HAL
open science

Computing the Volume of a Union of Balls: a Certified Algorithm

Frédéric Cazals, Kanhere Harshad, Sebastien Loriot

To cite this version:

Frédéric Cazals, Kanhere Harshad, Sebastien Loriot. Computing the Volume of a Union of Balls: a Certified Algorithm. [Research Report] RR-7013, INRIA. 2009, pp.25. inria-00409374

HAL Id: inria-00409374

https://inria.hal.science/inria-00409374

Submitted on 3 Sep 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Computing the Volume of a Union of Balls: a Certified Algorithm

Frédéric Cazals — Harshad Kanhere — Sébastien Loriot

N° 7013

August 2009

Thème BIO

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--7013--FR+ENG

Computing the Volume of a Union of Balls: a Certified Algorithm

Frédéric Cazals * , Harshad Kanhere † , Sébastien Loriot ‡

Thème BIO — Systèmes biologiques
Projet ABS

Rapport de recherche n° 7013 — August 2009 — 22 pages

Abstract: Balls and spheres are amongst the simplest $3D$ modeling primitives, and computing the volume of a union of balls is an elementary problem. Although a number of strategies addressing this problem have been investigated in several communities, we are not aware of any robust algorithm, and present the first such algorithm.

Our calculation relies on the decomposition of the volume of the union into convex regions, namely the restrictions of the balls to their regions in the power diagram. Theoretically, we establish a formula for the volume of a restriction, based on Gauss' divergence theorem. The proof being constructive, we develop the associated algorithm. On the implementation side, we carefully analyse the predicates and constructions involved in the volume calculation, and present a certified implementation relying on interval arithmetic. The result is certified in the sense that the exact volume belongs to the interval computed using the interval arithmetic.

Experimental results are presented on hand-crafted models presenting various difficulties, as well as on the 58,898 models found in the 2009-07-10 release of the Protein Data Bank.

Key-words: Union of balls, Voronoi diagrams, α-shapes, robustness, interval arithmetic, Van der Waals models, molecular shapes, structural biology.

* INRIA Sophia-Antipolis-Méditerranée, Algorithms-Biology-Structure; Frederic.Cazals@sophia.inria.fr
† IIT Bombay; harshad_kanhere@iitb.ac.in
‡ INRIA Sophia-Antipolis-Méditerranée, ABS; Sebastien.Loriot@sophia.inria.fr

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Calcul du volume d'une union de boules: un algorithme certifié

Résumé : Les sphères et les boules sont les primitives géométriques courbes les plus simples en 3D, de telle sorte que calculer le volume occupé par une union de boules est un problème élémentaire. Bien que divers algorithmes aient été développés pour le résoudre, à notre connaissance, aucun d'entre eux n'est robuste. Nous présentons donc le premier tel algorithme.

Notre calcul du volume repose sur une décomposition de celui-ci en régions convexes, à savoir la restriction des boules à leurs régions dans le diagramme de puissance. D'un point de vue théorique, nous établissons une formule de calcul basée sur le théorème de la divergence de Gauss. Sa preuve constructive donne directement un algorithme de calcul. D'un point de vue pratique, nous détaillons les prédicats et constructions impliqués dans ce calcul, ainsi qu'une implémentation certifiée à base d'arithmétique d'intervalle. La certification tient au fait que le volume exact appartient à l'intervalle calculé avec l'arithmétique d'intervalle.

Des résultats expérimentaux sont présentés, à la fois sur des cas d'école présentant diverses difficultés, mais aussi sur les 58,898 modèles de la Protein Data Bank, version du 10 Juillet 2009.

Mots-clés : Union de boules, diagrammes de Voronoi, α-shapes, calcul certifié, arithmétique d'intervalle, modèles de Van der Waals, molécules, biologie structurale.

1 Introduction

1.1 Modeling with Balls

Balls and spheres are the simplest curved primitives that can be used in 3D modeling, and have countless applications in science and engineering. Given a known 3D object, balls can be used to provide hierarchical approximations, for example to efficiently perform collision detection [1] or multi-scale visualization [20]. In particular, the volume of a union of balls can be used to evaluate the tightness of an approximation or of a bounding volume hierarchy. In a nearby vein, balls can be used to approximate an object by mimicking the medial axis transform, which is the bottom-line of say the power crust algorithm [3]. In the realm of approximation theory, where a 3D object is known through sample points on its surface, balls are also key for the inference of geometric and topological properties of the unknown object from the samples. In particular, one may mention the so-called boundary measure of a point cloud, from which singular points and sharp features of the sampled model can be estimated from the volume of restrictions of balls centered on the samples [10]. In molecular modeling, balls are central to manipulate Van der Waals or solvent accessible models, where each atom is represented by a ball whose radius depends upon the atom type and its covalent environment. In this context, the volume of a set of atoms directly encodes the packing properties of these atoms, a key parameter in structural analysis [16]. While attempts have been made at reporting the exact volume [18], another privileged route consist of estimating the volume by Monte Carlo and grid based techniques, for which algorithms are still being developed [19, 21].

These applications motivate our interest in the problem of computing the volume of unions of balls. In particular, extending previous analysis of molecular structures based on surface areas [6, 9, 17] requires the robust calculation of atomic volumes.

1.2 Computing the Volume of a Collection of Balls: Previous Work on Exact Methods

Assume that we wish to compute the volume of the domain $\bigcup B_i$ which is the union of n balls $\{B_i\}_{i=1,\ldots,n}$. Denote V_i the region of ball B_i in the power diagram[1] of the balls [4], and let the *restriction* R_i of the ball its intersection with its Voronoi region, that is $R_i = V_i \cap B_i$. The computation of the volume of $\bigcup B_i$ has been approached in several ways.

In [5], it is shown that the measure of the volume of the intersection of a ball and a simplex in $\mathbb{R}^d$ reduces to the computation of the intersection between the ball and at most d half-spaces. Applying this result to the partition of a Voronoi cell into simplices provides a direct way to compute the volume of a restriction. However, the calculation involves tricky integrals.

In computational geometry, the theory of α-shapes and α-complexes underpins algorithms on balls [13]. In particular, the volume of the union can be computed from inclusion-exclusion formula [14]. This approach has the advantage of localizing the volume computation to tuples of spheres which are found in the α-complex–up to quadruples in 3D. But it is global and does not provide volumes on a per-restriction basis.

In biophysics, the direct route is taken by Mc Conkey et al. in [18], since the volume calculation is reduced to the calculation of the volume of the restrictions R_i. This strategy has the advantage of providing volume information on a per-ball basis, and it also provides the molecular surface area along the way. Also, this natural partitioning and the simple calculations that ensue contrast with elaborate partitioning scheme requiring equally elaborate integration schemes [5, 14]. Yet, the algorithm of Mc Conkey et al. suffers from two drawbacks. First, the computation of the restriction is brute-force—it does not resort to the α-complex. Second and most importantly, numerical issues are overlooked, so that the algorithm is not robust.

1.3 Contribution and Paper Overview

Robust geometric algorithms. Developing robust geometric algorithms is by no means trivial, due to a subtle interplay between the geometry and the combinatorics. Classically and following the trend initiated around the Computational Geometry Algorithms Library, see `www.cgal.fr`, one distinguishes *predicates* and *constructions*. A predicate is a function taking as input geometric objects and returning a value within a finite

[1] Strictly speaking, the affine diagram of interest is indeed the power diagram. But in the sequel, we abuse terminology and speak of Voronoi cell / edge / vertex. In the same vein, we shall speak of Delaunay triangulation instead of regular triangulation, and of α-complex instead of weighted α-complex.

set. (An example predicate in 3D is the function stating whether a point lies inside, outside or on the sphere circumscribed to four points.) Predicates are key to traverse the decision tree of a given algorithm, so that their exact evaluation is mandatory if robustness is targeted. A construction is a function taking input geometric objects and returning new geometric objects. (An example in 3D is the computation of the coordinates of the center of the ball circumscribing four points.) The evaluation of a construction involved in a predicate clearly has to be under control. The same holds for a construction involved in the numerical output of the algorithm if one wishes to make a claim on the accuracy of this output. In our case, we shall encounter two types of constructions, namely *intermediate* constructions involved in predicates and in the numerical estimation of the volume, and *output* involved in the latter task only.

Contribution. This paper can be seen as a formalization of the paper of Mc Conkey and co-authors, since we also use the partitioning of the volume $\bigcup B_i$ into restrictions. First, we provide a proof the formula used in [18], a fact which is not completely obvious. The proof consists of applying Gauss' formula to the restrictions. It evidences the fact that the volume calculation only requires simple geometric constructions, in particular those of the vertices found on the boundary of the union $\partial \bigcup B_i$, and the weighted circumcenters of the tetrahedra. Second and most importantly, we present a detailed analysis of the numerics involved, together with the associated implementation which follows the spirit of CGAL. In particular, the robustness of the algorithm is controlled along three lines. As a pre-requisite, we take for granted topologically correct restrictions—which we achieve thanks to the Delaunay triangulation and the associated α-shape provided by CGAL. Then, we develop exact predicates to get a correct decision tree sorting out the cases faced along the volume calculation. Finally, we resort to interval arithmetic to obtain a volume estimate in the format of an interval. As a matter of fact, our implementation is the only robust one we are aware of.

As a final comment, notice that aside the volume calculation, we also compute the area of the boundary of the union of balls.

