Michel F. Sanner
Arthur J. Olson*
The Scripps Research Institute
10666 North Torrey Pines Road
La Jolla, California 92037, USA

Jean-Claude Spehner
Université de Haute Alsace
Laboratoire MAGE
F. S. T. 4 rue des Frères Lumière
Mulhouse Cedex 68093, France

Reduced Surface: An Efficient Way to Compute Molecular Surfaces

Because of their wide use in molecular modeling, methods to compute molecular surfaces have received a lot of interest in recent years. However, most of the proposed algorithms compute the analytical representation of only the solvent-accessible surface. There are a few programs that compute the analytical representation of the solvent-excluded surface, but they often have problems handling singular cases of self-intersecting surfaces and tend to fail on large molecules (more than 10,000 atoms). We describe here a program called MSMS, which is shown to be fast and reliable in computing molecular surfaces. It relies on the use of the reduced surface that is briefly defined here and from which the solvent-accessible and solvent-excluded surfaces are computed. The four algorithms composing MSMS are described and their complexity is analyzed. Special attention is given to the handling of self-intersecting parts of the solvent-excluded surface called singularities. The program has been compared with Connolly's program PQMS [M. L. Connolly (1993) Journal of Molecular Graphics, *Vol. 11, pp. 139–141] on a set of 709 molecules taken from the Brookhaven Data Base. MSMS was able to compute topologically correct surfaces for each molecule in the set. Moreover, the actual time spent to compute surfaces is in agreement with the theoretical complexity of the program, which is shown to be O[n log(n)] for n atoms. On a Hewlett-Packard 9000/735 workstation, MSMS takes 0.73 s to produce a triangulated solvent-excluded surface for crambin (1crn, 46 residues, 327 atoms, 4772 triangles), 4.6 s for thermolysin (3tln, 316 residues, 2437 atoms, 26462 triangles), and 104.53 s for glutamine synthetase (2gls, 5676 residues, 43632 atoms, 476665 triangles).*
© 1996 John Wiley & Sons, Inc.

INTRODUCTION

Molecules are often represented as a set of overlapping spheres M, each having the van der Waals radius of the constituent atoms. The *van der Waals surface* is defined as the topological boundary of this set of spheres. Lee and Richards[1] defined the *solvent-accessible surface* (SAS) using a sphere of radius r_p called the probe and used to represent a single solvent molecule (Figure 1). This surface is traced out by the center of the probe as it rolls over the spherical atoms. It can also be perceived as the topological boundary of a set of spheres M' obtained from M by increasing the radius of each sphere by the value r_p. Richards[2] defined the *smooth molecular surface,* which consists of spher-

Received May 30, 1995; accepted August 2, 1995.
*To whom correspondence should be addressed.
Biopolymers, Vol. 38, 305–320 (1996)
© 1996 John Wiley & Sons, Inc.
CCC 0006-3525/96/030305-16

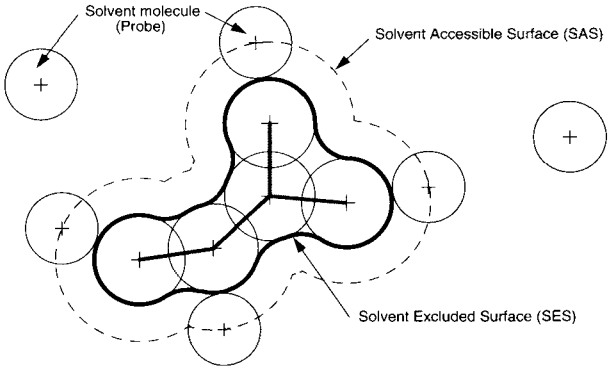


FIGURE 1 The solvent-accessible surface (SAS) is traced out by the center of the probe representing a solvent molecule. The solvent-excluded surface (SES) is the topological boundary of the union of all possible probes that do not overlap with the molecule.

ical patches lying either on the atoms or the probe, and toroidal patches defined by the probe when it rolls over pairs of atoms. Because such a description can lead to self-intersecting surfaces, we prefer to define it as the topological boundary of the union of all possible probes having no intersection with M. This is called the *solvent-excluded surface* (SES) as proposed by Greer and Bush.[3] Connolly[4] has given equations to compute the analytical description of that surface.

