Fast Analytical Computation of Richards's Smooth Molecular Surface

Amitabh Varshney and Frederick P. Brooks, Jr.
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175

Abstract

An algorithm for rapid computation of Richards's smooth molecular surface is described. The entire surface is computed analytically, triangulated, and displayed at interactive rates. The faster speeds for our program have been achieved by algorithmic improvements, parallelizing the computations, and by taking advantage of the special geometrical properties of such surfaces. Our algorithm is easily parallelizable and it has a time complexity of $O(k \log k)$ over n processors, where n is the number of atoms of the molecule and k is the average number of neighbors per atom.

1 Introduction

The smooth molecular surface of a molecule is defined as the surface which an exterior probe-sphere touches as it is rolled over the spherical atoms of that molecule. This definition of a molecular surface was first proposed by Richards [16]. This surface is useful in studying the structure and interactions of proteins, in particular for attacking the protein-substrate docking problem. An example of such a molecular surface for Felix with a probe-sphere radius of $1.4\mathring{A}$ can be seen in Figure 1.

Present systems to compute the surfaces of molecules are batch-oriented. They take a few minutes to compute the surface for a couple of thousand atoms. Our goal has been to compute and display these surfaces at interactive rates, by taking advantage of results from the field of computational geometry, making further algorithmic improvements, and parallelizing the computations.

Interactive computation of molecular surfaces should benefit biochemists in three important ways. First, the ability to change the probe-radius interactively helps one study the surface. Second, it helps in visualizing the changing surface of a molecule as its atom positions are changed. These changes in atom positions could be due to user-defined forces as the user attempts to modify a molecular model on a computer. Third, it assists in incorporating the effects of the solvent into the overall potential energy computations during the interactive modifications of a molecule on a computer.

2 Previous and Related Work

The first molecular surface computation algorithms were numerical in nature (ie. they were computed by sampling) Connolly [2], Greer [12]. Connolly [2] computes the sampled surface (also known as the *dot-surface*) by placing a probe tangent to either one atom, or two atoms, or three atoms and

Figure 1: Smooth Molecular Surface for Felix

checking to see if it intersects any of the other neighboring atoms. If it does not and it is tangent to (i) one atom, then a dot is placed at the point of tangency between the probe and that atom, (ii) two atoms, then a concave arc of dots connecting the two points of tangency is created, (iii) three atoms, then a concave spherical triangle of dots is created between the three points of tangency. This generates a dot-representation of the entire surface.

The analytic computation of the molecular surface was also first done by Connolly [3], [4]. Here a molecular surface is represented by a collection of spherical and toroidal patches as follows:

- The molecular surface for the regions of a molecule where the probe is in contact with a single atom are modeled by convex spherical patches.
- The molecular surface for the regions of a molecule where the probe is in simultaneous contact with two atoms are modeled by saddle-shaped toroidal patches.
- The molecular surface for the regions where the probe is in simultaneous contact with three atoms are modeled by concave spherical triangular patches.

The issues of triangulation of such surfaces are discussed by Connolly in [5].

Only recently have the issues of algorithmic complexity of these algorithms begun to be addressed. Let n be the number of atoms in a molecule and let k be the average number of *neighboring* atoms for an atom in the molecule. By *neighboring* we mean the atoms that are near enough to affect probe placement on a particular atom. Yip and Elber [18] present an algorithm for computation of the list of neighboring atoms that is linear in n. It is based on spatial subdivision by a global grid. Perrot *et al.* [13], [14] present a $O(kn)$ algorithm that generates an approximation to the solvent-accessible surface. In this approximation, every concave spherical triangular patch between three atoms is represented by a planar triangle with vertices at the centers of these three atoms. Saddle-shaped toroidal regions and convex spherical patches are ignored. In terms of sequential algorithmic complexity this is good, however some points remain unaddressed here. This algorithm is inherently sequential, as it always needs to start from some concave spherical triangular region of the molecule and from there it proceeds by adding an adjacent face at a time. Besides being hard to parallelize, it fails for the cases where the solvent-accessible surface folds back to intersect itself or where the molecule has two or more sub-parts connected by only two overlapping spheres. Also, it cannot generate the interior cavities of a molecule.