Paper overview. Section 2 present the theorem underlying our volume calculation. The proof of the theorem is preceded by a description of the geometry of a restriction, which takes for granted some familiarity with the α-complex [13]. Section 3 presents the algorithm, while section 4 presents implementation tests.

Our software, `Vorlume`, is made available from `cgal.inria.fr/abs/Vorlume`.

Notations. We consider a collection of balls $\{B_i\}_{i=1,\ldots,n}$; the union of balls and its boundary are denoted $\bigcup B_i$ and $\partial \bigcup B_i$. The sphere associated to B_i is denoted S_i, its center o_i and its radius r_i. Our calculation is based on the Voronoi (power) diagram of the balls. Recall that the power of a point p with respect to a sphere is defined by $\pi(p, S_i) = || \mathbf{o_i p} ||^2 - r_i^2$, and that the radical plane π_{ij} of two spheres is the locii of points such that $\pi(p, S_i) = \pi(p, S_j)$. The restrictions of the balls to their respective Voronoi regions are denoted $\{R_k\}_{i=1,\ldots,n}$.

The vector between two points a and b is denoted $\mathbf{ab}$, and its norm $|| \mathbf{ab} ||$. The dot product between two vectors $\mathbf{u}$ and $\mathbf{v}$ is denoted $\mathbf{u} \cdot \mathbf{v}$.

2 Volume Calculation: Theorem

2.1 Geometry of a Restriction

The boundary of the union of balls clearly contributes to the definition of the restrictions. In the following, we describe the geometry and topology of a restriction from these spherical caps as well as planar faces.

Geometry of restrictions. The geometry of a restriction is intimately related to the α-complex, a construction derived from the Delaunay triangulation and tightly coupled to the Voronoi diagram of the balls. We shall abuse terminology by speaking of α-complex, but will in fact only deal with the case $\alpha = 0$ [2]. Recall that for a fixed value of α, the α-complex is a simplicial complex extracted from the Delaunay triangulation as follows: a simplex defined as the convex hull of the centers of a subset of balls belongs to the α-complex iff the intersection of the restrictions of these balls is non-empty. Following the seminal work of H. Edelsbrunner, given a value of α, a simplex of the Delaunay triangulation is classified either as exterior, singular, regular, or interior [13].

[2] This abuse is justified as follows: to handle the 0-complex, we actually compute the α-complex, and classify its simplices for $\alpha = 0$.

The restriction R_i of ball B_i is a convex region whose boundary is composed of planar faces F_{ij} and spherical caps K_{ij} [3]. A planar face F_{ij} is located in the radical plane defined by ball B_i and a neighbor ball B_j in the α-complex. A spherical cap K_{ij} is a connected region of S_i found on the boundary of the union $\partial \bigcup B_i$. Note that the restriction of a vertex interior in the α-complex does not contain any cap, while a regular vertex may have several of them. Note also that a cap may contain holes. See Figs. 2, 3 and 4, respectively. We use the spherical caps K_{ij} and planar faces F_{ij} to define a partition of the restriction R_i in terms of generalized pyramids consistently denoted Q_{ij} and P_{ij} in the sequel. See Fig. 1 for an illustration.

First, to each spherical cap K_{ij} we associate the solid region Q_{ij} with apex o_i and subtended by the cap K_{ij}. The volume calculation for such a pyramid will be detailed in the proof of Theorem 1. Second, we associate to each face F_{ij} the solid whose apex is o_i and base F_{ij}. Note that the boundaries of face F_{ij} are straight line-segments e_k contained in Voronoi edges, or circular arcs A_k found on the intersection circle $S_i \cap S_j$. The faces of P_{ij} having vertex o_i in common are triangles corresponding to the straight edges on boundary of F_{ij}, conical patches corresponding to the small circular arcs making up the boundary of F_{ij} [4]. Denoting H_{ij} the height of pyramid P_{ij}, its volume satisfies

$$\texttt{Vol}(P_{ij}) = \texttt{Area}(F_{ij}) \cdot H_{ij}/3. \tag{1}$$

We shall need the following:

Definition 1. The *signed volume* $\texttt{SignedVol}(P_{ij})$ of a pyramid P_{ij} is defined as $\texttt{SignedVol}(P_{ij}) = \delta_{ij} \cdot \texttt{Vol}(P_{ij})$, where $\delta_{ij} = -1$ if o_i and the interior of R_i lie on the opposite sides of F_{ij}, and 1 otherwise.

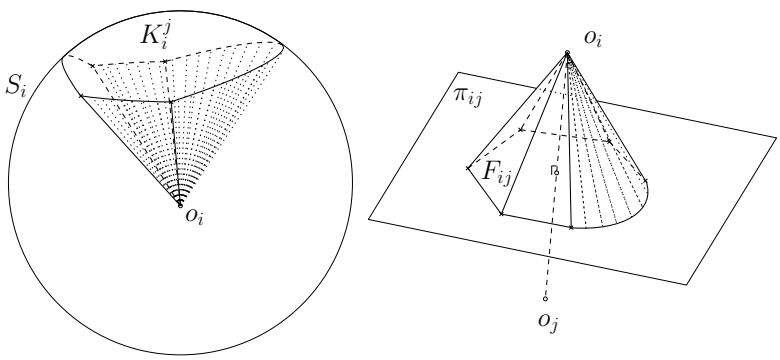

Figure 1: Generalized pyramids. Left: pyramid Q_{ij}; Right: pyramid P_{ij}.

[3]Note that index j in F_{ij} refers to a neighbor of ball i in the α-complex; this is not the case in K_{ij}, where index j just runs over all caps contributed by sphere S_i to $\partial \bigcup B_i$.

[4]If the circular arc lies on a great circle, then the patch is planar.

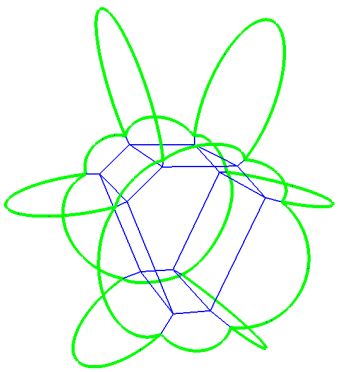

Figure 2: The restriction of an interior vertex with eight neighbors.

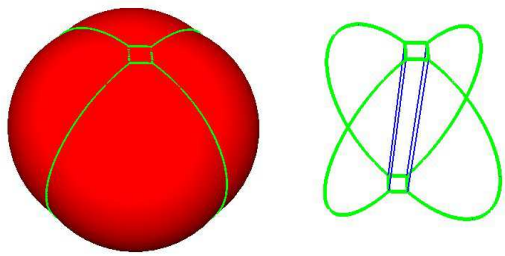

Figure 3: A central sphere, with four neighboring spheres arranged symmetrically around. The restriction of the central sphere has two spherical caps.

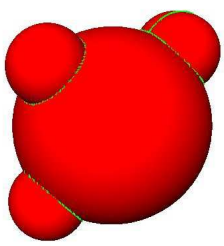

Figure 4: One small sphere delimits a spherical cap on the central sphere; the other two punch two holes on this spherical cap.

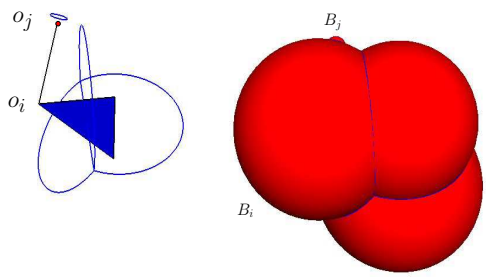

Figure 5: Four balls are depicted. Edge (o_i, o_j) is singular in the α-complex, and center o_j lies outside its restriction.

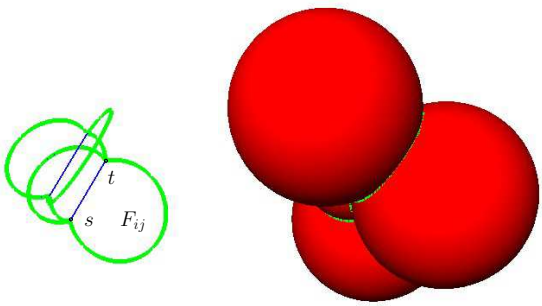

Figure 6: Face F_{ij} is delimited by a major arc and one line-segment.

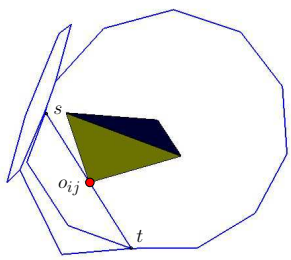

Figure 7: The center of face F_{ij} lies on edge (s, t) which bound the face.