Because the SAS area is widely used in protein modeling to describe hydration effects,[5-8] a number of algorithms to compute the analytical representation of this surface have been described. Recently Tuñòn et al.,[9] and Jackson and Sternberg,[10] argued that the SES area provides a better model for describing this effect. This surface is also used to visualize and study molecular properties. The SES of molecules forming complexes fit each other very well at the interface region and therefore this surface can be used to discriminate between different ways of docking the molecules.[11] We are aware of only two programs that can compute the analytical representation of the SES: Connolly's programs PQMS[12] and his earlier AMS.[4] Other efforts to compute triangulated solvent-excluded surfaces include the program GEOPOL[13] proposed by Silla et al. and Varshney's parallel algorithm.[14] One of the main difficulties in computing the SES is the detection and handling of the self-intersecting parts of the surface. These pathological cases, called *singularities,* have been described by Connolly[15] and Sanner.[16] Zauhar and Morgan[18] proposed a method to circumvent such singularities. Connolly[15] described a number of such singular cases and gave a rough outline of an algorithm to handle them. Some of the properties relative to the singular edges and points given by Sanner[16] are recalled here and an algorithm based on those properties is described.

We present here a set of algorithms that improve both the speed and the reliability of the computation of molecular surfaces (SES, SAS). These algorithms have been implemented in a program called MSMS, which can produce both the analytical and a triangulated representation of the SES. From the analytical description of the SES and the SAS, accurate molecular surface areas are computed using formulae that are the same or similar to the one given by Connolly.[4] These surface computations are based on the use of the reduced surface introduced by Sanner[16] and that is briefly described. This surface is equivalent to an α-shape with an alpha value equal to the probe radius as described by Edelsbrunner.[17] The main differences with the approach proposed by Perrot et al.[19] to compute analytically the SAS area are discussed. Our program has been tested on a set of 709 molecules taken from the Brookhaven Protein Data Bank with sizes ranging from 33 to 43,632 atoms. The results of MSMS are compared with those obtained with PQMS for the same set of molecules.

METHODS

MSMS consists of four algorithms. The first computes the reduced surface of a molecule from which the second algorithm builds an analytical representation of the solvent-excluded surface that may be self-intersecting. The third algorithm removes all self-intersecting parts. The last algorithm produces a triangulation of the SES.

Reduced Surface Definition

Richards defined the SES by rolling a probe over a set of atoms. When the probe is simultaneously in contact with three or more atoms it is in a *fixed position* because it cannot roll further without losing contact with at least one of the atoms. The polygon obtained by connecting with line segments the centers of the atoms on which the probe is lying in a fixed position is a *face of the reduced surface* (RS-face). The atom centers are *vertices of the reduced surface* (RS-vertices) and the line segments are *edges of the reduced surface* (RS-edges). Figure 2 shows a probe in a fixed position and the RS-face (c_1, c_2, c_3) of the reduced surface it defines. If the probe can roll over a pair of atoms without colliding with a third atom, the RS-edge joining the centers of these two atoms will be called a *free RS-edge* because it does not belong to any