In computational geometry, the α-hull has been defined as a generalization of the convex hull of point-sets by Edelsbrunner, Kirkpatrick, and Seidel [6], [8]. For $\alpha > 0$, the α-hull of a set of points P in two-dimensions is defined to be the intersection of all closed complements of discs with radius α that contain all points of P. If we generalize this notion of α-hulls over point-sets to the corresponding hulls over spheres of unequal radii in three-dimensions, we would get the molecular surface (along with the surface defining the interior cavities of the molecule). It has been shown in [8] that it is possible to compute the α-hulls from the Voronoi diagram of the points of P. For $\alpha = \infty$ the α-hull over the set of points P is the same as their convex hull. Richards [16] had also suggested computing the molecular surface by computing a 3D Voronoi diagram first and then using its faces to determine which nearby atoms to consider.

Edelsbrunner and Mücke [9] extend the definition of α-hulls to points in three-dimensions. Here an *α-shape* over a set of points P has been defined to be the polytope that approximates the α-hull over P, by replacing circular arcs of the α-hull by straight edges and spherical caps by triangles. An α-shape of a set of points P is a subset of the Delaunay triangulation of P. Edelsbrunner in [7], extends the concept of α-shapes to deal with weighted points (i.e. spheres with possibly unequal and non-zero radii) in three-dimensions. An α-shape of a set of weighted points P_w is a subset of the regular triangulation of P_w. Since these methods involve computing the entire triangulation first and then culling away the parts that are not required, their complexity is $O(n^2)$ in time. This is worst-case optimal, since an α-shape in three-dimensions could have a complexity of $O(n^2)$. We next discuss a different approach that works better for environments where the maximum density of P in a given volume is some constant smaller than n. Molecules are a good example of such environments.

3 Our Approach

Our goal has been to formulate a parallel analytical molecular surface algorithm that has expected linear complexity with respect to the total number of atoms of a molecule. For achieving this goal, we have avoided computation of the complete three-dimensional regular triangulation over the entire set of atoms - a process that takes time $\mathrm{O}(n^2)$, where n is the number of atoms in the molecule.

Molecules considered as a collection of weighted points in three-dimensions, where the coordinates of each point p_i correspond to the center of atom i and the weight r_i is the radius of atom i, have two interesting properties: (i) the minimum distance d_{ij} between any two points p_i and p_j is strictly positive, $d_{ij} > 0$ and (ii) the set of all the weights can be bounded from above and below by non-zero values, $r_{min} \leq r_i \leq r_{max}$. We take advantage of the first property to arrive at better running times for our algorithm. Stated simply, the first property says that the number of neighboring atoms within a fixed distance from any atom i, is always bounded from above by a constant k_{max} that depends on the minimum spacing between any two atoms. If the average number of neighbors for an atom is k, then we can just compute an approximation to the power cell (discussed in Section 3.1), which we call a *feasible cell*, by considering only these neighbors. Each feasible cell can be computed in parallel in time $O(k \log k)$. For n atoms, this task requires n processors, each processor computing the feasible cell for one atom.

3.1 Formal Notation

Here we consider the underlying space to be three-dimensional Euclidean Space $\Re^3$, although these results can be generalized to higher dimensions.

Let $M = \{S_1, \ldots, S_n\}$, be a set of spheres, where each sphere, S_i, is expressed as a pair $< c_i, r_i >$, c_i being the center of the sphere and r_i being the radius of the sphere.

Let x, y be two points. Define $d(x, y)$ to be the Euclidean distance between x and y. The *power of a point* x with respect to a sphere S_i is defined as $p(x, S_i) = d^2(x, c_i) - {r_i}^2$ Thus, $p(x, S_i) < 0, = 0, > 0$, depending on whether x lies inside S_i, on the boundary of S_i, or outside S_i, respectively.

