A Perturbation Scheme for Spherical Arrangements with Application to Molecular Modeling*

Dan Halperin† — Tel Aviv University

Christian R. Shelton‡ — Massachusetts Institute of Technology

December 14, 1996

Abstract

We describe a software package for computing and manipulating the subdivision of a sphere by a collection of (not necessarily great) circles and for computing the boundary surface of the union of spheres. We present problems that arise in the implementation of the software and the solutions that we have found for them. At the core of the paper is a novel perturbation scheme to overcome degeneracies and precision problems in computing spherical arrangements while using floating point arithmetic. The scheme is relatively simple, it balances between the efficiency of computation and the magnitude of the perturbation, and it performs well in practice. We report and discuss experimental results. Our package is a major component in a larger package aimed to support geometric queries on molecular models; it is currently employed by chemists working in 'rational drug design'. The spherical subdivisions are used to construct a geometric model of a molecule where each sphere represents an atom. We also give an overview of the molecular modeling package and detail additional features and implementation issues.

*This work has been supported in part by a grant from Pfizer Central Research. Dan Halperin has also been supported by an Alon Fellowship, and by ESPRIT IV LTR Project No. 21957 (CGAL).

†Department of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel. E-mail: danha@math.tau.ac.il.

‡Department of Computer Science, MIT, Cambridge, MA 02139. E-mail: cshelton@ai.mit.edu. Part of the work on this paper was carried out while Christian Shelton was at the Department of Computer Science, Stanford University

1 Introduction

Implementing geometric algorithms can be a difficult task. It has been found out and repeatedly rediscovered that there is a huge gap between geometric algorithms as they are described in most theoretical papers and their implementation in software. Two issues that are often ignored in the theoretical approach turn out to be critical in practice: Degeneracies and numerical precision. These issues are collectively referred to as "robustness". Surveys on the topic can be found in [7],[14],[16].

In theory degeneracies are often handled by assuming *general position*, namely assuming that degeneracies do not occur. Insolent as this may sound, the general position assumption had contributed significantly to the advancement of geometric algorithms by letting the researchers focus on the key (theoretical) problems while ignoring many technical issues. When implementing a geometric algorithm however, one must take degeneracies into consideration.

The numerical precision problem was solved in the theory of geometric algorithm by assuming infinite precision real arithmetic [17]. For certain algorithms and geometric objects this assumption is realizable in practice by using *exact arithmetic* [5],[8],[19],[23]. Computing with exact arithmetic is in general more costly than using floating point arithmetic, and in certain cases not realizable because of the geometric primitives that need to manipulated. Here again there is a gap between what could in theory be handled by exact arithmetic and what current technology offers [7].

The software package that we describe in this paper computes the boundary of a union of spheres, the surface area of the boundary, and the intersection pattern of any sphere with all the other spheres in a given set. We therefore have to compute the intersection of pairs and triples of spheres, various tangency points of great circles and *little circles*[1] on a sphere, and so on. Such computations would be difficult to carry out using exact arithmetic (all these operations require the solution of polynomial equations, so theoretically symbolic schemes could possibly be used here). Also floating point arithmetic has the obvious advantages of availability and efficiency. Our goal here is to devise robust algorithms that deal with intersecting spheres in $\mathbb{R}^3$ while using floating point arithmetic. Some examples of previous and related work on robust floating point geometric algorithms can be found in [10],[16],[20], and [21].

Our motivating application is geometric modeling of molecules. Our software package is part of a toolbox aimed to support the chemist in the drug design process [6]. The basic geometric model of a molecule that we use is the so-called *hard sphere model* where every atom is represented by a sphere at some fixed position relative to the other atom spheres in the molecule. Since the hard sphere model is an approximate model to begin with, we have the freedom to perturb the spheres slightly without much effect on the relevance of the model.

When computing with floating point arithmetic we cannot precisely determine degeneracies; we can only know that we are in a potentially degenerate situation. For example, we may not be able to determine for sure that four spheres in our collection meet at a single point, but we could detect that the point of intersection of three out of the four spheres is less than some $\varepsilon > 0$ away from a point of intersection of a different triple of these spheres. For a small parameter $\varepsilon > 0$ which we call the *resolution parameter*, we regard such a pair of intersection points as a *degeneracy*. (A "potential degeneracy" may be a more appropriate term, but for brevity we will refer to such a situation as degeneracy.) Our new scheme guarantees that for a given parameter $\varepsilon > 0$, all the features of the spherical arrangement are at least ε apart. The resolution parameter ε depends on the floating point precision and on the type of operations (e.g., computing the intersection points of three spheres). We assume here that ε is given. There are numerical analysis methods to compute

[1]Throughout the paper we use the term *little circle* to mean any circle on a sphere S that is the intersection of S with another sphere.

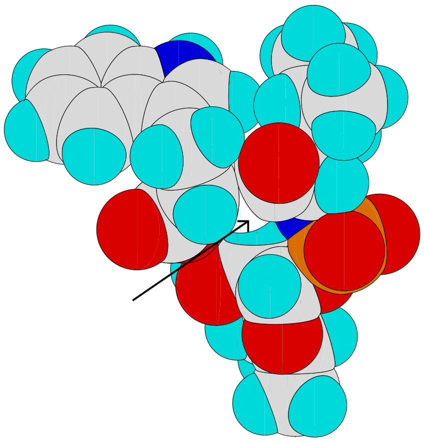


Figure 1: The sphere model of a molecule. The arrow points to the sphere for which the spherical arrangement is drawn in Figure 2(a).

useful bounds on ε (see [13, Chapter 4] for examples concerning linear objects). We point out that in our algorithms the 'depth of operations' (namely how many times in a row the result of one operation is the operand in another operation) is bounded by a small constant. Therefore one can obtain a bound on the resolution parameter that does not depend on the input size n—the number of spheres, in our case.

We present an efficient perturbation scheme for a collection M of n spheres in $\mathbb{R}^3$ that makes our geometric algorithms robust. We call the decomposition of $\mathbb{R}^3$ induced by the spheres the *arrangement of the spheres*; the subdivision of each sphere by the circles of intersection with other spheres we call a *spherical arrangement*. We denote the maximum number of spheres in M intersecting any single sphere in M by k (as was shown in [12], k is a constant for the hard sphere model of molecules; see also the next section). For any given resolution value $\varepsilon > 0$, we determine a parameter δ that depends on ε, on k, and on the maximum radius R of a sphere in M. We then present a scheme that perturbs each sphere by at most δ, resolves all the degeneracies in the arrangement of the spheres, and runs in $O(n)$ time.