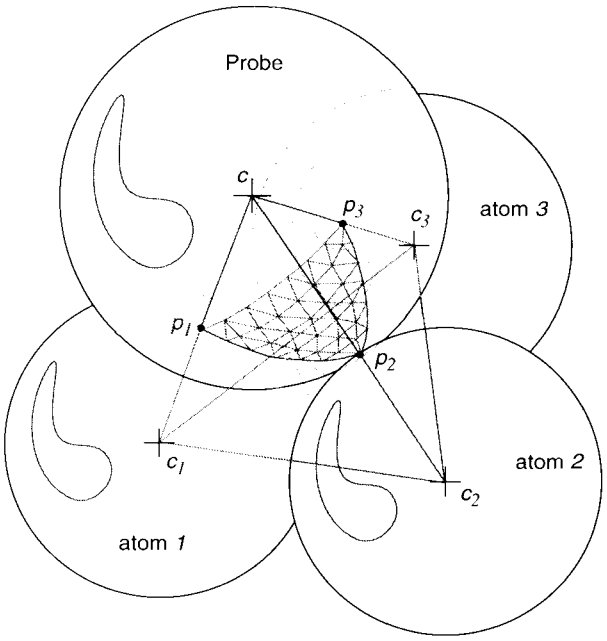


FIGURE 2 A probe in contact with three or more atoms is in a fixed position. Its center c is a vertex of the solvent-accessible surface. In such a position, a probe defines a face of the reduced surface (c_1, c_2, c_3) and a spheric reentrant face of the solvent-excluded surface (p_1, p_2, p_3). This face is a spherical triangle lying on the probe whose edges are defined by the intersection of the probe with the planes defined by the center of the probe c and the centers of the atoms c_1, c_2, and c_3.

face. Similarly, the center of an atom on which the probe can roll without touching any other atom will be called a *free RS-vertex*. The polygons associated with all fixed positions of the probe and the free edges and vertices define the reduced surface. Figure 3 shows the reduced surface of crambin. The reduced surface can be made of several closed surfaces connected by RS-vertices and/or free RS-edges. Such a combination of RS-elements is called an *RS-component*. Each free RS-vertex generates an RS-component made of just one point. Figure 4 shows a component of the reduced surface made of three closed surfaces connected to each other by one RS-vertex. Under the assumption that the probe is never in contact with more than three atoms at a time, the closed surfaces of the reduced surface are made of triangular faces. Geometrical properties of the reduced surface have been described previously.[16] In particular, one can show that for a *calibrated* set of n spheres and an *adequate* probe radius, the number of faces and edges of the reduced surface is of order n where n is the number of atoms in the molecule. A set of spheres is *calibrated* if the radius of the largest sphere (r_{max}) is less than twice the radius of the smallest sphere (r_{min}) and the distance between any of the sphere centers is greater than $r_{\min}$ divided by two. A probe radius is *adequate* if it is smaller than r_{max} and bigger than r_{min}. Most molecules consist of a calibrated set of spheres, and the usual value of the probe (1.5 Å) is adequate for such a set of spheres. Halperin[20] proved this result for any positive constants k and ρ such that $r_{max}/r_{min} < k$ and any ball concentric to a sphere of M of radius r_i and with radius $\rho \cdot r_i$ does not contain the center of any other sphere of M.

Reduced Surface vs Solvent-Excluded and Solvent-Accessible Surface

The reduced surface contains all the information necessary to build the SAS and the SES. Figure 2 shows an RS-face (c_1, c_2, c_3). The center c of the probe is a vertex of the SAS. In such a position, the probe generates a spheric reentrant face $F(p_1, p_2, p_3)$ of the SES. From any fixed position, the probe can roll over pairs of atoms connected by an RS-edge until it reaches its next fixed position (Figure 5). During this rotation, the center of the probe describes an arc of a circle (c, c'), which is an edge of the SAS, and the small geodesic arc connecting the contact points on the probe (p_2, p_3) sweeps out a piece of torus, called a *toric reentrant face* of the SES. The edges and the vertices of the SAS define the faces of the SAS, which are spherical patches lying on spheres centered on atomic positions, and whose radii have been augmented by the radius of the probe. For each face of the SAS, there is a contact face of the SES that is a spherical patch lying on the sphere representing an atom. The centers of spheres holding such faces are vertices of the reduced surface (Figure 6).

In the following we will denote by $a_i(c_i, r_i)$ the sphere of center c_i and radius r_i. The set of n spheres corresponding to a molecule is called $M = \{a_1(c_1, r_1), a_2(c_2, r_2), \ldots, a_n(c_n, r_n)\}$. The set of spheres obtained from M by increasing the radii by the radius r_p of the probe is called $M' = \{a'_1(c_1, r'_1), a'_2(c_2, r'_2), \ldots, a'_n(c_n, r'_n)\}$.