2.2 Statement of the Theorem

Our calculation of the volume is based on the application of Gauss' theorem to a restriction. More precisely, based on definition 1, we have the following theorem, which is implicitly used without a proof in [18]:

Theorem 1. *The volume of the restriction R_i is the sum of the signed volumes of the generalized pyramids P_{ij} whose bases are the faces F_{ij}, together with the volume of the generalized pyramids Q_{ij} subtended by the spherical caps K_{ij} of R_i. That is:*

$$\texttt{Vol}(R_i) = \sum_j \texttt{SignedVol}(P_{ij}) + \sum_j \texttt{Vol}(Q_{ij}). \tag{2}$$

Proof. of Theorem 1: Consider the vector field

$$\mathbf{F} = (\frac{x - x_i}{3}, \frac{y - y_i}{3}, \frac{z - z_i}{3}) \tag{3}$$

defined over R^3. One has

$$\text{div } \mathbf{F} = \nabla \cdot \mathbf{F} = \frac{\partial(\frac{x-x_i}{3})}{\partial x} + \frac{\partial(\frac{y-y_i}{3})}{\partial y} + \frac{\partial(\frac{z-z_i}{3})}{\partial z} = 1. \tag{4}$$

The volume of the restriction R_i is

$$\texttt{Vol}(R_i) = \int_{R_i} 1 dV = \int_{R_i} \nabla \cdot \mathbf{F} dV. \tag{5}$$

By Gauss' Divergence Theorem

$$\int_{R_i} \nabla \cdot \mathbf{F} dV = \oint_{\partial R_i} \mathbf{F} \cdot \mathbf{n} dS, \tag{6}$$

where $\mathbf{n}$ is unit vector along the outward normal of R_i. Therefore, denoting $\{F_{ij}\}_j$ and $\{K_{ij}\}_j$ the collections of planar faces and spherical caps bounding the restriction, we have:

$$\texttt{Vol}(R_i) = \oint_{\partial R_i} \mathbf{F} \cdot \mathbf{n} dS = \sum_j \int_{F_{ij}} \mathbf{F} \cdot \mathbf{n} dS + \sum_j \int_{K_{ij}} \mathbf{F} \cdot \mathbf{n} dS. \tag{7}$$

We now show that $\int_{F_{ij}} \mathbf{F} \cdot \mathbf{n} dS = \texttt{SignedVol}(P_{ij})$ and $\int_{K_{ij}} \mathbf{F} \cdot \mathbf{n} dS = \texttt{Vol}(Q_{ij})$.

Let $\mathbf{n}'$ be unit vector on the boundary of pyramid along the outward normal of P_{ij}. The surface integral $\int \mathbf{F} \cdot \mathbf{n}' dS$ on the triangular region or the conical region connecting a boundary of the face F_{ij} is zero since $\mathbf{n}'$ is normal to the vector field $\mathbf{F}$ of Eq. (3) on the region. Therefore, applying the divergence theorem to P_{ij} yields

$$\texttt{Vol}(P_{ij}) = \int_{P_{ij}} \nabla \cdot \mathbf{F} dV = \oint_{\partial P_{ij}} \mathbf{F} \cdot \mathbf{n}' dS = \int_{F_{ij}} \mathbf{F} \cdot \mathbf{n}' dS. \tag{8}$$

Now if o_i and the interior of R_i are on opposite sides $\mathbf{n}' = -\mathbf{n}$ otherwise $\mathbf{n}' = \mathbf{n}$. Therefore

$$\int_{F_{ij}} \mathbf{F} \cdot \mathbf{n} dS = \texttt{SignedVol}(P_{ij}).$$

Consider now a spherical cap on the boundary of R_i. Its boundaries are circular arcs and on the solid formed by joining the boundary arcs of the spherical cap to o_i the faces will be conical surfaces and the cap itself. By similar reasoning as above surface integral $\int \mathbf{F} \cdot \mathbf{n}' dS$ on conical surfaces will be zero. Therefore, the following concludes the proof:

$$\int_{K_{ij}} \mathbf{F} \cdot \mathbf{n} dS = \texttt{Vol}(Q_{ij}). \tag{9}$$

□

3 Algorithm

We have seen in section 2 that the calculation of the volume restriction reduces to the computation of the signed volumes of pyramids. These calculations are presented in section 3.2. Before and for the sake of completeness, section 3.1 recalls how the restrictions (whence their pyramids) of all balls can be computed at once from the α-complex.

3.1 Computing the Boundary of Restrictions

Computing spherical faces. A spherical face K_{ij} is defined by one or more closed cycles of circular arcs—it may not be simply connected. In the sequel, they are simply termed face when there is no ambiguity. The computation of these faces is carried out using the classification of simplices embedded in the α-complex [2] [5].

Since a singular vertex contributes a complete sphere and the contribution of an interior vertex is void, the only case deserving a discussion is that of a regular vertex, which contributes at least one face. For each such vertex v, consider all exterior tetrahedra incident on it. Term two such tetrahedra of *adjacent* if they share an exterior facet incident to v. Then, each maximal connected component of adjacent tetrahedra defines one face K_{ij}. Since such a face might not be simply connected, we need to identify the closed cycles defining its boundary. One actually finds one such cycle for each connected component of the link of vertex v in the α-complex [2].

Computing planar faces. Each planar face F_{ij} is included in the radical plane dual of an edge of the α-complex. Using the classification of edges in the α-complex, we face three cases: (i) a singular edge contributes a planar face bounded by the intersection circle of two vertex spheres; (ii) an interior edge contributes a planar face bounded by Voronoi segments and Voronoi vertices only; (iii) a regular edge contributes a planar face bounded by circular arcs and Voronoi segments bounded either by Voronoi vertices or intersection points of three spheres found on $\partial \bigcup B_i$. In any case, the boundary of the face is retrieved by processing the triangles incident to the corresponding α-complex edge.

Predicates and constructions.

▷ `Predicates.` Taking for granted the robust construction of the Delaunay triangulation and the α-complex, no other predicate is required.

▷ `Constructions.` As should be clear from the description of the geometry of a restriction R_i, two intermediate constructions are required to embed R_i in 3D: first, one needs to compute the intersection points found at the intersection of three spheres(one point per regular triangle, two per singular one); second, one needs to compute the weighted circumcenters of tetrahedra (one per non-exterior tetrahedron).

3.2 Computing the Volumes of Pyramids

3.2.1 Pyramids with Planar Base

Volume of pyramid. We are interested in computing the volume of a pyramid P_{ij} whose apex is o_i and base is the planar face F_{ij} contained in the plane π_{ij}. We define the *center* of F_{ij}, denoted o_{ij}, as the intersection between π_{ij} and the line through the centers o_i and o_j of the two balls B_i and B_j. Let $\mathbf{u_{ij}}$ be the unit vector from center o_i to center o_j. Denoting $d_{ij} = || \mathbf{o_i o_j} ||$, a simple calculation shows that point o_{ij} satisfies:

$$\mathbf{o_i o_{ij}} = \frac{d_{ij}^2 + r_i^2 - r_j^2}{2d_{ij}} \mathbf{u_{ij}} \equiv h_{ij} \mathbf{u_{ij}}. \tag{10}$$

We call quantity h_{ij} the *signed height* of the pyramid—compare to Eq. (1). The signed height h_{ij} is positive (respectively negative) when center o_i and the restriction R_i lie on the same (respectively opposite) side of face F_{ij}, and null when o_i lies on face F_{ij}.

Recalling the definition of signed volume introduced in Def. 1, we have:

$$\texttt{SignedVol}(P_{ij}) = \delta_{ij} \cdot \texttt{Vol}(P_{ij}) = h_{ij} \cdot \texttt{Area}(F_{ij})/3. \tag{11}$$

▷ `Predicates.` None.

▷ `Constructions.` The signed height of Eq. (10) and the area of Eq. (11) are output constructions.

[5] To handle homogeneously vertices inside and on the convex hull, our description assumes that the 3D Delaunay triangulation has been equipped with a vertex at infinity, thus providing a triangulation of the 3-sphere.

Surface area of its planar base. If the face F_{ij} is a full disk, the area computation is trivial. If not, the area is computed using a two-dimensional application of Gauss' theorem, by adding the signed areas of (pseudo) triangles with apex o_{ij} defined by all line-segments and circular arcs $\{e_k\}$ bounding the face F_{ij}.

The case of a line-segment. Consider the triangle defined by the line-segment e_k and point o_{ij}. Applying the Gauss' formula, its signed area is positive if o_{ij} and the interior of face F_{ij} lie on the same side of e_k, and negative otherwise.

▷ `Predicates.` To determine the sign of the area of a triangle, recall that the radical plane containing F_{ij} and the affine hull of its dual Delaunay edge are orthogonal. Consider a straight edge $e_k = (s, t)$, oriented such that when moving from s to t, and seen from the interior of F_{ij}, the normal induced points in the direction opposite of $\mathbf{o_i o_j}$. See Fig. 9(a) for an illustration. Point o_{ij} and face F_{ij} lie on the same side of edge e_k iff the sign of the following mixed product M is positive:

$$M = (\mathbf{o_{ij}t} \wedge \mathbf{o_{ij}s} \cdot)\mathbf{o_i o_j}. \tag{12}$$

Equivalently, one may resort to the following orientation predicate [6], which has the advantage of not using the constructed point (o_{ij}):

$$\texttt{sign}(M) = -\texttt{sign}(\texttt{Orient}(s, t, o_i, o_j)). \tag{14}$$

Practically, two of the points involved in Eq. (14) might be vertices of $\partial \bigcup B_i$, and in all generality, their coordinates belong to two different algebraic extensions. The problem of determining the sign of such a determinant, which is examined in section 7, is summarized in the following:

Observation 1. Let a, b, c, d be rational numbers. Determining the sign of Eq. (14) is tantamount to determining the sign of a rational number, or of an algebraic number of the form $a + b\sqrt{c}$, or of an algebraic number of the form $a + b\sqrt{\gamma_1} + c\sqrt{\gamma_2} + d\sqrt{\gamma_1\gamma_2}$ where γ_1 and γ_2 are two different algebraic extensions.