We also take care of degeneracies that result from a refinement of the spherical arrangements known as the *trapezoidal decomposition* (see Section 2.1). Since in the trapezoidal decomposition we are free to choose a direction for the 'poles' (two antipodal points on a sphere, such that all the added arcs in the refinement are portions of great circles through these poles), we choose the poles so that the angular separation of the added arcs will be above a certain threshold ω. Here also, we choose ω such that determining the poles could be done fast.

We have implemented this perturbation scheme and we report experimental results below. For example, letting $\varepsilon = 10^{-11}$, $\delta = 10^{-9}$, and $\omega = \arccos(1 - 10^{-11})$, loading a molecule with 2034 atoms (namely computing the boundary surface and all the individual spherical arrangements) takes 146.8 seconds (including the perturbation) on an SGI Indigo 250Mhz RS4400. Our experimental results show that in most cases the time taken up by the perturbation scheme is negligible (Section 6).

We point out that our software is part of a toolbox of data structures and algorithms aimed to support the chemist in the *rational drug design* process [1], and it is currently being used by chemists

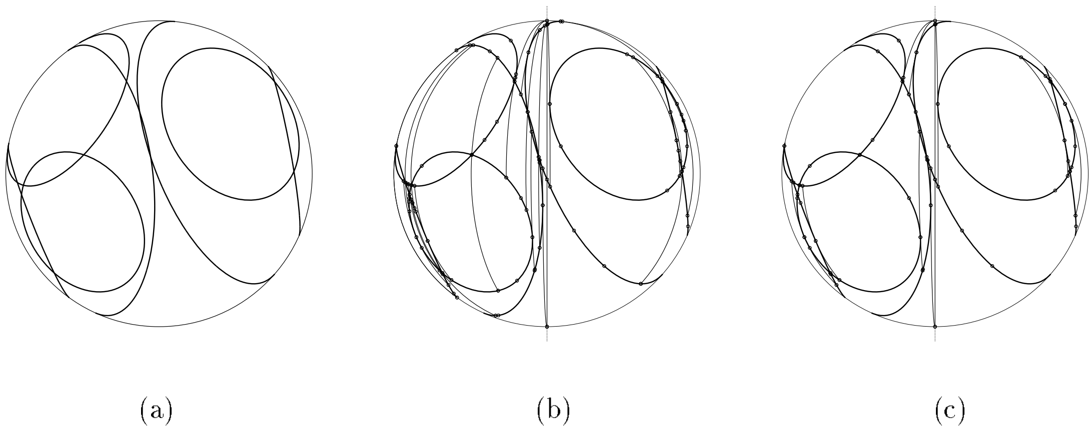


Figure 2: A spherical arrangement (a) its full trapezoidal decomposition (b) and partial trapezoidal decomposition (c).

in a pharmaceutical company. Further details on the larger software (the toolbox) can be found in [6]. Ours is definitely not the first implementation of an algorithm for computing molecular surfaces (see, e.g., [2],[18],[22]) and the recent survey by Connolly [3]). We believe that our approach stands out in its treatment of robustness issues and in the ease and flexibility of computing substructures of the collection of the spheres, such as the union of a subset of the spheres of a molecule consumed by another molecule.

The rest of the paper is organized as follows. In the next section we give more background details on spherical arrangements and on the hard sphere model of a molecule. In Section 3 we review our software package. The key ideas underlying our new perturbation scheme are presented in Section 4. Algorithmic details of the scheme and running time analysis are given in Section 5. In Section 6 we report experimental results. More practical issues concerning our implementation are discussed in Section 7. A brief summary and suggestions for future research on the topic are given in Section 8. In the Appendix we complete technical details concerning shapes and volumes that arise in the perturbation scheme.

2 Preliminaries: Spheres and Molecules

2.1 Little Circles on a Sphere

A collection of circles on a sphere S induces a partitioning of the sphere into vertices, edges and faces. We call such a partitioning a *spherical arrangement*; see Figure 2(a). If all the circles are great circles, then one can easily transform the spherical arrangement into a *planar arrangement of lines* [4], which is a simpler object to handle [9]. However, in our application the circles are not necessarily great circles. Each of the circles is the result of the intersection of S with another sphere. We refer to these more general circles as *little circles*.

The faces in the spherical arrangement need not be simply connected, and each face may have a large number of edges on its boundary. We can apply a standard refinement procedure, called a *trapezoidal decomposition* that will make each face homeomorphic to a disc and have at most four edges on its boundary. See Figure 2(b). We fix a pair of antipodal points as *poles*. We call the great circles through the poles *polar circles*, arcs of polar circles *polar arcs*, and any point on a little circle that is tangent to a polar circle we call a *polar tangency*. For every polar tangency of every circle in our collection, we extend a polar arc in either direction until it hits another little circle or reaches a pole. We do the same from every intersection point of a pair of little circles. We call this refinement the *full trapezoidal decomposition*.

If we are only concerned with making each face simply connected and making the graph of all the edges of the arrangement connected then using a *partial trapezoidal decomposition*, in which polar arcs are only extended from polar tangency points and not from intersections, suffices. See Figure 2(c).

2.2 The Hard Sphere Model

A common approach to representing the three-dimensional geometric structure of a molecule, is to represent each of its atoms by a "hard" sphere. In certain applications it is also assumed that the relative displacement of the spheres is fixed. There are recommended values for the radius of each atom sphere and for the distance between the centers of every pair of spheres. In this model, the spheres are allowed to interpenetrate one another, therefore it is sometimes referred to as the "fused spheres" model (see Figure 1). The envelope surface of the fused spheres may be regarded as a formal *molecular surface*. It is evident that various properties of molecules are disregarded in this simple model. However, in spite of its approximate nature, it has proven useful in many practical applications. For more background material and references, see for example the survey paper by Mezey [15].

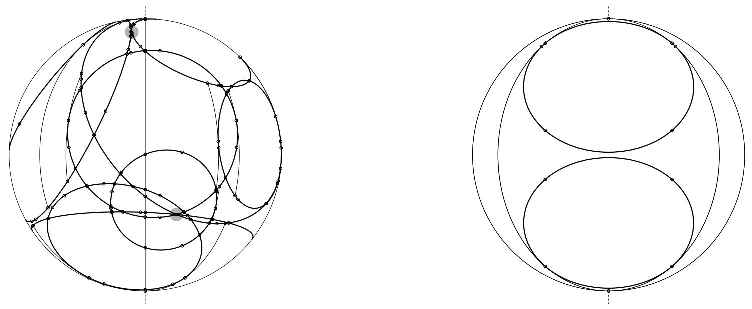
In [12] the hard sphere model is studied from a *computational geometry* point of view. Several observations are made in that paper showing that, because of certain special properties, the spheres in this model can be efficiently manipulated. We cite below the results that will be needed in later sections. Theorem 2.1, states the conditions that make the sphere model of a molecule favorable. Theorem 2.2 summarizes a hash-table based data structure that is constructed exploiting these conditions.