Reduced Surface Computation (Algorithm 1)

The reduced surface is computed using the algorithm proposed by Sanner.[16] The idea of "rolling" a probe over the atoms starting from an initial fixed position is similar to that used in the program MSEED proposed by Perrot[19] to compute the solvent-accessible surface. The algorithm takes as input a set M of n spheres and the radius of the probe r_p. It computes a triangulation of the reduced surface of M for r_p. During the initialization step of the algorithm an initial face $F(c_1, c_2, c_3)$ of the reduced surface is found. From this initial fixed position the probe can roll over every pair of atoms of F connected by an RS-edge, until it hits a third atom. This operation will be referred to as "the treatment of an RS-edge." Figure 5 shows the treatment of the edge (c_2, c_3) of the face (c_1, c_2, c_3). The atom centers c_2, c_3, c_4 define a new RS-face F', for which the edges c_2, c_4 and c_4, c_3 have to be treated. This recursive procedure stops when all RS-



FIGURE 3 Reduced surface of crambin (1crn) for a probe of radius 1.0 Å. The RS-vertices are the centers of the solvent-accessible atoms. This surface has a free RS-edge shown in green.

edges have been treated. The center c of the probe describes an arc of a circle (c, c') held by a circle C (not shown in the figure). All probes that are simultaneously in contact with atoms 2 and 3 have their center on C. For all atoms close enough to atoms 2 and 3, the program computes the center of the probe and selects that for which the arc (c, c') is smallest.

When this algorithm stops, a closed surface S of an RS-component has been identified. For each RS-vertex s_1 of S, (the center of atom a_1), the atoms close enough to a_1 to be bridged by the probe are selected. If there is such an atom a_2 whose center is not already connected to s_1 by an RS-edge, the program tries to find a third atom a_3 that does not already belong to this closed surface and such that s_1, s_2, s_3 form an RS-face. If such a face can be found, the algorithm restarts itself on that face and this RS-component will be *extended* by a new closed surface sharing only the vertex s_1 with S (Figure 4). If such a face cannot be found it is because the probe, tangent to the triple of atoms, is either always overlapping with another atom or because the probe can freely rotate around the RS-edge. In the latter case the two considered atoms form a free RS-edge that is added to the list of free edges of that RS-component (Figure 3). After an RS-component is extended or free RS-edges are created all new RS-vertices of the RS-component are tested for free RS-edges or component extension. This process is iterated until no more RS-vertices are added.

Free RS-vertices are atoms $a(c, r)$ such that the closest atom $a'(c', r')$ is further than $r + 2 \cdot rp + r'$ apart. Each such vertex generates a separate RS-component.

The user may specify 1, 2, or 3 atoms that the probe should be in contact with initially. If no atom is specified, the algorithm will automatically find a first face of the external RS-component as follows. The atom whose x coordinate minus its radius is minimal is an RS-vertex. Let us call this atom, $a_1(c_1, r_1)$, the left-most atom. To find a second atom $a(c, r)$ such that c_1, c is an RS-edge, we choose from the spheres of M' that intersect the sphere $a'_1(c_1, r_1 + r_p)$ that which has the left-most point on its intersection circle with a'_1 and we call it $a_2(c_2, r_2)$. For each atom, $a(c, r)$, close enough to c_1 and c_2, we check for both probes tangential to a_1, a_2, and a, if they overlap with any atom of M. As soon as a collision-free probe is found, the initial face is defined. If both positions of the probe are collision free, the one with the smallest x coordinate is used. If this method to find a first RS-face fails, the program tries the same method along another axis.



FIGURE 4 The reduced surface of a set of spheres is shown in cyan. This component is made of three closed surfaces connected to each other by one RS-vertex. The center of the spheres are shown in blue. The SES component corresponding to that RS-component is displayed in semitransparence. The violet spherical contact faces are lying on the spheres. There is one orange spheric reentrant face for each RS-face and one green toroidal reentrant face for each RS-edge. When the solvent-excluded surface intersects an RS-edge, singular points occur and when it intersects an RS-face singular edges occur. The contact faces held by the two atoms connecting pairs of closed surfaces of the reduced surface are bound by two distinct cycles of convex edges.

