurface is the boundary of a convex polytope by performing the following steps:

1. It is asserted that the surface is locally convex along all its ridges,

2. that the center of gravity o of the vertices of the output O is on the negative side of all facets, and

3. that a ray emanating from o through the center of gravity p of *any* of the facets f of O intersects only one facet, namely f.

Since our algorithm works in the dual setting, i.e. we are computing the intersection of halfspaces, we suggest the following strategy for a verification procedure after having computed a representation G of H:

1. Verify, that the output is locally convex at each vertex, i.e. that each neighbor w of a vertex $v \in V$ satisfies the boundary equations defining v.

2. For any of the vertices v calculate the vector

$$c := \sum_{k \in \mathrm{def}(v)} a_k,$$

which is locally an outer normal vector at v. Verify, that for all vertices $w \neq v$

$$c \notin \left\{ \sum_{k \in \mathrm{def}(w)} c_k a_k \text{ where } \forall k \in \mathrm{def}(w) : \; c_k \geq 0 \right\}$$

However, we cannot just "dualize" the proof given by Mehlhorn et al. (1996). Specifically, the output that our algorithm generates is only a labeled connected d-regular graph, so we cannot make the *a priori* assumption that it represents a valid hypersurface. Moreover, we cannot simply apply dualization, because at this point the concept of an interior point is not yet well-defined.

Instead, we will first prove that if an output O of our algorithm passes the tests stated above, then we can conclude that the vertices are in convex position. The key ingredient in this proof is the well-known Farkas-Lemma[4] from linear optimization, that states that a cost function x maximized at a vertex v of a convex polytope H can be represented as a positive combination of the outer normals of the facets incident to v. Therefore, we will partition the set of all possible directions S^{d-1} in such a way that we assign to each vertex v of G the set of directions that can be represented as a strictly positive combination of the outer normals at v. We will show that if G passes the test, then these sets fit nicely together to yield a tessellation of S^{d-1}. This allows us to have a well-defined notion of supporting hyperplanes at the vertices. Finally, we may conclude that these supporting hyperplanes indeed define the boundary of a convex polytope.

W.l.o.g., we assume that the input vectors a_i, $1 \leq i \leq n$, have been normalized.

Theorem 6 (Verification of intersection of halfspaces) *Let $d \in \mathbf{N}$, $d > 0$. Let $\{a_i\}_{1 \leq i \leq n} \subset S^{d-1}$, $a_i \neq a_j$ for all $i \neq j$, $\{b_i\}_{1 \leq i \leq n} \subset \mathbf{R}$. Let $G := (V, E)$ be a d-regular finite graph, where $V \subset \mathbf{R}^d$, $V \neq \emptyset$ and for each vertex $v \in V$ there is a label $\mathrm{def}(v) \subset \{1, \ldots, n\}$ with $|\mathrm{def}(v)| = d$. For $v \in V$ define*

$$\begin{aligned} \Delta(v) &:= \left\{ x = \sum_{j \in \mathrm{def}(v)} c_j a_j, \text{ where } 0 < c_j \text{ for all } j \in \mathrm{def}(v) \right\}, \\ \Lambda(v) &:= \Delta(v) \cap S^{d-1}. \end{aligned}$$

[4] Cf. Ziegler (1994), section 1.4

For an arbitrary but fixed vertex $v_0 \in V$ fix $x_0 \in \Lambda(v_0)$. The following conditions shall be true:

1. *For each vertex $v \in V$ the set $\{a_i, i \in \mathrm{def}(v)\}$ has linear rank d and $\langle a_i, v\rangle = b_i$ for all $i \in \mathrm{def}(v)$.*

2. *For each edge $e = \{v,w\} \in E$ we have $|\mathrm{def}(v) \setminus \mathrm{def}(w)| = |\mathrm{def}(w) \setminus \mathrm{def}(v)| = 1$.*

3. *For each pair of vertices $v \in V$ and $w \in \Gamma(v)$ the following holds:*

   (a) *For all $i \in \mathrm{def}(v)$ we have $\langle a_i, w\rangle \leq b_i$.*

   (b) *For $\{j\} = \mathrm{def}(v) \setminus \mathrm{def}(w)$ we have $\langle a_j, w\rangle < b_i$.*

4. *For all vertices $v \in V$, $v \neq v_0$ the relation $x_0 \notin \Lambda(v)$ holds.*

Then G is the skeleton graph of the convex polytope

$$H = \bigcap_{v \in V} \bigcap_{i \in \mathrm{def}(v)} \{x : \langle a_i, x\rangle \leq b_i\}.$$

We refer to conditions 1 to 3 as *local convexity* conditions. In the following argumentation, for any $v \in V$ we understand the boundary $\partial\Lambda(v)$ and the closure $\mathrm{cl}(\Lambda(v))$ relative to S^{d-1}.

We will prove the theorem with the help of two lemmas and corollaries, still referring to the notations introduced in the statement of the theorem.

Lemma 6 *For any $x \in S^{d-1}$ there exists a vertex $v \in V$ such that $x \in \mathrm{cl}(\Lambda(v))$.*

Proof: We will prove this lemma by induction on d. For $d > 2$ we assume the lemma and the theorem to hold for all lower dimensions $0 < d' < d$.

We will prove this lemma by contradiction. Let $d > 0$ be the minimum dimension such that the lemma is not true. Then there exists a direction $x_0 \in S^{d-1}$ such that for all $v \in V$ the condition $x_0 \notin \mathrm{cl}(\Lambda(v))$ holds. Let $\alpha(x,v) = \max_{y \in \mathrm{cl}(\Lambda(v))} \langle x, y\rangle$. Since $V \neq \emptyset$, there exist $v_0 \in V$ and $y \in \mathrm{cl}(\Lambda(v_0))$ such that

$$\langle x_0, y\rangle = \alpha(x_0, v_0) = \max_{v \in V} \alpha(x_0, v).$$

$y \in \Lambda(v_0)$ implies $x = y$, hence $y \in \partial\Lambda(v_0)$. We represent y as

$$y = \sum_{j \in \mathrm{def}(v_0)} c_j a_j, \tag{4.3}$$

where $0 \leq c_j$ for all $j \in \mathrm{def}(v_0)$. Then the set $C_0 = \{j : c_j = 0\}$ is non-empty.

If $|C_0| = 1$, then only one coefficient c_k in the representation 4.3 is zero. By duality, there exists a unique edge $\{v_0, w\}$ connecting v_0 to a neighbor of w such that $\mathrm{def}(w) \setminus \mathrm{def}(v_0) = k$. We might find a direction y' closer to x_0 in $\Lambda(w)$. If $|C_0| > 1$, then we could find a better direction at any of the vertices incident to the corresponding "$|C_0|$-face" F of the polytope. However, we have to be careful since at this point we have yet to show that indeed G represents a polytope. The idea will be to consider the situation projected onto the affine subspace spanned by F. We will apply the induction hypothesis restricted to the projected setting, and then lift everything back to the d-dimensional case. Formally, we have:

1. $|C_0| = 1$: By condition 2 in the statement of theorem there exists a unique $w \in \Gamma(v_0)$ such that $\mathrm{def}(v_0) \setminus \mathrm{def}(w) = C_0$. Let $\{i\} = \mathrm{def}(v_0) \setminus \mathrm{def}(w)$, $\{k\} = \mathrm{def}(w) \setminus \mathrm{def}(v_0)$ and let $B = B_\varepsilon(y) \cap S^{d-1}$ be a neighborhood of y in S^{d-1}. Let

   $$A = \mathrm{span}\left(\{a_j, j \in \mathrm{def}(v_0) \cap \mathrm{def}(w)\}\right),$$

   the vectorspace orthogonal to the vector $w - v_0$ along the edge $e = \{v_0, w\}$. The orthogonality follows from condition 1 in the statement of the theorem. By condition 3, $\langle w - v_0, a_i \rangle < 0$ and $\langle w - v_0, a_k \rangle > 0$, so a_i and a_k point away into different directions from A. See figure 4.10 for an illustration. Therefore, there exists $\varepsilon > 0$ such that for all $z \in B$ we have a representation

   $$z = \sum_{j \in \mathrm{def}(v_0) \cup \{k\}} c_j a_j,$$

   where $0 \leq c_j$ for all $j \in \mathrm{def}(v_0) \cup \{k\}$. Let $z_0 \in B$ such that $\langle x_0, z_0 \rangle = \max_{z \in B} \langle x_0, z \rangle$. Since we assumed $z_0 \notin \mathrm{cl}(\Lambda(v))$, we conclude $z_0 \in \Lambda(w)$, contradicting the maximality of y.

2. $|C_0| > 1$: Let

   $$D = \{\delta_1, \ldots, \delta_{d-|C_0|}\} = \mathrm{def}(v_0) \setminus C_0,$$

   and let

   $$A_D = \{x : \langle a_i, x \rangle = b_i \text{ for all } i \in D\}.$$

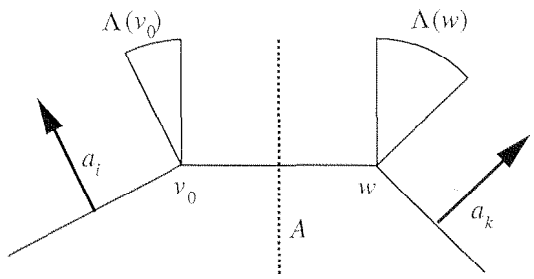


Figure 4.10: The edge $e = \{v_0, w\}$, which spans the orthogonal complement of the vector space A.

Let $\eta_1, \ldots, \eta_d$ be an orthonormal basis of $\mathbf{R}^d$ such that $\text{span}(\{a_i, i \in D\}) = \text{span}(\{\eta_{|C_0|+1}, \ldots, \eta_d\})$. Let $\Phi : \mathbf{R}^d \to \mathbf{R}^d$ be the mapping

$$\Phi : x \mapsto (\langle \eta_1, x \rangle, \ldots, \langle \eta_d, x \rangle)$$

and let ϕ be the restriction of the image of Φ on the first $|C_0|$ coordinates, which we identify with $\mathbf{R}^{|C_0|}$. Let F be the component of v_0 in the subgraph of G induced by the set $\{v \in V : D \subset \text{def}(v)\}$. Let $F' := (\phi(V(F)), E(F))$, and let $\{a'_i\}_{1 \le i \le n}$ and $\{b'_i\}_{1 \le i \le n}$ be defined as

$$a'_i = \frac{\phi(a_i)}{\|\phi(a_i)\|}$$

$$b'_i = \frac{(-1)^{d+1}}{\|\phi(a_i)\|} \cdot \frac{\begin{vmatrix} \langle a_i, \eta_{|C_0|+1} \rangle & \cdots & \langle a_i, \eta_d \rangle & b_i \\ \langle a_{\delta_1}, \eta_{|C_0|+1} \rangle & \cdots & \langle a_{\delta_1}, \eta_d \rangle & b_{\delta_1} \\ \vdots & \ddots & \vdots & \vdots \\ \langle a_{\delta_{|D|}}, \eta_{|C_0|+1} \rangle & \cdots & \langle a_{\delta_{|D|}}, \eta_d \rangle & b_{\delta_{|D|}} \end{vmatrix}}{\begin{vmatrix} \langle a_{\delta_1}, \eta_{|C_0|+1} \rangle & \cdots & \langle a_{\delta_1}, \eta_d \rangle \\ \vdots & \ddots & \vdots \\ \langle a_{\delta_{|D|}}, \eta_{|C_0|+1} \rangle & \cdots & \langle a_{\delta_{|D|}}, \eta_d \rangle \end{vmatrix}}.$$

$\{a'_i\}_{1 \le i \le n}$ and $\{b'_i\}_{1 \le i \le n}$ are the coefficients of the projections of the hyperplanes defined by $\{a_i\}_{1 \le i \le n}$ and $\{b_i\}_{1 \le i \le n}$ onto A_D, which we identified with $\mathbf{R}^{|C_0|}$. Note, that F', $\{a'_i\}_{1 \le i \le n}$ and $\{b'_i\}_{1 \le i \le n}$ satisfy the requirements for G, a_i and b_i, respectively, for $d = |C_0|$ as stated in the theorem.

We factor x_0 by Φ into image and kernel

$$x_0 = \sum_{i=1}^{|C_0|} c_i \eta_i + \sum_{i=1}^{|D|} c_{|C_0|+i} a_{\delta_i}.$$

Since y was chosen maximal, we have $c_i > 0$ for all $|C_0| < i \leq d$. By the induction hypothesis there exists a $v_D \in V(F')$ such that

$$\frac{\phi(x_0)}{\|\phi(x_0)\|} \in \mathrm{cl}\,(\Lambda(v_D)) \subset \mathbf{R}^{|C_0|}.$$

Note, that $\|\phi(x_0)\| = 0$ would imply $x_0 \in \mathrm{cl}\,(\Lambda(v_0))$, contradicting our assumptions. Therefore, $x_0 \in \mathrm{cl}\,(\Lambda(v))$, v being the preimage $\phi^{-1}(v_D) \subset V(F)$ leading to the desired contradiction and thus proving the lemma. □

Since the previous proof works independently for every connected component of G, we can formulate the following corollary:

Corollary 3 *For any $x \in S^{d-1}$ there exists a vertex v in every connected component of G such that $x \in \mathrm{cl}\,(\Lambda(v))$.*

Lemma 7 *Assume that G is connected and let $v, w \in V$, $v \neq w$, such that $\Lambda(v) \cap \Lambda(w) \neq \emptyset$. Then for any vertex $s \in V$ there exists a vertex $t \in V$, $s \neq t$, such that $\Lambda(s) \cap \Lambda(t) \neq \emptyset$.*

Proof: We show the lemma by proving that the claim holds for all neighbors of v. By the connectedness of G the statement follows for all vertices of G. Essentially, we show that whenever we cross a part of $\partial\Lambda(v)$ that is contained in $\Lambda(w)$ into a neighboring region $\Lambda(u)$, then $\Lambda(w)$ also overlaps a part of $\Lambda(u)$.

Let $R = \Lambda(v) \cap \Lambda(w)$. Then $\partial R \subset \partial\Lambda(v) \cup \partial\Lambda(w)$. W.l.o.g., let $e = \{v, u\}$ be an edge such that

$$\mathrm{cl}\,(\Lambda(v)) \cap \mathrm{cl}\,(\Lambda(u)) \cap \Lambda(w) \neq \emptyset.$$

Let $y \in \mathrm{cl}\,(\Lambda(v)) \cap \mathrm{cl}\,(\Lambda(u)) \cap \Lambda(w)$. As in case 1 of the proof of the previous lemma there exists an $\varepsilon > 0$ such that

$$B = B_\varepsilon(y) \cap S^{d-1} \subset \mathrm{cl}\,(\Lambda(v)) \cup \mathrm{cl}\,(\Lambda(u)).$$

B is the region where we "cross the boundary" from $\Lambda(v)$ to $\Lambda(u)$. Because $y \in \Lambda(w)$ and $\Lambda(w)$ is relativly open in S^{d-1}, there exists a neighborhood U of y with $U \subset \Lambda(w)$. Since $y \in B$ and $y \in U$, and both of these sets are open, we get $\Lambda(u) \cap \Lambda(w) \neq \emptyset$. □

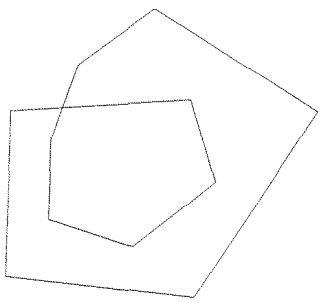


Figure 4.11: An output satisfying the local convexity conditions essentially corresponds to a surface with a winding number greater or equal to 1.

Corollary 4 *Assume that G is connected and let $v_1,\ldots,v_k \in V$, $v_i \neq v_j$ for all $1 \leq i < j \leq k$, such that $\bigcap_{i=1}^{k} \Lambda(v_i) \neq \emptyset$. Then for any vertex $s_1 \in V$ there exist vertices $s_2,\ldots,s_k \in V$, $s_i \neq s_j$ for all $i \neq j$, such that $\bigcap_{i=1}^{k} \Lambda(s_i) \neq \emptyset$.*

Proof: This follows by simple induction on k from the proof of the previous lemma. □

The two previous corollaries imply that there exists an integer $k > 0$ satisfying

$$k \cdot \omega_d = \sum_{v \in V} \mu_{d-1}\left(\Lambda(v)\right),$$

where μ_{d-1} denotes the $d-1$-dimensional Lebesgue-Measure and $\omega_d = \frac{2\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2})}$ is the surface area of the d-dimensional unit sphere.

Hence, an output O of the algorithm satisfying the local convexity conditions essentially looks as indicated in figure 4.11.

Proof of the theorem: We will prove the theorem by showing that the set of halfspaces $h_i = \{\langle a_i, x\rangle \leq b_i\}$, $i \in \bigcup_{v \in V} \mathrm{def}(v)$ form a non-redundant set of supporting hyperplanes of the polytope H.

From condition 4 and corollary 3 we conclude that G is a connected graph. By corollary 4, all $\Lambda(v)$, $v \in V$ are disjoint. Moreover, also by corollary 4, if there exists an $x \in S^{d-1}$ such that the set $C := \{v \in V : x \in \mathrm{cl}(\Lambda(v))\}$ has cardinality $|C| > 1$, then the induced subgraph $G(C) = (C, E(C))$ forms a connected component of G, and all $v \in C$ are contained in a common affine subspace given by $\{\langle a_i, x\rangle = b_i : i \in \bigcap_{v \in C} \mathrm{def}(v)\}$.

Let $x \in S^{d-1}$. A vertex $v \in V$ maximizes $\langle x, v\rangle$ if and only if $x \in \mathrm{cl}(\Lambda(v))$. This follows from the Farkas-Lemma[5], stating that a cost function x maximized at a

[5]Cf. Ziegler (1994), section 1.4

vertex v can be represented as a positive combination of the outer normals a_i at v, where $i \in \mathrm{def}(v)$.

Therefore, the $h_i = \{\langle a_i, x\rangle = b_i\}$, $i \in \bigcup_{v \in V} \mathrm{def}(v)$ form a set of supporting hyperplanes. In fact, for any $x \in S^{d-1}$ and a vertex $v \in V$ such that $x \in \mathrm{cl}(\Lambda(v))$, the set $\{y : \langle x, y\rangle \leq \langle x, v\rangle\}$ is a supporting hyperplane at v. This implies that H is bounded. Hence, we ontain $V = V(H)$, since for any $v \in V(H) \setminus V$ lemma 6 would imply $\Lambda(v) = \emptyset$.

Since we assumed the input coordinates to be non-degenerate, each vertex is defined by a unique set of hyperplanes from the input. This implies that the edge set E is determined uniquely by the sets $\mathrm{def}(v), v \in V$.

All in all, we have shown that

$$H := \bigcap_{v \in V} \bigcap_{i \in \mathrm{def}(v)} \{x : \langle a_i, x\rangle \leq b_i\}$$

□

To show that G represents

$$H = \bigcap_{i=1}^{n} h_i,$$

we have to maintain for each redundant halfspace h_i, $1 \leq i \leq n$, a witness $v \in V$ such that $a_i \in \Delta(v)$. These witnesses can be obtained by a trivial modification of the intersection algorithm.

4.6 Post-processing

At this point, we have shown how to compute an explicit representation of an additively weighted Voronoi cell. However, for practical purposes, this representation is still not sufficient. For visualization, we have to break up the surface patches into triangular meshes[6], because the majority of current 3D hardware is limited to accepting triangles as input. At the same time, since the cells are star-shaped, a triangular surface mesh yields an approximation of the cell as a collection of simplices spanned by the individual triangles of the surface mesh and the center of

[6]Though there are visualization libraries appearing that can handle continuous surface patches, such as the OpenGL Optimizer or Direct Model — actually, these libraries perform the triangulation themselves. Indeed, we would have used OpenGL Optimizer, if the first release of that software only had been reliable enough.

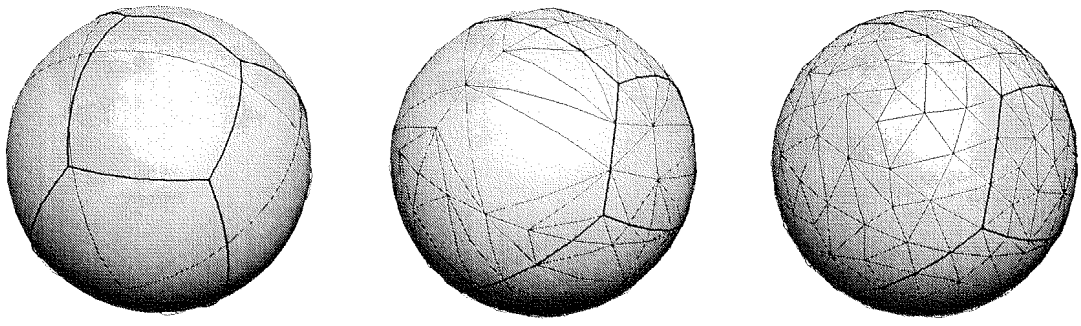


Figure 4.12: The subdivision of the surface is triangulated, then refined to a Delaunay triangulation, which is further refined by introducing additional points.

the defining sphere. These simplices can be used to obtain a good approximation of the volume of a cell.

We construct a *refined constraint Delaunay triangulation* describing the cell boundary. For general background on surface meshing, the reader is referred to the extensive survey by Bern and Eppstein (1995). This computations is done in the following steps, see also figure 4.12:

1. In a first step, we compute a straight approximation of the spherical subdivision representing the combinatorial structure of the cell. The main problem to solve in this step is to guarantee that the straight approximation will have the same topological structure as the original subdivision.

2. Then, for each face, resulting simple spherical polygons are triangulated. We propose a very simple heuristically motivated algorithm that tries to exploit the fact that most of the polygons in our setting are "almost convex". This algorithm turns out to be pretty fast in practice.

3. Using the standard Lawson-flip (1977), this triangulation is transformed into a constrained Delaunay triangulation on the sphere. The Delaunay property can be formulated either in terms of the parameter space, i.e. on the sphere, or in terms of the actual surface.

4. Finally, similar to Chew (1989, 1993) and Ruppert (1995), circumcenters of large or skinny triangles are added to the triangulation to obtain triangles that are nicely shaped. Again, this process can be performed either with respect to triangles in the parameter space or with respect to the lifted triangles at the actual boundary of the AWV cell.

4.6.1 Computing the straight approximation

Let $G=(V,E)$ denote the graph describing the spherical subdivision we have computed so far. To compute a straight approximation of $\pi(G)$, we have to determine for each edge $e=\{u,w\}\in E$ of the spherical subdivision how many additional vertices $v_1,\ldots,v_r(e)$, $r(e)\geq 0$, we have to introduce in order to guarantee that the final approximation is topologically correct.

Definition 17 *Let $e=\{u,w\}$ be an edge with a straight approximation u, $v_1,\ldots,v_{r(e)}$, w. We call this approximation a δ (-straight) approximation, if the arc length along $\pi(e)$ between any pair of consecutive points from the sequence $\langle\pi(u),\pi(v_1),\ldots,\pi(v_r),\pi(w)\rangle$ is less than δ.*

Our goal is to determine for each edge $e\in E$ a value δ_e, such that the following holds: For each $e=\{u,w\}\in E$ we fix an arbitrary δ_e-straight approximation $\pi(H_e)$. Then $\pi(H)$, where $V(H)=\bigcup_{e\in E}V(H_e)$, $E(H)=\bigcup_{e\in E}E(H_e)$, $\pi(V(H))=\bigcup_{e\in E}\pi(V(H_e))$, and $\pi(E(H))=\bigcup_{e\in E}\pi(E(H_e))$, is a straight approximation of $\pi(G)$.

We determine δ_e, $e\in E$ by examining each pair of edges $e_1,e_2\in E$ that are incident to a common face $f\in F(G)$ of the spherical subdivision. Let us begin by considering two special cases:

- If e_1 and e_2 form a diangle, that is, they have both endpoints in common, then we have to introduce at least one additional point on each edge.
- Let c_1 and c_2 denote the circles such that $\pi(e_1)\subset c_1$ and $\pi(e_2)\subset c_2$. If both c_1 and c_2 are great circles on S^2, then their approximations will not interfere if $\delta_{e_1},\delta_{e_2}\leq\frac{\pi}{2}$. Hence we assume that at least one of c_1 and c_2 is not a great circle.