Theorem 2.1 [12] *Let $M = \{B_1, \ldots, B_n\}$ be a collection of n balls in 3-space with radii $r_1, \ldots, r_n$ and centers at $c_1, \ldots, c_n$. Let $r = \min_i r_i$ and let $R = \max_i r_i$. Also let $S = \{S_1, \ldots, S_n\}$ be the collection of spheres such that S_i is the boundary surface of B_i. If there are positive constants k, ρ such that $\frac{R}{r} < k$ and for each B_i the ball with radius $\rho \cdot r_i$ and concentric with B_i does not contain the center of any other ball in M (besides c_i), then:* **(i)** *for each $B_i \in M$, the maximum number of balls in M that intersect it is bounded by a constant, and* **(ii)** *The maximum combinatorial complexity of the boundary of the union of the balls in M is $O(n)$.*

Theorem 2.2 [12] *Given a collection M of n balls as defined in Theorem 2.1, one can construct a data structure using $O(n)$ space, to answer intersection queries for balls whose radii are not greater than $r_{\max}$, in $O(1)$ time. The expected preprocessing time of the structure is $O(n)$.*

3 Overview of the Software Package

Our software package is divided into two main parts: the core computation engine and the interface. The computation engine consists of the code necessary to compute the surface area of any arbitrary subset of the spheres, to answer intersection queries, and to create decompositions of the spheres. A number of different interfaces have been constructed. Two primary ones are currently being used. One is a batch processor which reads in a list of molecules (or a list of sets of molecules) to process and outputs a surface file for each molecule (or set) consisting of general statistics such as name and number of atoms, the surface area and void areas, and the surface area of one set of atoms in one molecule consumed by another set of atoms in another molecule (for multiple molecule sets).



type I type II

Figure 3: Examples of the two types of degeneracies. Type I degeneracies are marked by small shaded circles.

The other interface is a graphical interface. It displays the general statistics of the molecule along with boundary surface and void area computations. In a separate window, it displays a rendering of a three-dimensional model of the spheres which can be rotated, translated, and scaled. This model can further be colored and shaded per atom by various calculations from the computational engine such as surface area, surface percentage, element type, molecule, and number of intersecting atoms. This interface also allows the user to select any atoms in the molecule. The selected atoms are displayed separately with a full view of the spherical arrangement on the atom (we also call it the *atom map*) and the exact statistics on the atom's contribution in terms of surface area to the molecule's outer boundary and inner voids.

This paper will detail the computational engine of the package. In particular, we will concentrate on robust computation of the spherical decomposition. Higher level data structures for keeping track of atoms within multiple molecules and lower level code for computing intersections and keeping track of the geometry of points and arc segments will be assumed for our discussion.

For representing the atom map, we chose a variant of the quad-edge data structure [11]. This structure makes updating the subdivision simple. Each arc is stored as a plane and two points (thus the arc is the intersection of the plane and the sphere between the two points). Four operations are needed: adding an arc, removing an arc, splitting an arc, and merging two connected arcs (namely two arcs that belong to and are adjacent along a single circle). The maintenance of this data structure is fairly standard and we omit the details here.

The data structure representing the spherical arrangement assumes that the graph of arcs is connected. This means that all of the regions must be simply connected. To insure this, we break up the surface with additional arcs in order to create a trapezoidal decomposition, as described in Section 2.1.

We have presented so far the ingredients that are necessary to understanding the perturbation scheme. Further details on our software are given below in Section 7.

4 The Perturbation Scheme

We distinguish two types of degeneracies that arise in a collection of intersecting spheres and atom maps as we compute them. Recall that we are concerned with floating point arithmetic and that we define a degeneracy using a resolution parameter $\varepsilon > 0$. In this section we give a precise definition of the various degeneracies. We describe the two types as they occur on a single sphere. See Figure 3.

Type I Two intersection points each being the intersection of three spheres are close together (see below for numerical values), two little circles are tangent (or almost tangent), or a little circle is too small. Had we been using exact arithmetic these (potential) degeneracies would correspond respectively to the following (exact) degeneracies in the three-dimensional arrangement of spheres: Four spheres intersect in a point, three spheres intersect in a single point, or two spheres are tangent to one another.

Type II The angle between the planes containing two distinct polar arcs is too small (it is below some threshold ω).

Type I is inherent to the arrangement of spheres, whereas type II is an artifact of our decomposition method.

We wish to perturb the spheres slightly so that all the features of the arrangement will be at least ε apart, for a given resolution parameter ε. We would like the perturbation procedure to be efficient and at the same time that the perturbation magnitude will be as small as possible. For a given ε our procedure will determine a perturbation value δ that will guarantee that the procedure will take $O(n)$ time for n spheres.

We use a two-step perturbation scheme. At the first step we remove type I degeneracies by an incremental procedure where we add the spheres one by one and if a degeneracy occurs we only perturb the last sphere that has been added. We then choose the orientation of the poles (namely the direction to which all the planes containing polar arcs will be parallel) that will eliminate degeneracies of type II.

4.1 Removing Type I Degeneracies

Let $S_1, S_2, \ldots, S_n$ be an ordering of the spheres in M. For a pair of spheres S_i and S_j let ρ_{ij} denote the distance between their centers. Let M_t denote the set $\cup_{i=1}^{t}\{S_i\}$. After the completion of stage t, the incremental procedure will maintain the following invariants:

I_1 The center C_i of any sphere in M has been moved by at most δ from its original placement (δ is a constant to be determined below).

I_2 For every pair of spheres S_i and S_j, $i \neq j \leq t$, and $r_j \geq r_i$, we have $|\rho_{ij} - r_i - r_j| > \varepsilon$ and $|\rho_{ij} + r_i - r_j| > \varepsilon$.

I_3 For every triple of spheres S_i, S_j and S_k, $i \neq j \neq k \leq t$, the circle $C_{ij} := S_i \cap S_j$ and the sphere S_k are not tangent, and are at least ε away from being tangent. (The formal definition of this invariant is given in the Appendix.)