▷ `Constructions.` The surface area of a triangle is an output construction.

The case of a circular arc. Consider a circular arc A_k on the intersection circle between the spheres S_i and S_j, and let r_{ij} be the radius of this circle. The area of interest is naturally $r_{ij}^2 \cdot \theta_k/2$, with

$$r_{ij}^2 = r_i^2 - h_{ij}^2. \tag{15}$$

Define

$$\gamma_k = \arccos \frac{\mathbf{o_{ij}s} \cdot \mathbf{o_{ij}t}}{\| \mathbf{o_{ij}s} \| \| \mathbf{o_{ij}t} \|}. \tag{16}$$

Let a circular arc of measure strictly less than π be a minor arc, and a major arc otherwise. For a minor and major arc respectively, one has $\theta_k = \gamma_k$ and $\theta_k = 2\pi - \gamma_k$. See Fig. 9(b) for an illustration.

▷ `Predicates.` The cases between minor and major arcs need to be sorted out. It can be checked that a minor arc corresponds to a positive sign of the mixed product M of Eq. (12), and a major arc to a negative or null sign. Therefore, the decision follows the case analysis of Eq. (14).

▷ `Constructions.` The radius given by Eq. (15) and the angle θ_k are output constructions.

Remark 1. In Mc Conkey et al. [18], the planar base area computation is described as a three-stage process: first, the area of the convex hull of the boundary points is computed; second, the area of each sector subtended by two consecutive points p_i, p_{i+1} found on $\partial \bigcup B_i$ is added; third, the surface area of each triangle defined by the face center and p_i, p_{i+1} is removed. For a major arc, this latter area should be added and not subtracted. See Fig. 8.

[6] Denoting $P_i = (x_i, y_i, z_i)$ the coordinates of point P_i, the orientation predicate of four points consists of evaluating the sign of the following 4×4 determinant:

$$\texttt{Orient}(P_1, P_2, P_3, P_4) = Sign \left(\begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix} \right) \tag{13}$$

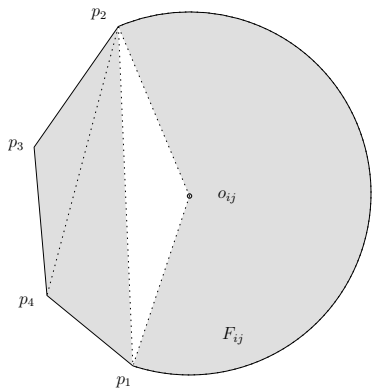

Figure 8: In Mc Conkey et al. [18], the area of face F_{ij} is computed as the sum of area of triangles $p_2p_3p_4$ and $p_2p_4p_1$, minus that of triangle $o_{ij}p_2p_4$, plus area of the sector defined by $o_{ij}p_1p_2$.

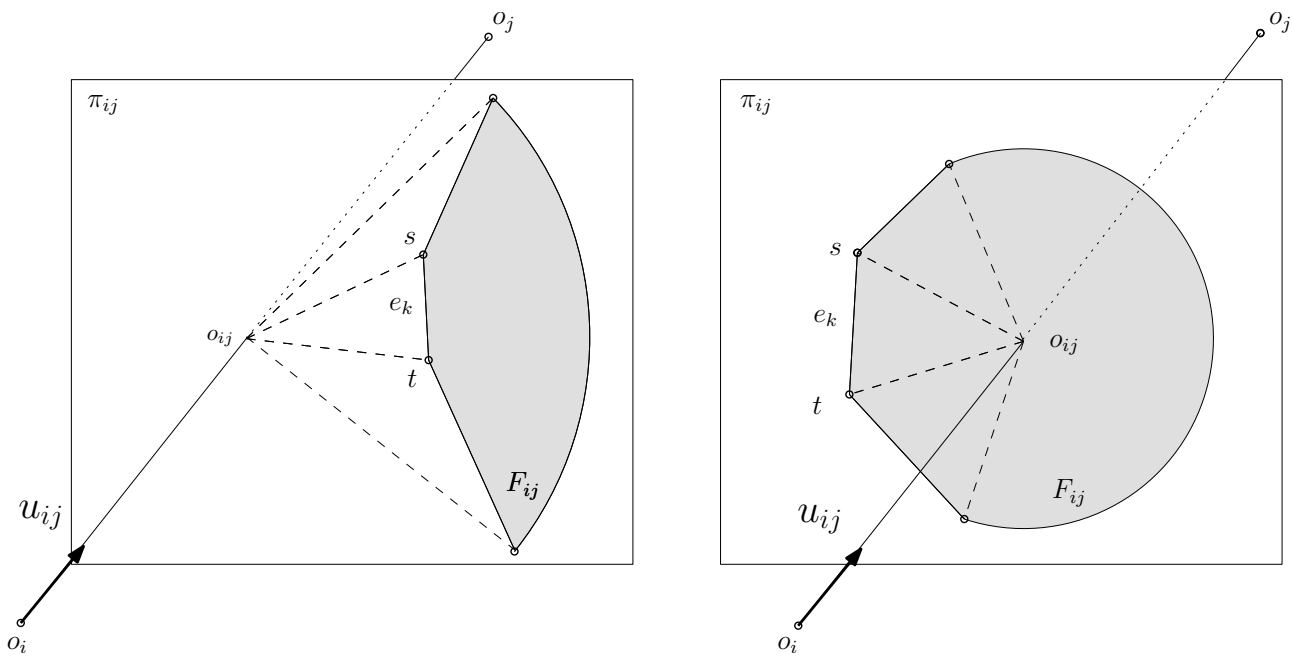

Figure 9: Computing the surface area of planar face F_{ij}. Left: The face does not contain its center o_{ij}; Right: Face with a major arc.

3.2.2 Pyramid with Spherical Base

From Eq. (9) in the proof of Theorem 1, the volume of a spherical pyramid Q_{ij} is given by:

$$\texttt{Vol}(Q_{ij}) = r_i \cdot \texttt{Area}(K_{ij})/3.$$

We compute the area $\texttt{Area}(K_{ij})$ using the Gauss-Bonnet Theorem [12]. If the face K_{ij} is not simply connected, one boundary cycle defines a simply connected region $\texttt{cap}(K_{ij})$ of the sphere S_i containing the face, while the remaining ones define holes. The calculation [7] consists of subtracting the areas of the holes from that of $\texttt{cap}(K_{ij})$, so that the primitive calculation if that of the surface area of a simply connected region. Therefore, assume w.l.o.g. that K_{ij} is such a region, and denote $\{A_k\}_{k=0,\ldots,m-1}$ the m oriented circular arcs defining its boundary—oriented so as to leave the interior of the face to the left. For $k \in \{0, \ldots, m-1\}$, let v_k be the target

[7] From a computational standpoint, this is tantamount to applying Gauss-Bonnet on the non-simply connected region, as changing the orientation of the holes is equivalent to changing the sign of the geodesic curvatures of its circular arcs.

vertex of A_k and θ_k be the external angle between A_k and $A_{k+1 \bmod m}$. Finally let s be a parametrization by arc length of the boundary of K_{ij}—with same orientation. Using the Gauss-Bonnet Theorem, we have:

$$\sum_{k=1}^{m} \int_{v_{k-1}}^{v_k \bmod m} k_g(s)ds + \frac{\texttt{Area}(K_{ij})}{r_0^2} + \sum_{k=0}^{m-1} \theta_k = 2\pi, \tag{17}$$

where $k_g(s)$ is the geodesic curvature of the oriented circular arcs bounding K_{ij}. Notice that this quantity is constant on a given oriented circular arc [8], and its value is 0 if the underlying circle is a great circle.

▷ `Predicates.` The determination of the sign of the geodesic curvature needs a predicate. For a non-great circle, using the definition of the geodesic curvature based upon the covariant derivative [8, 12], one gets that the geodesic curvature on an oriented circle is positive if the orientation induces the smallest part of the sphere, and negative otherwise. Given two spheres S_i and S_j intersecting along a non-great circle C_i, C_i describes the largest part of S_i if

$$d_{ij}^2 + r_i^2 - r_j^2 \tag{18}$$

is positive, and the smallest part otherwise.

▷ `Constructions.` The arc length, external angle and unsigned geodesic curvature are output constructions.

4 Robust Implementation

In this section, we explain how the robust predicates and constructions are practically handled.

Robust geometric implementations. The analysis presented in section 3 shows that apart from the standard predicates involved with the Delaunay triangulation and the α-shape of CGAL, our calculation actually requires two predicates. The first one is used to determine the orientation of a quadruple of points, two of which are constructed points. The second one gives the sign of the geodesic curvature, and is tantamount to determining whether a site lies in its Voronoi cell, see Eq. (18) and Eq. (10).

On the constructions side, the only intermediate constructions are the weighted circumcenters of tetrahedra in the α-complex and boundary points of $\partial \bigcup B_i$.

Input data and number types. The input geometric objects are spheres whose coordinates and squared radii are given as floating point numbers. Our algorithm represents this input exactly. Using a fixed precision floating point number type, the result of an operation on these numbers cannot in general be represented exactly, so that the result returned is an approximation of the true result. To avoid this approximation, we resort to an interval number type. Upon performing an operation, the bounds of the interval returned delimit the range containing the unknown exact result. Output constructions are directly computed using the interval number type as they only influence the precision of the result.