We distinguish the following cases:

1. c_1 and c_2 do not intersect. If c_1 and c_2 are on different hemispheres, i.e. there exists a plane α containing the origin that separates c_1 and c_2, then any straight approximations of e_1 and e_2 will not interfere as they are separated by α. Hence, we may assume that c_1 has a larger radius than c_2, and that c_2 is contained inside c_1 with respect to any hemisphere containing c_1.

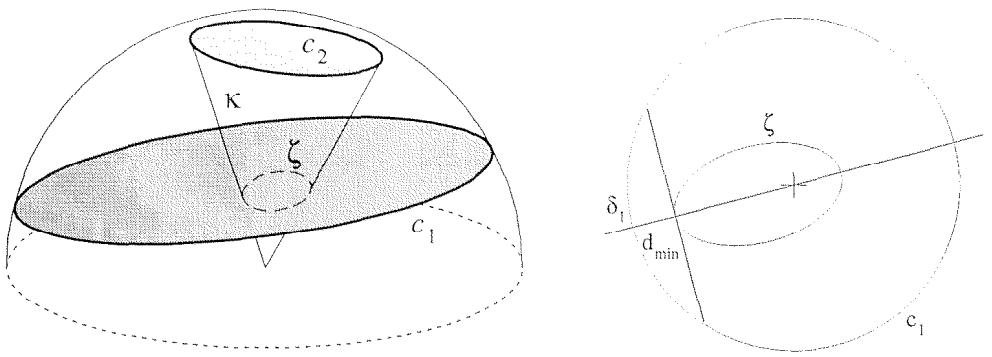


Figure 4.13: Finding the arc length $\delta_1 = \delta_{e_1}$ when the two circles do not intersect. In the left picture, we see a perspective view of the configuration. The smaller circle c_2 is contained within circle c_1. The cone κ, spanned by c_2 from the origin, intersects the plane γ_1, the plane defining c_1, in an ellipse ζ. As shown in the right picture, the long axis of ζ contains the origin of c_1. The most restrictive constraint on δ_1 is given by the nearest point of ζ to c_1. Hence, the arc length of the segment of c_1 defined by the cap of height $d_{\min}$, the minimum distance between c_1 and ζ, should be taken as an upper bound on δ_1.

Let a be an arc on a great circle on S^2 connecting its endpoints $u, v \in S^2$. We call the set of all rays emanating from the origin through a point $p \in a$ the *curtain* $\mathrm{cur}(a)$ spanned by a. Given two points $u, v \in S^2$, the curtain $\mathrm{cur}(u, v)$ spanned by u and v is the curtain spanned by the shortest great arc connecting u and v.

We will determine a value δ_{e_1}, such that for all points $u, v \in c_1$ with an arc distance $d(u, v) < \delta_{e_1}$ we have $\mathrm{cur}(u, v) \cap c_2 = \emptyset$. Observe, that any straight approximation of e_2 is contained inside the cap bounded by c_2.

Let κ denote the cone spanned by c_2 with its tip at the origin. Let ζ denote the intersection $\kappa \cap \gamma_1$, γ_1 being the plane such that $\gamma_1 \cap S^2 = c_1$. According to our assumptions, ζ is an ellipse. Given two points $u, v \in c_1$, we have $\mathrm{cur}(u, v) \cap c_2 = \emptyset$ if and only if the segment $\overline{uv} \cap \zeta = \emptyset$.

By symmetry considerations, the long axis of ζ contains the origin of c_1. Hence, we obtain a bound for δ_{e_1} by taking the arc length l of the circular segment whose height is the distance $d_{\min}$ between ζ and c_1, which can be easily computed using elementary geometry. See figure 4.13 for an illustration.

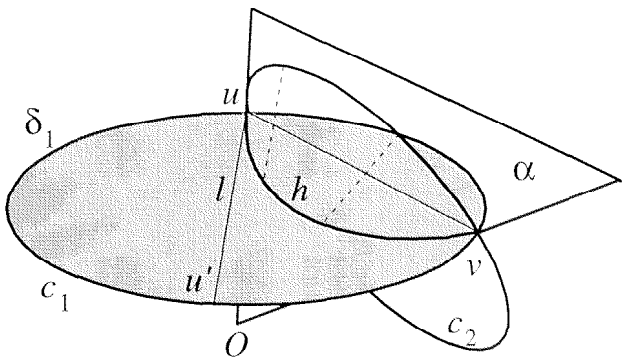


Figure 4.14: Finding the arc length $\delta_1 = \delta_{e_1}$ when the two circles c_1 and c_2 do intersect in the points u and v. The hyperbolic arc h is the intersection of the cone spanned by c_2 from the origin with the plane containing c_1. l is the tangent to h at u and has a second intersection u' with c_1. δ_1 must be bounded by the length of the arc from u to u'. The curtain α spanned by u and v separates straight approximations on c_1 and c_2 in the upper right diangle.

2. c_1 and c_2 have a non-empty intersection in two vertices u and v, see figure 4.14. We will only discuss the restrictions imposed on $\delta_1 = \delta_{e_1}$ due to the presence of e_2. The case for δ_{e_2} is symmetric. Let β be the plane such that $\beta \cap S^2 = c_2$. β partitions the space and hence c_1 into two halves, one of them containing the origin. In accordance with the picture we call the half containing the origin the *left* and the other one the *right* half. Similarly the plane γ, defined by $c_1 = S^2 \cap \gamma$, divides c_2 into an *upper* and a *lower* part. Obviously, e_1 is either contained completely in the left or in the right half, as e_2 is either contained in the *upper* or in the *lower* part of c_2. We can distinguish the following cases:

   (a) e_1 is contained in the right part of c_1 and e_2 is contained in the upper part of c_2: Then the curtain $\alpha = \mathrm{cur}(u, v)$ separates any straight approximations of e_1 and e_2, as long as the rule for diangles is respected.

   (b) e_1 is contained in the left part of c_1 and e_2 is contained in the lower part of c_2. Then the edges imply no restrictions upon each other, since any straight approximation of e_1 is in the upper part while any straight approximation of e_2 is in the lower part.

   (c) e_1 is in the left part and e_2 is in the upper part. Let κ denote the (infinite) cone spanned by c_2 from the origin. Any straight approximation

of e_2 is contained in the interior of κ. Let h denote the hyperbolic arc given as the intersection of the plane defining c_1 with κ. A straight approximation of e_1 does not interfere with e_2 if no curtain spanned by to adjacent vertices intersects h. By continuity, we only have to check the restrictions imposed on δ_1 at two locations: First, we have to look at the apex of h, which is similar to the elliptic intersection we discussed previously. Second, we have to check the extremal locations near the intersections u and v. Precisely, let l be the tangent line at κ in u within γ. Besides u, l has another intersection point u' with c_1. The length of the arc uu' is an upper bound on δ_1.

Let us sketch the computation of u': Let r denote the radius of c_2, C its center. Then every point $X = (x, y, z)$ of κ fulfills the equation

$$f(X) = \|X \times C\| - \frac{r}{\|C\|}\langle X, C\rangle = 0,$$

since $\tan\alpha = \frac{r}{\|C\|}$, where

$$\cos\alpha = \frac{\langle X, C\rangle}{\|X\|\,\|C\|}, \quad \sin\alpha = \frac{\|X \times C\|}{\|X\|\,\|C\|}.$$

Hence, the tangent plane to κ at u, which contains the origin, is given by a linear equation

$$\frac{\partial}{\partial x} f|_u\, x + \frac{\partial}{\partial y} f|_u\, y + \frac{\partial}{\partial z} f|_u\, z = 0,$$

that, together with the linear equation of γ yields an expression for l.

(d) e_1 is in the right part and e_2 is in the lower part. This is the same case as before with the roles of e_1 and e_2 interchanged.

4.6.2 Triangulating simple polygons

In this subsection we deal with the problem of computing a triangulation of a simple polygon on a sphere, where all edges are embedded on subsets of great circles on the sphere. Since these polygons are contained in one octant of the coordinate system, this problem is trivially equivalent to triangulating a simple polygon in the plane.

There is a vast amount of literature devoted to triangulating simple polygons, such as Caray et al. (1978), Asano et al. (1986), or Atallah and Goodrich (1986).

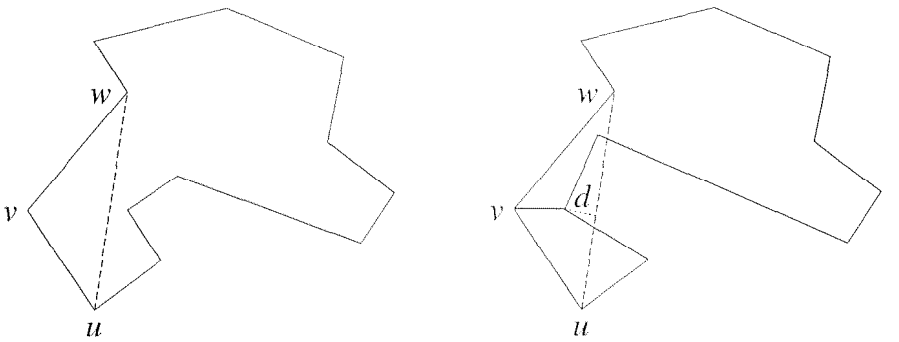


Figure 4.15: Proof of the existence of a triangulation of a simple polygon P. Let wvu be a convex triangle along the boundary of P. If no boundary edge of P intersects uw, then uw can be chosen as an edge of the triangulation, as shown in the left picture. Otherwise, there is a vertex v' inside this triangle that maximaizes the distance d to the edge uw. Then, as shown in the right picture, vv' can be chosen as an edge of the triangulation. Adapted from de Berg et al. (1997).

An optimal yet rather complicated solution running in linear time was given by Chazelle (1990). Perhaps the most practical solution is the algorithm given by Seidel (1991) running in time $O(n \log^* n)$. However, all these algorithms are based on some vertical decomposition of the input domain, either implicitly in terms of a sweep line algorithm, or even explicitly by computing a trapezoidal decomposition of the input domain, from which the actual triangles are then extracted in a second step.

Since we wanted to avoid using vertical decompositions, and considering that almost all our polygons have less than 30 vertices, we questioned whether one of these algorithms would be an appropriate choice for our problem. We decided to implement a very simple heuristically motivated algorithm, that tries to exploit the fact that most of the polygons in our setting are "almost convex". The algorithm is based on the well known proof of the fact that a simple polygon admids a triangulation, see for example the textbook by de Berg et al. (1997). See figure 4.15 for a short review of this proof.

The algorithm works in three phases to triangulate a simple polygon P: an optimistic phase, a cautious phase, and a "panic" phase.

- **Optimistic phase.** The algorithm proceeds as if it had to triangulate a convex polygon. Starting at an arbitrary edge along the boundary, the algorithm seeks a convex corner wvu of P and pushes wu as candidate edge on a stack.

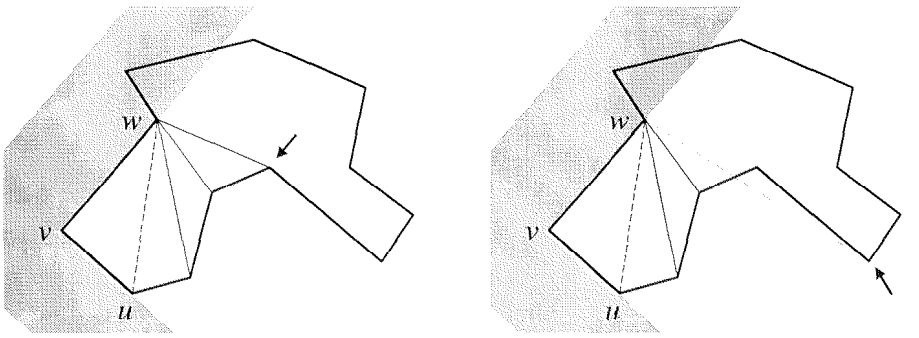


Figure 4.16: Left picture: During the optimistic phase, the algorithm walks along the boundary and tries to push as many candidate edges as possible onto the stack. All these edges have w as one endpoint and have their other endpoint in the cone spanned by vw and vu. When the first boundary vertex p is reached such that wp cannot be pushed onto the stack, the algorithm enters the cautious phase, as shown in the right picture. All candidate edges, that are not located clockwise with repect to wp are popped of the stack.

Then, it proceeds traversing the boundary as long as the following two conditions are fulfilled:

- The current boundary vertex p is contained in the positive cone spanned by vw and vu.
- The edge on top of the stack is located clockwise with respect to the ray wp.

If these conditions are true, wp is pushed as new candidate edge on top of the stack. For an illustration, see the left picture of figure 4.16.

- **Cautious phase.** The algorithm continues to walk along the boundary of P, but it checks which edges have to be removed from the stack. For each boundary vertex p, such that p is contained in the positive cone spanned by vw and vu, the algorithm pops all edges from the stack that are located counterclockwise with respect to wp. When the algorithm is to pop off the initial edge wu, it enters panic mode. For an illustration, see the right picture of figure 4.16.

- **Panic mode.** The algorithm continues walking along the boundary of P until the initial vertex w is reached again. Along its way it remembers the

vertex that maximizes the distance d with respect to the edge uw as shown in the right picture of figure 4.15.

After having traversed the boundary of P, the algorithm either creates all candidate edges that are still on the stack, or, if the stack is empty, creates the edge vv' to the maximal violator of wu. The pseudo code of this algorithm is given in figures 4.17 (optimistic and cautious phase) and 4.18 (panic mode). The following functions and notions have not been introduced yet:

1. curtain($vertex_1, vertex_2$) computes a an oriented plane α defining the great circle c, such that both $vertex_1$ and $vertex_2$ are located on c. According to our conventions, the normal snrm(α) points to the left with respect to the oriented line from $vertex_1$ through $vertex_2$.

2. *vertex*.coordinates $\in$ *space* tests if the embedding of *vertex* is contained in a halfspace. For this notion to be well-defined, we have to remind the convention that a halfspace H is defined as a set $\{x : \langle a,x\rangle \leq b\}$, a being the normal of the oriented boundary plane of H.

Implementation note. *The pseudo code of the algorithm is simplified in that the actual implementation decides these tests combinatorially where appropriate to avoid problems of degeneracy. These might otherwise occur along refinements of edges that have been (almost!) great circles in the spherical subdivision before the approximation process.*

4.6.3 Computing the Delaunay triangulation

To turn the triangulation so far computed into a Delaunay triangulation, we use the well-known Lawson (1977) flip that flips edges in the triangulation if they violate a locally formulated in-circle predicate.

Constraint Delaunay trianglation of the sphere. Let us first define what we mean by an arc between two points on a sphere:

Definition 18 *Let $u, v \in S^2$, u, v being not antipodal points. Then the arc uv is the shortest circular arc from u to v within the surface S^2.*

```
triangulate-polygon(halfedge):
    exit if number of edges less than 4.

    Find halfedges ancor, first, and second, such that
    ancor.next = first, first.next = second, and
    (ancor.vertex, first.vertex, second.vertex) is a correctly oriented triangle.

    push(second, curtain-plane(second.vertex, ancor.vertex)
    first-space := curtain(ancor.vertex, first.vertex)
    second-space := curtain(first.vertex, second.vertex)
    iterator := second.next, last-edge := NIL

    while iterator.next ≠ ancor ∧
          iterator.vertex.coordinates ∈ top.space ∧
          iterator.vertex.coordinates ∉ first-space ∧
          iterator.vertex.coordinates ∉ second-space
    do
          push (iterator, curtain (iterator.vertex, ancor.vertex))
          iterator := iterator.next
    end

    while iterator ≠ ancor do
          if iterator.vertex.coordinates ∉ first-space ∧
                iterator.vertex.coordinates ∉ second-space
          then
                while iterator.vertex.coordinates ∉ top.space do
                      if stack.elements > 1 then
                            pop
                      else
                            Handle maximal violator and return, see figure 4.18.
                      end
                end
          end
          iterator := iterator.next
    end

    for entry ∈ stack do
          link(ancor, entry.edge, new face)
    end
    triangulate-polygon(last created edge)
```

Figure 4.17: A simple algorithm for triangulating simple polygons on a sphere.

```
Handle maximal violator in triangulate-polygon(halfedge):
    max-value := ⟨top.space.normal, iterator.vertex.coordinates⟩
    last-edge := iterator, iterator := iterator.next

    loop
        if iterator.vertex.coordinates ∉ first-space ∧
            iterator.vertex.coordinates ∉ second-space ∧
            ⟨top.space.normal, iterator.vertex.coordinates⟩ > max-value
        then
            max-value := ⟨top.space.normal, iterator.vertex.coordinates⟩
            last-edge := iterator
        end

        iterator := iterator.next
        exit if iterator = ancor
    end

    new-edge := link(first, last-edge, new face)
    triangulate-polygon(new-edge.twin)
    triangulate-polygon(new-edge)
    return
```

Figure 4.18: Inner part of algorithm for triangulating simple polygons on a sphere.

Observe, that any arc defined in this way is a subset of a great circle on S^2. The following definitions try to transfer the corresponding definition from the planar setting[7] onto the sphere:

Definition 19 (Triangulation) *Let G be a planar graph, $\pi(G)$ a straight embedding of G on the unit sphere S^2. Let $T = (V,E)$ be a planar graph with $V = V(G)$, $E \subset E(G)$ and its straight embedding $\pi(T)$, such that $\pi(T)|_G = \pi(G)$. If each face induced by $\pi(T)$ is a triangle then we call $\pi(T)$ a triangulation of $\pi(G)$.*

Definition 20 (Constraint Delaunay triangulation) *Let $\pi(G)$ be the straight embedding of a graph G on the unit sphere S^2. Assume that the convex hull $\mathrm{conv}(\pi(V(G)))$ contains the origin inside its interior. Let $\pi(T)$ be a triangulation of $\pi(G)$. Then we call $\pi(T)$ a constraint Delaunay triangulation of $\pi(G)$ if the following conditions hold:*

1. *For all edges $e = \{u,v\} \in E$ we have that u and v are not antipodal points on S^2.*

2. *For all edges $e = \{u,v\} \in E \setminus E(G)$ let $\alpha_{u,v}$ be the oriented plane $\{x : \langle a,x\rangle = b\}$ with normal $a_{u,v} = \frac{u+v}{2}$ and $b_{u,v} = \langle u, a_{u,v}\rangle$. If there is another vertex w embedded on $\pi(w)$ such that $\langle \pi(w), a_{u,v}\rangle > b_{u,v}$, then there is an edge $e' = \{s,t\} \in E(G)$, such that $\pi(e')$ cuts the great circular arcs uw resp. vw.*

Note that the intersection of $\alpha_{u,v} \cap S^2$ is a circle, so this definition basically specifies an *in-circle* test. A related definition, *constraint convex hulls*, was given by Akkiraju (1996) in his thesis.

The flip algorithm, however, uses an even simpler predicate: Let abc, cbd be two adjacent oriented triangles. Let α_{abc} be the oriented halfspace containing $\pi(a), \pi(b)$, and $\pi(c)$ on its boundary plane, such that $\mathrm{snrm}(\alpha_{abc})$ points away from the center of S^2. The flip algorithm replaces the triangles abc and cbd by adc, abd, if $\{bc\} \notin E(G)$ and $\pi(d) \notin \alpha_{abc}$. If all points $\pi(a), \pi(b), \pi(c)$ are contained in the interior of a single hemisphere, then this criterion has the following "in-circle" formulation: Let D denote the open disc $S^2 \cap \neg\alpha_{abc}$. The flip is performed if $\{bc\} \notin E(G)$ and $\pi(d) \in D$. Currently, this is the default "Delaunay predicate" implemented in the algorithm.

[7] Cf. Bern and Eppstein (1995)

Constraint Delaunay triangulation of the boundary surface. It is also possible to formulate an in-circle predicate with respect to the actual boundary surfaces of the AWV cell. To find a suitable definition of what a "Delaunay triangulation" on a curved surface might be, Chew (1993) proposed to do the definition the other way around: First define a suitable concept of what a "circle" should be, then plug this definition in the form of an in-circle test into the flip algorithm:

Definition 21 (Chew (1993)) *Given three vertices on a curved surface, consider the infinite set of spheres through the three vertices. The centers of all these spheres lie on a single line. We choose the sphere whose center is on the surface and define the circumcircle of the three vertices to be the set of points where this sphere intersects the surface.*

As Chew points out, the advantage of this definition is that finding the circumcenter is basically equivalent to finding the intersection of the surface with a line, while the in-circle test is reduced to checking the distance of a vertex to the circumcenter in 3D. He notes that the normals on the portion of the surface that is within the union of the circumcircles should not vary by more than $\frac{\pi}{2}$. Therefore, starting from a suitably refined triangulation, we can plug this predicate into the flip algorithm to compute a CDT of the surface of the cell.

4.6.4 Refinement of the triangulation

The algorithms by Chew (1993) and Ruppert (1995) for refining constraint Delaunay triangulations are basically very simple: First, the algorithm needs a predicate that can be applied to a triangle to tell if the triangle has to be processed further or if it fulfills the specified requirements. These requirements can state, for example, that the minimum angle of the triangle has to be greater than 20^o and that its area must not exceed a certain amount.

The algorithm maintains a queue of these "bad" triangles. As long as this queue is not empty, the algorithm selects a triangle t based on some strategy, such as the largest triangle, or the triangle with the worst angle. Let c denote the circumcenter of t. The algorithm tries to insert c as a new vertex into the triangulation and reestablishes the Delaunay property. However, if c happens to be close to a prespecified edge that has to be maintained, than the algorithm may decide to split that boundary edge instead. Ruppert (1995) proposes to split such an edge e if c is located inside the diametrical circle of e, i.e. the smallest circle containing

e. Chew (1993), on the other hand, suggests to split an edge only if it has to be crossed when traversing the triangulation from t to c.

In the present implementation, we handle each face separately and hence do not split boundary edges. We define a triangle to be bad, if its area is greater than a certain parameter A_1 and it has a smallest angle less than a parameter ω, or if its area is greater than another parameter value A_2 with $A_2 > A_1$. In this way, the algorithm is always guaranteed to stop.

4.7 The graphical user interface

To make the algorithm and its implementation described in this thesis accessible to a non-expert audience, a graphical user interface was created. See figure 4.19 for a screen shot. Each molecule is displayed in a split frame window hosting two panes with different views of the molecule. The tree control to the left displays the hierarchical structure of the molecule as chains comprised of residues having individual atoms. The window to the right shows a 3-dimensional image of the covalent structure of the molecule. Atoms can be selected either by point-and-click into the right window, or by selecting atomic groups in the tree control.

Having selected a set of atoms, the user can choose to calculate a graphical representation of the AWV cells of the selected atoms, which is then included in the graphics window. Moreover, the volumes of the AWV cells of the selected atoms can be calculated. The volume information is captured into tables that are displayed in separate windows. These tables can be stored onto disk for further processing using another program, most notably a spreadsheet application such as Microsoft Excel. Besides AWV volumes, the program can also compute atomic volumes using Richard's B method, the radical plane method, and, of course, the volume of the unweighted Voronoi cells defined by the atomic centers.

A dedicated window allows the user to specify rules that radii to the specific atoms, see figure 4.20. These rules are specified using to mappings, the first assigning a hybridization code to a residue/atom specification, and a second mapping assigning a radius to each hybridization code.