I_4 Let τ_i and τ_j each be an intersection point of three spheres in M_t; then $d(\tau_i, \tau_j) > \varepsilon$.

We call a perturbation scheme that satisfies the above invariants at the end of each stage (and in particular after the nth stage) a *valid perturbation scheme*.

Suppose the procedure has been carried out successfully for the first t stages. We next describe how we add the sphere S_{t+1} so that M_{t+1} will maintain the invariants above. We denote the center of sphere S_i for $i \leq t$ after the completion of step t by C'_i. Let $B(C_{t+1}, \delta)$ be the ball of radius δ around the original placement of the center C_{t+1} of S_{t+1}. We will place the center of S_{t+1} inside $B(C_{t+1}, \delta)$ and this will guarantee the invariant I_1. The invariants I_2, I_3 and I_4 define forbidden loci F_2, F_3 and F_4 respectively for the center of S_{t+1}. We will choose the new placement C'_{t+1} of the center of S_{t+1} to be in $G_{t+1} := B(C_{t+1}, \delta) \setminus (F_2 \cup F_3 \cup F_4)$. If such a placement exists, we call it a *valid placement* of the center of S_{t+1}.

The region F_2 is the union of *spherical shells*[2] of two types. Each sphere $S_i \in M_t$ induces two spherical shells according to whether the tangency is external (namely, at the tangency point the interior of the spheres are disjoint) or internal. One shell has its center in C_i' and its radii are $r_i + r_t - \varepsilon$ and $r_i + r_t + \varepsilon$. This spherical shell represents the loci of placement of the center of S_t that result in the spheres S_i and S_t being tangent, or almost tangent. The other type of shells is defined similarly.

We postpone a detailed explanation of the shape of the objects contributing to F_2, F_3, and F_4 to the Appendix. We just mention that F_4 is also the union of spherical shells, whereas F_3 is the union of *toric shells*. In the Appendix we derive upper bounds on the volume taken up by the regions F_2, F_3, and F_4. Let $VF := \text{Volume}(F_2 \cup F_3 \cup F_4)$ and let $VB := \text{Volume}(B(C_{t+1}, \delta))$.

Our goal is to choose δ such that $VB > 2 \cdot VF$. If we do that, we are guaranteed that a point chosen uniformly at random inside $B(C_{t+1}, \delta)$ has probability $> \frac{1}{2}$ to be a valid placement for the center of S_{t+1}. Our calculations show that we need to choose $\delta = f(k, \varepsilon, R) = 2k\varepsilon^{1/3}R^{2/3}$, where R is the maximum radius of a sphere in M, and assuming $k \geq 10$. Not surprisingly, our experimental results show that this is a very conservative bound. The theoretical bound is a crude worst-case bound, and even as such it shows that our approach does not conceal any very large constant.

4.2 Removing Type II Degeneracies

After we have removed all type I degeneracies, we wish to choose a direction for the poles such that no type II degeneracy arises, when using the **partial** vertical decomposition. (As we remark below the method can be extended to deal with full vertical decomposition.) We will handle each sphere separately, namely choose the direction for the poles independently for each sphere.

The situation here is significantly different from the situation with type I degeneracies: It is impossible to guarantee an ε-separation of the polar arcs. Polar arcs can get arbitrarily close to one another, and they can coincide in the poles. Therefore our goal is to obtain a good angular separation, which we call *ω-separation*, and is defined as follows: Once the direction of the poles has been determined, for any pair of distinct polar arcs the angle between the planes containing them is at least ω. We wish to determine the biggest ω that will allow us to run Step 2 in $O(n)$ time. If ω-separation has been obtained for an ω value which is greater than the floating point resolution, polar arcs can be safely and consistently distinguished during computation.

We next describe this procedure for a single sphere S_t, after Step 1 has been completed (and type I degeneracies have been removed). Fix two little circles C_{it} and C_{jt} on S_t (each being the intersection of S_t with another sphere in M), and let Φ be a great circle on S_t that is tangent to C_{it} and C_{jt} (it is one of at most four great circles tangent to this two little circles). The circle Φ is the locus of pole locations that will cause the planes containing the polar arcs extending from the tangency points of Φ with C_{it} and C_{jt} to coincide (namely have zero angular separation).

Let $\Pi := \Pi(\Phi)$ denote the plane containing Φ. Let Π' and Π'' be two planes parallel to Π, each on a different side of Π, and such that any plane Π_0 passing through the center of S_t and tangent to the circle $\Pi' \cap S_t$ (or the circle $\Pi'' \cap S_t$) makes an angle ω with Π. See Figure 4 for an illustration. We call the portion of S_t between $\Pi' \cap S_t$ and $\Pi'' \cap S_t$ the *ω-strip of* Φ, and denote it by $\sigma(\omega, \Phi)$. It is easily verified that if the poles are chosen outside $\sigma(\omega, \Phi)$ then the polar arcs that correspond to the tangencies of Φ with C_{it} and C_{jt} are at least ω apart.

Every tangent circle Φ of every pair of circles on S_t defines an ω-strip $\sigma(\omega, \Phi)$. Assume without loss of generality that the radius of S_t is 1. The area of any such strip is $4\pi \sin \omega$. The maximum number of strips that need to be considered on S_t is $4\binom{k}{2}$. For Step 2 to run in $O(n)$ time we wish

[2] A spherical shell is the region enclosed between two concentric spheres. It is the three-dimensional analog of an annulus.

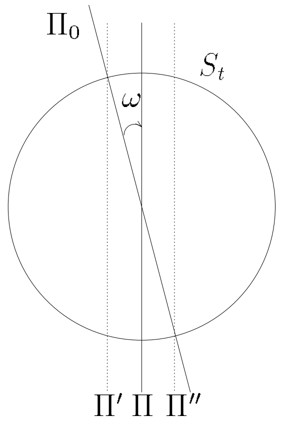


Figure 4: A cross-section of S_t.

that at least half of the surface area of S_t will not be covered by ω-strips (any constant fraction will do; the smaller the fraction the higher the constant factor in the expected running time bound). Therefore we choose ω such that $2 \cdot 4\binom{k}{2}4\pi \sin\omega < 2 \cdot 4\pi \implies \sin\omega < 1/(2k(k-1))$.