Henceforth, we shall be assuming that the atoms of a molecule are represented as spheres and will be using the terms *atom* i and S_i interchangeably. Let the radius of the probe-sphere be R. Then we define the *extended-radius sphere* for atom i to be $\Psi_i =< c_i, r_i + R >$. This extended-radius sphere Ψ_i is the locus of the possible centers of the probe-sphere when it is in contact with atom i.

Define a *chordale* Π_{ij} of two spheres Ψ_i and Ψ_j as $\Pi_{ij} = \{x | p(x, \Psi_i) = p(x, \Psi_j)\}$. Π_{ij} is a plane perpendicular to the line joining c_i and c_j. If Ψ_i and Ψ_j overlap, Π_{ij} passes through the circle that is common to the boundaries of Ψ_i and Ψ_j. If Ψ_i and Ψ_j do not overlap, it lies between c_i and c_j. Define the halfspace H_{ij} as $H_{ij} = \{x | p(x, \Psi_i) < p(x, \Psi_j)\}$.

Thus, whereas $\Pi_{ij} = \Pi_{ji}$, $H_{ij} \neq H_{ji}$. Define the *feasible cell*, F_i, for atom i as $F_i = \cap_j H_{ij}$.

So far, the definitions that we have given for a *feasible cell* correspond exactly to the *power cells* of a power diagram as defined by Aurenhammer [1]. The reason we prefer to use a different term for these will be made clear in Section 3.3.

3.2 Determination of Neighboring Atoms

Determination of neighboring atoms can be done by spatial grid subdivision into voxels, and assigning atoms to appropriate voxels. We define the *region of influence*, ρ_i, for atom i to be the sphere $< c_i, r_i + 2R + \max_{j=1}^{n} r_j >$. Then for computing the list of neighboring atoms, N_i, for atom i, one needs to find all the atoms that are close enough to affect probe placement on atom i. Formally, $N_i = \{j | d(c_i, c_j) < r_i + 2R + r_j\}$, or equivalently, $N_i = \{j | \Psi_i \cap \Psi_j \neq \phi\}$. Intuitively, an atom j is a neighbor to atom i, if it is possible to place a probe such that it is in contact with both S_i and S_j (without considering any hindrance due to other atoms). Since, by definition $N_i \subseteq \rho_i$, to compute the list of neighboring atoms for atom i, one needs to look at all the atoms that lie in the voxels that intersect ρ_i. Let the average number of neighboring atoms be k. Note that k grows as R^3 assuming that the atoms are uniformly distributed. In Figure 2 atoms j_1 and j_3 are neighbors to atom i, but atom j_2 is not.

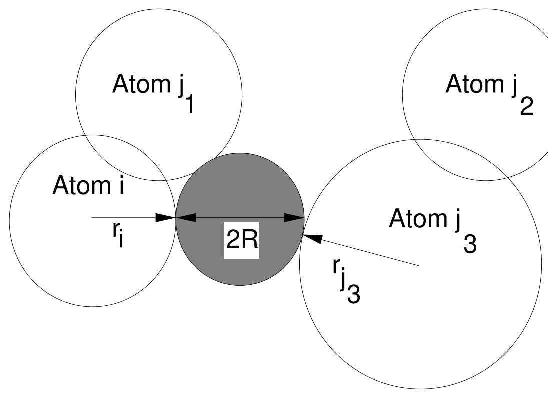

Figure 2: Determination of Neighboring Atoms

3.3 Determination of Surface Atoms

Here the aim is to determine the atoms that are buried in the interior of the molecule and would not therefore directly participate in the final definition of the smooth molecular surface. This step is not crucial to the linear time complexity of the overall algorithm but it helps in improving the execution times.

For every atom i, we first compute N_i as described in Section 3.2. Then we compute $F_i = \cap_{j \epsilon N_i} H_{ij}$. If $F_i = \phi$, atom i is totally buried and cannot be a surface atom. This checking for nullity is done by Seidel's randomized linear programming algorithm that has linear time and is quite fast in practice [17].