Finally, and this feature is rather relevant for applications in biochemistry, the user can request the program cut off the computed AWV cells at a specified distance d_0. The two cells shown in the screenshot in figure 4.19 have been calculated this way. To calculate these restricted cells, the algorithm computing the AWV cell of

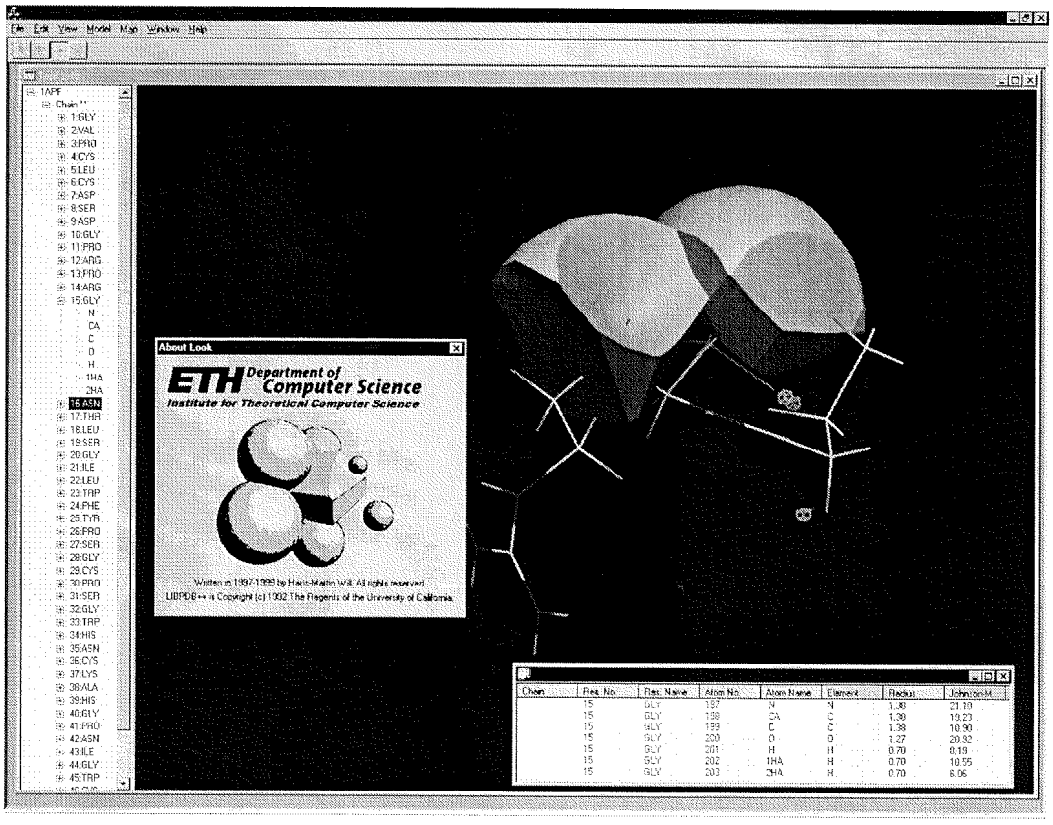


Figure 4.19: The graphical user interface implemented around the algorithm. The right pane shows the AWV cells of two atoms that have been cut off at a distance of 1.4Å from the atomic surfaces.

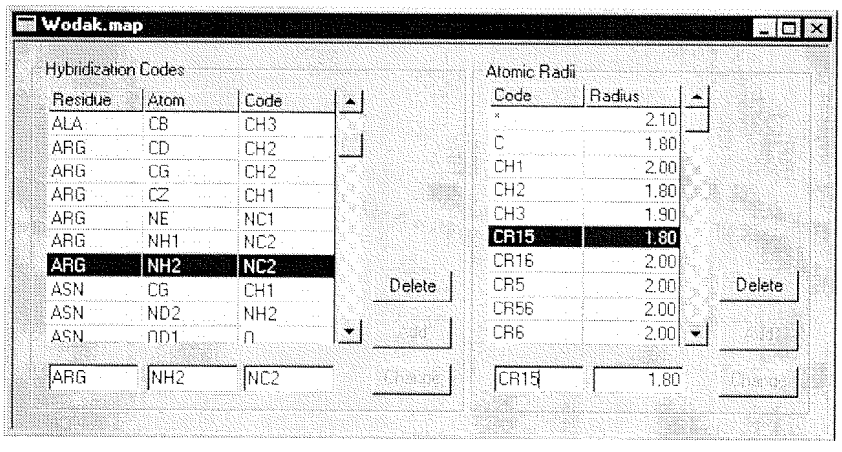


Figure 4.20: Setting up radius rules.

a single sphere σ is slightly modified in the following way: Remember that the boundary of the projection of each bisector surface ϕ onto the parameter space S^2 is a circle. The algorithm represents this circle bounding the domain of ϕ using an oriented plane. The set of points

$$\{x \in \phi : d(x, \sigma) = d_0\}$$

is also either empty or a circle. Hence, by adjusting the planes used to trim the domains of the bisector surfaces, the algorithm can compute the intersection of the AWV cell $V(\sigma)$ with as sphere $(c_\sigma, r_\sigma + d_0)$.

Conclusions

In this chapter, we have described the implementation of an algorithm for computing AWV cells. We chose an engineering approach to this problem, trying to focus on a practical solution that could be implemented with a reasonable effort. We incorporated this implementation into an intuitive graphical user interface for application by non-expert users.

When working on the implementation, we found it very surprising that the area of meshing, something we considered as trivial post-processing in the first place, still lacks a rigorous understanding as soon as our input is more complex than a planar straight line graph. Considering the numerous possible applications of meshing algorithm ranging from numerical mathematics to computer graphics, we believe that designing meshing algorithms for non-linear input in non-planar domains will remain an important and active area of research.

Chapter 5

Practical considerations and experimental results

In this chapter, we want to examine how our approach to computing AWV cells behaves in practice. We will focus on two central issues. In the first part, we will take a close look at the combinatorial complexity of the individual AWV cells as they arise from computations on biological macromolecules. Of special interest will be the relationship between the combinatorial complexity of an AWV cell and the combinatorial complexity of the corresponding 4-dimensional power cell as given by Aurenhammer's lifting procedure.

The second part of this chapter is dedicated with questions related to numerical robustness and computational resources required by our implementation. After a short overview of different approaches relevant to deal with numerical errors and degenerate input configurations, we will present and discuss the engineering approach we chose for our implementation. We will provide experimental evidence to provide a profound argumentation in favor of this decision.

Additionally, we will describe our experience with other approaches to compute AWV cells: We will describe the behavior of an implementation the direct extraction of an AWV cell from its 4-dimensional power cell, and we will describe a simulation of the numerical behavior of a vertical decomposition approach to compute the spherical subdivision describing an AWV cell. Moreover, an appendix provides detailed information on the setup used for the experiments reported in this chapter.

5.1 Cell complexities

In the previous chapters, we related the running time and space requirements of our algorithms to the combinatorial complexity of the computed cells. Therefore, we begin by examining the complexities of AWV cells as they arise in the intended domain of application. All the results to be presented in the following discussion have been derived from a distinct set of 10 molecule entries selected from the Brookhaven Protein Data Bank (PDB)[1] having a total number of 17196 atoms. Detailed information on the choice of these entries is provided in the appendix of this chapter.

First, we will examine the distribution of the combinatorial complexities of AWV cells as defined by these data sets. Then, we will relate these values to the location of the atoms within the molecule, that is, if the defining atom of the cell is located on the outer surface, on the surface of a cavity or in the interior of the molecule. Finally, we will look at the ratio of the combinatorial complexity of the 4-dimensional power cell of an atom as defined by Aurenhammer's lifting procedure divided by the combinatorial complexity of the AWV cell. We will see that this ratio is related to the radius of the defining atomic sphere.

5.1.1 Overall combinatorial complexity of AWV cells

The following statistics were obtained by processing the specified data sets with the implementation described in the previous chapter. After each computation of a cell V_i, $1 \leq i \leq n$, n the number of atoms of the molecule, the total number of combinatorial vertices $n_V^{(i)}$, edges $n_E^{(i)}$ and faces $n_F^{(i)}$ together with the identification of the defining atom were recorded. We point out, that faces refined by helper edges were glued together for this counting. We write

$$\#AWV_i = n_V^{(i)} + n_E^{(i)} + n_F^{(i)}$$

for combinatorial complexity computed in this way.

Figure 5.1 shows the distribution of the overall complexity of the AWV cells as defined by our data sets. We consider the smoothness of this graph to be a strong indicator of the staistical relevance of the results.

The average value of the overall combinatorial complexity of an AWV cell we computed a value of 82.17. Only 39 out of all 17196 cells, that is less than 0.227%,

[1] The main site of the PDB is currently located at `http://www.rcsb.org/`.

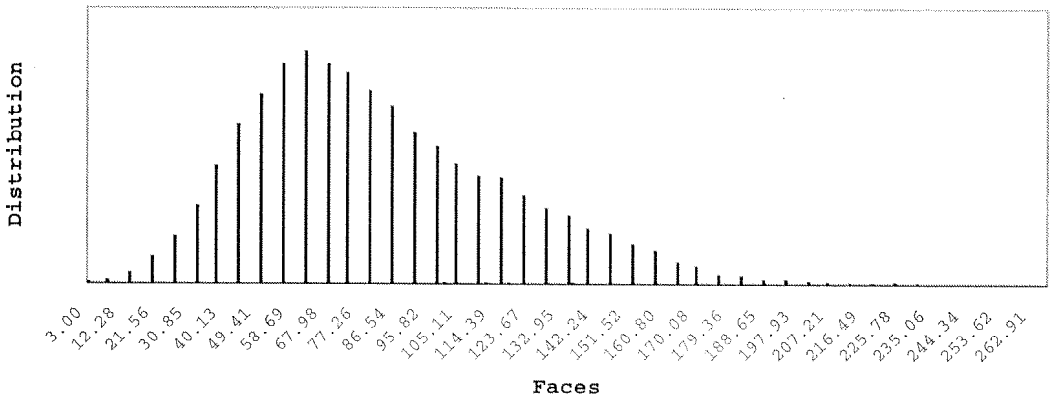


Figure 5.1: Distribution of the total complexities of the AWV cells as defined by our sample data set taken from the PDB.

Type	Atoms	Average	Std. Dev.
Internal	13252	86.58	36.90
Cavity	550	71.61	19.62
External	3121	64.91	20.63

Table 5.1: Dependency of the average overall combinatorial complexity of AWV cells on the location of the defining atom.

have more than 200 faces. The most complex cell has 266 faces, the simplest cells are lens-shaped with 3 faces of all dimensions — two 2-dimensional faces separated by a closed edge. From a practical algorithm designer's point of view, it is therefore admissible to assume AWV cells arising in the application domain have rather moderate combinatorial complexities.

An interesting question is if there is any relationship between the combinatorial complexity of an AWV cell and the location of the defining atom within the molecule. Using the classification as defined by Kleywegt and Jones (1994), we labeled each atom either as surface, cavity or interior atom. An interior atom is an atom that cannot by touched by a probe sphere of radius $R = 1.4$Å, the radius of a water molecule, without the probe sphere intersecting any other neighboring atomic sphere. A cavity atom is an atom that can be touched by the probe sphere, and that is bounding a compartment of space that either already is closed of the external solvent volume, or that would be closed off, if the atomic radii were to be increased. The remaining atoms are external atoms.

Table 5.1 shows how the overall cell complexity relates to the location of the defin-

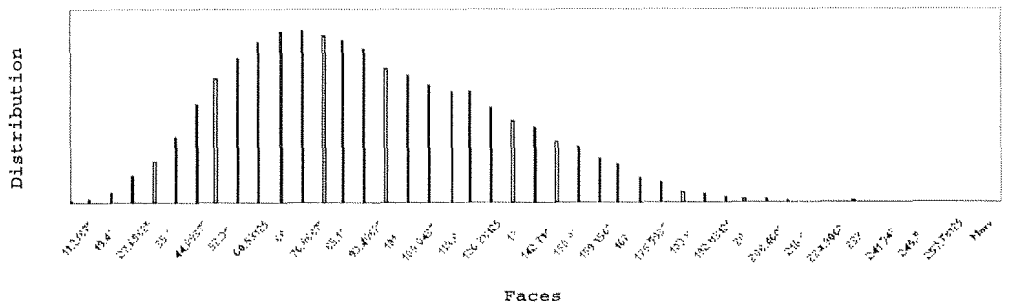


Figure 5.2: Distribution of the total complexities of the AWV cells of cells defined by internal atoms.

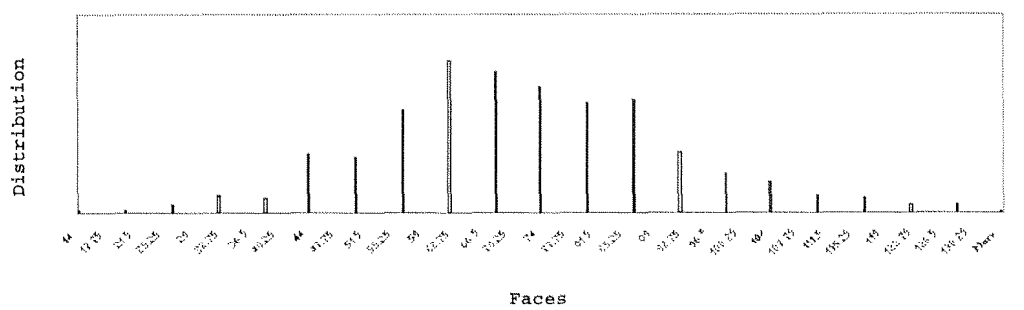


Figure 5.3: Distribution of the total complexities of the AWV cells defined by cavity atoms.

ing atom within the molecule. The average complexities of cells whose defining atom is located on the surface of a cavity and those located on the outer surface show an average complexity of 71.61 and 64.91, respectively, values that are significantly lower than the overall average value of 82.17. See also figures 5.2, 5.3, and 5.4.

Observe, that for interior atoms, the following argument gives a constant worst-case upper bound on the combinatorial complexity of the cell. Halperin and Overmars (1994) give a similar argument to bound the complexity of the description of the solvent accessible surface of an atom.

Proposition 8 *Let* $M = \{\sigma_1, \ldots, \sigma_n\}$ *be the collection of atomic spheres of a molecular model. The combinatorial complexity of a single cell* $V(\sigma)$, $\sigma \in M$,

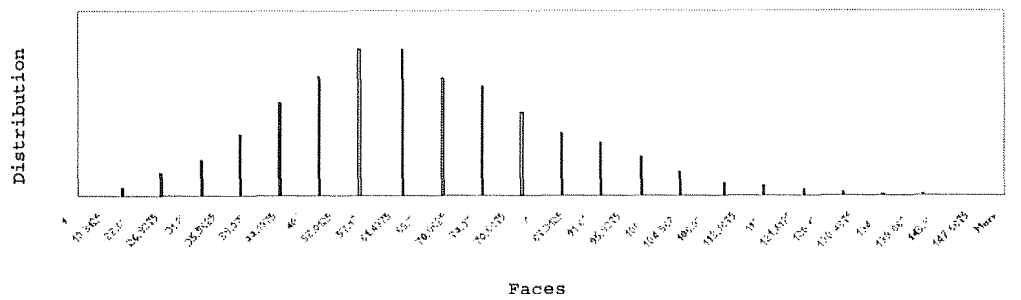


Figure 5.4: Distribution of the total complexities of the AWV cells defined by external atoms.

i.e. the total number of vertices, nodes and faces incident with $V(\sigma)$, σ *being a sphere taken from the interior of* M, *is bounded by a constant depending on the radius* R *of the probe sphere, the minimum distance* $d_{\min}$ *between atomic centers and maximum radius* $r_{\max}$ *of all atomic spheres.*

Proof: Let $T := \{\tau \in M : \operatorname{cl}(V(\sigma)) \cap \operatorname{cl}(V(\tau)) \neq \emptyset\}$. Since σ is interior, we have $d(c_\sigma, \tau) \leq 2R + r_{\max}$, which implies $c_\tau \subset B_{2(R+r_{\max})}(c_\sigma)$ for all $\tau \in T$. Since $d(c_i, c_j) \geq d_{\min}$ for all $1 \leq i \neq j \leq n$, we get

$$|T| \leq \left(\frac{4R + 6r_{\max}}{d_{\min}}\right)^3$$

□

Yet, we see that cells not located in the interior of the molecule tend to have even lower complexity, decreasing with the amount the defining atom is exposed.

We also found a relation between the atomic radii and the cell complexity. Smaller atoms tend to have less complex AWV cells, and especially the least complex cells are the lens-shaped cells that occurred around the atomic spheres of smallest radius, namely the hydrogen atoms.

5.1.2 Relation between AWV cells and 4D power cells

Let σ_i and σ_j, $1 \leq i \neq j \leq n$, be two distinct spheres from the input data. Lemma 1 implies that whenever V_i and V_j have a common face then so have the corresponding power cells P_i and P_j obtained by Aurenhammer's lifting procedure. However,

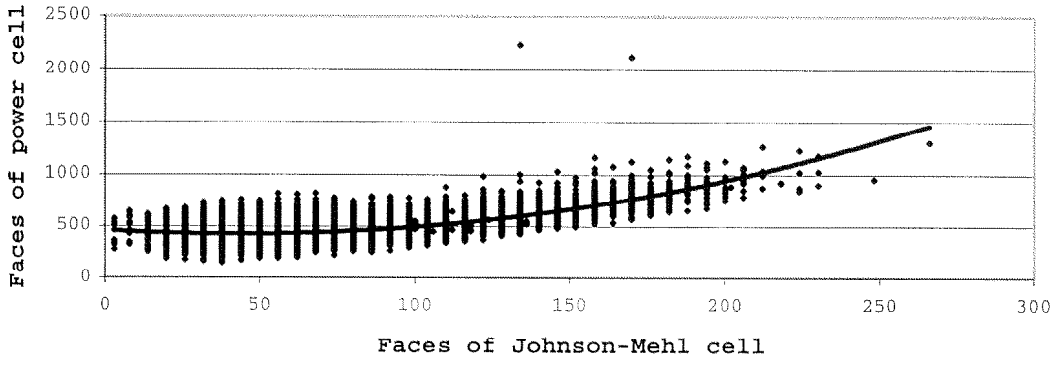


Figure 5.5: Complexity of power cell used in lifting construction versus actual complexity of additively weighted Voronoi cell.

the opposite direction of this implication is generally not true. Therefore, an interesting question is the relation between the combinatorial complexity $\#AWV_i$ of an AWV cell and the combinatorial complexity $\#PC_i$ of its corresponding power cell. As shown earlier, from the theoretical point of view, both a 4-dimensional power cell as well as a 3-dimensional AWV cell can realize up to $\Theta(m^2)$ faces of all dimensions, if m is the number of neighbors.

To examine this relation between $\#AWV_i$ and $\#PC_i$, we modified our implementation to record these two values for each cell computed for an atom of our specified data set. Figure 5.5 shows a plot relating these two numbers for each cell computed. Of primary interest was the ratio $\rho_i = \frac{\#PC_i}{\#AWV_i}$. We computed an average value $\bar{\rho}$ of this ratio as $\bar{\rho} = 6.486$. As we can see from the plot, the dependency of $\#PC_i$ as a function of $\#AWV_i$ is not simply linear. In fact, in the figure a least squares fit of degree 2 is shown. The fit is approximately the function

$$x \mapsto \frac{1}{4}x^2 - \frac{17}{10}x + 561.$$

However, a per-se quadratic relationship between these two numbers seemed to be rather inprobable. As it turns out, the atoms showing the largest values of ρ_i are all hydrogen atoms, whose AWV cells have very few faces but whose power cells have a rather average complexity. Table 5.3 on page 101 shows the data for those cells with the highest ratio ρ_i. To make the dependency of the complexity ration on the atomic radius even more explicit, we identified all atoms with $\rho_i > \bar{\rho}$ and sorted them by their radius. Table 5.2 gives the counts we obtained for the individual atom types.

The zinc ions (ZN) have to be treated as special case: ZN does not occur as part

Element	Radius	Count
H	0.70	5064
C	1.38	222
S	1.55	3
ZN	2.10	2

Table 5.2: Counts of cells with an above average ratio of ρ_i sorted by their element type.

AWV cell				Power cell				
Vertices	Edges	Faces	Total	Vertices	Edges	Ridges	Facets	Total
0	1	2	3	127	254	159	32	572
0	1	2	3	119	238	150	31	538
0	1	2	3	112	224	142	30	508
0	1	2	3	105	202	127	28	462
0	1	2	3	80	160	103	23	366
0	1	2	3	75	150	97	22	344
0	1	2	3	72	144	92	20	328
0	1	2	3	60	120	79	19	278
2	3	3	8	144	288	178	34	644
2	3	3	8	141	282	177	36	636

Table 5.3: Atoms with the highest ratio of the complexity of the power cell versus the complexity of the additively weighted Voronoi cell. All atoms are hydrogen atoms from the interior of the molecules.

of an amino acid. Rather, these ions are located in very specific chemical and geometric environments. Obviously, the ratio depends highly on the radius of the defining sphere. Or, putting it the other way around: The combinatorial complexity of a power cells $\#PC_i$ depends much less on the atomic radius than does the combinatorial complexity $\#AWV_i$ of the corresponding AWV cell.

5.2 Numerical behavior and robustness

All previous discussions about geometric computations were based on two simplifications: First, we assumed that we could perform exact computations over the real numbers that could be evaluated at unit cost per operation. Second, we

imposed certain non-degeneracy conditions on the input data. However, real computers can perform only arithmetic of finite precision at unit cost, and degenerate input data *does* occur in practice. In this section, we will discuss different approaches to deal both with the limitations imposed by real world hardware and the problems posed by degenerate input data. Schirra (1998) gives an up-to-date survey on robustness and precision issues in geometric computations.

5.2.1 Exact computation

In implementations of geometric algorithms, exact arithmetic over the real numbers is commonly replaced by using the machine's finite precision floating-point arithmetic. In fact, most workstations and personal computers on the market today provide hardware implementations of floating-point operations as defined by the IEEE 754 standard[2]. Goldberg (1991) gives a thorough introduction to IEEE floating-point arithmetic. Yet, floating-point computations suffer from numeric round-off errors that can lead to incorrect results or even may crash the algorithm due to internal inconsistencies.

More precisely, most geometric algorithms can be formulated in terms of purely combinatorial objects and operations in conjunction with certain Boolean predicates, that are sign evaluations of functions in the coordinates of the input objects. When discussing our algorithms, we already made these predicates explicit. Evaluation of these functions in the input coordinates using floating-point arithmetic may lead to the situation that the sign of the corresponding value is not determined correctly. A wrong branch of the program — compared to an ideal implementation using exact arithmetic — might be taken, leading to undesired behavior such as wrong results or even program crashes and "core dumps".

We *define* an implementation of the geometric predicates to be exact if at each step the same decision is taken by the implementation of the algorithm compared to the theoretical counterpart formulated over the real numbers. Note, that this does *not* imply that for all numerical values exact representations have to be computed.

In the following, we give a brief overview of techniques proposed to implement exact computation on existing computer hardware.

[2]Cf. IEEE (1985)

Exact representations

Infinite precision libraries for integer and rational arithmetic. A large number of geometric predicates used within computational geometry algorithms are purely rational expressions. In fact, typical textbooks on computational geometry exclusively deal with problems whose geometric primitives can be written in terms of $+,-,*,/$ and sign determinations. Since a rational number can be represented as an integer, library packages for computing with arbitrary long integer numbers can be used to provide exact implementations of these predicates. Common libraries are BigNum by Serpette et al. (1989), GNU MP[3], PARI by Cohen (1993), or the `integer` and `rational` number types of the LEDA library, see Mehlhorn and Näher (1998). Of these packages, PARI and GNU MP are tuned for applications in computer algebra. For most problems, it is even possible to avoid division operations by embedding the Euclidean problem into projective space[4]. In this case, the geometric predicates can be formulated as taking the sign of the evaluation of a multi-variate polynomial.

However, the evaluation of geometric predicates implemented in this way is much more expensive than using built-in floating-point arithmetic. Karasick et al. (1997) report a slowdown factor of up to 10000.

Compiled multi-precision code. Fortune and van Wyk (1996) noticed that the bit-lengths of the integers involved in geometric calculations are rather small compared to those arising in computer algebra. For this reason, they developed a preprocessor LN ("little numbers") specifically designed to generate exact implementations of geometric predicates. Input to LN is a description of the input data types in terms of coordinates and bit-lengths, and the formulas defining the intermediate results and predicates the user wishes to compute. In a first step, LN computes for each expression the required maximum bit-length. Then, LN generates program code that evaluates the expressions using the required number of bits. LN also introduces several optimizations such as static floating-point filters (see below) and overlapping representations of intermediate results.

However, LN is not generally available and it has never been developed up to the point to be useful for a general audience. For further experience with LN see the

[3]GNU MP 2 was finished and released by TMG Datakonsult, Sodermannagatan 5, 11623 Stockholm, Sweden, in cooperation with the IDA Center for Computing Sciences, USA.

[4]Cf. Stolfi (1991)

paper by Chang and Milenkovic (1993).