Remarks. (1) The ω-strip is only an inscribing region of the forbidden region corresponding to any Φ. The actual forbidden region induced by any tangency is in fact smaller, and it has a more complicated boundary. (2) Extending the method to the full vertical decomposition requires handling several more cases involving up to four circles C_{it} each. This extension is straightforward and we omit further details here. We have implemented this extension and we will report experimental results for it in the full version. (3) In practice, it is convenient to impose the same pole direction for all the spheres in the arrangement (for ease of coding, debugging and visualization). To this end we draw all the ω-strips on the (unit) sphere of directions $\mathcal{S}^1$, and the constraint on ω becomes $\sin\omega < 1/(2nk(k-1))$. The experimental results described below are for this choice of pole directions. (4) The construction of the forbidden regions on S_t is diametrically symmetric, namely a point on S_t is free (i.e., represents a pole that will induce an ω-separation) if and only if its antipodal point is free. (5) In our implementation we also check to make sure that a polar arc does not pass within ω of any intersection point.

5 Algorithmic Details and Complexity Analysis

In this section we describe how the perturbation scheme is carried out efficiently.

The incremental procedure of Step 1 will move the center of each sphere by at most δ from its original placement. We expand each of the original spheres S_t into a sphere S'_t whose radius is $r_i+\delta$, and let M' be the set of the expanded spheres S'_t. We construct a data structure as described in Theorem 2.2 to support range queries on the spheres in M'. This structure enables us to find in $O(1)$ time all the spheres in M' that intersect a given query sphere S'_t.

When looking for a perturbation of the center of S_t during Step 1 of the procedure we proceed as follows. Although constructing the subdivision of $B(C_t, \delta)$ into free and forbidden regions could be done in maximum constant time, it would be an extremely difficult task that may introduce new precision problems. Instead we do something much simpler. We choose a point p uniformly at random in the ball $B(C_t, \delta)$. We next check whether p is a valid perturbation by checking that it lies outside all the forbidden regions F_2, F_3 and F_4. For example, to check whether p lies outside

number	file name	size	max k	mean k	number	file name	size	max k	mean k
1	estradiol.mol2	44	17	6.81	8	4pti.pdb	454	10	5.79
2	clofazimine.mol2	55	13	6.67	9	1bzm.pdb	2034	10	5.74
3	michellamine_b.mol2	104	13	6.81	10	2pka.pdb	3598	11	5.79
4	288d.pdb	120	9	6.25	11	2ace.pdb	4143	12	6.05
5	conocurvone.mol2	127	14	6.67	12	1sdk.pdb	4384	12	6.00
6	245d.pdb	240	9	6.17	13	1nok.pdb	6759	11	5.73
7	1ppt.pdb	301	10	5.92	14	7at1.pdb	7106	12	5.70

Table 1: Listing of the molecules used for testing. Molecules with file names ending in pdb can be found on the Protein Databank at http://www.pdb.bnl.gov/. Molecules with file names ending in mol2 can be found at the Center for Molecular Modeling at http://cmm.info.nih.gov/modeling/. The size column gives the number of atoms in the molecule, and k is the number of atom spheres intersecting any single atom sphere.

F_2 we check whether it lies outside each of the at most $2k$ spherical shells defining F_2. Each of these tests is simple and fast and their overall number is bounded by a constant. See below for experimental results. Recall that δ was chosen such that with probability $> \frac{1}{2}$, the point p will represent a valid perturbation. Thus the expected number of trials before we find a valid p is ≤ 2, and the overall running time of Step 1 is $O(n)$.

Similar arguments apply to Step 2 procedure. For any given sphere S_t, we choose a point p uniformly at random on the boundary of $\mathcal{S}^1$. We then test all the ω-strips relevant to S_t to check whether p lies outside all those strips. By the choice of ω the expected number of trials before we find a valid p is ≤ 2, and the overall running time of Step 2 is $O(n)$ as well.

We summarize the performance of the perturbation scheme in the following theorem.

Theorem 5.1 *Given a collection M of n spheres as described in Theorem 2.1, and a resolution parameter $\varepsilon > 0$, a valid perturbation of the spheres in M can be computed in $O(n)$ time, by moving each sphere by at most δ from its original placement, where δ is a parameter that depends on ε, on the maximum number of spheres in M intersecting any other sphere in M, and on the maximum radius of a sphere in M, and such that all the degeneracies are resolved. In $O(n)$ time we can also find a direction for the poles on each sphere so that all the polar arcs in the trapezoidal decomposition of the spherical arrangements lie on planes the angle between any pair of which is at least ω, for ω that depends on the maximum number of spheres in M intersecting any single sphere in M.*

We emphasize again that the theoretical bounds that we obtain on δ and ω are crude and in practice (as shown below) these quantities are much smaller.

6 Experimental Results

For timing experiments, we chose fourteen molecules as listed in Table 1. We feel these are a good sample of typical molecules (from a geometric computing point-of-view). The molecules from the Protein Databank did not specify the positions of hydrogen atoms whereas the others did. This accounts for most of the difference in k values. All timings were done on an SGI Indigo with a 250Mhz MIPS RS4400 CPU.

For our timings, we ran the program on each of the molecules in our set for a varying ϵ and δ (δ on $[10^{-6}, 10^{-10}]$ and ϵ on $[10^{-9}, 10^{-12}]$). Figure 5 shows the results of these timing for the portion of the code not involved with computing perturbations. Figure 6 details the the time spent in the perturbation code.

It should be noted that almost all of this time ($90 - 98\%$) is spent finding the pole direction. Figure 2 shows the fraction of the δ-spheres which are free and Figure 3 shows the fraction of the pole directions that are free for a typical molecule. To obtain the former chart, we modified the program to attempt (and discard) 1000 perturbations each time the center of an atom was positioned. The values shown in the table are the average over all centers of the fraction of these perturbations that were valid. The latter was obtained with a similar sampling method for pole directions.

The reason for the dramatic difference between the time spent on pole location and the time spent on center location can be attributed to a few additional constraints we placed on the poles. In order to simplify certain coding and debugging procedures, we insisted that all spheres share the same pole direction. In addition, the poles were required to be chosen such that no intersection point of a polar arc and a little circle would be within ϵ of any other point and that no polar arc be within ω of the special great circle that is added as part of our "simplified point location" procedure (which we detail in the full version of our paper).

We also reperformed the experiment of Table 2 for a purposefully degenerate collection of spheres (Table 4). For this set, we arranged 27 spheres in a cube on a unit regular grid. Each sphere had a radius of 1.05 thus producing 2 to 12 degeneracies where 3 little circles intersect at one point on each sphere. Even in this wildly degenerate case, most of the δ-sphere was free even when the ratio of δ to ϵ was 10.