Interestingly, such a number type is also an efficient alternative for the exact evaluation of predicates, and also to handle intermediate constructions.

Lazy evaluation of predicates. To see how the exact evaluation of a predicate can be carried out using an interval number type, assume that we wish to test the sign (positive, negative, null) of an expression involving input data [9]. If the interval computed for this expression is sufficient to conclude—it does not contain zero, we are done at a modest overhead since the cost of maintaining the interval is about twice that of performing a fixed precision floating point calculation. If not, the exact calculation must be carried out, using the exact representation of the parameters.

The case of predicates involving intermediate constructions is handled thanks to lazy constructions.

Lazy constructions. To deal with intermediate constructions, we resort to a lazy strategy based on two complementary representations of the construction. First, an approximate representation of the construction is computed using an interval number type. Then, storing a reference to relevant input data, the exact version of the construction can be worked out upon failure of a predicate. As we do not deal with cascaded constructions,

[8] If a circular arc is supported by a circle $C = (c_C, r_C)$, then its geodesic curvature on a sphere $S = (c_S, r_S)$ is $\frac{\|c_C c_S\|}{r_C r_S}$.

[9] We do not mention classical filtering strategies such as static filters since our predicates involve constructed objects.

our approach is simpler than that of [15], where an history directed acyclic graph of each constructions is maintained.

All predicates are evaluated using this lazy scheme. In particular the orientation predicate `Orient` is the only one involving intermediate constructions. More details on its implementation are given in section 7. Note that the overall efficiency of such a scheme is related to the number of approximate predicates/constructions failing.

Practical matters. Practically, we use `CGAL::Interval_nt` as interval number type and `CGAL::Gmpq` as exact type, both from the CGAL library.

The different versions of intermediate constructions are built using functors from the CGAL Spherical Kernel [11] for boundary points, and from the CGAL Kernel for weighted circumcenters [7] with different number type instantiations: the interval number type for the approximate version and the exact number type for the exact version.

Amongst output constructions, the computation of volume of pyramids and area of faces of $\partial \bigcup B_i$ requires the *acos* function. An interval certified approximation of this function has been used, based on the upper and lower rounded versions of that function provided in the CRLIBM library [10].

5 Experiments

5.1 Setup

Contenders and accuracy. Our experimental study is concerned with two contenders:
– `McC-et-al`: the volume computation algorithm [11] by Mc Conkey et al. [18], which is a standard tool used for structural biology studies. The volume returned is denoted V_{MCC}. This program is a good illustration of what one may expect for a straight implementation of a geometric algorithm relying on fixed precision floating point operations.

– `Vorlume`: our implementation of the algorithm presented in section 3. Since this implementation uses interval arithmetic, the volume returned V_i is also an interval which we represent in the form `interval_median` ± `interval_width`.

To establish the accuracy of these programs, we challenge them with a volume estimate obtained by a Monte-Carlo procedure called `MonteCarlo`. Practically, we compute a bounding cube of the union of balls and then generate uniformly at random points in the cube. The fraction of points within $\bigcup B_i$ multiplied by the volume of cube is the estimated volume V_{MC}.

Models used. Our experiments are threefold. First, we use a set of hand-crafted models featuring various difficulties: one model with an interior vertex (Fig. 2); one model featuring a restriction with two caps (Fig. 3); one model with a spherical cap with holes (Fig. 4); one model with a sphere center outside its restriction (Fig. 5); one model with a face involving a major arc (Fig. 6).

Second, we present statistics on molecular models extracted from the Protein Data Bank. Each such model is processed in the solvent accessible model, that is, crystal water molecules are discarded and so are hydrogen atoms, and the atomic radii of the heavy atoms are expanded by 1.4 Å. The atomic radii used are taken from [22].

On the one hand, we present detailed statistics on one antiboby-antigen complex featuring 2731 atoms—PDB code 1vfb. Since the implementation of Mc Conkey et al. [18] reports volumes for exposed and buried atoms, we use these two categories for a comparison. In addition, we provide additional statistics generated by `Vorlume`. On the other hand, we present a large scale study on the $N_{PDB} = 58,898$ models of the tenth of July 2009 release of the Protein Data Bank. For these models, we compare the accuracies of `Vorlume` versus `McC-et-al`, and also investigate the running time of our software.

All computation reported were run on one core of an Intel(R) Xeon(R) CPU X5482 @ 3.20GHz running under Linux 2.6 x86_64.

[10] Correctly-rounded library of basic double-precision transcendental elementary functions; see `http://lipforge.ens-lyon.fr/www/crlibm/links.html`.

[11] We run it with options options `-all` and `-planedef X`.

5.2 Results

Contenders and accuracy on hand-crafted models. Table 1 reports the volume computed by the three algorithms on the hand-crafted models of section 2. Program `McC-et-al` crashes in 3 cases out of 6, and provides erroneous results in the other cases which are exclusively made up of exposed atoms only—see comments on buried vs. exposed atoms below. Regarding the results produced by `Vorlume`, we note that the median of the interval reported matches the volume estimated by the Monte-Carlo procedure up to two digits.

Model			`Vorlume`	`MonteCarlo`			`McC-et-al`
-	#B	#E	V_i	V_{MC}	#B	#E	V_{MCC}
capwithholes	0	5	$585.48 \pm 2.11\ 10^{-11}$	585.48	2	3	480.66
majorarcs	0	4	$66.58 \pm 9.60\ 10^{-4}$	66.58	0	4	53.39
center-outside-restriction	1	4	$36.12 \pm 6.54\ 10^{-13}$	36.12			segfault
interiorvertex	1	8	$16.91 \pm 7.75\ 10^{-12}$	16.91			segfault
twocaps	0	5	$6.11 \pm 3.26\ 10^{-7}$	6.11			segfault
face-center-collinear	0	4	29,841,383 ± 0.05	29,840,952	0	4	37,475,493

Table 1: Results of testing on small hand-crafted cases. #B, #E represent the number buried and exposed atoms, respectively, of $\bigcup B_i$. For program `MonteCarlo`, the volume is estimated with 10^9 points.

Antibody-antigen complex. To better understand the specificities of the two contenders, we examine in detail the results on the antibody-antigen complex. As seen from Table 2, while program `McC-et-al` is correct for buried atoms [12], whose restrictions are bounded by planar faces only, this is not the case for exposed atoms.

Interestingly, model 1vfb presents the various types of difficulties faced in sections 2 and 3. Out of the 2731 atoms, 66 were found with a ball center outside the restriction; a total of 10,998 faces with face-center outside the face were reported; and 28 faces with major-arcs were found; finally, out of 1614 spherical caps, 12 of them were listed as non-simply connected. These statistics evidence the need to cover all cases in a robust way.

Algo.	#atoms	Vol. of exposed atoms	Vol. of buried atoms	Total volume
`Vorlume`	2,731	44,214.15±0.06	$19{,}887.79 \pm 4.10^{-09}$	64,101.95±0.06
`McC-et-al`	2,731	39,181.76	19,887.67	59,069.43
`MonteCarlo`	2,731			64,102.1

Table 2: Comparing the three programs on PDB file 1vfb. For program `MonteCarlo`, volume estimated with 10^8 points.

Calculations on the whole Protein Data Bank. In doing a large scale study on the PDB, we are interested in assessing the robustness, the precision and the execution speed of our algorithm.

Robustness. The main purpose of this paper is to present a robust application to compute the volume of a union of balls. While program `Vorlume` processes successfully the N_{PDB} models of the PDB, `McC-et-al` faces three types of errors. The first one is a simple segmentation fault, probably a wrong way taken in the decision tree; this happens for 104 files, ranging from 9 to 94,970 atoms.

The second one is the wrong classification of atoms as exposed or buried. This was observed on more than the two third of the files. The mean number of miss-classifications is about one percent of the number of atoms, with values up to 15 percents.

The third type of issue is about erroneous volumes. For a given PDB file, we consider separately the exposed and buried atoms, an information read from the α-complex. For each such class, we compute the fraction of atoms such that the volume V_{MCC} computed by `McC-et-al` is outside the interval reported by `Vorlume`. Gathering these two values for each file of the PDB allows us to plot their distribution for both

[12] The volume V_{MCC} computed by `McC-et-al` belongs to the interval V_i computed by `Vorlume`; two digits are used for the precision.

classes of atoms. As seen from Fig. 10, the mode is about 60% of volumes outside the range for exposed atoms, but it is below 5% for buried atoms.

For a value V_{MCC} lying outside the interval, define the error as the least distance of V_{MCC} to the bounds of interval V_i, and let the normalized error be this error divided by the median of the interval. On selected examples where `McC-et-al` classifies correctly all atoms as exposed or buried, we investigated the distributions of such errors. One such example is PDB code 3fob, which corresponds to an oxidoreductase with 6529 atoms. We observe that for the 3,503 surface atoms, 3,498 volume values are outside the interval we report, and only 125 for the 3,026 interior atoms. The distribution of these normalized errors, on Fig. 11, shows that for exposed atoms, most of the values reported by `McC-et-al` incur a normalized error of about 20%. For buried atoms, the distribution is more favorable, but error beyond one (i.e. 100%) are frequently encountered.