Modular arithmetic. Brönnimann et al. (1997) describe an approach to compute the sign of an integer number using the Chinese remainder theorem[5]. Let $m_1, \ldots, m_k$ be a collection of pairwise relatively prime natural numbers, let $m = \prod_{i=1}^{k} m_i$, and assume that m is even. Then the system of modular equations

$$x \equiv x_i \bmod m_i, \ 1 \leq i \leq k$$

has a unique solution for $x \in [-m/2, m/2)$. Let $\mathrm{frac}(x) = x - \lfloor x \rfloor$. The basic idea of Brönnimann et al. (1997) for computing the sign of a large integer number x is to compute an approximation of the value

$$S = \frac{x}{m} = \mathrm{frac}\left(\sum_{i=1}^{k} \frac{(x_i w_i) \bmod m_i}{m_i} \right)$$

using a fixed number of b bits, where $v_i = m/m_i$ and $w_i \equiv v_i^{-1} \bmod m_i$. The authors give an error bound ε_k depending on b and k. They show that either $|S| > \varepsilon_k$, which implies that the sign of S is the same as the sign of x, or that $|x| < \prod_{i=1}^{k-1} m_i$, in which case the computation can be reduced to the case $k-1$. The authors implemented two extended versions of this basic method to compute the signs of determinants with integer coefficients. They report that these methods perform well compared to LN, especially if $|x|$ is small.

It has been observed repeatedly, that in most cases when the error in evaluating a geometric predicate exceeds the computed value, then the actual value is really zero[6]. Computer algebra systems routinely apply modular arithmetic to evaluate and check integer equalities[7]. Hence, it might make sense to check first for $x = 0$ before going into the approximation loop. Yet, the main disadvantage of using modular arithmetic is the fact that an implementation is rather involved.

Representation of algebraic numbers. If the geometric predicates not only involve the four arithmetic operations $+, -, *, /$ but also require the computation of roots, then more sophisticated techniques have to be applied to realize an exact implementation of the predicate. A general technique originating from computer

[5] Cf. Lang (1992)

[6] Cf. Schirra (1998)

[7] Cf. Geddes et al. (1992)

algebra is to implement an algebraic number system, where each number is represented by its defining minimal polynomial and an isolating interval[8]. The aforementioned arithmetic operations can be expressed using subresultants, and the separating intervals can be refined using binary search.

However, the bit-lengths of the polynomial coefficients arising in this representation can be very large[9]. A more practical approach is a kind of "simulation" of the subresultant evaluations, and to compute only *separation bounds* for numerical approximations of the numbers. This has been done in Real/Expr and the LEDA reals, and will be described in more detail below.

Exact computation on the sphere. Andrade and Stolfi (1998) presented a framework for performing oriented geometry on the sphere. They devised a scheme to compute the orientation test on circles on the sphere (see equation 4.2 on page 57) using integer arithmetic, given that all input circles are defined using integer coefficients. Andrade also implemented these predicates in Modula-3 based on the GNU-MP library. If all input coefficients have a maximal bit-length b, then we calculated a required bit-length of $10b + 14$ to evaluate the orientation predicate. As we will see in the next section, we have $b > 50$ for our implementation.

Adaptive computations

Adaptive or lazy approaches try to deliver exact results with minimal computational effort. Hence, these approaches choose a costly high-precision evaluation of a predicate only after checking that a cheaper method could not give the right answer. This laziness can speed up geometric computations significantly.

Floating point filters. The idea of a floating-point filter is to use floating-point arithmetic to compute the predicate. However, contrary to a naive implementation, the absolute value of the computed approximate result is compared to an error bound. When the absolute value is outside the error interval, then the sign of the floating-point approximation is known to be the exact sign. If the absolute value is within the error interval, then a more expensive method has to be used to evaluate the predicate.

[8] Cf. Loos (1983), Mishra (1993)

[9] Cf. Mishra (1993)

Implementations of floating-point filters vary in the type of error bounds used. In the simplest case, the error bound is derived from a static worst-case analysis of the predicate. This is, for example, the approach implemented by the LN preprocessor.

Another approach is to compute the error bound dynamically at run-time using the well known equations to compute numerical round-off errors. If the machine presicion is denoted ε, then the bound error(ϕ) of a floating-point expression ϕ can be computed recursively via

$$\begin{aligned} \text{error}(a \oplus b) &= \text{error}(a) + \text{error}(b) + \varepsilon|a \oplus b| \\ \text{error}(a \otimes b) &= \text{error}(a)|b| + \text{error}(b)|a| + \varepsilon|a \otimes b| \\ &\quad + \text{error}(a)\text{error}(b) \\ \text{error}(\text{sqrt}(a)) &= \varepsilon\sqrt{a} + \sqrt{\text{error}(a)} \end{aligned} \tag{5.1}$$

If these bounds are computed using floating-point arithmetic, then additional correction factors of the form $(1+\varepsilon)$ have to be taken into account after each floating-point operation. For IEEE double precision arithmetic, we have $\varepsilon = 2^{-53}$. Floating point filters of this kind have been implemented in LEDA or by Fortune and Van Wyk (1996).

Lazy evaluation schemes. Shewchuk (1996) suggested an adaptive evaluation scheme that reuses the results from an evaluation with a lower precision in the computation of the next, more precise evaluation of the predicate. He implemented an adaptive evaluation scheme for planar sidedness and in-circle tests using the multiprecision techniques proposed by Dekker (1971) and Priest (1991). This approach can be seen as a hand-tuned approach to creating predicate implementations similar to those generated by LN.

Numerical approximations based on separation bounds. As mentioned earlier, it is possible to provide a complete implementation algebraic number fields when roots have to be calculated. Each number α is represented by its minimal polynomial p_α over the rational numbers and an isolating interval identifying one root of p_α. Since the algebraic operations to compute these polynomials are rather involved, another approach has turned out to be quite practical: Instead of trying to compute the algebraic representation, the implementation computes a high-precision floating-point approximation. The required precision,

however is derived from root separation bounds that essentially capture the degree and complexity of the algebraic representation that *would have to be* computed[10]. To compute the approximations, the dag (directed acyclic graph) describing the expression is stored as a data structure, and it is evaluated operator-node by operator-node as required. Combined with filters using built-in floating-point operations only, this approach has been implemented in the library Real/Expr by Dubé et al. (1996), and as the number type `leda_real` contained in the LEDA library, see Burnikel et al. (1996).

5.2.2 Degenerate configurations

Fortune (1989) requires an algorithm to always compute the correct topology, which means that the algorithm must cope with all possible degenerate input data. Note, that exact computation is a prerequisite to detecting and handling degenerate configurations.

However, very often the combinatorics of dealing with all possible degeneracies turns out to be rather complicated. Then the implementation of the algorithm might gain significantly from the simplifications achieved by the assumption of the input being in general position. Hence, a large number of techniques to remove degeneracies from the input have been proposed. On the other hand, Burnikel (1996) shows in his thesis how to implement algorithms for planar Voronoi diagrams and line segment intersections that cope with degenerate configurations.

Symbolic perturbation schemes. A very popular approach to deal with degenerate configurations is to a apply a symbolic perturbation, effectively changing the input coordinates by an infinitesimally small amount ε. All intermediate results are then elements of the field $\mathbf{Q}(\varepsilon)$. Symbolic perturbation schemes were introduced to computational geometry by Edelsbrunner and Mücke (1988), and have been refined and extended by Yap (1990), Emiris and Canny (1995), and Emiris et al. (1997). These techniques require exact evaluation of the geometric predicates. The major objective against using symbolic perturbation schemes is the fact that these algorithms do not compute the topologically correct solution to a specific instance of the geometric problem, but rather the solution of the limit $\varepsilon \to 0$.

[10] Cf. Burnikel et al. (1997)

Numerical perturbation of input data. In the context of computing van der Waals surfaces of biological macromolecules, Halperin and Shelton (1997) proposed to perturb the input data numerically to avoid degenerate configurations. This approach is viable since the geometric coordinates are imprecise estimates obtained from statistical measurement procedures. Hence, a perturbation within the accuracy of the measurements does not harm.

Other approaches. Fortune (1989) calls an algorithm parsimonious, if the algorithm never evaluates a geometric predicates whose value can be deduced logically from previous predicate evaluations and the axioms underlying the domain of the algorithm. A parsimonious algorithm can never reach an inconsistent state, even if the predicate evaluations would be replaced by a random process, since for each branching taken by the algorithm a set-theoretic model m can be created. A model can be identified with a concrete geometric input, such that the branching taken by the algorithm corresponds to the branching that the algorithm would have reached if it had been processing the model m as input. Knuth (1992) presents a parsimonious algorithm for computing planar convex hulls that is derived from the well-known lower bound construction on the number of comparisons needed for sorting.

Sugihara and Iri (1994) provide a conceptually similar approach they call *topologically oriented*, where the model theoretic view is somewhat relaxed. For example, their algorithm for computing planar Voronoi diagrams is guaranteed to produce a planar graph yet the embedding computed by the algorithm may actually be non-planar.

Also quite similar, Schorn (1991) proposes what he calls an *axiomatic approach*, that modifies the problem to solve in such a way that a convenient axiomatic system can be found. For example, instead of computing the closest pair of a set of points, he rather devises an algorithm determining only the smallest distance between any two points from the set.

5.2.3 The implemented strategy

To decide for a specific strategy to deal with numerical precision and degeneracy issues, we first formulated a number of premises:

1. Out software is intended to be run routinely on large data sets. Therefore,

machine-provided arithmetic should be exploited as much as possible.

2. Within a molecule, the spheres are more or less evenly distributed. Taking into account experience gained from earlier implementations, we decided that numerical problems would occur only seldomly.

3. The proper identification and representation of degenerate configurations is of no importance to the intended applications.

4. As already noted by Halperin and Shelton (1997), molecular data is gained from experiments and henceforth inaccurate data. Small changes of the coordinates within the error of the physical measurements preceding the calculations are acceptable.

5. The implementation of the numeric predicates should be kept as simple as possible, since already the combinatorial part posed challenging implementation problems.

From these premises, we decided for the following strategy: In our implementation, all predicates are implemented using the built-in double precision IEEE 754 arithmetic provided by the underlying hardware. However, all computations influencing the branching of the algorithm are performed with a dynamic error analysis according to equations 5.1. Additional correction factors account for the fact that the error bounds themselves are subject to round-off errors. See figure 5.13 on page 121 for an excerpt of the actual implementation. Whenever a branch decision is taken, the actual value is compared against the computed error bound. If the implementation cannot guarantee the correctness of the taken branch, then an exception is thrown by the algorithm. Obviously, the implementation cannot distinguish degenerate configurations from round-off errors.

The exception handler is located in the outermost loop controlling the computation of the individual cells. When an exception occurs, the computation of that specific cell is aborted and all data computed so far for this cell is thrown away. A small numeric perturbation within the precision of the data is applied, and the computation for that specific cell is started again.

5.2.4 Structure and precision of input data

A common format used for the representation molecular structures is the Brookhaven Protein Data Bank (PDB) format. A PDB data record describing a

single atom within a molecule has the following format:

```
ATOM   27  QD  ARG  1    -9.145  -0.560  11.890  0.00  2.69
```

The first number specifies the sequential number of the atom within the molecule, the second number specifies the number of the residue. The next three numbers are the atomic coordinates in Å. Since natural molecules are of limited size, we can assume that the coordinates of all other atoms relevant to the computation of a single cell are contained in the interval $[-128.000\text{Å}, 128.000\text{Å})$ around the center of the atom of interest. Hence, $b_c = 17$ data bits and one sign bit are sufficient to store the input coordinates after translating the center of the atom whose cell is to be computed to the origin. The radii are all bounded by 8.0Å, and they can be represented using $b_r = 13$ bits. In fact, as we will discuss in the next chapter, for practical purposes the region of interest around an atom is typically restricted even more.

As we have shown in chapter 2, for a sphere σ of radius 0 centered at the origin the equation describing the projection of an edge generated by two neighboring spheres $\sigma_i = (c_i, r_i)$ and $\sigma_j = (c_j, r_j)$ within the spherical map is given by the equality

$$\begin{aligned} \langle a_{i,j}, p \rangle &= b_{i,j}, \text{ where} \\ a_{i,j} &= c_i \cdot \left(c_j^2 - r_j^2\right) - c_j \cdot \left(c_i^2 - r_i^2\right) \\ b_{i,j} &= r_i \cdot \left(r_j^2 - c_j^2\right) - r_j \cdot \left(r_i^2 - c_i^2\right) \end{aligned}$$

c_k^2, $k = i, j$, is strictly positive, as is r_k^2. Therefore, we have

$$\begin{aligned} |a_{i,j}| &\leq 2 \cdot \left(2^{b_c} \cdot 2^{2b_c}\right) = 2^{3b_c+1} = 2^{52}, \\ |b_{i,j}| &\leq 2 \cdot \left(2^{2b_c} \cdot 2^{b_r}\right) = 2^{2b_c+b_r+1} = 2^{48}. \end{aligned}$$

Hence, we can store the coordinates of the defining planes of an edge of the spherical map exactly using double precision IEEE 754 floating-point numbers.

All further computations of the geometric objects according to the formulas as given in previous chapters are performed with the dynamic error analysis as shown in Figure 5.13 in the appendix of this chapter.

Figure 5.6 shows the distribution of the relative errors when running the algorithm on the specified test data set. The algorithm shows a very nice behavior with respect to the distribution of the numerical errors. Only about once every 300

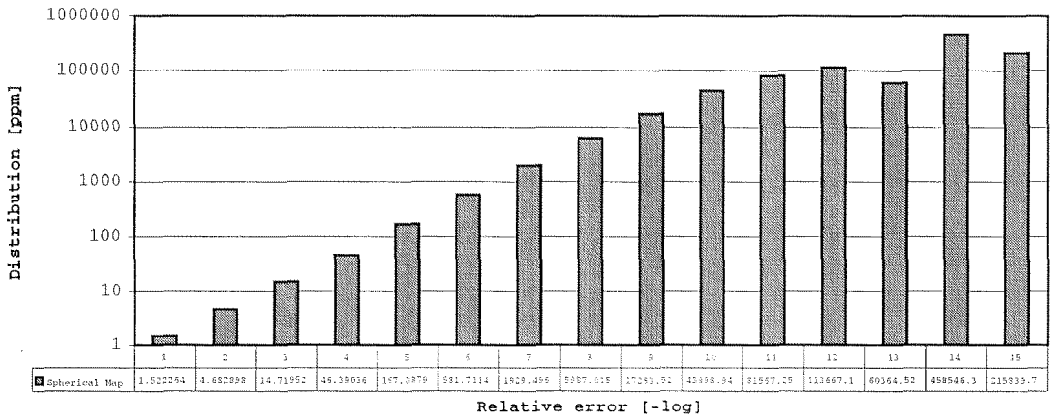


Figure 5.6: Distribution of numerical errors using ordinary IEEE 754 floating-point arithmetic. Both the relative error and the distribution of the error values in sign comparisons are plotted using a logarithmic scale.

cells an exception triggering the numeric perturbation has to thrown. These results demonstrate very clearly the suitability of our approach to computing AWV cells in practice.

5.3 Running times

We compared the running time of our implementation with two other alternatives and collected the results into table 5.4 and figure 5.7:

1. `double` is the algorithm using built in IEEE double precision numbers with significants of 53 bits.

2. `filter` is the algorithm compiled using the code for dynamic error analysis as shown in figure 5.13.

3. `leda_real` is the algorithm compiled with the corresponding data type from the LEDA library[11].

In all cases, the underlying implementation is precisely as described in the previous chapter, and only the arithmetic base type employed by the geometric primitives is changed.

[11] Cf. Burnikel et al. (1996)

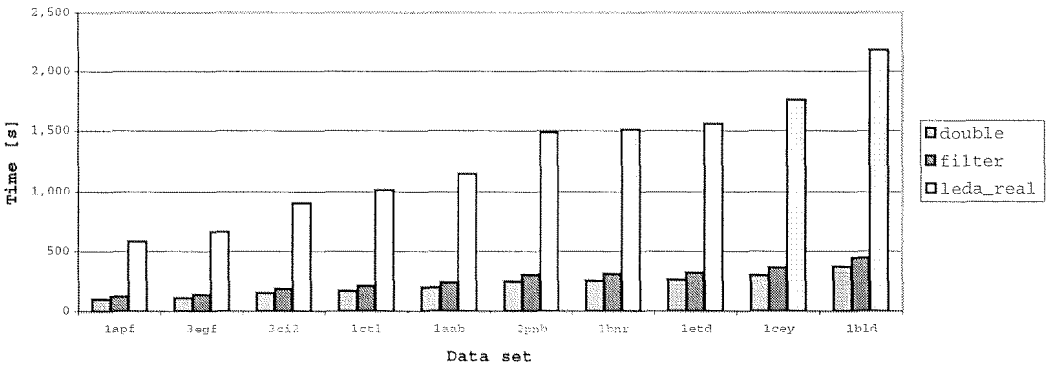


Figure 5.7: Running times for computing additively weighted Voronoi cells using different types of arithmetic.

In addition, we tried to use floating-point filters similar to `leda_floatf`[12], which also perform a dynamic error analysis at runtime, but are much simpler than the code shown in figure 5.13. As it turned out, the bounds computed by these filters were so bad that not a single cell could be computed without the filters signaling a numerical underflow[13]. This mismatch of the error bounds is caused by the high polynomial degree of the predicates to be evaluated.

As we can see, the dynamic error analysis imposes an overhead of about 25% compared to the simple floating-point implementation. The implementation using `leda_reals`, on the other hand, is about six times slower, since the expression dags of our predicates are rather complicated, and they have to be built up and destroyed for each single evaluation.

5.4 Experiences with other approaches

5.4.1 Extracting an explicit representation using Aurenhammer's method

As mentioned earlier on page 14, we could not find a reference to a previous implementation of an algorithm extracting the geometry of an AWV cell directly from the corresponding power cell. Hence, we provided our own implementation

[12]Cf. Mehlhorn and Näher (1994)

[13]We emphasize that `leda_floatf` is different from the dynamic error analysis used internally by `leda_real`. `leda_floatf` is used in the planar line sweep algorithm of LEDA.

		double		filter			leda_real		
PDB	Atoms	Time [s]	T_c [ms]	Time [s]	ratio	T_c [ms]	Time [s]	ratio	T_c [ms]
1apf	709	97.94	138.14	122.77	1.25	173.16	583.33	5.97	824.16
3egf	794	110.42	139.07	135.21	1.22	170.29	663.56	6.01	835.72
3ci2	1061	149.07	140.50	184.06	1.23	173.48	902.60	6.05	850.70
1ctl	1214	166.71	137.32	207.60	1.25	171.00	1014.17	6.08	835.39
1aab	1357	192.52	141.87	232.96	1.21	171.67	1146.29	5.95	844.68
2pnb	1699	244.75	144.06	297.99	1.22	175.39	1486.89	6.08	875.16
1bnr	1727	249.20	144.30	305.60	1.23	176.95	1505.26	6.04	871.62
1etd	1779	257.83	144.93	315.91	1.23	177.57	1563.11	6.06	878.64
1cey	1979	291.91	147.50	355.76	1.22	179.77	1761.13	6.03	889.91
1bld	2443	365.03	149.42	446.53	1.22	182.78	2182.57	5.98	893.40
Total	14762	2125.77	143.98	2604.39	1.23	176.43	12809.89	6.03	876.76

Table 5.4: Running times for computing additively weighted Voronoi cells using different types of arithmetic. The columns labeled "ratio" contain the ratio of the running time compared to the naive implementation using built-in floating-point numbers.

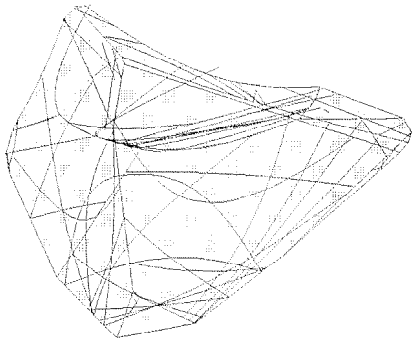


Figure 5.8: Result of direct extraction of an additively weighted Voronoi cell from its corresponding power cell. The additional hyperbolic arcs are artifacts from the elimination process used to solve the equation systems. Moreover, the numerical sensitivity of the elimination process is visible.

of an algorithm along the outline given on page 38.

A typical result of applying this algorithm is shown in figure 5.8. For reference, the same cell computed by the implementation as described in the previous chapter is shown in figure 5.9. Distinctive features are:

- First, we can see many additional arcs within the individual faces of the cell. These arcs are artifacts from the elimination process and inherent to all variants of cylindrical algebraic decompositions. Of course, it is possible to glue suitable patches together in a post-processing step.

- Second, we can see from the glitches and imprecisions that the algorithm suffers from numerical errors when implemented using floating-point arithmetic. We performed a dynamic error analysis to verify this claim. The resulting error distribution is shown in figure 5.10.

- Concerning running time, our implementation of the direct extraction method is definitely not competitive with the algorithm from the previous chapter. To achieve a running time within the theoretical worst-case bound, we have to compute a triangulation of the power cell and then call the extraction algorithm for each simplex. However, due to the numerical problems which already showed up in the simple setting, the more advanced approach does not appear to be viable: We were not able to compute a single cell without significant errors using floating-point arithmetic. This observation is what was to be expected, since the coefficients of the addi-

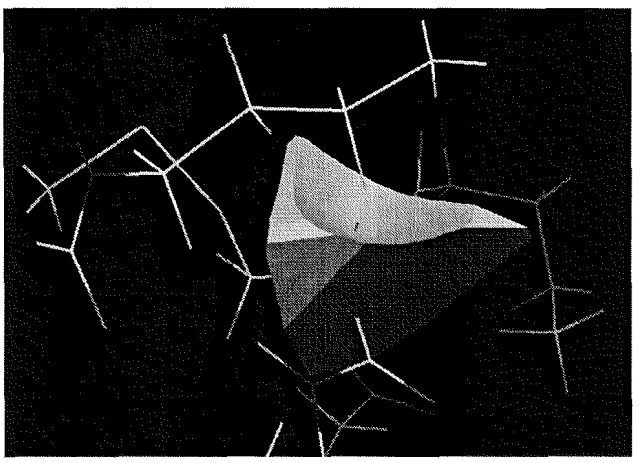


Figure 5.9: The same cell as shown in figure 5.8 computed using the algorithm based on spherical maps described in section 5.3. The cell is shown within the molecular neighborhood defining its shape.

tional planes introduced in the triangulation process are 4×4-determinants in 4×4-determinants of numbers from the input data.

As a preliminary result, the algorithm performing the triangulation needs about 2 minutes for the cell shown in the picture (yet, of course, producing significant errors in the output). So even if the numerical issues could be solved, it is doubtful if the running time could be lowered by a significant factor.

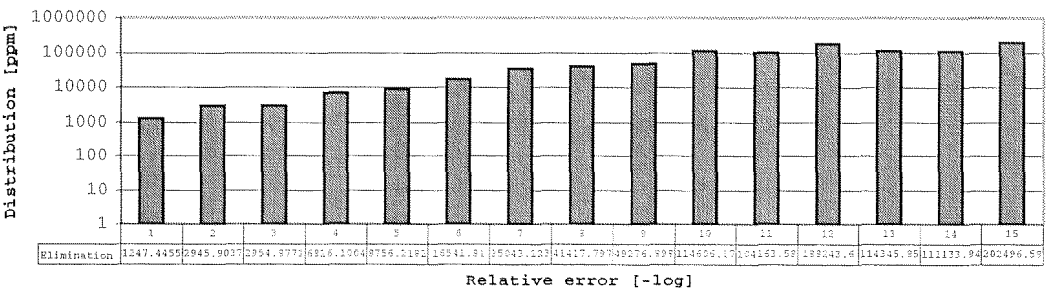


Figure 5.10: Distribution of numerical errors when computing the cell shown in figure 5.8 using floating-point arithmetic. Both the relative error and the distribution of the error values in sign comparisons are plotted using a logarithmic scale.

5.4.2 Simulating a vertical decomposition scheme

To examine the influence of a vertical decomposition scheme on the numerical sensitivity of the algorithm, we compared our previous results with the following variant of the algorithm: Again, we assume the sites $\sigma_1, \ldots, \sigma_n$ to be given in random order:

1. *Initialization:* Construct an initial spherical subdivision P_0 by cutting the unit sphere S^2 with the three coordinate planes $x = 0$, $y = 0$ and $z = 0$.

   Compute initial conflict information between each vertex, edge and face of P_0 and each site σ_i, $1 \leq i \leq n$.

2. *Incremental step:* For each $i = 1 \ldots n$ perform the following operations:

   (a) Create new edges due to edge conflicts generated by σ_i.

   (b) Remove redundant old edges which no longer separate different faces. Rejoin chains of edges into single edges.

   (c) Create new edges:

      i. Process disc conflicts generated by σ_i.

      ii. Process ring conflicts generated by σ_i.

   (d) Update conflict information.