7 Additional Issues in Computing Spherical Arrangements

In our implementation we offer solutions to additional problems concerning geometric computing. In the incremental construction of spherical arrangements we need to perform *point location* queries. In the full version of the paper we report experimental results with a simplified point-location structure that we have devised. Also in the full version we discuss the advantages and disadvantages of using the *partial* trapezoidal decomposition, in contrast to using the *full* vertical decomposition. We have implemented both decompositions and support our claims with experimental results. For lack of space we omit a detailed discussion of these issues in this abstract.

8 Conclusion

We have presented a perturbation scheme for a collection of spheres in three-dimensional space. Our scheme is suitable for computing with finite precision arithmetic and we have presented experimental results while using standard floating point arithmetic. For a given resolution parameter ε we perturb the spheres such that features of the three dimensional arrangement of the spheres (and hence of the two-dimensional spherical arrangement on each sphere) are at least ε apart. Our scheme balances between the size of the perturbation (which we aim to minimize) and the expected running time of the scheme (the smaller the magnitude of the perturbation the longer the expected time it may take to compute a valid perturbation).

Our motivation to develop this scheme is a software package that we have devised aimed to support geometric queries on molecular models. The new scheme has made our algorithms and data structures robust with hardly any effect on the running and reaction time of the system. We have also presented experimental results showing this small effect. Our software package is currently used by chemists working in rational drug design.

The main direction for further research that we propose is to extend the scheme to other types of arrangements of geometric objects. An obvious limitation of our approach, that may make it unsuitable for certain applications, is that we actually move the input geometric objects from their given placement. However, we believe that there are applications where a small bounded perturbation of the input objects is permissible, since often the precision of the input objects is limited to start with (due to measurement limitations, for example). Moreover, applications that use floating point arithmetic without further adaptations, already make approximate computations, and a small enough perturbation may not have any noticeable effect.

Acknowledgment

The molecular modeling package described in the paper has been developed as part of a joint project between the Computer Science Robotics Laboratory at Stanford University and Pfizer Central Research in Sandwich, England. The authors acknowledge the assistance of Jean-Claude Latombe who initiated and supervised the project, Paul Finn of Pfizer Central Research who provided useful guidance, and the help and support of the other participants at Stanford: Lydia Kavraki (now at Rice University), Rajeev Motwani, and Suresh Venkatasubramanian.

References

[1] L. Balbes, S. Mascarella, and D. Boyd. A perspective of modern methods for computer aided drug design. In *Reviews in Computational Chemistry*, volume 5, pages 265–294. VCH Publishers Inc., 1994.

[2] M. Connolly. Analytical molecular surface calculation. *Journal of Applied Crystallography*, 16:548–558, 1983.

[3] M. Connolly. Molecular surfaces: A review. *Network Science*, 1996. http://www.awod.com/netsci/Issues/Apr96/feature1.html.

[4] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Heidelberg, West Germany, 1987.

[5] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9:66–104, 1990.

[6] P. Finn, D. Halperin, L. Kavraki, J.-C. Latombe, R. Motwani, C. Shelton, and S. Venkatsubramanian. Geometric manipulation of flexible ligands. In M. C. Lin and D. Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 67–78. Springer, 1996.

[7] S. Fortune. Ronustness issues in geometric algorithms. In M. C. Lin and D. Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 9–14. Springer, 1996.

[8] S. Fortune and C. J. Van Wyk. Static analysis yields efficient exact arithmetic for computational geometry. *ACM Trans. on Graphics*, 25(3):223–248, 1996.

[9] M. Goldwasser. An implementation for maintaining arrangements of polygons. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages C32–C33, 1995.

[10] D. H. Greene and F. F. Yao. Finite-resolution computational geometry. In *Proc. 27th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 143–152, 1986.

[11] L. J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graph.*, 4:74–123, 1985.

[12] D. Halperin and M. H. Overmars. Spheres, molecules, and hidden surface removal. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 113–122, 1994.

[13] C. Hoffmann. *Geometric and Solid Modeling.* Morgan Kaufmann, San Mateo, California, 1989.

[14] C. Hoffmann. The problems of accuracy and robustness in geometric computation. *IEEE Computer*, 22(3):31–41, Mar. 1989.

[15] P. Mezey. Molecular surfaces. In K. Lipkowitz and D. Boyd, editors, *Reviews in Computational Chemistry*, volume 1, pages 265–294. VCH Publishers Inc., 1990.

[16] V. Milenkovic. Verifiable implementations of geometric algorithms using finite precision arithmetic. *Artif. Intell.*, 37:377–401, 1988.

[17] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, NY, 1985.

[18] M. F. Sanner, A. J. Olson, and J.-C. Spehner. Fast and robust computation of molecular surfaces. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages C6–C7, 1995.

[19] J. R. Shewchuk. Robust adaptive floating-point geometric predicates. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 141–150, 1996.

[20] K. Sugihara. On finite-precision representations of geometric objects. *J. Comput. Syst. Sci.*, 39:236–247, 1989.

[21] K. Sugihara and M. Iri. Two design principles of geometric algorithms in finite-precision arithmetic. *Appl. Math. Lett.*, 2(2):203–206, 1989.

[22] A. Varshney, F. B. Jr., and W. Wright. Computing smooth molecular surfaces. *IEEE Computer Graphics and Applications*, 14:19–25, 1994.

[23] C. Yap and T. Dubé. The exact computation paradigm. In D. Du and F. Hwang, editors, *Computing in Euclidean Geometry*, pages 452–492. World Scientific, 1995.

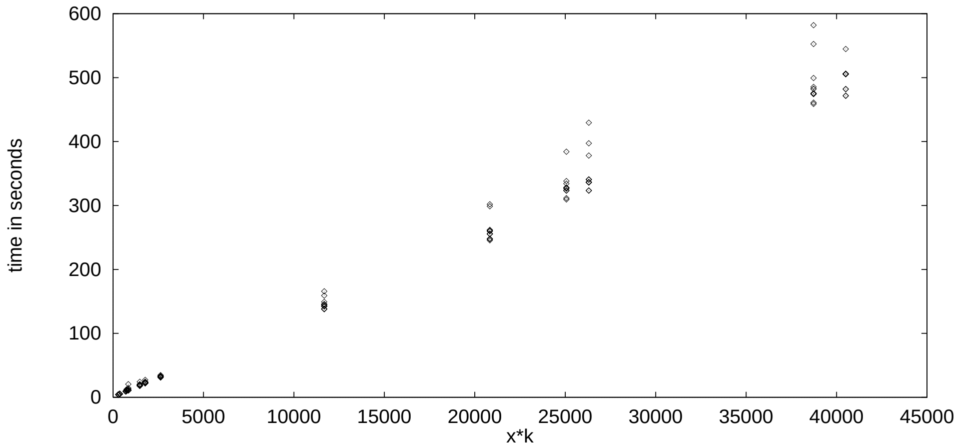


Figure 5: Time to build the molecules shown in Table 1 excluding perturbation time. The horizontal coordinate is size-of-molecule $\times$ average-k.