Precision. Let us now focus on the precision of program `Vorlume`, which is encoded in the width of intervals returned. More precisely, for all atoms of a class (exposed, buried) in a given file, consider the sum of the interval width normalized by the interval median. As seen from Fig. 12, for buried atoms, the error is almost a linear function in the number of atoms, with a precision always smaller than 4.10^{-12}. This observation does not hold for exposed atoms, yet, the precision obtained is always below three percent of the total volume. This is related to the fact that the volume of a restriction with spherical caps involves points whose coordinates are algebraic numbers of degree two—as opposed to points with rational coordinates for weighted circumcenters defining planar faces of a buried restriction. This again sheds light on the difficulty of reporting a correct volume estimate using a fixed precision floating point arithmetic.

Computation time. Having examined the question of accuracy, one may question the cost of the overhead faced by our implementation. As seen from Fig. 13, which compares the execution times of `Vorlume` and `McC-et-al`, a mere factor of ten is faced by our implementation. With an average of 3,500 atoms processed per second, the robust volume calculation is quite affordable.

Performances versus accuracy. Finally, let us investigate more precisely the connexion between the performances and the precision on the output. In section 3, we presented the lazy approach used to evaluate exact predicates. We also noticed that in case of failure, the approximate representations were refined from the exact calculation of the constructions. These refined representations are re-used for later predicate evaluations, but also for the volume calculation. In other words, these exact calculations are costly, but improve the precision of the output. To investigate this trade-off precisely, consider the following three strategies, which distinguish the processing of our intermediate constructions (boundary points versus weighted circumcenters of tetrahedra):
– 1: approximating boundary points and weighted circumcenters using interval arithmetic;
– 2: approximating boundary points from their exact counterparts;
– 3: approximating boundary points and weighted circumcenters from their exact counterparts.

We applied them on two PDB files, the one with the worst absolute error and the one with the worst relative error, the results being presented on Table 3. First, we observe that the precision loss comes from the boundary points. Second, with respect to strategy 1, strategy 2 provides a significant precision increase at a mere overhead of two regarding the execution time. Incidentally, this also evidences the importance of the lazy scheme to save computation time.

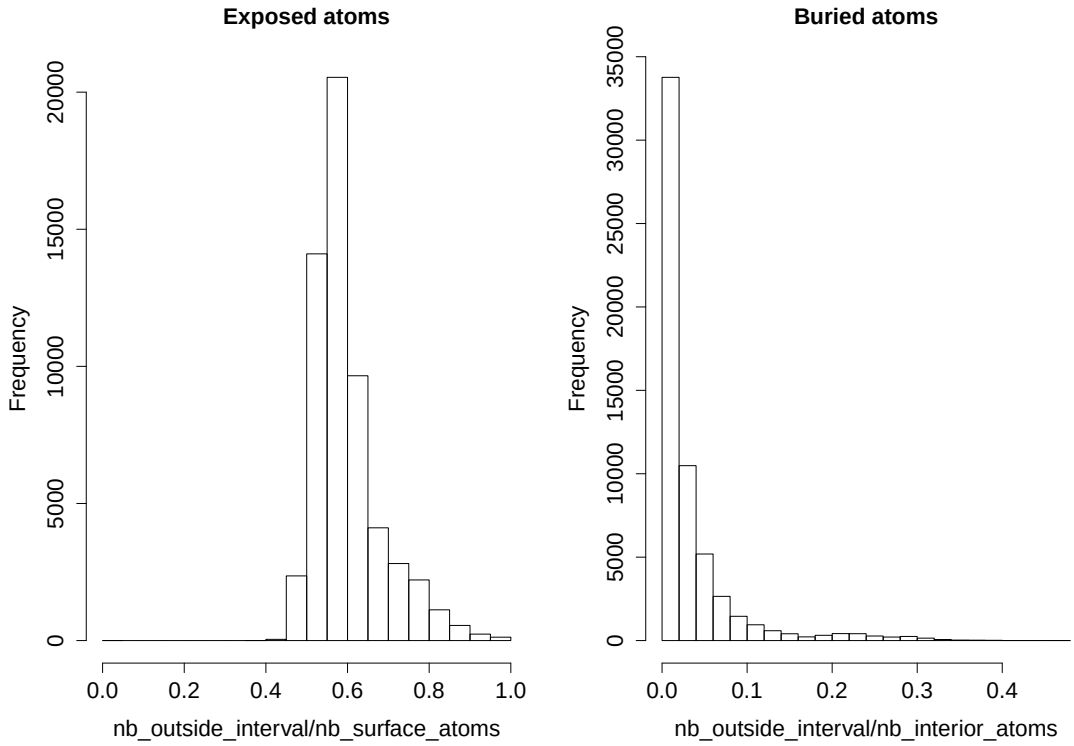

Figure 10: Distribution of the fraction (computed on a per-file basis) of atoms whose volume reported by `McC-et-al` is outside the range reported by `Vorlume`. Left: exposed atoms; right: buried atoms.

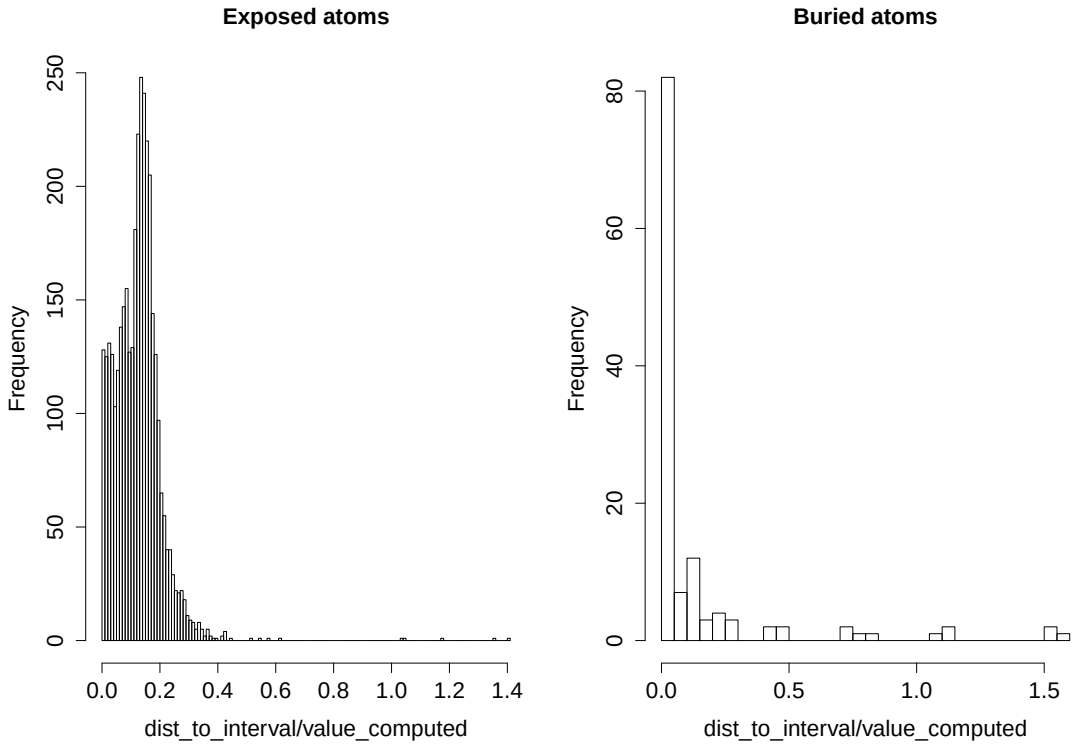

Figure 11: PDB file 3fob. Distribution of the normalized errors, encoding the discrepancy between V_{MCC} and V_i for all atoms of a class (exposed, buried). See text for details.

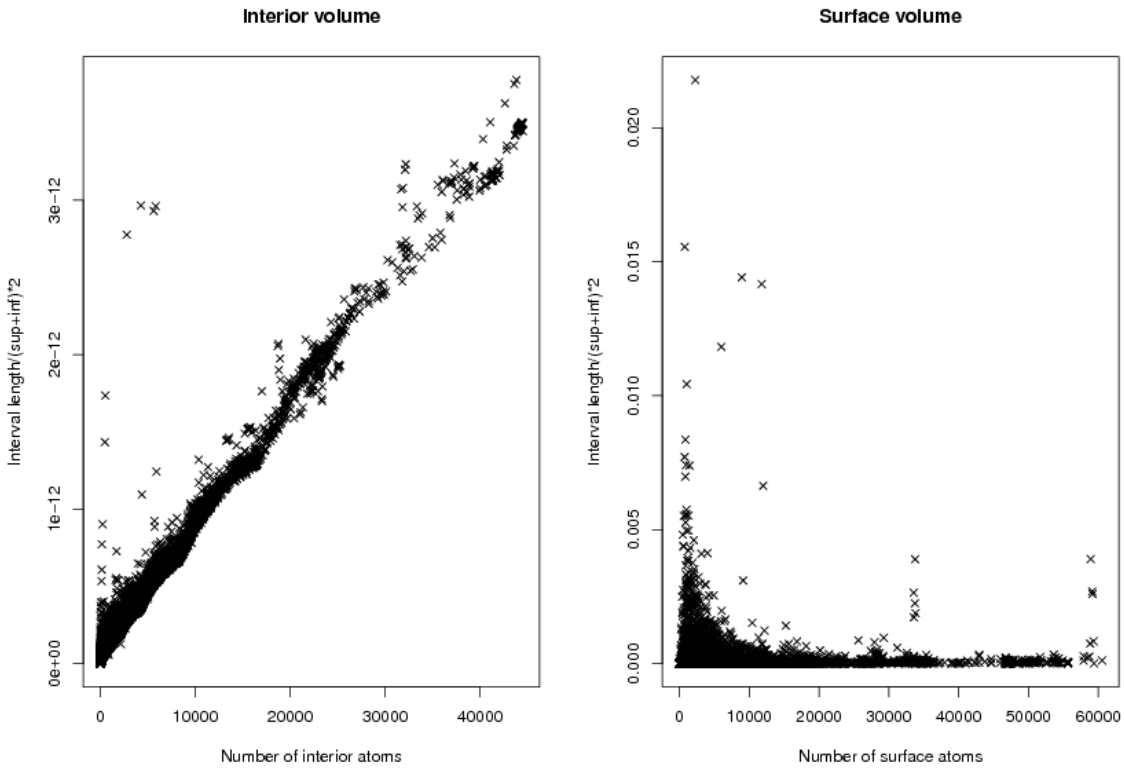

Figure 12: Precision of the volumes computed for the 58,898 models in the PDB. Values depicted are the interval lengths of total (interior or surface) volumes divided by the interval midpoint value.