   (e) *Let E_i be the set of all end vertices and vertical tangencies of edges in P_i. Construct a tree T_i containing all elements from E_i sorted by their degree of longitude.*

   (f) *Insert all newly created conflicts into T_i and record the numerical errors during this insertion process.*

Observe, that only the errors when inserting the new conflicts were recorded.

Figure 5.11 shows a plot of the error distribution when running this algorithm on the specified data set. For comparison, the error distribution collected in the experiments previously described is also shown. As we can see, the simulation of the vertical decomposition shows an error distribution very similar to the basic non-refining algorithm. However, there is a significant increase of relatively large errors, which looked to systematic to be overlooked. Of course, our first suspicion was that this is an artifact of our simulation process. We examined manually those input data sets exhibiting these unexpectedly large errors.

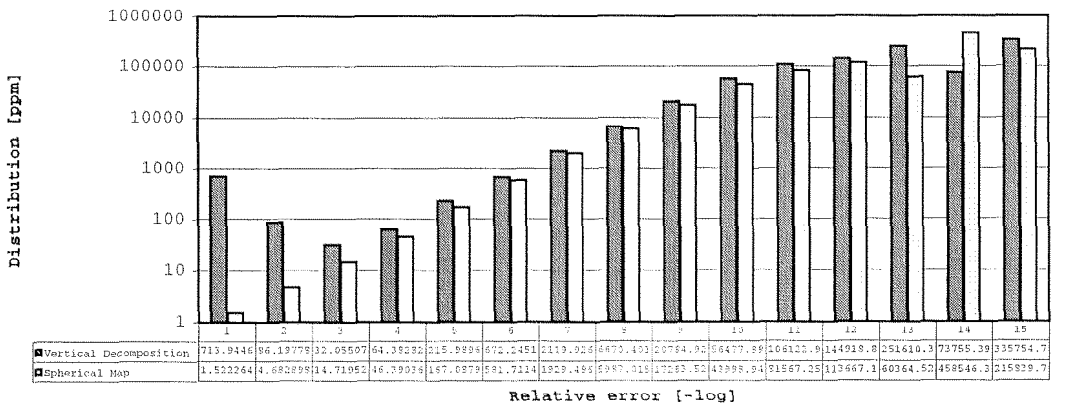


Figure 5.11: Distribution of numerical errors when computing additively weighted Voronoi cells using floating-point arithmetic. Both the errors and the distribution are plotted using a logarithmic scale.

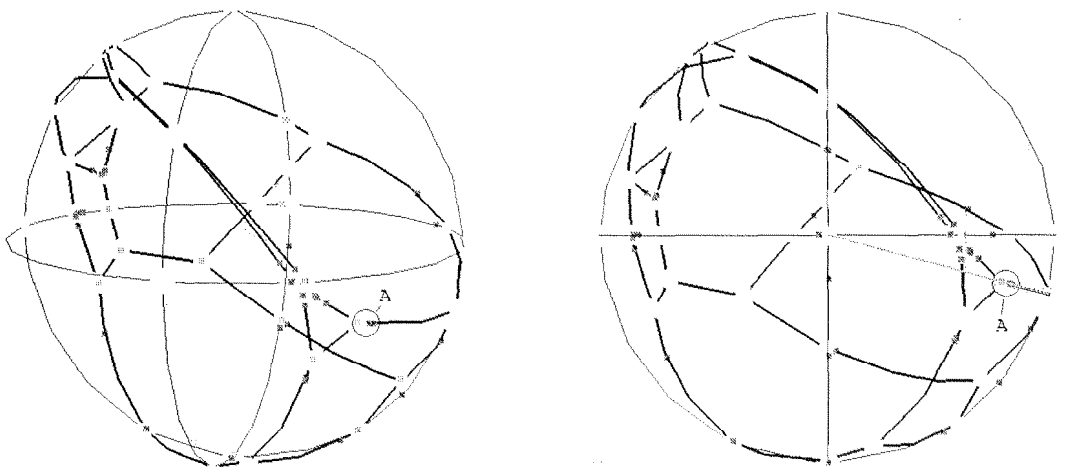


Figure 5.12: A typical constellation causing numerical problems for algorithms using vertical decomposition. The left picture shows a perspective view of the spherical map. The x-axis extends to the right, the y-axis to the top, and the z-axis towards the reader. The right picture shows the projection of the configuration on the x,y-plane. Conflicts are marked as gray dots. The problematic area is labeled A: Three conflicts are located on an edge which is part of a circle defined by a plane *almost* parallel to the sweep plane, whose projection is drawn in light gray.

As it turned out, the errors are systematic. Yet, they are not artifacts of our simulation but are inherent to vertical decompositions: Figure 5.12 shows a typical example. The numerical problems are caused by edges, which are *almost* fragments of a great circles through the z-pole. Determining the circular order of any conflicts located on these edges around the z-axis results in large errors.

The question is of course, why do these errors occur that often? An edge of a AWV cell that is defined by three spheres of equal radius is a line segment as in the unweighted case. Hence, its projection onto the parameter space is a great circle. The problematic configurations occur if the three spheres are located on a plane that is *almost* parallel to the x,y-plane. In fact, all the configurations we examined were caused by carbon atoms which occurred in chains and rings of the molecules.

We conclude that vertical decomposition schemes exhibit inherent problems concerning numerical stability.

5.5 Conclusions

In this chapter we examined the practicality of our approach to compute AWV cells. We saw that the data sets arising in molecular biology have specific characteristics that lead to distinct distributions of the combinatorial complexities of the cells. Moreover, the more or less even distribution of the atoms within the molecule leads to a favorable numerical behavior of our implemented algorithm.

We saw that within our domain of application, the ratio between the complexity of an AWV cell compared to that of its corresponding 4-dimensional power cell tends to depend on the radius of the atom; smaller atoms tend to have a significantly less complex AWV cell.

However, we feel that the main lesson learned from our experiments is the influence of the combinatorial design on the numerical behavior of an algorithm. We believe that this influence cannot be underestimated. Especially any kind of projection along coordinate axes seems to be a dangerous operation from the numerical point of view.

Finally, the overhead we introduced to deliver a robust implementation is not too high compared to a naive implementation using simple built-in floating-point arithmetic only.

Appendix

Equipment used

All experiments where performed on a PC with a 266MHz Pentium II Processor with 128MB of main memory. All programs where compiled using Microsoft Visual C++ 5.0SP3 using maximum optimization. Since our algorithms are embedded within a graphical user interface, the algorithms where linked to the multithreaded DLL versions of the runtime environment. We used the SGI STL version 2.0 and the STL by DinkumWare provided with MSVC. The LEDA release used was 3.7R.

Selection of test data sets

The data sets taken from the PDB are not random samples but where selected according to the following criterion: It is important to know that there are basically two different methods to measure atom coordinates, X-ray crystallography and NMR spectroscopy. The first method, however, cannot determine the locations of hydrogen atoms. Using AWV cells instead of unweighted cells only makes sense if the atomic radii vary over a sufficiently large range. There are basically two ways how these larger ranges occur: First, larger radii may be assigned according to quantum chemical considerations, and these radii are then typically assigned to groups of atoms, such as a methyl group. Second, the inclusion of hydrogen atoms, which are rather small, leads to the typical effects distinguishing AWV diagrams from unweighted ones, such as closed elliptic edges and disconnected faces separating a cell from the same neighboring atom. Since there is no commonly agreed set of rules for the first case, we chose all samples from NMR data. Whenever a data set contained more than one coordinate set, we used the first model defined.

Dynamic error analysis

The error analysis is performed using a modified version of the code as given in Fig. 12 of the paper by Fortune and Van Wyk (1996). The changes are the following: The code for multiplication provided in the paper lacks an error term, which we added. Second, we do not store the absolute values of the floating-point

PDB	Atoms	Description
1apf	709	ANTHOPLEURIN-B, NMR
3egf	794	EPIDERMAL GROWTH FACTOR
3ci2	1061	CHYMOTRYPSIN INHIBITOR 2
1ctl	1214	AVIAN CYSTEINE RICH PROTEIN
1aab	1357	HMGA DNA-BINDING HMG-BOX DOMAIN A OF RAT HMG1
2pnb	1699	PHOSPHATIDYLINOSITOL 3-KINASE
1bnr	1727	BARNASE
1etd	1779	MURINE ETS-1 TRANSCRIPTION FACTOR
1cey	1979	CHEY COMPLEXED WITH MAGNESIUM
1bld	2443	BASIC FIBROBLAST GROWTH FACTOR (FGF-2) MUTANT

Table 5.5: PDB entries used in the experiments.

Element	H	C	N	O	P	S	Fe
Radius [Å]	0.7	1.38	1.38	1.27	1.61	1.55	1.0

Table 5.6: Atomic radii used in the experiments.

numbers explicitly, but use the `fabs`-function to calculate the information on demand. On our machine, this bit-flipping operation is much faster than loading a value from memory into the FPU. Finally, the error analysis itself is done using floating-point arithmetic and is subject to round-off errors. We take this fact into account using appropriate correction factors. Figure 5.13 gives an excerpt of our code.

```
CT_FloatFilter operator+(const CT_FloatFilter& rcco_other) const
  {
  double r8_result = r8_Value + rcco_other.r8_Value;
  double r8_correction =
    1.0 + 2.0 * std::numeric_limits<double>::epsilon();
  return CT_FloatFilter (r8_result, (r8_Error + rcco_other.r8_Error +
    fabs(r8_result) * std::numeric_limits<double>::round_error()) *
    r8_correction);
  }

CT_FloatFilter operator*(const CT_FloatFilter& rcco_other) const
  {
  double r8_result = r8_Value * rcco_other.r8_Value;
  double r8_correction =
    1.0 + 4.0 * std::numeric_limits<double>::epsilon();
  return CT_FloatFilter (r8_result,
    (r8_Error * rcco_other.r8_Error +
      r8_Error * fabs(rcco_other.r8_Value) +
      rcco_other.r8_Error * fabs(r8_Value) +
      fabs(r8_result) * std::numeric_limits<double>::round_error())
    * r8_correction);
  }

CT_FloatFilter co_Sqrt() const
 {
  double r8_value = sqrt(r8_Value);
  double r8_correction =
    1.0 + 2.0 * std::numeric_limits<double>::epsilon();
  double r8_error = sqrt(r8_Error);
  return
    CT_FloatFilter (r8_value, (r8_value + r8_error) * r8_correction);
  }

int i_Sign() const
  {
  double r8_eps = r8_Error * std::numeric_limits<double>::epsilon();
  if (r8_Value > r8_eps)       return 1;
  else if (r8_Value < -r8_eps) return -1;
  throw CT_NumberUnderflow ();
  }
```

Figure 5.13: An excerpt of the code used to accumulate error bounds at runtime. This is the version used in the "production" version of our algorithm, which raises an exception to perturb the input in case of a degeneracy.

Chapter 6

Standard volumes of amino acids and their constituent atoms

In this chapter, we want to use our new algorithm to compute AWV cells of the individual atoms of proteins. We will derive a set of standard volumes for the different amino acid residues. Additionally, we will compare the volumes computed using the AWV with previous approaches to assign volumes to individual atoms from crystal structures. All these previous approaches — Voronoi, Richards' B and radical planes — use planar bisector surfaces to separate the individual cells.

The structure of this chapter is as follows: In the first section, we will present a short review of the experiences made with the previous approaches. These experiences also guided the setting for our own experiments. In the following two sections, we will discuss the results obtained from our experiments. First, we will discuss the results for complete amino acid residues, then we will discuss the volumes computed for the individual atomic types. In the next section, we will report some preliminary results concerning the packing density of proteins. Finally, in the last section, we provide all information necessary to reproduce our experiments.

6.1 Introduction

6.1.1 Previous approaches

Previous studies on the volume of individual atoms in molecular structures have shown that the individual volumes depend greatly on the method chosen for the calculation. Comparing his "B" method with the Voronoi method, Richards (1974) reports that

> "[...] the mean volume for an atom with a small van der Waals radius (i.e. a carbonyl oxygen) tends to decrease, while those with large radii (i.e. $-CH_3$) tend to increase. [...]"

This obervation was also confirmed in the study by Gellatly and Finney (1982). Gerstein et al. (1995) summarize these observations as follows:

> "[...] Bisection systematically misallocates volume inside of a protein, producing larger variance in the volume for any particular atom type. [...]"

The latter authors propose two different approaches to obtain a better partitioning scheme:

1. A hybrid approach that uses Richards' B method to position the plane between protein atoms and to ignore the radii in all other cases.
2. An approach based on spherical bisector surfaces that bend around the smaller atom.

However, in later studies the authors did not use these partitioning schemes again, see Gerstein and Chothia (1996) and Tsai et al. (1999).

Concerning the radical plane method, Goede et al. (1997) remark that the

> "[...] principal advantage of the radical plane method separation scheme is the passage of the dividing plane through the intersection circle of both atoms. [...]"

This property is of special importance when the local density of an atom is calculated as the van der Waals volume contained inside its cell divided by the cell

volume. The Voronoi and Richards' B method do not have this property, and hence part of the van der Waals volume of a large atom next to a small atom is neither allocated to the small nor to the large atom. Goede et al. computed a total loss of van der Waals volume of 8% and 6%, respectively, for the two methods. On the other hand, for bonded atoms (at a distance of about 1.4Å), even the center of an atom is not part of the volume assigned to the atom itself by the radical plane method if the difference of the radii is at least 0.6Å. Therefore, Goede et al. propose to use AWV cells for volume and density calculations, because this method

> "[...] unifies the advantages of earlier approaches by
>
> 1. keeping the spirit of the geometrically rational partitioning implied by the Richards' method,
> 2. avoiding vertex error and meeting the intersection circle between atoms like the radical plane method, and
> 3. using non-planar boundaries like Gerstein et al. [...]"

Yet, as mentioned in the introduction to this thesis, they could not devise an efficient and practical algorithm for the computation of AWV cells.

6.1.2 Statistical parameters

As we can see from the previous citations, the variance of the computed volumes has been used as a quality measure of the different methods. Let $X = X_1, \ldots, X_n$, $X_i \in \mathbf{R}$ for all $1 \leq i \leq n$, be a sequence of observations. We write the *mean value* as

$$\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i.$$

The *standard deviation* can be calculated as

$$\sigma(X) = \sqrt{\frac{n \sum_{i=1}^{n} X_i - \sum_{i=1}^{n} X_i^2}{n^2}}.$$

Since we are interested in the relative errors of our calculations, we use the *variational coefficient* $\frac{\sigma(X)}{\bar{X}}$ as index value of our results. As it is common practice, we multiply this number by a factor of 100 and denote this *percentage deviation* by $\frac{\sigma(X)}{\bar{X}}\%$.

Usually, an estimation method is considered superior if the variational coefficient is lower, which is regarded as an indicator to what extend the distribution is concentrated around the mean value. The percentage deviation has been considered as quality measure for volume computations by most previous authors[1]. Pontius (1997), on the other hand, tries to relate variations of the atomic volumes to physical properties. She argues that the lower deviation observed for Richards' B might result in part from the "vertex error" this method exhibits, small tetrahedral volumes that are not assigned to any of the atoms.

6.1.3 Boundary conditions

A consequence of modeling atoms as cells of a tessellation is that surface atoms, being incompletely surrounded by other atoms, have the possibility of being very large or even unbounded. Published studies vary widely in how atoms near the surface were treated. The easiest approach is to exclude all those atoms from the calculation that have a significant surface area accessible to the solvent. This was the approach chosen in the early studies by Chothia (1975). Later authors such as Harpaz et al. (1994), Pontius et al. (1996) and Pontius (1997) went even further by excluding all atoms that have any solvent accessible surface area.

Another approach is soak the crystal structure into a hypothetical solvent. Richards (1974) assigned solvent atoms to positions on a cubic lattice surrounding the protein. Finney (1975) placed solvent molecules at all surface sites that could accommodate a hypothetical "solvent" molecule of radius 1.7Å. This was done irrespective of possible overlap between the individual solvent molecules placed. Gerstein et al. (1995) used molecular dynamics simulations to position water molecules.

In our studies, we considered only completely buried atoms, i.e. atoms with no accessible surface. However, we think that modeling volume distributions near the molecular surface using the AWV method is an interesting area of future research.

6.2 Volumes of amino acid residues

For each buried atom of the chosen data set we computed the volume of the cell associated with the atom using the four different methods: AWV, Richards' B,

[1] Cf. Richards (1974), Gellatly and Finney (1982), Gerstein et al. (1995)

Residue		AWV		Richards' B		Voronoi		Radical	
Type	#	$\bar{V}[\text{Å}^3]$	$\sigma/\bar{V}\%$	$\bar{V}[\text{Å}^3]$	$\sigma/\bar{V}\%$	$\bar{V}[\text{Å}^3]$	$\sigma/\bar{V}\%$	$\bar{V}[\text{Å}^3]$	$\sigma/\bar{V}\%$
ALA	300	91.4	**4.81**	90.9	*4.85*	91.7	5.43	91.4	4.91
ARG	8	203.8	**3.70**	201.9	3.81	197.1	3.96	202.6	*3.81*
ASN	22	132.3	5.58	132.1	**5.54**	138.7	*5.57*	133.7	5.61
ASP	25	122.9	*4.73*	122.9	4.75	132.7	**4.60**	124.7	4.86
CYS	66	108.6	**7.38**	108.0	*7.53*	108.1	8.25	108.6	7.54
GLN	8	152.8	**4.92**	152.6	*5.03*	159.1	5.82	154.2	5.19
GLU	7	149.7	2.76	149.6	*2.72*	159.8	2.83	151.6	**2.71**
GLY	220	65.5	5.39	65.0	**5.26**	67.2	5.81	65.8	*5.38*
HIS	19	163.0	**3.63**	161.9	*3.75*	163.2	4.49	163.0	3.78
ILE	221	166.8	**3.35**	166.0	3.39	163.1	3.73	166.0	*3.37*
LEU	222	166.9	**3.73**	166.2	*3.78*	164.1	4.32	166.3	3.82
LYS	4	173.0	1.23	170.6	*1.20*	162.5	1.94	170.9	**1.15**
MET	55	169.3	**4.34**	168.4	*4.45*	166.5	5.43	168.7	4.49
PHE	82	196.5	**3.78**	196.3	*3.85*	199.4	4.35	197.1	3.89
PRO	24	125.8	**4.61**	124.6	*4.70*	122.4	5.55	125.1	4.71
SER	106	97.8	**5.05**	97.1	*5.15*	101.5	6.07	98.4	5.30
THR	56	122.8	**4.04**	122.2	*4.08*	125.6	4.82	123.2	4.20
TRP	24	232.5	*2.93*	232.1	**2.91**	236.2	3.50	233.1	2.98
TYR	26	202.0	**4.13**	201.9	*4.27*	208.7	5.09	203.4	4.36
VAL	293	141.4	**3.32**	140.6	*3.34*	138.6	3.73	140.7	3.35

Table 6.1: The mean volumes $\bar{V}$ of the different amino acid residue types as calculated with different methods. For each residue, the second column labeled # specifies the number of buried occurrences in the data set. The methods employed are AWV, Richards' B, unweighted Voronoi, and radical planes. The values given are the mean volume $\bar{V}$ and the percent deviation $\frac{\sigma(X)}{\bar{X}}\%$. For each row, the lowest value of the percentage deviation is typeset using a bold font, and the second lowest value is typeset using an italic font.

Voronoi, and radical plane. We recorded the mean value and the percentage deviation of these volumes for each atomic position within each residue type. Tables 6.3 to 6.6 list the values computed. For each residue type we computed the mean and percentage deviation of the volume based on all buried occurrences of the residue type in the data set. A residue is buried if all its constituent atoms are so, and the volume of the residue is computed as the sum of the volumes of the constituent atoms. In table 6.1, we give the average volumes we computed for the individual amino acid residues using the different methods.

The overall differences of the mean residue volumes as computed with the different methods are not too big. Almost all standard deviations are less than 6.0% with

the exception of Cys, where we did not differentiate between thiol form and occurrences in disulfide bonds. In addition, the Voronoi volume for Ser has a standard deviation of 6.07%. Since the data set contained only four buried occurrences of Lys, the computed values cannot be considered statistically relevant. Indeed, for all four methods one of the four computed volumes deviates significantly from the other three volumes.

The most significant differences are between the unweighted Voronoi method and the other three weighted methods. Specifically, for Arg, the mean volume computed with the Voronoi method is approximately 5Å^3 lower than those computed with the other three methods. This is due to the fact that the Voronoi method assigns relatively smaller volumes to large atoms types, such as carbon, and relatively larger volumes to small atom types, such as oxygen and nitrogen. In the case of Arg, C_β, C_γ, and C_δ receive significant less volume by the Voronoi method. A similar effect can be observed for the aliphatic residues Leu and Iso, Lys, and slightly less visible for Met and Val.

For Asn, on the other hand, the Voronoi volume is approximately 6Å^3 larger than that computed using the other three methods. This is mainly due to the much larger volumes associated with the small polar atoms $N_{\delta 2}$ and especially $O_{\delta 1}$. Asp, Gln, and Glu show a quite similar behavior. The larger volume assigned to Thr can also be related to this effect. Finally, the Voronoi methods assigns higher volumes to aromatic ring systems, such as Phe, Trp, and Tyr.

The residue volumes computed with the AWV method appear to be rather similar to those computed using radical planes, and both are quite similar to the volumes computed using Richards' B method. In general, the volume computed by the AWV method is slightly higher than the corresponding volume computed with Richards' B method. This can be attributed to the fact that AWV calculations do not suffer from the effect of vertex errors.

Regarding the deviations of the computed volumes, the AWV computations clearly stand out. In 14 out of 20 residue types, the AWV volumes exhibit the lowest percentage deviation, and for two further types, Asp and Trp, this value is second lowest among the four methods. In the remaining cases, when AWV has only the third lowest deviation, there is a significant gap to the fourth method while the gap to the second best methods is rather small. For Asn, we have 5.57 : 5.58 : 5.61 for Voronoi:AWV:Radical; for Glu 2.72 : 2.76 : 2.83 for Radical:AWV:Voronoi; for Gly 5.38 : 5.39 : 5.81 for Radical:AWV:Voronoi; and finally, even for Lys, we have 1.20 : 1.23 : 1.94 for Richards' B:AWV:Voronoi. On the other hand, ignoring

the values for AWV, Richards' B method shows the best behavior with respect to deviations among all the three methods using planar bisector surfaces only.

The rather large deviations of the AWV volume for Glu stem from the large deviations computed for the atoms of the terminal carboxy-group, compare the data in table 6.4. One reason might be the somewhat problematic assignment of the atomic types. The older rule set by Gerstein et al. (1995) had a specific atomic type O1O2H for carboxy groups, which the authors dropped in the more recent study by Tsai et al. (1999). In the present calculations, the carboxy atoms are assigned the same type as the backbone carbonyl atom. On the other hand, as we will discuss below in more detail, the electronic distribution for hydrogen bonded atoms is directed towards the other atoms taking part in the hydrogen bond. Hence, the distributions for polar atoms might be significantly better if the hydrogen atoms are included in the data set. The data set used was obtained from X-ray diffraction patterns, and hence does not include any coordinates for hydrogen atoms.

The rather high deviation of the AWV volume as computed for Asn is rather surprising as all the volume computations performed for the individual atoms of this residue show rather favorable low deviations, see table 6.3. One reason might be a rather stronger correlation between the individual atomic volumes. Or, the selected residues and their environments are not completely representative; stripping off all water molecules might have distorted those volumes[2].

6.2.1 Comparison of computed residue volumes with previous studies.

Table 6.2 shows a comparison of the residue volumes we computed using the AWV method with previously published values. The studies by Chothia (1975), Harpaz et al. (1994), and Tsai et al. (1999) were based on Richards' B method. The older studies by Chothia and Harpaz et al. used slightly different radius sets. Pontius et al. (1996) used unweighted Voronoi cells and the same data set as our study. However, we were more restrictive in selecting the atoms to include in the statistics.