Data Structure. The atomic coordinates and radii of the molecule to be surfaced are stored in an array. Since all RS-vertices are atom centers, an index into this array is stored for each RS-vertex along with a list of pointers to the edges that this vertex belongs to. The RS-edges are stored in a linked list of C structures containing for each edge the index of the two vertices it connects, pointers to the two faces it belongs to, the center and the radius of the torus described by the probe when it rolls about the two atoms whose centers are the vertices of that edge. If the major radius of this torus is smaller than the radius of the probe, a *radial singularity* occurs in the solvent-excluded surface (Figure 7); the intersection points between the RS-edge and the probe are computed and stored with the edge. The edge structure also contains the rotation angle between the starting and ending positions of the probe, and the centers and radii of the two circles describing the contact points between the probe and the two atoms. The RS-faces are stored in a linked list of C structures containing, for each face, three pointers to its edges, three vertex indices, a vector normal to the face, and the coordinates of the probe center associated with that face.

Computational Complexity. The first vertex c_1, center of the atom a_1, can be found in $O[n]$ operations. To find the second vertex requires finding a_1's neighbors, which are the atoms close enough to a_1 to be touched by the probe as it rolls over a_1. Those are all atoms whose center is inside a sphere of center c_1 and radius $R = r_1 + (2 \cdot r_p) + \max(r_{i=1,n})$. Because atoms cannot come too close together, the number of such atoms has an upper bound independent of the size of the molecule.[21] To retrieve these neighbor atoms we use a data structure called a Binary Spatial Division (BSD-Tree), inspired by the tree-structured hierarchical subdivision of space proposed by Barnes and Hut.[22] This data structure is described in the section BSD-Tree where we give statistics for the number of neighbors and show that the average complexity to retrieve the neighbor atoms is $O[\log(n)]$. Since the number of neighbors is constant, the second atom can be found in $O[\log(n)]$ operations. To find a_3 one has to find all atoms that can be touched by the probe when it rolls over the two atoms a_1 and a_2. These atoms have their centers inside a sphere C of center c_t and radius $R' = r_t + r_p + \max(r_{i=1,n})$, where c_t and r_t are, respectively, the center and the major radius of the torus described by the probe when it rolls over a_1 and a_2. Again, the number of such atoms can be considered as constant and they can be found with an average number of operations in $O[\log(n)]$. This means that the average complexity to find the initial RS-face is $O[n]$.

Each RS-edge defines a torus of center ct and a sphere $S_{(ct,R')}$. To treat an edge requires finding the q atoms inside the sphere S, which can be done with an average complexity in $O[\log(n)]$. The selection of the atom such that the rotation angle of the probe about the edge is minimal is done in $O[1]$. Since all faces are triangles, this has to be done 1.5 times for each RS-face and since the number of RS-faces of a calibrated set of spheres such as a molecule is $O[n]$,[16] the complexity to compute the initial closed surface S of the outer RS-component is in $O[n \log(n)]$. Each RS-vertex of S is then tested for free edges or RS-component extension in $O[\log(n)]$. So the outer RS-component can be found in $O[n \log(n)]$ operations. Free RS-vertices can also be found in $O[n \log(n)]$ operations.

Analytical Solvent-Excluded Surface Computation (Algorithm 2)

After the reduced surface has been computed, the second algorithm builds the analytical representation of the