It might be useful to point out how the feasible cells F_i we are computing are different from the power cells described by Aurenhammer [1]. The difference arises from the fact that each of these cells F_i are being computed independently of the other, and for each cell F_i we consider only the hyperplanes whose indices occur in N_i. This is best illustrated by Figure 3, where the two types of cells are shown. Thus, for construction of F_i we use only those chordales Π_{ij}, as defined in section 3.1, that arise from overlapping extended-radius spheres Ψ_i and Ψ_j. However, for forming the power cells C_i, we use all the chordales Π_{ij} regardless of whether Ψ_i and Ψ_j intersect or not.

In Figure 3, we show these differences for power cells and feasible cells defined over circles. The power cell C_3 contains two edges and one vertex as does the corresponding feasible cell F_3. However, whereas the power cells C_1 and C_2 have two edges and one vertex, the corresponding feasible cells F_1 and F_2 have only one edge each, with no vertices.

We should also point out here that the feasible cells that occur on the boundary atoms are not compact (actually, they are neither closed nor bounded). Although the details of triangulation of these surfaces will appear elsewhere we would like to mention here that we use the vertices of F_i

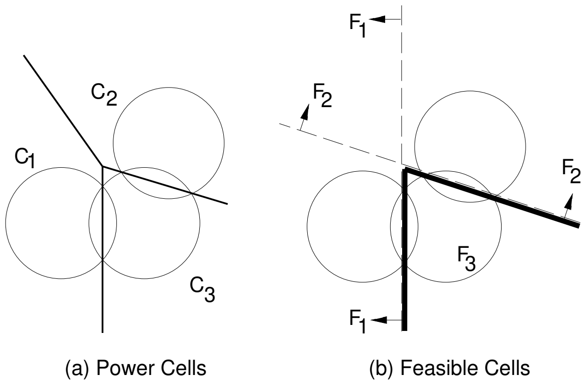

Figure 3: Difference between Power Cells and Feasible Cells

during triangulation. For this purpose we would like all F_i to be compact (closed and bounded) polytopes. To achieve this we bound the entire molecule by four extra planes forming a regular tetrahedron. These are included in N_i, for all i.

3.4 Determination of Surface Patches

Determination of the vertices defining the convex spherical, concave spherical and toroidal patches is the most crucial (and time-consuming) part of the whole algorithm.

If one computes a three-dimensional α-shape polytope for the set of atoms in a molecule, with $\alpha =$ probe-radius, then the torii occur along the edges, the concave spherical triangular patches correspond to the faces, and the convex spherical patches correspond to the vertices of this polytope. The method given by Edelsbrunner [7] finds these edges by first computing the entire three-dimensional regular triangulation, an $O(n^2)$ approach. We show here an equivalent method for computing the three-dimensional α-hulls for molecules in parallel time $O(k \log k)$ over n processors

To compute F_i, we compute the convex hull of the points dual to the H_{ij} in the dual-space, as described in [15]. This is an $O(k \log k)$ time process. After this, we compute the dual of the convex hull to get the feasible cell F_i. The intersection of the feasible cell F_i with Ψ_i gives rise to a set of closed components on Ψ_i. Since F_i is convex, these components will be non-intersecting. Each of these closed components ∂c_j,divide Ψ_i into two connected regions, say R_{j_0} and R_{j_1}. For one of these, say R_{j_m}, it will be true that $R_{j_m} = R_{j_m} \cap F_i$. We define R_{j_m} to be the interior of the closed component ∂c_j. To find these components, we determine the edges and faces of F_i that intersect Ψ_i.

After a connected component has been determined on Ψ_i, it is easy to generate the surface patches. The arcs defining the boundary of this component determine the locus of the center of the probe while it is in contact with two atoms. These therefore are used to generate the toroidal patches. The vertices of this component, where two arcs intersect, define the positions of the center of the probe while it is in contact with three atoms. These are used to generate the concave spherical triangular patches. The interiors of the components on Ψ_i, correspond to the positions of the center of the probe while it is tangent to only atom i. These are used to generate the convex spherical patches.

This is shown in the Figure 4 for a component defined by three hyperplanes. The interior of this component is shown unshaded.