We observe that the AWV volumes computed for hydrophobic residues are surprisingly similar to the early results published by Chothia (1975), while Chothia's

[2]Cf. Pontius (1997)

Residue	Residue Volume [Å^3]				
	this study	Tsai et al.	Pontius et al.	Harpaz et al.	Chothia
ALA	91.4	90.0	91.5	90.1	91.5
ARG	203.8	194.0	196.1	192.8	—
ASN	132.3	124.7	138.3	127.5	135.2
ASP	122.9	117.3	135.2	117.1	124.5
CYH	108.6$^+$	113.7	114.4	113.2	117.7
CYS	108.6$^+$	103.3	102.4	103.5	105.6
GLN	152.8	149.4	156.4	149.4	161.1
GLU	149.7	142.2	154.6	140.8	155.1
GLY	65.5	64.9	67.5	63.8	66.4
HIS	163.0	160.0	163.2	159.3	167.3
ILE	166.8	163.3	162.6	164.9	168.8
LEU	166.9	164.0	163.4	164.6	167.9
LYS	173.0	167.3	162.5	170.0	171.3
MET	169.3	167.0	165.9	167.7	170.8
PHE	196.5	191.9	198.8	193.5	203.4
PRO	125.8	122.9	123.4	123.1	129.3
SER	97.8	95.4	102.0	94.2	99.1
THR	122.8	121.5	126.0	120.0	122.1
TRP	232.5	228.2	237.2	231.7	237.6
TYR	202.0	197.0	209.8	197.1	203.6
VAL	141.4	139.0	138.4	139.1	141.7

Table 6.2: Comparison of volumes of amino acid residues as given by different authors. The studies are Tsai et al. (1999), Pontius et al. (1996), Harpaz et al. (1994), and Chothia (1975). ($^+$) In the present study, we do not differentiate between disulfide-bonded and non-bonded cysteine.

volumes for polar residues are higher. Hydrophobic residues tend to aggregate in the interior of the molecule, and hence a statistically meaningful sample was already present at that time. Polar residues, on the other hand, are much more likely to be located near the molecular surface. At the time of Chothia's study, only very few were contained in the interior of the available protein crystal structures. Therefore, he had to include even residues with a non-zero yet small solvent accessible surface. Even our data set contains only very few acidic and basic residues that are completely buried in the interior of the molecules.

Comparing the volumes we computed for the Voronoi method with the values published by Pontius et al. yields another interesting observation: Both calculations were performed on the same data set using the same method of allocating space to the individual atoms. The only difference between the two calculations is that we stripped off all non-protein atoms from the data sets before starting the calculation. However, we get the same volumes when exactly the same residues were considered, such as for Lys, so this difference is not due to an error in either calculation. We may conclude that volume calculations of this type are extremely sensitive to the selection of the data set and to the boundary conditions such as the inclusion/exclusion of water and cofactors[3].

Considering the rather small differences between the volumes we computed with the AWV and the Richards' B method on our data set, it is reasonable to assume that the differences of our volumes to the volumes computed by Tsai et al. stem mainly from the different data sets used.

6.3 Volumes for individual atom types

In this section, we discuss the volume distributions obtained for the individual atom types in more detail.

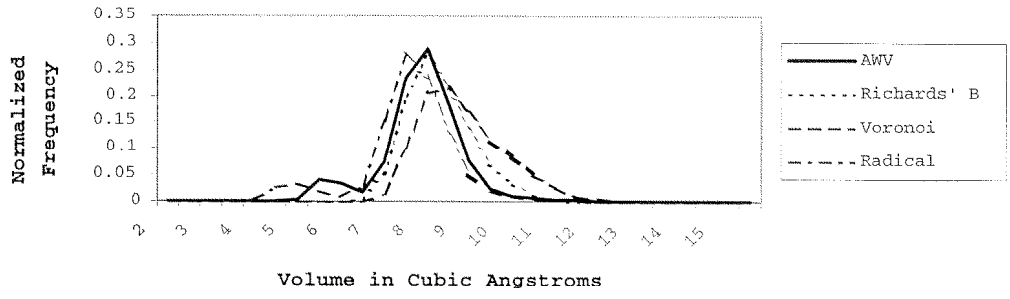
6.3.1 Carbon atoms

Figures 6.1 and 6.2 show the distributions of the atomic volumes computed for different types of carbon atoms.

C3 is the atom type assigned to planar configurations of carbon as found in the carbonyl group along the backbone and as branching point in ring systems such

[3]Cf. also Gerstein et al. (1995), Pontius (1997)

C3



C3H

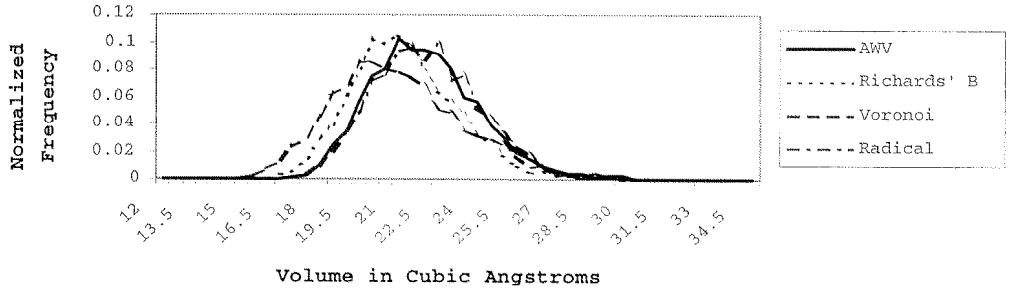


Figure 6.1: Distribution of cell volumes of carbon atoms with planar orbital configurations as found in aromatic and carbonyl groups.

as found in Trp, Tyr, Phe, and His. As we can see from the topmost picture in figure 6.1 and the detailed volume data in tables 6.3 to 6.6, both the radical plane and the AWV method distinguish C3 atoms along the backbone from those in ring systems. The smaller volume assigned to ring atoms shows up as "shoulder" to the left of the main peak. Richards' B and Voronoi, on the other hand, do not show this distinction.

C3H is the type assigned to carbons in aromatic rings. Especially for AWV, the distribution is almost perfectly Gaussian as can be seen in the middle picture in figure 6.1.

C4H is an sp^3-carbon bonded to a hydrogen, most notably the C_α carbons along the backbone of the protein, but also the branching points in aliphatic side chains, such as Val, Leu, and Iso. Comparing the top picture of figure 6.2 with the detailed data given in tables 6.3 to 6.6, we see that both the AWV and the radical plane method assign significantly lower volumes to these atoms in side chains than along the backbone. This might be an indicator for a less tight packing of the backbone than previously assumed. Richards' B shows almost no differences for the two occurrence modes.

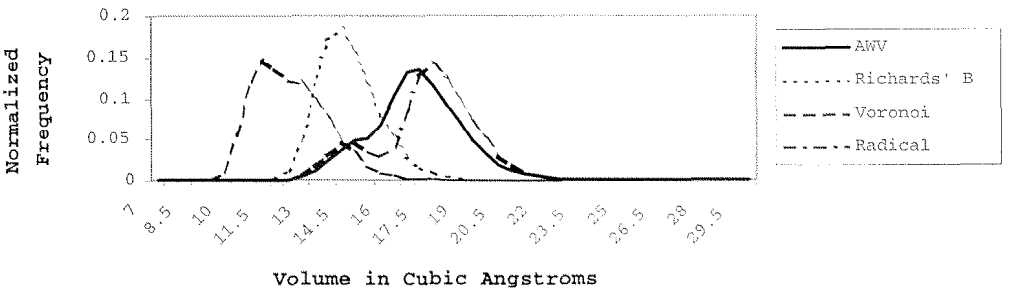
C4HH is the atomic type assigned to $-CH_2-$ groups in side chains. Both radical planes and AWV assign significantly different volumes depending on the radius of the neighboring atoms along the chain. Hence, C4HH atoms next to polar groups, such as C_ε of Met, C_β of Ser, C_β of Asp or C_γ of Glu, show larger volumes than those in aliphatic chains. The overlay of these individual distributions leads to the long tail of the joint volume distribution depicted in the middle picture of figure 6.2 for C4HH towards higher volumes. For the Richards' B and the Voronoi method, this dependency is much less distinctive.

Methyl carbons, which are assigned type C4HHH, exhibit an almost Gaussian volume distribution for all of the four methods. However, the distributions obtained for Richards' B and AWV look smoother than those obtained for the radical planes and the Voronoi method.

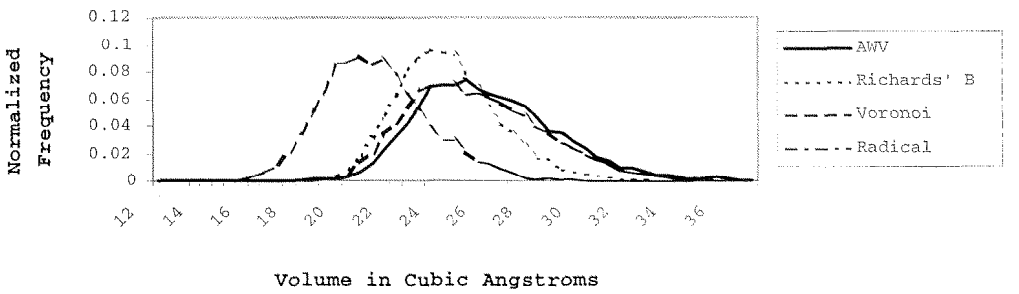
6.3.2 Nitrogen atoms

Figure 6.3 shows the volume distributions for the two types of nitrogen atoms. N3 is only assigned to $N_{\delta 1}$ of His. All methods show similar rough volume distributions. Following the argumentation by Pontius (1997), this behavior might result from the fact that we stripped off all non-protein atoms from the data set. Moreover, she observed variances of the cell volumes depending on the number

C4H



C4HH



C4HHH

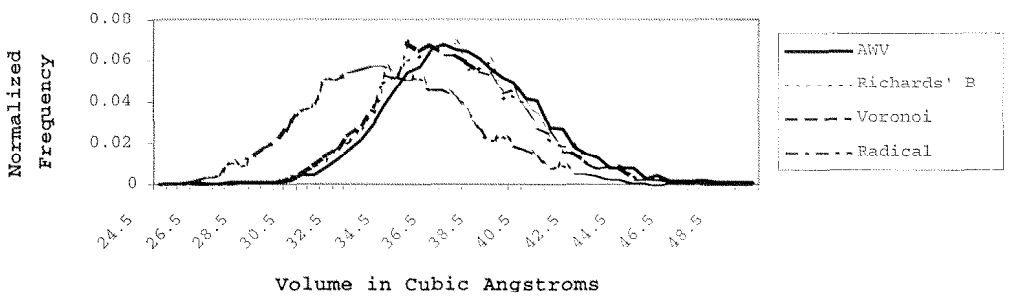


Figure 6.2: Distribution of cell volumes of tetrahedral carbons.

of hydrogen bonds formed in the structure.

For type N3H, all methods show an overlay of two distributions. Comparing with tables 6.3 to 6.6, we cannot separate this distribution into two distributions based on a distinction by residue type and atomic position within the residue alone. Again, the number of hydrogen bonds formed in the structure might be an explanation. Both Voronoi and Richards' B method assign relatively larger volumes to this atom type.

The atomic type N3HH is assigned to the basic groups of Asn, Gln, and to the terminal nitrogens of Arg. Similar to N3, all methods show similar rough volume distributions. Again, this behavior might result from the fact that we stripped off all non-protein atoms from the data set.

6.3.3 Oxygen atoms

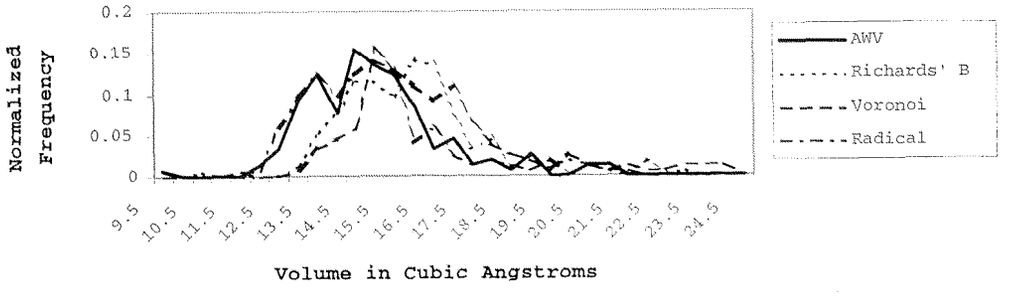
The atomic type O1 is assigned to backbone oxygens, and to atoms located in acidic and basic side chains. The volume distributions for this type as computed with the four different methods are shown in the top picture of figure 6.4. All methods yield similar smooth distributions. However, the volumes assigned by the Voronoi method are significantly larger than those derived using the other three methods. The very few cells of relatively high volumes (more than 37 $Å^3$) might result from missing water molecules.

O2H is the type assigned to the terminal hydroxyl groups of Tyr, Ser, and Thr. Again, we observe the rough distributions resulting from hydrogen bonds as was the case with N3 and N3HH.

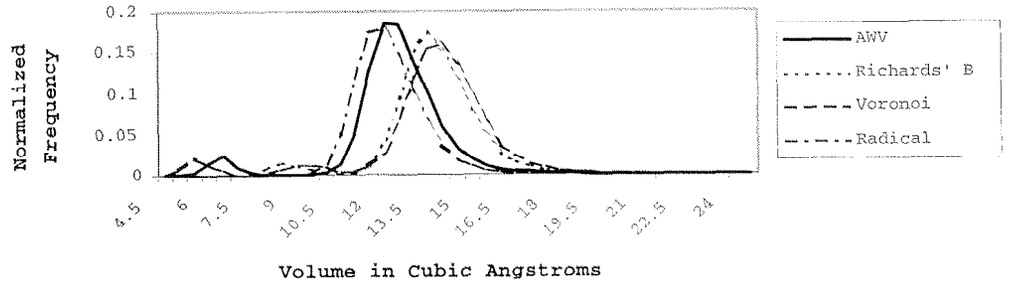
6.3.4 Sulfur atoms

Atomic type S2 is found both in Met and in the terminal position of Cys. Although we did not distinguish between the disulfide-bonded and the thiol form of Cys, none of the methods yields a volume distribution that can be clearly seperated into two smooth overlayed distributions. Comparing with tables 6.3 to 6.6, the peak at 25.5$Å^3$ can be assigned to S_δ of Met, the peak at 27.0$Å^3$ would correspond to disulfide bonds, and the tail to the right would correspond to the behavior as exhibited by the hydrogen bonds previously discussed.

N3



N3H



N3HH

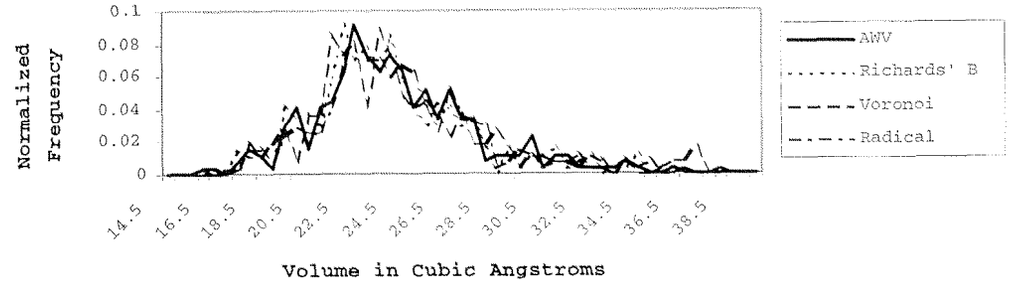
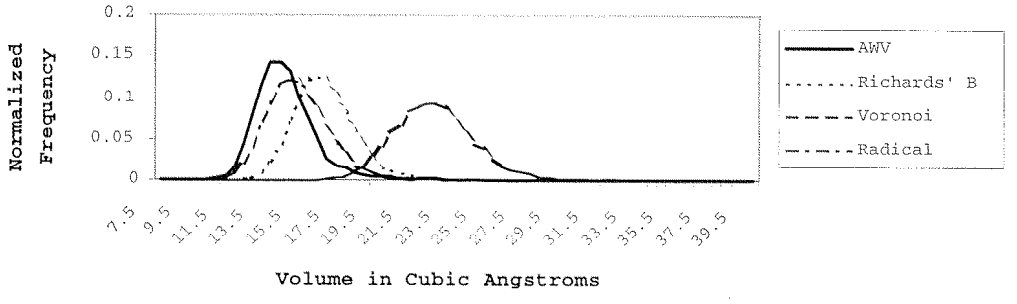


Figure 6.3: Distribution of cell volumes of nitrogen atoms as computed with the different methods.

O1



O2H

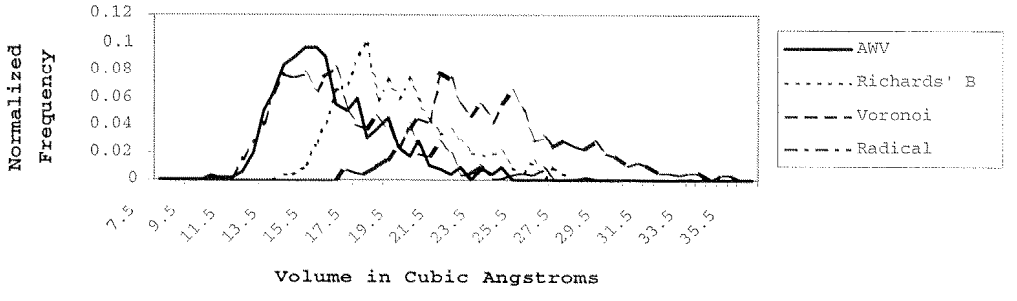


Figure 6.4: Distribution of cell volumes of oxygen atoms as computed with the different methods.

S2

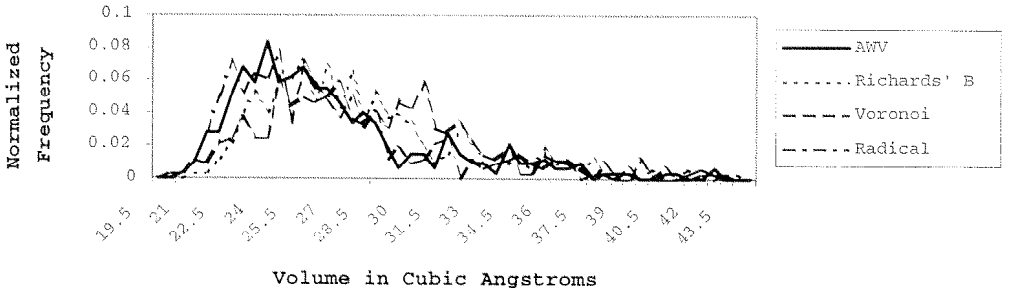


Figure 6.5: Distribution of cell volumes of sulfur atoms as computed with the different methods.

Atom			AWV		Richards' B		Voronoi		Radical	
Res.	Atm.	#	$\bar{V}$[Å^3]	$\sigma/\bar{V}$%	$\bar{V}$[Å^3]	$\sigma/\bar{V}$%	$\bar{V}$[Å^3]	$\sigma/\bar{V}$%	$\bar{V}$[Å^3]	$\sigma/\bar{V}$%
ALA	C	1108	8.1	**6.01**	8.2	6.87	8.6	7.84	7.9	*6.58*
ALA	CA	792	18.2	*7.78*	14.9	8.36	12.4	10.56	18.8	**7.04**
ALA	CB	431	38.0	*7.94*	37.1	8.00	33.0	9.58	37.1	**7.92**
ALA	N	892	12.7	**9.44**	14.1	*9.57*	14.5	10.45	12.3	10.21
ALA	O	670	14.4	12.17	16.7	*11.61*	22.8	**10.96**	15.3	13.43
ARG	C	473	7.9	**6.74**	8.0	7.37	8.5	8.18	7.7	*7.25*
ARG	CA	325	17.4	*8.02*	14.1	8.41	11.7	10.17	17.9	**7.10**
ARG	CB	178	24.2	8.30	23.4	*8.17*	20.3	8.82	23.5	**7.84**
ARG	CD	119	25.8	*10.94*	23.3	11.31	20.3	12.97	25.6	**10.34**
ARG	CG	178	24.5	*9.98*	23.8	10.12	20.9	11.67	23.9	**9.76**
ARG	CZ	211	8.3	**8.08**	9.0	*8.50*	9.6	8.57	8.3	8.80
ARG	N	415	12.4	**8.49**	13.7	*8.56*	14.1	9.43	12.0	9.15
ARG	NE	131	14.0	*10.42*	15.7	**9.95**	16.3	10.44	13.7	11.10
ARG	NH1	76	23.1	**11.16**	22.9	11.41	23.1	12.51	23.2	*11.40*
ARG	NH2	53	24.5	**10.16**	24.2	10.53	23.9	12.32	24.6	*10.41*
ARG	O	289	14.1	10.79	16.3	*10.36*	22.3	**9.53**	14.9	11.77
ASN	C	554	8.0	**7.00**	8.2	7.83	8.5	8.85	7.8	*7.51*
ASN	CA	357	17.5	6.88	14.0	*6.61*	11.2	7.81	18.0	**5.74**
ASN	CB	142	27.2	*8.89*	24.3	8.90	20.1	11.08	26.8	**8.29**
ASN	CG	323	8.5	**7.38**	8.8	7.98	9.3	8.22	8.3	*7.85*
ASN	N	536	12.6	*8.85*	13.7	**8.74**	13.8	9.77	12.1	9.44
ASN	ND2	80	25.0	**16.72**	24.9	17.15	25.5	18.51	25.2	*17.05*
ASN	O	302	14.3	*10.85*	16.4	**10.80**	22.3	11.10	15.1	12.25
ASN	OD1	164	14.7	*12.99*	16.9	**12.66**	22.0	13.03	15.4	14.36
ASP	C	709	8.0	**6.70**	8.1	7.16	8.4	8.39	7.8	*7.08*
ASP	CA	352	17.6	*7.01*	14.2	7.11	11.3	7.80	18.1	**5.93**
ASP	CB	166	28.1	*7.74*	24.8	7.91	20.7	9.54	27.6	**7.30**
ASP	CG	273	10.2	8.27	9.0	*7.76*	9.1	**7.61**	10.2	7.76
ASP	N	604	12.7	**9.20**	13.9	*9.41*	14.0	10.93	12.2	10.01
ASP	O	362	14.4	13.24	16.6	*12.65*	22.4	**12.04**	15.2	14.49
ASP	OD1	170	14.2	15.44	16.3	**14.96**	20.9	*15.01*	14.8	16.93
ASP	OD2	139	14.8	19.14	16.9	*18.77*	21.6	**17.78**	15.4	20.92
CYS	C	261	8.0	**6.23**	8.1	*6.54*	8.5	7.44	7.7	6.68
CYS	CA	222	17.3	*7.62*	14.0	7.80	11.5	9.03	17.9	**6.54**
CYS	CB	147	27.0	*9.29*	25.4	9.47	21.9	11.00	26.4	**8.94**
CYS	N	211	12.8	**10.29**	14.0	*10.30*	14.3	11.21	12.3	11.07
CYS	O	163	14.4	11.35	16.5	*11.30*	22.4	**10.98**	15.2	12.73
CYS	SG	151	28.2	*18.43*	28.9	**18.28**	28.5	19.26	27.9	18.75
GLN	C	378	7.9	**6.57**	8.0	7.13	8.5	8.13	7.7	*7.12*
GLN	CA	219	17.2	*6.38*	13.9	6.85	11.5	8.80	17.8	**5.70**
GLN	CB	148	24.4	7.78	23.4	*7.68*	20.0	8.92	23.7	**7.46**
GLN	CD	168	8.5	**7.26**	8.9	7.85	9.6	8.32	8.4	*7.80*
GLN	CG	107	26.6	*7.92*	23.9	8.36	20.2	10.26	26.3	**7.49**
GLN	N	353	12.4	**9.17**	13.6	*9.19*	13.9	9.86	11.9	9.79
GLN	NE2	63	24.1	**14.07**	24.1	*14.48*	24.8	16.59	24.4	14.71
GLN	O	216	14.3	12.48	16.5	*12.18*	22.3	**11.96**	15.1	13.98
GLN	OE1	71	15.3	16.61	17.6	*15.69*	23.7	**14.48**	16.2	18.09

Table 6.3: Volumes of individual atoms as calculated with different methods. For each residue, the second column specifies the number of buried occurrences in the data set. The methods employed are AWV, Richards' B, unweighted Voronoi, and radical planes. The values given are mean volume and percentage deviation. The lowest deviation is typeset in bold, the second lowest value in italic.