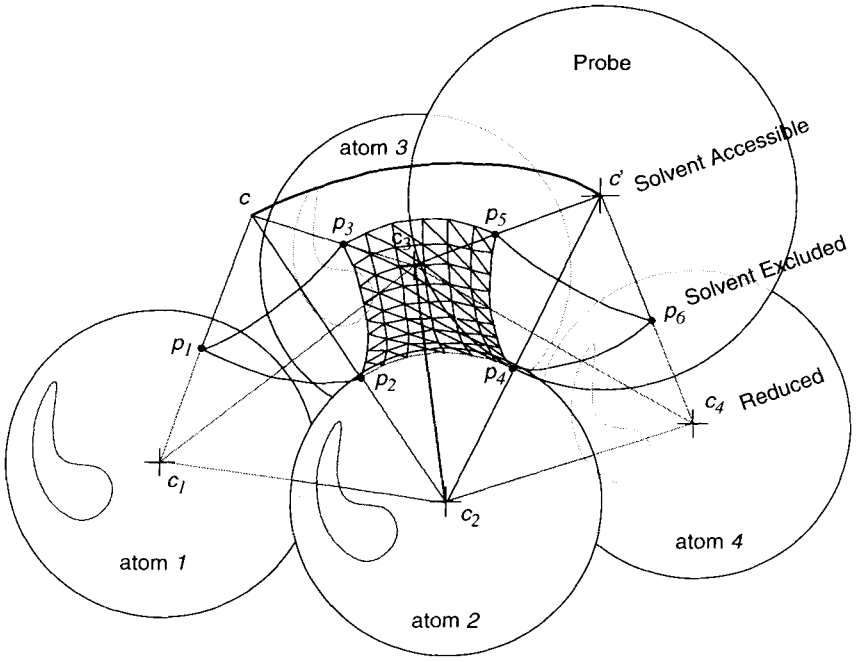


FIGURE 5 From a given fixed position c the probe can roll over pairs of atoms (here atoms 2 and 3). The geodesic arc p_2, p_3 sweeps out a part of a torus of axis c_2, c_3. The probe stops rolling over these two atoms when it hits a third atom (atom 4) that will define with atom 2 and 3 a new RS-face (c_2, c_4, c_3) and a new spherical reentrant face (p_4, p_6, p_5) for the SES. The center of the probe describes an arc of a circle (c, c') that is an edge of the SAS.

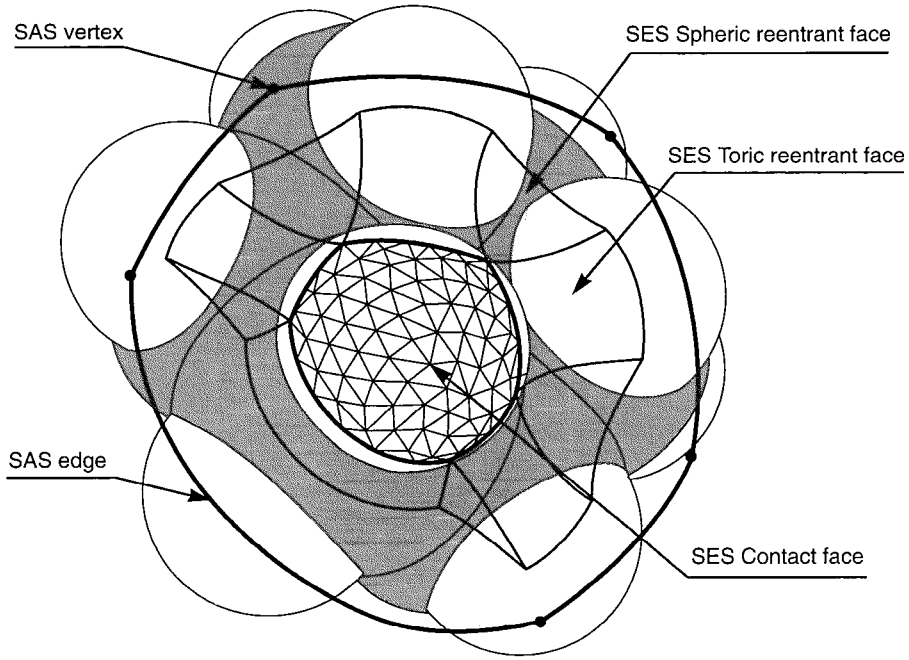


FIGURE 6 The boundary of the contact face whose triangulation is shown is defined by cycles of convex edges of the toroidal reentrant SES faces. Each vertex of the reduced surface (center of an atom) is associated to one or more SES contact faces and SAS faces.