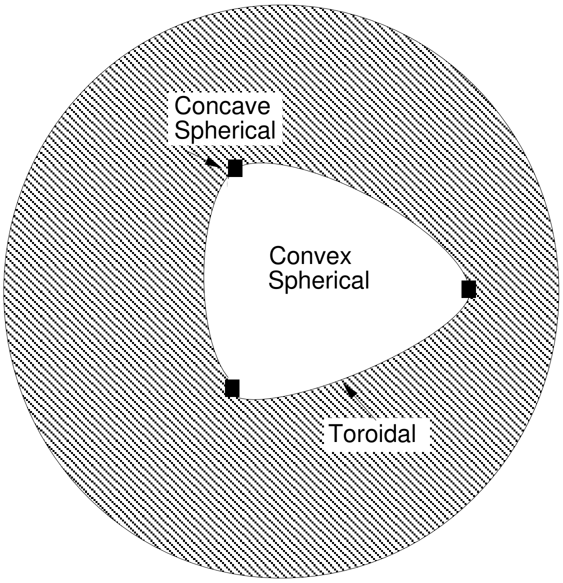

Figure 4: Determination of Molecular Surface Patches from Components

3.5 Parallelization

Our approach to computing the smooth molecular surface can be parallelized over all the atoms of the molecule. Since each of the steps as described above can be carried out independently for each atom, the complexity of our algorithm over n processors would be $O(k \log k)$. If the number of available processors $p < n$, we can allocate $\frac{n}{p}$ atoms per processor to get a time complexity of $O(\frac{nk \log k}{p})$. These bounds hold in a CREW PRAM model of parallel computation.

3.6 Robustness

In the algorithms for computing the convex hull of a set of points, it is assumed that the points are in a general position, ie. no more than d points lie on the same $d-1$ dimensional hyperplane. In reality this assumption often fails to hold, leading to problems. For example, planar benzene rings occur often in proteins, causing six carbon and six hydrogen atoms to be all coplanar.

One of the recent approaches to solving this problem has been to perturb the input point set slightly to avoid these degeneracies. We are using the deterministic perturbation scheme proposed by Emiris and Canny [10], which perturbs the j^{th} dimension of the i^{th} point as:

$$p_{i,j}(\epsilon) = p_{i,j} + \epsilon(i^j \bmod q) 1 \leq i \leq n, 1 \leq j \leq d \tag{1}$$

where ϵ is a symbolic infinitesimal and q is the smallest prime greater than n.

4 Results

Our implementation has been done on Pixel-Planes 5 [11], although it is general enough to be easily portable to any other parallel architecture. Table 1 shows our timings for computation and display of the molecular surface for various molecules for a probe-radius of $1.4\mathring{A}$. For these results we were using either a configuration of 13 or 26 Intel i860 graphics processors as shown below. At present, we are representing the molecular surface by triangles and the column *No. of Triangles* in the table below refers to the complexity of the computed surface. As can be seen the value of k is fairly constant for a given probe-radius over different molecules.

Molecule	No. of atoms	Times (in sec)		k	No. of Triangles
		13 procs	26 procs		
Crambin	396	0.46	0.25	45	18.5K
Felix	613	0.76	0.41	41	36.5K
Dihydrofolate reductase	3123	3.23	1.63	44	95.5K
Superoxide dismutase	4386	4.51	2.25	46	125.4K

Table 1: Smooth Molecular Surface Generation Times

Figure 5 shows the smooth molecular surfaces for Crambin with probe-sphere radii of (a) $1.0\mathring{A}$, (b) $1.4\mathring{A}$, (c) $2.8\mathring{A}$, and (d) $5.0\mathring{A}$. Figure 6 shows the smooth molecular surface for Dihydrofolate reductase with probe-sphere radius of $1.4\mathring{A}$. Figure 7 shows the smooth molecular surface for Superoxide dismutase with probe-sphere radius of $1.4\mathring{A}$.

5 Conclusions and Future Work

We have presented a parallel algorithm for computing the molecular surfaces in parallel time $O(k \log k)$ over n processors. This is sufficiently general enough to be used for computation of α-hulls and α-shapes for a given value of α as long as no two points are arbitrarily close. Our algorithm would give an order of magnitude improvement over the best known previous algorithms for molecules with large values of n - both in sequential as well as in parallel implementations.