Atom			AWV		Richards' B		Voronoi		Radical	
Res.	Atm.	#	$\bar{V}$[Å^3]	$\sigma/\bar{V}$%	$\bar{V}$[Å^3]	$\sigma/\bar{V}$%	$\bar{V}$[Å^3]	$\sigma/\bar{V}$%	$\bar{V}$[Å^3]	$\sigma/\bar{V}$%
GLU	C	654	7.9	**6.55**	8.0	*6.91*	8.4	7.73	7.7	6.98
GLU	CA	350	17.5	*7.11*	14.2	7.93	11.7	10.13	18.0	**6.42**
GLU	CB	155	24.6	7.94	23.7	*7.87*	20.2	9.15	23.9	**7.59**
GLU	CD	133	10.2	8.14	9.0	8.30	9.2	*8.09*	10.2	**7.75**
GLU	CG	105	28.1	*11.37*	25.1	11.65	21.4	12.58	27.6	**10.60**
GLU	N	521	12.4	*8.50*	13.7	**8.44**	14.2	9.23	12.0	9.10
GLU	O	275	14.1	11.75	16.3	*11.25*	22.3	**10.23**	14.9	12.89
GLU	OE1	48	14.7	15.51	17.0	*13.97*	22.4	**12.96**	15.5	16.53
GLU	OE2	52	15.4	17.36	17.7	*16.09*	23.4	**15.77**	16.3	18.94
GLY	C	803	8.5	**7.70**	9.1	8.61	9.4	9.41	8.3	*8.43*
GLY	CA	321	29.3	*8.51*	24.4	9.48	20.0	11.46	29.3	**7.72**
GLY	N	695	13.2	**10.04**	14.9	*10.14*	14.9	11.16	12.8	10.79
GLY	O	574	14.4	12.24	16.6	*11.95*	22.5	**11.47**	15.2	13.58
HIS	C	350	7.9	**7.21**	8.1	*7.37*	8.4	8.66	7.7	7.92
HIS	CA	235	17.4	*7.37*	14.1	7.74	11.4	8.91	18.0	**6.35**
HIS	CB	175	26.8	*8.48*	24.2	8.79	20.4	10.54	26.6	**8.03**
HIS	CD2	160	23.2	*10.50*	21.1	11.34	19.8	13.61	23.4	**10.47**
HIS	CE1	76	22.7	*9.26*	20.3	9.68	19.2	12.07	22.9	**9.16**
HIS	CG	275	6.2	**5.97**	8.9	7.97	10.2	*7.95*	5.5	8.43
HIS	N	313	12.5	**8.95**	13.7	*9.11*	13.9	10.03	12.0	9.57
HIS	ND1	156	15.0	**11.83**	16.0	*12.05*	16.6	13.92	14.9	12.54
HIS	NE2	103	15.2	*14.09*	17.3	**13.79**	18.0	15.55	14.9	15.12
HIS	O	230	13.9	10.81	16.0	**10.45**	21.6	*10.60*	14.6	12.19
ILE	C	783	7.7	**6.89**	7.8	*6.90*	8.3	7.87	7.5	7.49
ILE	CA	624	16.8	*7.02*	13.8	7.48	11.4	9.54	17.4	**6.34**
ILE	CB	639	14.6	7.54	14.3	*7.32*	13.0	7.85	14.3	**7.22**
ILE	CD1	409	38.2	**9.13**	37.9	9.19	35.9	10.47	37.7	*9.14*
ILE	CG1	505	24.7	*7.96*	24.4	8.02	22.5	9.36	24.3	**7.89**
ILE	CG2	428	36.8	*8.03*	36.3	8.08	33.5	9.17	36.1	**7.97**
ILE	N	704	12.5	*7.91*	13.7	**7.76**	14.2	8.25	12.0	8.43
ILE	O	556	14.3	10.92	16.5	*10.59*	22.6	**9.75**	15.1	12.11
LEU	C	1129	8.0	**6.49**	8.1	7.09	8.6	7.80	7.8	*7.00*
LEU	CA	978	17.1	*6.43*	13.9	6.68	11.6	7.95	17.7	**5.63**
LEU	CB	776	23.8	7.13	23.2	*7.05*	20.8	8.00	23.3	**6.93**
LEU	CD1	659	37.9	**8.39**	37.7	8.52	35.7	10.00	37.5	*8.51*
LEU	CD2	578	37.5	**8.47**	37.2	8.73	35.3	10.51	37.0	*8.65*
LEU	CG	1024	15.1	*8.67*	14.9	8.73	13.8	9.96	14.8	**8.55**
LEU	N	1072	12.4	**8.20**	13.6	*8.27*	14.0	8.90	12.0	8.81
LEU	O	754	14.1	11.57	16.3	*11.23*	22.4	**10.32**	14.9	12.85
LYS	C	778	7.9	**7.24**	8.0	8.00	8.5	8.78	7.7	*7.85*
LYS	CA	437	17.2	*6.97*	14.0	7.48	11.6	9.19	17.8	**6.20**
LYS	CB	207	24.0	*7.93*	23.3	8.00	20.6	9.69	23.4	**7.84**
LYS	CD	104	24.8	*8.87*	24.2	8.88	21.6	10.97	24.2	**8.73**
LYS	CE	40	27.2	*9.62*	24.5	10.26	21.6	12.61	26.9	**9.27**
LYS	CG	175	23.9	9.33	23.5	*9.23*	21.1	10.10	23.4	**8.91**
LYS	N	692	12.5	**8.76**	13.7	*8.87*	14.1	9.61	12.0	9.41
LYS	NZ	14	22.8	*16.98*	24.5	**16.16**	23.3	17.34	22.1	17.29
LYS	O	393	14.2	11.52	16.4	*11.11*	22.4	**10.27**	15.0	12.72

Table 6.4: *Cont.* Volumes of individual atoms as calculated with different methods.

Atom			AWV		Richards' B		Voronoi		Radical	
Res.	Atm.	#	$\bar{V}[Å^3]$	$\sigma/\bar{V}\%$	$\bar{V}[Å^3]$	$\sigma/\bar{V}\%$	$\bar{V}[Å^3]$	$\sigma/\bar{V}\%$	$\bar{V}[Å^3]$	$\sigma/\bar{V}\%$
MET	C	268	8.0	**6.78**	8.1	7.51	8.6	8.30	7.8	*7.33*
MET	CA	196	17.3	*7.22*	14.1	7.61	11.7	9.62	17.9	**6.51**
MET	CB	168	24.3	8.54	23.6	*8.45*	21.1	9.24	23.8	**8.28**
MET	CE	125	39.7	**10.59**	38.4	10.90	35.3	12.91	39.1	*10.62*
MET	CG	171	26.7	7.31	25.6	*7.28*	23.1	8.53	26.3	**7.06**
MET	N	241	12.6	*16.94*	13.9	**16.83**	14.3	18.24	12.2	18.27
MET	O	192	14.3	12.75	16.5	*12.04*	22.7	**11.36**	15.1	14.17
MET	SD	175	25.5	*9.65*	27.1	**9.62**	28.2	10.47	25.4	10.09
PHE	C	514	8.0	**7.68**	8.1	8.75	8.5	9.99	7.7	*8.40*
PHE	CA	392	17.6	*7.06*	14.2	7.61	11.7	9.68	18.2	**6.32**
PHE	CB	341	26.8	*7.75*	24.5	7.83	21.2	9.07	26.6	**7.33**
PHE	CD1	391	21.7	**9.38**	20.9	9.81	20.5	11.56	21.9	*9.52*
PHE	CD2	369	22.1	**8.37**	21.3	8.59	20.9	9.83	22.3	*8.38*
PHE	CE1	335	21.6	**9.57**	21.7	*9.62*	22.0	10.72	21.7	9.79
PHE	CE2	305	21.9	**9.28**	22.0	*9.31*	22.3	10.35	22.0	9.43
PHE	CG	546	5.7	**4.92**	8.8	7.29	10.3	*7.24*	4.8	7.75
PHE	CZ	319	21.8	**9.92**	21.9	*10.02*	22.2	11.51	21.9	10.25
PHE	N	469	12.5	**8.23**	13.7	*8.31*	14.0	9.09	12.0	8.91
PHE	O	361	14.4	10.25	16.6	*9.93*	22.6	**9.47**	15.2	11.43
PRO	C	501	8.0	*6.43*	8.1	**6.39**	8.5	6.98	7.8	6.70
PRO	CA	335	17.5	*7.30*	14.5	7.55	12.1	9.14	18.1	**6.53**
PRO	CB	148	26.0	*8.27*	25.5	8.69	23.0	10.51	25.5	**8.24**
PRO	CD	156	25.4	*8.00*	23.6	8.46	20.6	10.79	25.2	**7.74**
PRO	CG	127	26.8	**9.84**	26.4	9.94	24.2	11.94	26.3	*9.86*
PRO	N	573	6.5	11.00	8.8	*9.55*	9.6	**8.89**	5.8	13.43
PRO	O	247	14.6	13.60	16.8	*13.04*	22.9	**12.11**	15.5	14.84
SER	C	794	8.1	**6.69**	8.2	7.31	8.4	8.75	7.8	*7.21*
SER	CA	478	18.0	*7.47*	14.4	8.06	11.6	10.14	18.4	**6.62**
SER	CB	231	29.4	*8.55*	24.8	9.20	21.0	11.62	29.1	**8.12**
SER	N	600	13.0	*9.35*	14.3	**9.09**	14.4	10.05	12.6	9.89
SER	O	460	14.2	12.15	16.3	*11.92*	22.0	**11.75**	15.0	13.51
SER	OG	237	15.2	*14.69*	19.1	**14.64**	23.8	15.15	15.5	17.04
THR	C	787	7.9	**7.20**	8.0	*7.50*	8.4	7.64	7.6	7.59
THR	CA	512	17.3	*7.11*	14.0	7.51	11.3	9.61	17.9	**6.29**
THR	CB	299	18.5	*8.38*	15.2	9.82	13.0	11.21	18.6	**7.54**
THR	CG2	171	36.6	*8.14*	35.8	8.20	31.8	9.53	35.7	**8.05**
THR	N	623	12.8	*8.54*	13.9	**8.24**	14.0	8.61	12.3	8.92
THR	O	429	14.1	10.58	16.2	*10.52*	22.0	**10.26**	14.8	11.88
THR	OG1	209	15.2	*15.24*	18.3	**14.73**	23.4	15.30	15.4	17.57

Table 6.5: *Cont.* Volumes of individual atoms as calculated with different methods.

Atom			AWV		Richards' B		Voronoi		Radical	
Res.	Atm.	#	$\bar{V}[Å^3]$	$\sigma/\bar{V}\%$	$\bar{V}[Å^3]$	$\sigma/\bar{V}\%$	$\bar{V}[Å^3]$	$\sigma/\bar{V}\%$	$\bar{V}[Å^3]$	$\sigma/\bar{V}\%$
TRP	C	201	8.0	**7.72**	8.1	8.92	8.5	10.21	7.7	*8.51*
TRP	CA	164	17.7	*6.75*	14.3	6.77	11.8	8.37	18.3	**5.87**
TRP	CB	128	27.5	*7.37*	24.9	7.52	21.5	8.98	27.3	**7.00**
TRP	CD1	102	23.5	*8.09*	21.3	8.64	20.2	10.35	23.7	**7.95**
TRP	CD2	181	8.6	**6.39**	9.9	*6.85*	10.8	6.85	8.5	7.23
TRP	CE2	176	7.9	**8.21**	9.3	9.07	10.3	*8.44*	7.7	9.43
TRP	CE3	155	22.2	**7.87**	21.0	8.21	20.9	9.67	22.4	*7.94*
TRP	CG	201	6.6	**5.18**	9.2	6.40	10.4	*6.37*	6.0	6.96
TRP	CH2	110	21.4	**9.57**	21.5	*9.57*	21.5	10.26	21.4	9.61
TRP	CZ2	94	22.9	*9.14*	21.6	9.32	21.0	10.51	23.0	**9.05**
TRP	CZ3	136	21.7	**8.43**	21.8	*8.50*	22.2	10.06	21.8	8.73
TRP	N	169	12.7	**8.56**	14.0	*8.61*	14.3	9.43	12.3	9.18
TRP	NE1	92	16.6	*11.86*	17.6	**11.85**	18.3	12.83	16.6	12.39
TRP	O	142	14.2	12.09	16.3	*11.51*	22.2	**10.45**	14.9	13.23
TYR	C	515	8.0	**7.71**	8.1	8.71	8.5	9.77	7.7	*8.45*
TYR	CA	384	17.4	6.97	14.0	*6.91*	11.4	8.54	17.9	**6.03**
TYR	CB	302	27.0	*8.02*	24.7	8.17	21.3	9.51	26.8	**7.60**
TYR	CD1	339	21.3	**8.76**	20.5	9.06	20.1	10.62	21.5	*8.84*
TYR	CD2	311	21.6	**8.88**	20.8	9.26	20.2	10.96	21.8	*9.00*
TYR	CE1	234	22.4	*9.12*	21.1	9.28	20.3	10.90	22.5	**9.11**
TYR	CE2	214	22.7	*8.97*	21.3	9.09	20.4	10.12	22.8	**8.84**
TYR	CG	521	5.7	**5.14**	8.8	7.57	10.3	*7.30*	4.8	8.11
TYR	CZ	396	8.1	**7.20**	9.3	7.83	10.1	*7.40*	7.9	8.13
TYR	N	430	12.5	**8.19**	13.7	*8.39*	13.9	9.00	12.0	8.84
TYR	O	336	14.2	10.88	16.3	*10.51*	22.1	**10.12**	15.0	12.05
TYR	OH	111	17.9	**13.76**	19.7	*13.86*	24.8	15.13	18.9	15.18
VAL	C	1174	7.8	**7.14**	7.9	*7.35*	8.4	8.30	7.6	7.74
VAL	CA	937	16.9	*6.79*	13.8	7.38	11.4	9.70	17.5	**6.20**
VAL	CB	907	14.9	7.93	14.6	*7.80*	13.3	8.70	14.7	**7.67**
VAL	CG1	610	37.2	**7.40**	36.7	7.50	33.7	8.90	36.5	*7.44*
VAL	CG2	655	36.8	*7.07*	36.4	7.13	33.5	8.35	36.2	**7.07**
VAL	N	1031	12.5	*7.69*	13.7	**7.62**	14.2	8.14	12.1	8.23
VAL	O	756	14.4	9.78	16.5	*9.45*	22.7	**8.39**	15.2	10.72

Table 6.6: *Cont.* Volumes of individual atoms as calculated with different methods.

6.4 Packing densities

We also computed the packing densities of the amino acid residues. The packing density of a residue is defined as the quotient $\frac{V_{vdW}}{V_{AWV}}$ of the van der Waals volume of the residue divided by the residue volume computed using the AWV method. Again, we used the set of radii proposed by Tsai et al. (1999), and only residues from the interior without any accessible surface were considered in the calculation. See table 6.7 for the detailed results.

We computed an overall average packing density of 64.10%. For the individual amino acid residues, the average densities range from 62.73% for Leu up to 66.56% for Lys. Along the backbone the average packing density is 70.33%, which is significantly higher than the average packing density of side chains, which we computed as 59.13%. Aliphatic side chains tend to be slightly less tightly packed than polar ones.

Apparently, these densities are much smaller than those values given by Richards (1974). Most notably, Richards calculated an average packing density of 75% for the interior residues of lysozyme and ribonuclease S. This number is also cited extensively in the biochemical literature, such as the textbooks by Creighton (1993) or Kyte (1995).

We found two sources for the large discrepancy between our results and those numbers published by Richards: First of all, the actual set of radii used in the calculation influences the computed van der Waals volume of the residues. As we have noted earlier, the partitioning of space given by the AWV method does not change if all radii are increased by a common additive constant Δr. When comparing the radii set we used with the set of radii used by Richards (1974), we observe that most radii assigned by Richards are larger than the corresponding radii given by Tsai et al. (1999). See also table 6.9. Therefore, we did the same computation using Richards' set of radii. This also implied using a slightly smaller probe sphere with a radius of only 1.4Å. See table 6.8 for the densities calculated using these parameters. The average packing density increased to 69.70%, with an average density of 75.14% along the backbone, and 65.62% for the side chains. Yet, while these values are much higher than those obtained using the new set of radii by Tsai et al. (1999), they are still well separated from those values reported by Richards.

We compared the van der Waals volumina computed by our program to those given by Liang et al. (1998). We found that the volumes computed by our program

Residue		Density $\frac{V_{vdW}}{V_{AWV}}$ %		
Name	#	tot.	backb.	sidec.
ALA	299	64.27	70.25	56.08
ARG	8	63.23	68.94	61.18
ASN	22	63.26	69.23	59.26
ASP	25	64.35	69.58	60.48
CYS	65	66.56	70.69	62.86
GLN	8	64.96	73.24	60.87
GLU	7	62.93	69.58	59.26
GLY	220	65.72	65.72	0.00
HIS	19	65.88	69.55	64.11
ILE	219	63.23	71.88	59.42
LEU	221	62.73	71.32	58.90
LYS	4	65.93	73.48	62.82
MET	55	63.41	71.36	59.98
PHE	82	63.12	70.43	60.47
PRO	24	65.43	71.42	61.98
SER	106	65.59	69.81	60.71
THR	56	65.31	71.74	60.68
TRP	24	64.38	71.04	62.47
TYR	26	64.01	71.85	61.33
VAL	292	63.31	71.42	58.68

Table 6.7: Average packing densities of amino acid residues computed using the radius set by Tsai et al. (1999). The packing density is defined as the quotient of the van der Waals volume V_{vdW} divided by the volume occupied by the residues' AWV cells V_{AWV}. The table gives for each residue the overall density, and separately the density of the polypeptide group along the backbone and the density of the side chain.

Residue		Density $\frac{V_{vdW}}{V_{AWV}}$ %		
Name	#	tot.	backb.	sidec.
ALA	281	70.04	75.17	63.41
ARG	8	67.81	73.29	65.85
ASN	17	67.88	73.08	64.26
ASP	20	69.24	74.00	65.73
CYS	57	71.73	75.32	68.61
GLN	8	69.72	78.32	65.52
GLU	7	67.90	73.95	64.59
GLY	194	71.07	71.07	0.00
HIS	17	69.92	74.66	67.62
ILE	208	69.30	76.40	66.33
LEU	213	68.76	76.10	65.65
LYS	5	71.83	78.19	69.52
MET	52	69.04	76.01	66.15
PHE	65	66.51	74.73	63.34
PRO	21	71.74	75.71	69.57
SER	95	71.66	74.99	68.07
THR	47	71.41	76.16	68.17
TRP	16	68.55	75.74	66.38
TYR	26	67.39	75.62	64.51
VAL	267	69.22	76.13	65.48

Table 6.8: Average packing densities of amino acid residues computed using the radius set by Richards (1974). The packing density is defined as the quotient of the van der Waals volume V_{vdW} divided by the volume occupied by the residues' AWV cells V_{AWV}.

are up to 3% less than those values published by the latter authors. This implies that all the densities computed by our software are approximately 2% below the correct values. The reason for this difference turned out to be the rather naive approximation of the spherical parts of the van der Waals surface of the molecule by polyhedra from the interior. As already discussed in section 4.6, the algorithm could be modified to take this error into account while refining the triangulation approximating the surfaces of the individual AWV cells. Until this has been done, we would like to consider the results in this section as preliminary.

6.5 Conclusions

To summarize, we have demonstrated that the atomic and especially the overall residue volumes computed using the AWV method have a significantly better distribution than those computed using Richards' B and radical planes, not to mention the Voronoi method. In a way, the radical planes method tends to yield the least percentage distribution for atoms of different sizes that are farther away, while Richards' B is especially good at bonded atoms of different types. The AWV method seems to combine this in a favorable way.

The comparison with previous studies essentially confirmed the sensitivity of volume computations on the exact choice of the data set and the boundary conditions. All the previous authors cited in this section already observed this.

Finally, neither method seems to yield adequate results for hydrogen bonded atoms. Hydrogen bonds are highly directed, and a purely geometric division of space based on Euclidean distance alone cannot capture this directionality. A possible solution might be the explicit inclusion of hydrogens in the data set. Another solution might be a partition of space based on orbitals instead of atomic coordinates only.

We believe that these results provide enough experimental evidence to justify further studies of molecular volumes and density distributions using the AWV method. In our opinion, studies of the packing density near the molecular surface, and a detailed account on the side chain packing in the interior of the molecule are interesting topics of future research.

6.6 Experimental setup

6.6.1 Methods employed

In our study, we compared the volumes of the cells as defined by an AWV tessellation to the corresponding tessellations computed using the Voronoi method, the radical plane method, and Richards' B method. The cells of the AWV tessellation were computed using the implementation as described in chapter 4 of this thesis. The polyhedral cells of the other three tessellations were computed using the intersection algorithm for halfspaces described in section 4.5. As in the previous studies by Harpaz et. al (1994), Gerstein et al. (1995), and Tsai et al.(1999), the polyhedra for Richards' B were computed without special treatment of aromatic rings.

6.6.2 Radius set

To assign radii to the individual atom types, we used the set of radii that was recently proposed by Tsai et al. (1999), see also table 6.9. The assignment of atom types to individual atoms was done using the rules given by the same authors. Tables 6.10 and 6.11 list these type assignments in detail.

6.6.3 Data set

We performed the volume and density calculations on the same data set as Ponitius et al. (1996). This data set consists of 64 high-resolution structures selected from the PDB. These structures were selected by the cited authors because they had been refined at a resolution of 2Å or better and to an R-factor of at most 0.20. In addition, they include representatives of different fold families as described by Orengo et al. (1993). Table 6.12 gives a detailed list of the PDB identification codes of these entries.

When reading the data sets, all cofactors and water molecules were stripped off and only atoms belonging to one of the 20 standard amino acid residue types were retained. We included only atoms and residues in the statistics that are completely buried. An atom is considered as buried, if a probe sphere cannot touch it without intersecting one of the other retained atoms. We used a probe sphere of radius $R = 1.5$Å in our calculations.

Type	Radius [Å]		
	Tsai	Chothia	Richards
C3	1.61	1.76	1.70
C3H	1.76	1.76	1.70
C4H	1.88	1.87	2.00
C4HH	1.88	1.87	2.00
C4HHH	1.88	1.87	2.00
N3	1.64	1.50	1.70
N3H	1.64	1.65	1.70
N3HH	1.64	1.65	1.60
N4HHH	1.64	1.50	2.00
O1	1.42	1.40	1.40
O2H	1.46	1.40	1.60
S2	1.77	1.85	1.80
S2H	1.77	1.85	—

Table 6.9: Radii assigned to the specific atom types. The first set of radii was recently proposed by Tsai et al. (1999), and was used in the present study. The other two columns specify the radii sets as given by Chothia (1975) and Richards (1974).

Residue	Atom	Type
GLY	O	O1
GLY	C	C3
GLY	CA	C4HH
GLY	N	N3H
ALA	O	O1
ALA	C	C3
ALA	CA	C4H
ALA	N	N3H
ALA	CB	C4HHH
VAL	O	O1
VAL	C	C3
VAL	CA	C4H
VAL	N	N3H
VAL	CB	C4H
VAL	CG1	C4HHH
VAL	CG2	C4HHH
LEU	O	O1
LEU	C	C3
LEU	CA	C4H
LEU	N	N3H
LEU	CB	C4HH
LEU	CG	C4H
LEU	CD1	C4HHH
LEU	CD2	C4HHH
ILE	O	O1
ILE	C	C3
ILE	CA	C4H
ILE	N	N3H
ILE	CB	C4H
ILE	CG1	C4HH
ILE	CG2	C4HHH
ILE	CD1	C4HHH
MET	O	O1
MET	C	C3
MET	CA	C4H
MET	N	N3H
MET	CB	C4HH
MET	CG	C4HH
MET	SD	S2
MET	CE	C4HHH
PRO	O	O1
PRO	C	C3
PRO	CA	C4H
PRO	N	N3H
PRO	CB	C4HH
PRO	CG	C4HH
PRO	CD	C4HH

Residue	Atom	Type
HIS	O	O1
HIS	C	C3
HIS	CA	C4H
HIS	N	N3H
HIS	CB	C4HH
HIS	CG	C3
HIS	ND1	N3
HIS	CD2	C3H
HIS	CE1	C3H
HIS	NE2	N3H
PHE	O	O1
PHE	C	C3
PHE	CA	C4H
PHE	N	N3H
PHE	CB	C4HH
PHE	CG	C3
PHE	CD1	C3H
PHE	CD2	C3H
PHE	CE1	C3H
PHE	CE2	C3H
PHE	CZ	C3H
TYR	O	O1
TYR	C	C3
TYR	CA	C4H
TYR	N	N3H
TYR	CB	C4HH
TYR	CG	C3
TYR	CD1	C3H
TYR	CD2	C3H
TYR	CE1	C3H
TYR	CE2	C3H
TYR	CZ	C3
TYR	OH	O2H
TRP	O	O1
TRP	C	C3
TRP	CA	C4H
TRP	N	N3H
TRP	CB	C4HH
TRP	CG	C3
TRP	CD1	C3H
TRP	NE1	N3H
TRP	CD2	C3
TRP	CE2	C3
TRP	CE3	C3H
TRP	CZ3	C3H
TRP	CZ2	C3H
TRP	CH2	C3H

Table 6.10: Rule set used to associate atomic types to individual atoms. Part I, aliphatic residues, methinoine, proline, aromatic residues and histidine.