Appendix: Shells of Degenerate Placements

In this appendix we give more details on the shape and volume of the shells defining the regions F_2, F_3 and F_4 as mentioned in Section 4.

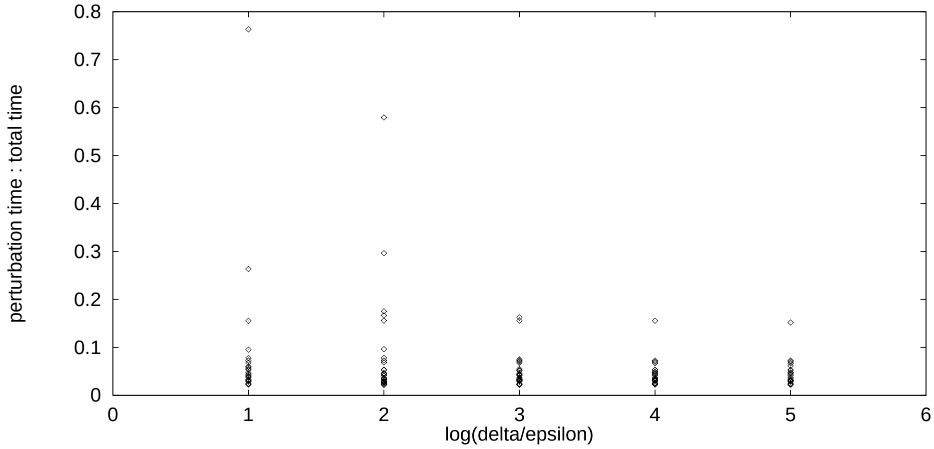


Figure 6: Ratio of perturbation time to total time versus ratio of δ to ϵ.

	1e-3	1e-4	1e-5	1e-6	1e-7	1e-8	1e-9
1e-2	0.993837	0.999558	0.999958	0.999994	0.999998	1.000000	1.000000
1e-3		0.999669	0.999967	0.999997	1.000000	1.000000	1.000000
1e-4			0.999778	0.999977	0.999997	0.999999	1.000000
1e-5				1.000000	1.000000	1.000000	1.000000
1e-6					1.000000	1.000000	1.000000
1e-7						1.000000	1.000000
1e-8							1.000000

Table 2: Valid fraction of the δ-sphere for molecule 10 from Table 1. ϵ varies across the table and δ varies downwards.

$1-\cos(\omega)$				
1e-9	1e-10	1e-11	1e-12	1e-13
0.285	0.675	0.890	0.950	0.975

Table 3: Valid fraction of pole positions for molecule 8 from Table 1.

	1e-3	1e-4	1e-5	1e-6	1e-7	1e-8	1e-9
1e-2	0.625565	0.937450	0.993346	0.999296	0.999923	1.000000	1.000000
1e-3		0.626231	0.937365	0.993377	0.999350	0.999923	1.000000
1e-4			0.626281	0.937365	0.993381	0.999331	0.999954
1e-5				0.626281	0.937365	0.993362	0.999338
1e-6					0.626281	0.937358	0.993358
1e-7						0.626281	0.937358
1e-8							0.626281

Table 4: Valid fraction of the δ-sphere for degenerate set of spheres described in Section 6. ϵ varies across the table and δ varies downwards.

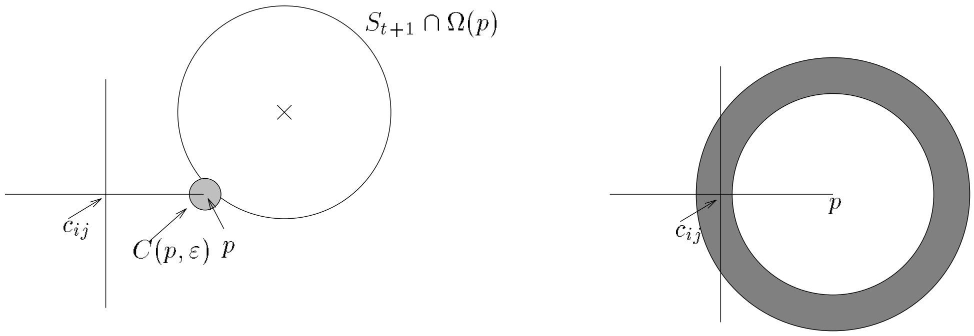


Figure 7: A cross-section of C_{ij} and S_{t+1} on the plane $\Omega(p)$.

The region F_2. F_2 consists of placements of the center of S_{t+1} that induce tangency or near-tangency of S_{t+1} and another sphere. There are two types of tangencies: external and internal. We first describe the shell for external tangency. For a sphere S_i, an exact tangency is induced by placing the center C_{t+1} of S_{t+1} at distance exactly $r_i + r_{t+1}$ away from the center of S_i, namely, $\rho_{i(t+1)} - r_i - r_{t+1} = 0$. We define the potential degeneracy of this type when using floating point with resolution ε as $-\varepsilon \leq \rho_{i(t+1)} - r_i - r_{t+1} \leq \varepsilon$, which is a spherical shell (i.e. the region sandwiched between two concentric spheres) with center at C_i and radii $r_i + r_{t+1} - \varepsilon$ and $r_i + r_{t+1} + \varepsilon$. Its volume is $\frac{4}{3}\pi[(r_i + r_{t+1} + \varepsilon)^3 - (r_i + r_{t+1} - \varepsilon)^3]$. Similarly the volume of a shell corresponding to internal tangency is (assuming $r_{t+1} > r_i$) $\frac{4}{3}\pi[(r_{t+1} - r_i + \varepsilon)^3 - (r_{t+1} - r_i - \varepsilon)^3]$.

The region F_3. This region F_3 is the union of *toric shells* each defined for a pair of distinct sphere $S_i, S_j \in M_t$. Let C_{ij} denote $S_i \cap S_j$ after both spheres have been possibly perturbed. The toric shell for the pair S_i, S_j is the loci of placements of the center of S_{t+1} that will cause S_{t+1} to be tangent, or almost tangent to C_{ij}. Thus it represents the placements of S_{t+1} that may cause two circles on a spherical arrangement to be tangent or almost tangent.