At present we are not using any incremental temporal information in constructing these surfaces. Thus, if the atoms move slightly from their positions, the whole surface has to be recomputed from the beginning. Assuming the atoms of the molecule move along continuous trajectories, it should be possible to compute such surfaces (and indeed α-hulls and α-shapes) incrementally by using the information from previous time steps.

6 Acknowledgements

We would like to acknowledge the valuable discussions we had with Dr. William V. Wright, Dr. Dinesh Manocha, Dr. David C. Richardson, and Dr Jan F. Prins during various stages of this work. This work was supported by NIH National Center for Research Resources grant number 5-P41-RR02170.

(a)

(b)

Figure 5: Smooth Molecular Surfaces for Crambin

(c)

(d)

Figure 5: Smooth Molecular Surfaces for Crambin(contd)

Figure 6: Smooth Molecular Surface for Dihydrofolate reductase

Figure 7: Smooth Molecular Surface for Superoxide dismutase

References

[1] F. Aurenhammer. Power diagrams: Properties, algorithms and applications. *SIAM Journal of Computing*, 16(1):78–96, 1987.

[2] M. L. Connolly. Master's thesis, University of California at Berkeley, Berkeley, USA, 1981.

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

[4] M. L. Connolly. Solvent-accessible surfaces of proteins and nucleic acids. *Science*, 221(4612):709–713, 1983.

[5] M. L. Connolly. Molecular surface triangulation. *Journal of Applied Crystallography*, 18:499–505, 1985.

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

[7] H. Edelsbrunner. Weighted alpha shapes. Technical Report UILU-ENG-92-1740, Department of Computer Science, University of Illinois at Urbana-Champaign, 1992.

[8] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, IT-29(4):551–559, 1983.

[9] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. Technical Report UILU-ENG-92-1714, Department of Computer Science, University of Illinois at Urbana-Champaign, 1992.

[10] I. Emiris and J. Canny. An efficient approach to removing geometric degeneracies. In *Eighth Annual Symposium on Computational Geometry*, pages 74–82, Berlin, Germany, June 1992. ACM Press.

[11] H. Fuchs, J. Poulton, J. Eyles, T. Greer, J. Goldfeather, D. Ellsworth, S. Molnar, G. Turk, B. Tebbs, and L. Israel. Pixel-planes 5: A heterogeneous multiprocessor graphics system using processor-enhanced memories. In *Computer Graphics: Proceedings of SIGGRAPH'89*, volume 23, No. 3, pages 79–88. ACM SIGGRAPH, 1989.

[12] J. Greer and B. L. Bush. Macromolecular shape and surface maps by solvent exclusion. In *Proceedings of the National Academy of Sciences USA*, volume 75, No. 1, pages 303–307, 1978.

[13] G. Perrot, B. Cheng, K. D. Gibson, J. Vila, K. A. Palmer, A. Nayeem, B. Maigret, and H. A. Scheraga. Mseed: A program for the rapid analytical determination of accessible surface areas and their derivatives. *Journal of Computational Chemistry*, 13(1):1–11, 1992.

[14] G. Perrot and B. Maigret. New determinations and simplified representations of macromolecular surfaces. *Journal of Molecular Graphics*, 8:141–144, 1990.

[15] F.P. Preparata and M.I. Shamos. *Computational Geometry - an Introduction*. Springer-Verlag, 1985.

[16] F. M. Richards. Areas, volumes, packing and protein structure. *Ann. Rev. Biophys. Bioengg.*, 6:151–176, 1977.

[17] R. Seidel. Linear programming and convex hulls made easy. In *Sixth Annual ACM Symposium on Computational Geometry*, pages 211–215, Berkeley, California, June 1990. ACM Press.

[18] V. Yip and R. Elber. Calculations of a list of neighbors in molecular dynamics simulations. *Journal of Computational Chemistry*, 10(7):921–927, 1989.