Residue	Atom	Type
CYS	O	O1
CYS	C	C3
CYS	CA	C4H
CYS	N	N3H
CYS	CB	C4HH
CYS	SG	S2
SER	O	O1
SER	C	C3
SER	CA	C4H
SER	N	N3H
SER	CB	C4HH
SER	OG	O2H
SER	OG1	=OG
THR	O	O1
THR	C	C3
THR	CA	C4H
THR	N	N3H
THR	CB	C4H
THR	OG1	O2H
THR	CG2	C4HHH
THR	CG	=CG2
ASN	O	O1
ASN	C	C3
ASN	CA	C4H
ASN	N	N3H
ASN	CB	C4HH
ASN	CG	C3
ASN	OD1	O1
ASN	ND2	N3HH
GLN	O	O1
GLN	C	C3
GLN	CA	C4H
GLN	N	N3H
GLN	CB	C4HH
GLN	CG	C4HH
GLN	CD	C3
GLN	OE1	O1
GLN	NE2	N3HH

Residue	Atom	Type
ASP	O	O1
ASP	C	C3
ASP	CA	C4H
ASP	N	N3H
ASP	CB	C4HH
ASP	CG	C3
ASP	OD1	O1
ASP	OD2	O1
GLU	O	O1
GLU	C	C3
GLU	CA	C4H
GLU	N	N3H
GLU	CB	C4HH
GLU	CG	C4HH
GLU	CD	C3
GLU	OE1	O1
GLU	OE2	O1
LYS	O	O1
LYS	C	C3
LYS	CA	C4H
LYS	N	N3H
LYS	CB	C4HH
LYS	CG	C4HH
LYS	CD	C4HH
LYS	CE	C4HH
LYS	NZ	N4HHH
ARG	O	O1
ARG	C	C3
ARG	CA	C4H
ARG	N	N3H
ARG	CB	C4HH
ARG	CG	C4HH
ARG	CD	C4HH
ARG	NE	N3H
ARG	CZ	C3
ARG	NH1	N3HH
ARG	NH2	N3HH

Table 6.11: Rule set used to associate atomic types to individual atoms. Part II, polar residues, basic and acidic residues.

1bbp	1cob	1csc	1cse	1ctf	1fkf	1fxd	1gd1
1gp1	1hoe	1ifc	1mba	1mbc	1paz	1r69	1rbp
1rnh	1rop	1snc	1tgs	1thb	1trb	1ubq	2alp
2aza	2ca2	2cdv	2ci2	2er7	2fb4	2fcr	2fx2
2gbp	2ovo	2rhe	2rsp	2sar	2scp	2sga	2sic
2trx	2tsc	2wrp	3blm	3chy	3ebx	3grs	3lzm
4bp2	4cla	4enl	4icb	4ptp	5p21	5rub	6tmn
6xia	7aat	8abp	8acn	8dfr	9pap	9rnt	9wga

Table 6.12: Structure set used in calculations. This selection of 64 entries from the PDB has also been used in the study by Pontius et al. (1996).

Chapter 7

Summary

So, what have we achieved in this thesis? We started with a review of different methods to calculate volumes and densities of individual atoms in molecular ensembles. Since none of the previous methods worked completely satisfactory both from the mathematical as the chemical point of view, we considered the AWV tessellation as possible alternative, as proposed in the article by Goede et al. (1997).

To derive an algorithm to compute AWV cells, we went back to study their geometric properties and came up with the surprising result that the hyperbolic and elliptic edges of a cell project as circular arcs onto the defining sphere of cell. Since circles on a sphere can be handled very conveniently by an algorithm, this observation became the very foundation of the algorithms derived in the remainder of the thesis. In addition, we gave a new tight lower bound on the worst-case complexity of a single AWV three dimensional cell defined by n spheres. This bound of $\Theta(n^2)$ demonstrates that AWV cells are significantly different from ordinary Voronoi cells, which can attain in three dimensions only a maximum complexity of $\Theta(n)$.

Because our goal was to devise a practical solution, we decided to design a randomized algorithm that would work without elimination along the coordinate axes. This lead to the theoretical algorithm presented in chapter 3 that used a triangulation of convex polytopes to represent an individual AWV cell. This algorithm computes one such cell amidst n other spheres in expected time $O(n^2 \log n)$. Since we showed the upper bound of $O(n^2)$ on the complexity such a cell to be tight, this is optimal up to a logarithmic factor. However, the experimentally observed behavior of the complexity of these cells is linear in n. In this case, this algorithm would perform the task in expected time $O(n \log^2 n)$.

Motivated by experimental experience, we implemented a slightly simplified version of the algorithm. Besides the core algorithm for computing an analytical representation of an AWV cell, this implementation also includes several pre- and post-processing steps. We used controlled floating point arithmetic combined numeric perturbation techniques to deal with issues of degenracies and numerical round-off errors. As argued by Halperin and Shelton (1997), this is a viable approach for applications in molecular biology because the data sets suffer from experimental imprecision anyway. All these design decisions were verified in a sequence of experiments where we applied the implementation to data sets taken from the domain of application.

Finally, we studied the performance of the AWV method compared to other methods for assigning volumes and densities to individual atoms and amino acid residues of a molecule. As it turned out, the variances of volumes calculated with the AWV method are almost uniformly lower than those obtained for the Richards' B, the radical planes, or the Voronoi method. These results suggest that indeed the AWV method might be the method of choice for this type of calculations.

On the other hand, several questions have been left open, and we would like to indicate some directions of future research:

1. To our best knowledge, the exact worst-case complexity of the single AWV cells in odd dimensions $d \geq 5$ and the complete AWV *diagram* for even dimensions $d > 2$ is still open. The lower bound construction we gave in this thesis might suggest that the AWV diagram can achieve a higher complexity in even dimensions $d > 2$ than is possible for the unweighted diagram.

2. We found it very surprising that the area of meshing, something we considered as trivial post-processing in the first place, still lacks a rigorous understanding as soon as the input is more complex than a planar straight line graph. Considering the numerous possible applications of meshing algorithm ranging from numerical mathematics to computer graphics, we believe that designing meshing algorithms for non-linear input in non-planar domains will remain an interesting and active area of research.

3. Finally, we think that a detailed account on the packing of side chains in the interior of molecular structures using the AWV method might provide new and interesting insights into the problem of protein folding and molecular recognition. In addition, studies of the packing density of proteins near the molecular surface using AWV could prove useful.

Bibliography

Adamy, U. and Seidel, R. (1998). On the Exact Worst Case Query Complexity of Planar Point Location. In *Proc. 9th ACM-SIAM Sympos. Discrete Algorithms.*

Akkiraju, N. (1996). *Molecule Surface Triangulation from Alpha Sphapes.* Dissertation, University of Illinois at Urbana-Champaign.

Alard, P. and Wodak, S. J. (1991). Detection of cavities in a set of interpenetrating spheres. *J. Comp. Chem.*, 12:918–922.

Andrade, M. V. and Stolfi, J. (1998). Exact Algorithms for Circles on the Sphere. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*

Asano, T., Asano, T. and Imai, H. (1986). Partitioning a Polygonal Region into Trapezoids. *J. ACM*, 33:290–312.

Atallah, M. J. and Goodrich, M. T. (1986). Efficient plane sweeping in parallel. In *Proc. 2nd Annu. ACM Sympos. Comput. Geom.*, pp. 216–225.

Atkins, P. W. and Friedman, R. S. (1997). *Molecular Quantum Mechanics.* Oxford University Press, 3rd Edition.

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

Aurenhammer, F. (1991). Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405.

Avrami, M. (1939). Kinetics of phase change I. *J. Chem. Phys.*

Benz, W. (1992). *Geometrische Transformationen.* BI-Wissenschafts-Verlag.

Bern, M. and Eppstein, D. (1995). Mesh generation and optimal triangulation. In Du, D.-Z. and Hwang, F. K., Ed., *Computing in Euclidean Geometry*, Volume 4 of *Lecture Notes Series on Computing*, pp. 47–123. World Scientific, Singapore, 2nd Edition.

Boissonnat, J.-D. and Dobrindt, K. (1992). Randomized construction of the upper envelope of triangles in R^3. In *Proc. 4th Canad. Conf. Comput. Geom.*, pp. 311–315.

Boissonnat, J.-D. and Dobrindt, K. (1996). On-line construction of the upper envelope of triangles and surface patches in three dimensions. *Comput. Geom. Theory Appl.*, 5:303–320.

Boissonnat, J.-D. and Yvinec, M. (1998). *Algorithmic Geometry*. Cambridge Univerisy Press.

Brönnimann, H., Emiris, I., Pan, V. and Pion, S. (1997). Computing exact geometric predicates using modular arithmetic with single precision. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pp. 174–182.

Burnikel, C. (1996). *Exact Computation of Voronoi Diagrams and Line Segment Intersections*. Ph.D thesis, Universität des Saarlandes.

Burnikel, C., Fleischer, R., Mehlhorn, K. and Schirra, S. (1997). A strong and easily computable separation bound for arithmetic expressions involving square roots. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pp. 702–709.

Burnikel, C., Mehlhorn, K. and Schirra, S. (1996). The LEDA class real number. Technical Report MPI-I-96-1-001, Max-Planck Institut Inform., Saarbrücken, Germany.

Chang, J. and Milenkovic, V. (1993). An experiment using LN for exact geometric computations. In *Proc. 5th Canad. Conf. Comput. Geom.*, pp. 67–72.

Chazelle, B. (1990). Polygon Triangulation. In *Proceedings of FOCS*.

Chazelle, B. (1993). An optimal convex hull algorithm in any fixed dimension. *Discrete Comput. Geom.*, 10:377–409.

Chazelle, B., Edelsbrunner, H., Guibas, L. J. and Sharir, M. (1991). A singly-exponential stratification scheme for real semi-algebraic varieties and its applications. *Theoret. Comput. Sci.*, 84:77–105.

Chew, L. (1997). Guaranteed quality Delaunay meshing in 3D. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pp. 391–393.

Chew, L. P. (1989). Guaranteed-quality triangular meshes. Technical Report TR-89-983, Dept. Comput. Sci., Cornell Univ., Ithaca, NY.

Chothia, C. (1975). Structural invariants in protein folding. *Nature*, 254:304–308.

Clarkson, K. L. and Shor, P. W. (1989). Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421.

Cohen, H. (1993). *A Course in Computational Algebraic Number Theory*. Springer Verlag.

Collins, G. E. (1975). Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. In *Proc. 2nd GI Conference on Automata Theory and Formal Languages*, Volume 33 of *Lecture Notes Comput. Sci.*, pp. 134–183. Springer-Verlag, Berlin, West Germany.

Connolly, M. L. (1981). Molecular surface program. *QCPE Bull.*

Connolly, M. L. (1983). Analytical molecular surface calculation. *J. Appl. Crystallogr.*, 16:548–558.

Creighton, T. E. (1993). *Proteins*. W. H. Freemann and Company, 2nd Edition.

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

Dekker, T. J. (1971). A floating-point technique for extending the available precision. *Numerische Mathematik*, 18:224–242.

Doucet, J.-P. and Weber, J. (1996). *Computer-Aided Molecular Design : Theory and Applications*. Academic Press.

Dubé, T., Ouchi, K. and Yap, C. (1996). Tutorial for Real/Expr.

Edelsbrunner, H. (1992). Weighted alpha shapes. Technical Report UIUCDCS-R-92-1760, Dept. Comput. Sci., Univ. Illinois, Urbana, IL.

Edelsbrunner, H., Guibas, L. J. and Stolfi, J. (1986). Optimal point location in a monotone subdivision. *SIAM J. Comput.*, 15(2):317–340.

Edelsbrunner, H. and Mücke, E. P. (1988). Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pp. 118–133.

Emiris, I. and Canny, J. (1995). A general approach to removing degeneracies. *SIAM J. Comput.*, 24:650–664.

Emiris, I. Z., Canny, J. F. and Seidel, R. (1997). Efficient Perturbations for Handling Geometric Degeneracies. *Algorithmica*, 19(1–2):219–242.

Finney, J. L. (1975). Volume, occupation, environment, and accessibility in proteins. The problem of the protein surface. *Journal of Molecular Biology*, 96:721–732.

Fortune, S. (1989). Stable maintenance of point set triangulations in two dimensions. In *Proc. 30th Annu. IEEE Sympos. Found. Comput. Sci.*, pp. 494–505.

Fortune, S. and Van Wyk, C. J. (1996). Static Analysis Yields Efficient Exact Integer Arithmetic for Computational Geometry. *ACM Trans. Graph.*, 15(3):223–248.

Garey, M. R., Johnson, D. S., Preparata, F. P. and Tarjan, R. E. (1978). Triangulating a simple polygon. *Inform. Process. Lett.*, 7(4):175–179.

Geddes, K. O., Czapor, S. R. and Labahn, G. (1992). *Algorithms for Computer Algebra*. Kluwer Academic Publishers.

Gellatly, B. J. and Finney, J. L. (1982). Calculation of protein volumes: An alternative to the Voronoi procedure. *Journal of Molecular Biology*, 161:305–322.

Gerstein, M. and Chothia, C. (1996). Packing at the Protein-Water Interface. *Proc. Nat. Acad. Sci.*, 93:10167–10172.

Gerstein, M., Tsai, J. and Levitt, M. (1995). The Volume of Atoms on the Protein Surface: Calculated from Simulation, using Voronoi Polyhedra. *Journal of Molecular Biology*, 249:955–966.

Goede, A., Preißner, R. and Frömmel, C. (1997). Voronoi Cell - A new method for the allocation of space among atoms. *Journal of Computational Chemistry*.

Goldberg, D. (1991). What Every Computer Scientist Should Know About Floating-Point Arithmetic. *ACM Comput. Surv.*, 23(1):5–48.

Guibas, L. J. and Sedgewick, R. (1978). A dichromatic framework for balanced trees. In *Proc. 19th Annu. IEEE Sympos. Found. Comput. Sci.*, Lecture Notes Comput. Sci., pp. 8–21. Springer-Verlag.

Halperin, D. and Overmars, M. H. (1994). Spheres, Molecules, and Hidden Surface Removal. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pp. 113–122.

Halperin, D. and Shelton, C. (1997). A perturbation scheme for spherical arrangements with application to molecular modeling. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pp. 183–192.

Halperin, D. and Shelton, C. R. (1998). A perturbation scheme for spherical arrangements with application to molecular modeling. *Comput. Geom. Theory Appl.*, 10:273–287.

Harpaz, Y., Gerstein, M. and Chothia, C. (1994). Volume changes on protein folding. *Structure*, 2:641–649.

IEEE (1985). *IEEE Standard for binary floating point arithmetic, ANSI/IEEE Std* 754 – 1985. IEEE Computer Society, New York, NY. Reprinted in SIGPLAN Notices, 22(2):9–25, 1987.

Johnson, W. A. and Mehl, R. F. (1939). Reaction cinetics in processes of nucleation and growth. *Trans. Am. Inst. Min. Engrs.*, 135.

Karasick, M. S., Lieber, D., Nackman, L. R. and Rajan, V. T. (1997). Visualization of Three-Dimensional Delaunay Meshes. *Algorithmica*, 19(1–2):114–128.

Kirkpatrick, D. G. (1983). Optimal search in planar subdivisions. *SIAM J. Comput.*, 12(1):28–35.

Kleywegt, G. T. and Jones, T. A. (1994). Detection, delineation, measurement and display of cavities in macromolecular structures. *Acta Crystallographica*, D50:178–185.

Knuth, D. E. (1992). *Axioms and Hulls*, Volume 606 of *Lecture Notes Comput. Sci.* Springer-Verlag, Heidelberg, Germany.

Kolmogoroff, A. N. (1937). Statistical theory of crystallization of metals. *Bull. Acad. Sci. USSR Mat. Ser.*, 1:355-359.

Kyte, J. (1995). *Structure in Protein Chemistry*. Garland Publishing.

Lang, S. (1992). *Algebra*. Addison Wesley, 3rd Edition.

Lawson, C. L. (1977). Software for C^1 surface interpolation. In Rice, J. R., Ed., *Math. Software III*, pp. 161–194. Academic Press, New York, NY.

Liang, J., Edelsbrunner, H., Fu, P. and Sudhakar, P. V. (1998). Analytical shape computation of macromolecules. *Proteins*, 33:1–17.

Loos, R. (1983). Computing in algebraic extensions. In Buchberger, B., Collins, G. E., Loos, R. and Albrecht, R., Ed., *Computer Algebra: Symbolic and Algebraic Computation*, pp. 173–187. Springer-Verlag.

Mahin, W. K., Hanson, K. and Morris, J. W. (1980). Comparative analysis of the cellular and Johnson-Mehl microstructures through computer simulation. *Acta Metallurgica*, 28:443-453.

McMullen, P. (1970). The maximal number of faces of a convex polytope. *Mathematika*, 17:179–184.

Mehlhorn, K. and Näher, S. (1994). Implementation of a sweep line algorithm for the straight line segment intersection problem. Report MPI-I-94-160, Max-Planck-Institut Inform., Saarbrücken, Germany.

Mehlhorn, K. and Näher, S. (1998). *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, New York.

Mehlhorn, K., Näher, S., Schilz, T., Schirra, S., Seel, M., Seidel, R. and Uhrig, C. (1996). Checking Geometric Programs or Verification of Geometric Structures. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pp. 159–165.

Meijering, J. L. (1953). Interface area, edge length, and number of vertices in crystal aggregates with random nucleation. *Philips Research Report*.

Mishra, B. (1993). *Algorithmic Algebra*. Springer Verlag.

Møller, J. (1992). Random Johnson-Mehl tesselations. *Adv. Appl. Prob.*, 24:814–844.

Møller, J. (1995). Generation of Johnson-Mehl crystals and comparative analysis of models for random nucleation. *Adv. Appl. Prob.*, 27:367–383.

Mulmuley, K. (1989). An efficient algorithm for hidden surface removal. *Comput. Graph.*, 23(3):379–388.

Mulmuley, K. (1994a). An Efficient Algorithm for Hidden Surface Removal, II. *Journal of Computer and Systems Sciences*, 49:427–453.

Mulmuley, K. (1994b). *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice Hall, Englewood Cliffs, NJ.

Okabe, A., Boots, B. and Sugihara, K. (1992). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Chichester, UK.

Orengo, C. A., Flores, T. P., Taylor, W. R. and Thornton, J. M. (1993). Identification and classification of protein fold families. *Protein Engineering*, 6:485–500.

Pontius, J. (1997). *Atomic volumes in protein crystallographic structures and their use in structure validation*. Dissertation, Université Libre de Bruxelles.

Pontius, J., Richelle, J. and Wodak, S. (1996). Deviations from Standard Atomic Volumes as a Quality Measure for Protein Crystal Structures. *Journal of Molecular Biology*, 264:121–136.

Priest, D. (1991). Algorithms for arbitrary precision floating point arithmetic. In *Proc. 10th Symp. on coputer arithmetic*, pp. 132–143.

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

Richards, F. M. (1977). Areas, volumes, packing and protein structure. *Annu. Rev. Biophys. Bioeng.*, 6:151-176.

Ruppert, J. (1995). A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation. *Journal of Algorithms*, 18(3):548–585.

Sanner, M. F., Olson, A. J. and Spehner, J.-C. (1995). Fast and Robust Computation of Molecular Surfaces. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pp. C6–C7.

Sarnak, N. and Tarjan, R. E. (1986). Planar point location using persistent search trees. *Commun. ACM*, 29(7):669–679.

Schirra, S. (1998). Robustness Issues. In Sack, J.-R. and Urrutia, J., Ed., *Handbook of Computational Geometry*. Elsevier Science Publishers B.V. North-Holland, Amsterdam.

Schorn, P. (1991). *Robust algorithms in a program library for geometric computation*. Ph.D. thesis, ETH Zürich, Switzerland. Report 9519.

Seidel, R. (1981). A convex hull algorithm optimal for point sets in even dimensions. M.Sc. thesis, Dept. Comput. Sci., Univ. British Columbia, Vancouver, BC. Report 81/14.

Seidel, R. (1991). A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom. Theory Appl.*, 1(1):51–64.

Seidel, R. (1993). Backwards Analysis of Randomized Geometric Algorithms. In Pach, J., Ed., *New Trends in Discrete and Computational Geometry*, Volume 10 of *Algorithms and Combinatorics*, pp. 37–68. Springer-Verlag.

Serpette, B., Vuillemin, J. and Hervé, J. C. (1989). BigNum: a portable and efficient package for arbitrary-precision arithmetic. Technical report, INRIA.

Sharir, M. (1985). Intersection and closest-pair problems for a set of planar discs. *SIAM J. Comput.*, 14:448–468.

Shewchuk, J. (1996a). Adaptive precision floating point arithmetic and fast robust geometric predicates. Technical Report CMU-CS-96-140, Carnegie Mellon Univ., Pittsburgh, PA.

Shewchuk, J. R. (1996b). Robust Adaptive Floating-Point Geometric Predicates. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pp. 141–150.

Stolfi, J. (1991). *Oriented Projective Geometry: A Framework for Geometric Computations*. Academic Press, New York, NY.

Stoyan, D., Kendall, W. S. and Mecke, J. (1995). *Stochastic Geometry and its Applications*. John Wiley & Sons, 2nd Edition.

Sugihara, K. and Iri, M. (1994). A robust Topology-Oriented Incremental algorithm for Voronoi diagrams. *Internat. J. Comput. Geom. Appl.*, 4(2):179–228.

Szabo, A. and Ostlund, N. S. (1996). *Modern Quantum Chemistry*. Dover Publishing.

Tsai, J., Taylor, R., Chothia, C. and Gerstein, M. (1999). Radii and Volumes for Atomic Groups in Proteins. to appear.

van der Waerden, B. L. (1955). *Moderne Algebra II*. Springer Verlag, 3rd Edition. The chapter on Kronecker elimination was removed in later editions.

van Lint, J. H. and Wilson, R. M. (1992). *A Course in Combinatorics*. Cambridge University Press.

Varshney, A., Brooks, F. P. and Wright, W. V. (1994). Computing smooth molecular surfaces. *IEEE Comp. Graph. Appl.*, 14:19–25.

Voronoi, G. M. (1908). Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième Mémoire: Recherches sur les parallélloèdres primitifs. *J. Reine Angew. Math.*, 134:198–287.

Weiler, K. (1985). Edge-Based Data Structures for Solid Modeling in a Curved Surface Environment. *IEEE Comput. Graph. Appl.*, 5(1):21–40.

Yap, C. K. (1990). Symbolic treatment of geometric degeneracies. *J. Symbolic Comput.*, 10:349–370.

Ziegler, G. M. (1994). *Lectures on Polytopes*, Volume 152 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg.

Curriculum Vitae

Name:	Hans-Martin Will
Date of Birth:	January 6th, 1970
Place of Birth:	Waiblingen
Nationality:	German

1976–1980	Karolinger Elementary School, Waiblingen
1980–1989	Staufer Gymnasium, Waiblingen
1989–1990	Military Service
1990–1992	University of Osnabrück Studies in Applied Systems Sciences and Mathematics
1992–1995	University of Bonn Studies in Mathematics and Biology Diploma in Mathematics
1995–1996	Free University of Berlin Ph.D. Student in Computer Science
1996–1999	ETH Zürich, Dept. Computer Science Ph.D. Student in Computer Science

1985–1989	TRADOS GmbH, Stuttgart Software Developer
1989–1992	Maas High Software GmbH, Stuttgart Software Developer
1992–1994	Research Institute for Discrete Mathematics, Bonn Student Researcher
1995–1996	TRADOS GmbH, Stuttgart Consultant
1996–1998	ETH Zürich, Dept. Computer Science Research Assistant
since Oct. 1998	TRADOS Benelux S.A., Brussels Software Engineer

1991–1995	Scholar of the Studienstiftung des Deutschen Volkes
1995–1996	Scholar of the DFG Graduate School "Computational Discrete Mathematics", Berlin