We denote the center of circle C_{ij} by c_{ij} and its radius by r_{ij}. At a point p of tangency between C_{ij} and S_{t+1} there is a plane $\Pi(p)$ that is tangent to S_{t+1}, and such that its intersection with the plane containing C_{ij} is a line $L(p)$ tangent to C_{ij}. If we fix the point p on C_{ij} and hence the line $L(p)$ there is a pencil of planes through $L(p)$ where each plane defines a possible osculation placement of the sphere and the circle at p. The induced loci of forbidden placements for the center of S_{t+1} is a circle with center at p, lying on a plane $\Omega(p)$ orthogonal to $L(p)$ and having radius r_{t+1}. We now extend the definition to near-tangency at p and restrict ourselves to the plane $\Omega(p)$.

We assume that C_{t+1} lies on $\Omega(p)$ and hence $S_{t+1} \cap \Omega(p)$ is a great circle of S_{t+1}. We expand a circle $C(p, \varepsilon)$ of radius ε around p. The forbidden placements for C_{t+1} are now all the placement where $S_{t+1} \cap \Omega(p) \cap C(p, \varepsilon) \neq \emptyset$—the shaded annulus in Figure 7.

We repeat this for every point p on C_{ij} and obtain a toric shell. In fact it is only toric-like but for brevity we call it a toric shell. It would have been a real toric shell (the region sandwiched between two tori with the same center, axis of rotation, and major radius) were it not self intersecting as it may possibly be here.

To bound the volume of the toric shell, we will first look at the solid of revolution obtained by rotating an external quarter of the annulus around the line through C_{ij} and orthogonal to the plane containing C_{ij}. The obtained volume, which we denote by $V_{1/4}$, is clearly an upper bound on quarter the volume of the toric shell. Let $V_{1/4}(r)$ denote the volume of a quarter torus with major radius r_{ij} and minor radius r. Then $V_{1/4} = V_{1/4}(r_{t+1}+\varepsilon) - V_{1/4}(r_{t+1}-\varepsilon)$.

$$\begin{aligned} V_{1/4}(r) &= 2\pi \int_0^r (u + r_{ij})\sqrt{r^2 - u^2}du \\ &= 2\pi[\frac{1}{3}(r^2 - u^2)^{3/2} + \frac{u}{2}\sqrt{r^2 - u^2} + \frac{r^2}{2}\arcsin\frac{u}{r}] \mid_0^r \\ &= -\frac{2}{3}\pi r^3 + \pi^2 r_{ij}\frac{r^2}{2} \end{aligned}$$

It follows that $V_{1/4} = 2\pi^2 r_{ij} r_{t+1}\varepsilon + 4\pi r_{t+1}^2\varepsilon + \frac{4}{3}\pi\varepsilon^3$. Assuming that both r_{ij} and r_{t+1} are bounded by R, we get that the volume of the toric shell is bounded by $4\pi(2\pi+1)R^2\varepsilon + \frac{16}{3}\pi\varepsilon^3$.

The region F_4. The region F_4 is the union of spherical shells each centered at a point of intersection p of three distinct spheres in M_t and having radii $r_{t+1}-\varepsilon$ and $r_{t+1}+\varepsilon$. The spherical shell here is the loci of placements of the center of S_{t+1} that will cause S_{t+1} to pass through an intersection point of three distinct spheres in M_t or very close to this intersection point.

Let p denote such an intersection point. The loci of placements of C_{t+1} that will cause S_{t+1} to go through p is the sphere centered at p with radius r_{t+1}. It is easily verified that if we wish that p will be ε away from S_{t+1} than the forbidden placements are the spherical shell centered at p with radii $r_{t+1}-\varepsilon$ an $r_{t+1}+\varepsilon$. Its volume is $\frac{4}{3}\pi[(r_{t+1}+\varepsilon)^3 - (r_{t+1}-\varepsilon)^3]$.

Bounding the volume of F_2, F_3, and F_4 Let M be a collection of spheres as defined in Theorem 2.1. Also let $R := \max_{i=1}^n r_i$, and let k denote the maximum number of spheres of M intersecting a single sphere of M. Given a parameter $\varepsilon > 0$, if we choose $\delta = f(k,\varepsilon,R) = 10k\varepsilon^{1/3}R^{2/3}$, then we get that the volume of the forbidden region inside $B(C_{t+1},\delta)$ is less than half the volume of $B(C_{t+1},\delta)$. Let $VB :=$ Volume$(B(C_{t+1},\delta))$, and let $VF :=$ Volume$(F_2 \cup F_3 \cup F_4)$.

Note that there are at most k external spherical shells defining the region F_2 and k internal shells, at most $\binom{k}{2}$ toric shells defining the region F_3, and at most $\binom{k}{3}$ spherical shells defining the region F_4. Therefore we obtain the following bounds on the volume of these regions:

$$\begin{aligned} \text{Volume}(F_2) &\leq k\frac{4}{3}\pi[(2R+\varepsilon)^3 - (2R-\varepsilon)^3] + k\frac{4}{3}\pi[(R+\varepsilon)^3 - (R-\varepsilon)^3] \\ &\leq \frac{4}{3}\pi k(30R^2\varepsilon + 4\varepsilon^3) \end{aligned}$$

$$\text{Volume}(F_3) \leq \binom{k}{2}[4\pi(2\pi+1)R^2\varepsilon + \frac{16}{3}\varepsilon^3] \leq \frac{1}{2}k^2[4\pi(2\pi+1)R^2\varepsilon + \frac{16}{3}\pi\varepsilon^3]$$

$$\text{Volume}(F_4) \leq 2\binom{k}{3}\frac{4}{3}\pi[(R+\varepsilon)^3 - (R-\varepsilon)^3] \leq \binom{k}{3}\frac{4}{3}\pi(6R^2\varepsilon + 2\varepsilon^3) \leq \frac{4}{9}k^3\pi(6R^2\varepsilon + 2\varepsilon^3)$$

$$\begin{aligned} VF &\leq \pi R^2\varepsilon[40k + (4\pi+2)k^2 + \frac{8}{3}k^3] + \varepsilon^3[\ldots] \\ &\leq 5\pi R^2\varepsilon k^3 \text{ (assuming } k \geq 10 \text{ and } \varepsilon \ll k, R) . \end{aligned}$$

Finally, we choose δ such that $VB = \frac{4}{3}\pi\delta^3 > 2VF$. Thus we let $\delta := 2k\varepsilon^{1/3}R^{2/3}$.