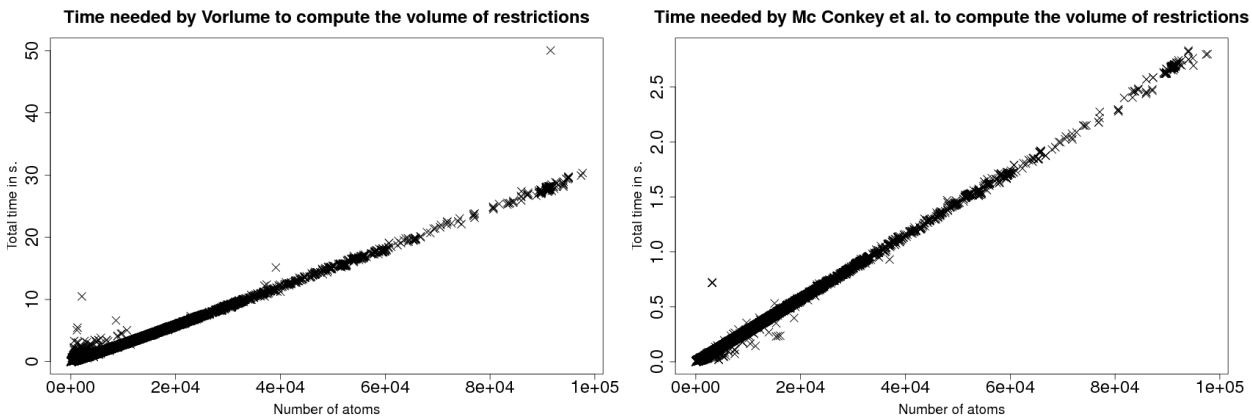

Figure 13: Total time required to compute the volume of the restriction of each balls to its cell, for the 58,898 models in the PDB. On the left our software, on the right the implementation of Mc Conkey et al. [18]. A factor of 10 is observed between our software (robust and providing guarantees on the value of the volume) and the floating point version.

pdbcode	#atoms	p	Volume of Exposed atoms	Volume of Buried atoms	*time*
1vow	91,013	1	1,754,197.19 ± 3,380.96	383416.36 ± 6.12 10^{-07}	27.44
		2	1,753,461.16 ± 0.00139	383416.36 ± 6.12 10^{-07}	43.32
		3	1,753,461.16 ± 0.00139	383416.36 ± 6.12 10^{-07}	64.54
1tnv	2,328	1	73,485.44 ± 704.16	643.95 2.22 ± 10^{-11}	0.70
		2	73,204.51 ± 1.38 10^{-5}	643.95 2.24 ± 10^{-11}	1.38
		3	73,204.51 ± 1.38 10^{-5}	643.95 1.81 ± 10^{-11}	1.64

Table 3: Trade off between precision and computation time in seconds. See text for details.

6 Conclusion

This paper presents the first algorithm computing the volume of a union of balls in a certified fashion. The algorithm reads directly from a proof of the volume calculation derived from Gauss' divergence theorem. The certification stems from two facts. First, the so-called predicates which are involved in the decision tree of the algorithm are evaluated exactly. Second, numerical primitives involved in the volume calculation are evaluated resorting to interval arithmetic. This includes the construction of points with algebraic coordinates, the construction of points with rational coordinates, and the evaluation of inverse trigonometric functions. Overall, an interval containing the exact volume is returned.

From a practical standpoint, our C++ implementation is generic, and using interval arithmetic as a building block results in running times which incur a mere factor of 10 w.r.t. a non-robust calculation based on fixed precision floating point arithmetic.

While previous work on this problem by pioneers resulted in non-robust programs, we believe that our software, made available to the community, should be instrumental to derive precise volume estimates in a number of applications. We also believe that our analysis should help to settle equivalent problems.

Acknowledgements. B.J. Mc Conkey is acknowledged for making his implementation available.

References

[1] P. Agarwal, L. Guibas, A. Nguyen, D. Russel, and L. Zhang. Collision detection for deforming necklaces. *Computational Geometry: Theory and Applications*, 28:137–163, 2004.

[2] N. Akkiraju and H. Edelsbrunner. Triangulating the surface of a molecule. *Discrete Applied Mathematics*, 71(1):5–22, 1996.

[3] N. Amenta, S. Choi, and R. Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry : Theory and Applications*, 19(2):127–153, 2001.

[4] F. Aurenhammer. Power diagrams: properties, algorithms and applications. *SIAM Journal on Computing*, 16:78–96, 1987.

[5] D. Avis, B. K. Bhattacharya, and H. Imai. Computing the volume of the union of spheres. *The Visual Computer*, 3:323–328, 1988.

[6] B. Bouvier, R. Grunberg, M. Nilges, and F. Cazals. Shelling the voronoi interface of protein-protein complexes reveals patterns of residue conservation, dynamics and composition. *Proteins: structure, function, and bioinformatics*, 76(3):677–692, 2009.

[7] H. Brönnimann, A. Fabri, G.-J. Giezeman, S. Hert, M. Hoffmann, L. Kettner, S. Schirra, and S. Pion. 2D and 3D geometry kernel. In C. E. Board, editor, *CGAL User and Reference Manual*. 3.4 edition, 2008.

[8] F. Cazals, F. Chazal, and T. Lewiner. Molecular shape analysis based upon the morse-smale complex and the connolly function. *Proc. 19th Annual Symposium on Computational Geometry*, pages 351–360, 2003.

[9] F. Cazals and S. Loriot. Computing the exact arrangement of circles on a sphere, with applications in structural biology. *Computational Geometry: Theory and Applications*, 42:551–565, 2009.

[10] F. Chazal, D. Cohen-Steiner, and Q. Mérigot. Stability of boundary measure. Research Report 6219, INRIA, 2007. `http://hal.inria.fr/inria-00154798`.

[11] P. M. M. de Castro and M. Teillaud. 3D spherical geometry kernel. In C. E. Board, editor, *CGAL User and Reference Manual*. 3.4 edition, 2008.

[12] M. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall Englewood Cliffs, 1976.

[13] H. Edelsbrunner. Weighted alpha shapes. Technical Report UIUCDCS-R-92-1760, Dept. Comput. Sci., Univ. Illinois, Urbana, IL, 1992.

[14] H. Edelsbrunner. The union of balls and its dual shape. *Discrete and Computational Geometry*, 13:415–440, 1995.

[15] A. Fabri and S. Pion. A generic lazy evaluation scheme for exact geometric computations. In *Proc. 2nd Library-Centric Software Design*, 2006.

[16] M. Gerstein and F. Richards. Protein geometry: volumes, areas, and distances. In M. G. Rossmann and E. Arnold, editors, *The International Tables for Crystallography (Vol F, Chap. 22)*, pages 531–539. Springer, 2001. Updated version of the 1977 paper.

[17] S. Loriot, S. Sachdeva, K. Bastard, C. Prevost, and F. Cazals. On the characterization and selection of diverse conformational ensembles. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2009. To appear. INRIA Research Report 6503. `http://hal.inria.fr/inria-00252046`.

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

[19] K. Rother, P. Hildebrand, A. Goede, B. Gruening, and R. Preissner. Voronoia: analyzing packing in protein structures. *Nucleic Acids Research*, 37(Database issue):D3933, 2009.

[20] S. Rusinkiewicz and M. Levoy. QSplat: a multiresolution point rendering system for large meshes. *Proc. 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 343–352, 2000.

[21] M. Till and G. Ullmann. McVol-A program for calculating protein volumes and identifying cavities by a Monte Carlo algorithm. *Journal of Molecular Modeling*, 2009. Online First.

[22] J. Tsai, R. Taylor, C. Chothia, and M. Gerstein. The packing density in proteins: standard radii and volumes. *Journal of Molecular Biology*, 290(1):253–266, 1999.

7 Exact determination of `Orient`

Orientation predicate involving constructions. Denoting (p_x, p_y, p_z) the coordinates of input point p, the orientation predicate of four points `Orient(p,q,r,s)` consists of evaluating the sign of the following 4×4 determinant:

$$\texttt{Orient(p,q,r,s)} = Sign\left(\begin{vmatrix} 1 & p_x & p_y & p_z \\ 1 & q_x & q_y & q_z \\ 1 & r_x & r_y & r_z \\ 1 & s_x & s_y & s_z \end{vmatrix}\right) \tag{19}$$

Or equivalently, using

$$D = \begin{vmatrix} p_x - r_x & q_x - r_x & s_x - r_x \\ p_y - r_y & q_y - r_y & s_y - r_y \\ p_z - r_z & q_z - r_z & s_z - r_z \end{vmatrix}, \tag{20}$$

one has:

$$\texttt{Orient(p,q,r,s)} = Sign\,(D)\,. \tag{21}$$

In our case, as seen from Eq. (14), points p and q are intermediate constructions, while r and s are input points. Recall that an intermediate construction is either a weighted circumcenter or a point found on $\partial \bigcup B_i$—called boundary point for short in the sequel. In the former case, its coordinates are rational numbers. In the latter, its coordinates are algebraic numbers of degree two in the same extension. Each such coordinate can thus be written as $a + b\sqrt{c}$, with a, b, c three rational numbers. Denoting $Sign(x - y)$ as $Compare(x, y)$, the sign of such a number is $Sign(a)$ if $Sign(a) = Sign(b)$ and $Sign(a)Compare(a^2, -b^2c)$ otherwise [13].

Since we assume that points r and s have rational coordinates, determining the sign of D faces three cases: (i) p and q are weighted circumcenters; (ii) amongst p and q, one is a weighted circumcenter and the other one is a boundary point, and (iii) both p and q are boundary points.

Case (i). Since all points have rational coordinates, the sign of D can be directly computed using a rational number type.
Case (ii). Number D is an algebraic number of degree two, and its sign can be evaluated as explained above.
Case (iii). The first and second column of D involve algebraic numbers in the same extension, but these two extensions are different. Denoting γ_1 and γ_2 for these extensions, D can be written as

$$D = a + b\sqrt{\gamma_1} + c\sqrt{\gamma_2} + d\sqrt{\gamma_1\gamma_2} \tag{22}$$

Proceeding as above for the sign evaluation, comparing two algebraic numbers in different extensions, that is $Compare(a_1 + b_1\sqrt(\gamma_1), a_2 + b_2\sqrt(\gamma_2))$, is an easy task. Using this primitive, the evaluation of the sign of D consists in reducing it to a comparison between two algebraic numbers of degree two, which requires a case study involving the signs of a, b, c, d. This case-study is presented in the sequel. Note that due to the symmetric role played by b and c, we do not list induced symmetric cases. For now assume $Sign(a) \neq 0$, $Sign(b) \neq 0$, $Sign(c) \neq 0$ and $Sign(d) \neq 0$

Four identical signs.

- if $Sign(a) = Sign(b) = Sign(c) = Sign(d)$ then $Sign(D) = Sign(a)$.

Three identical signs.

- if $Sign(a) = Sign(b) = Sign(c)$ and $Sign(a) \neq Sign(d)$ then:

$$Sign(D) = Compare(d\sqrt{\gamma_1\gamma_2}, -a - b\sqrt{\gamma_1} - c\sqrt{\gamma_2})$$

$$\Longleftrightarrow$$

$$Sign(D) = Sign(d)Compare(d^2\gamma_1\gamma_2, a^2 + b^2\gamma_1 + c^2\gamma_2 + 2ab\sqrt{\gamma_1} + 2ac\sqrt{\gamma_2} + 2bc\sqrt{\gamma_1\gamma_2})$$

$$\Longleftrightarrow$$

$$Sign(d)Compare(e_1, e_2)$$

with $e_1 = d^2\gamma_1\gamma_2 - a^2 - b^2\gamma_1 - c^2\gamma_2 - 2bc\sqrt{\gamma_1\gamma_2}$ and $e_2 = 2ab\sqrt{\gamma_1} + 2ac\sqrt{\gamma_2}$.

[13] cases $a = 0$ or $b = 0$ are even more trivial.

If $Sign(e_1) \neq Sign(e_2)$ then:

$$Sign(D) = Sign(d)Sign(e_1)$$

otherwise:

$$Sign(D) = Sign(d)Sign(e_1)Compare(e_1^2, e_2^2).$$

- if $Sign(a) = Sign(c) = Sign(d)$ and $Sign(a) \neq Sign(b)$ then:

$$Sign(D) = Compare(b\sqrt{\gamma_1}, -a - c\sqrt{\gamma_2} - d\sqrt{\gamma_1\gamma_2})$$

$$\Longleftrightarrow$$

$$Sign(D) = Sign(b)Compare(b^2\gamma_1, a^2 + c^2\gamma_2 + d^2\gamma_1\gamma_2 + 2ac\sqrt{\gamma_2} + 2ad\sqrt{\gamma_1\gamma_2} + 2cd\gamma_2\sqrt{\gamma_1})$$

$$\Longleftrightarrow$$

$$Sign(D) = Sign(b)Compare(e_1, e_2)$$

with $e_1 = b^2\gamma_1 - a^2 - c^2\gamma_2 - d^2\gamma_1\gamma_2 - 2ad\sqrt{\gamma_1\gamma_2}$ and $e_2 = 2ac\sqrt{\gamma_2} + 2cd\gamma_2\sqrt{\gamma_1}$.
If $Sign(e_1) \neq Sign(e_2)$ then:

$$Sign(D) = Sign(b)Sign(e_1)$$

otherwise:

$$Sign(D) = Sign(b)Sign(e_1)Compare(e_1^2, e_2^2).$$

- if $Sign(b) = Sign(c) = Sign(d)$ and $Sign(a) \neq Sign(b)$ then:

$$Sign(D) = Compare(a, -b\sqrt{\gamma_1} - c\sqrt{\gamma_2} - d\sqrt{\gamma_1\gamma_2})$$

$$\Longleftrightarrow$$

$$Sign(D) = Sign(a)Compare(a^2, b^2\gamma_1 + c^2\gamma_2 + d^2\gamma_1\gamma_2 + 2bc\sqrt{\gamma_1\gamma_2} + 2bd\gamma_1\sqrt{\gamma_2} + 2cd\gamma_2\sqrt{\gamma_1})$$

$$\Longleftrightarrow$$

$$Sign(D) = Sign(a)Compare(e_1, e_2)$$

with $e_1 = a^2 - b^2\gamma_1 - c^2\gamma_2 - d^2\gamma_1\gamma_2 - 2bc\sqrt{\gamma_1\gamma_2}$ and $e_2 = 2bd\gamma_1\sqrt{\gamma_2} + 2cd\gamma_2\sqrt{\gamma_1}$.
If $Sign(e_1) \neq Sign(e_2)$ then:

$$Sign(D) = Sign(a)Sign(e_1)$$

otherwise:

$$Sign(D) = Sign(a)Sign(e_1)Compare(e_1^2, e_2^2).$$

Two identical signs.

- if $Sign(a) = Sign(b)$ and $Sign(c) = Sign(d)$ and $Sign(a) \neq Sign(c)$ then:

$$Sign(D) = Compare(a + b\sqrt{\gamma_1}, -c\sqrt{\gamma_2} - d\sqrt{\gamma_1\gamma_2})$$

$$\Longleftrightarrow$$

$$Sign(D) = Sign(a)Compare(a^2 + b^2\gamma_1 + 2ab\sqrt{\gamma_1}, c^2\gamma_2 + d^2\gamma_1\gamma_2 + 2cd\gamma_2\sqrt{\gamma_1})$$

- if $Sign(a) = Sign(d)$ and $Sign(b) = Sign(c)$ and $Sign(a) \neq Sign(b)$ then:

$$Sign(D) = Compare(a + d\sqrt{\gamma_1\gamma_2}, -b\sqrt{\gamma_1} - c\sqrt{\gamma_2})$$

$$\Longleftrightarrow$$

$$Sign(D) = Sign(a)Compare(a^2 + d^2\gamma_1\gamma_2 + 2ad\sqrt{\gamma_1\gamma_2}, b^2\gamma_1 + c^2\gamma_2 + 2bc\sqrt{\gamma_1\gamma_2})$$

Finally, if amongst the signs of a, b, c, d one or more are zero, then if the sign of D can not be directly deduced, we can resort to the cases enumerated in **Two identical signs**, considering the sign of the zero as positive or negative to match the sign criterion. Note that in cases enumerated in **Three identical signs**, since a, b, c, d are different from zero, e_1 and e_2 are always different from zero.

Contents

1 Introduction **3**
1.1 Modeling with Balls . . . 3
1.2 Computing the Volume of a Collection of Balls: Previous Work on Exact Methods . . . 3
1.3 Contribution and Paper Overview . . . 3

2 Volume Calculation: Theorem **4**
2.1 Geometry of a Restriction . . . 4
2.2 Statement of the Theorem . . . 6

3 Algorithm **7**
3.1 Computing the Boundary of Restrictions . . . 8
3.2 Computing the Volumes of Pyramids . . . 8
3.2.1 Pyramids with Planar Base . . . 8
3.2.2 Pyramid with Spherical Base . . . 10

4 Robust Implementation **11**

5 Experiments **12**
5.1 Setup . . . 12
5.2 Results . . . 13

6 Conclusion **18**

7 Exact determination of `Orient` **20**

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399