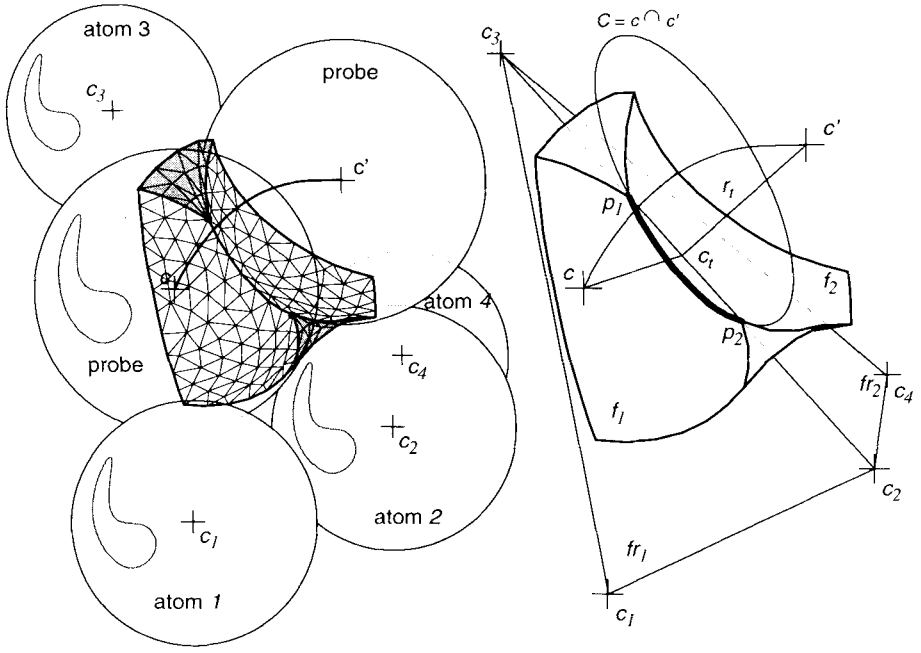


FIGURE 7 When the major radius r_t of the torus is smaller than the probe radius, the circle $C = c \cap c'$ intersects the RS-edge c_2, c_3 generating a radial singularity. The spheric reentrant faces f_1 and f_2 generated by the faces of the reduced surface fr_1 and fr_2 share a singular edge p_1, p_2 lying on C. The toric reentrant face associated with the edge of the reduced surface c_2, c_3 is split in two.

solvent excluded surface. This algorithm creates a spheric reentrant face for each RS-face. Figure 2 shows such a face (p_1, p_2, p_3) that is a spherical triangle lying on the probe whose position is defined by the RS-face (c_1, c_2, c_3). The vertices of the spheric reentrant faces are the contact points between the probe and the atoms. The edges are geodesic arcs defined by the intersection of the probe with the plane containing the probe center and the atom centers; for instance, p_1, p_2 is defined by the intersection of the probe with the plane c, c_1, c_2. For each RS-edge, this algorithm will create a toric reentrant face. Figure 5 shows a face (p_2, p_4, p_5, p_3) that is associated with the RS-edge c_2, c_3. These faces have concave edges (p_2, p_3 and p_4, p_5) that they share with spheric reentrant faces, and convex edges (p_2, p_4 and p_5, p_3) that they share with contact faces. Radial singularities as shown in Figure 7 are detected during the computation of the reduced surface and the singular points p_1 and p_2 are computed and stored in the structure describing an RS-edge. When this type of edge is treated, the created toric reentrant face is made of two triangular faces connected by a singular vertex or a singular edge. Once all the spheric reentrant and toric reentrant faces have been constructed, the cycles of convex edges that belong to toric reentrant faces define the contact faces (Figure 6).

Data Structure. The SES is stored using linked lists of C structures for the vertices, edges, and faces. For each vertex we store its coordinates, surface normal, and the index of the closest atom. An edge structure has pointers to its two vertices and its two faces, a vector orthogonal to the plane containing the edge, and the center and radius of the circle on which the edge lies. For the convex edges we also store a pointer to the corresponding RS-edge. An SES face structure includes the face type (spheric reentrant, toric reentrant, contact), a pointer to the associated element in the reduced surface (vertex, edge, face), an array of pointers to edges and vertex structures, and an array of booleans indicating the orientation of each edge in that face.

Computational Complexity. The complexity of this algorithm is a function of the number of faces and edges of the reduced surface. Since the reduced surface of a molecule made of n atoms has $O[n]$ faces and edges, the number of operations required to build the SES from the reduced surface is a linear function of the number of atoms.

Treatment of Singularities (Algorithm 3)

Sanner[16] showed that the only singular points that do not belong to the edges of spheric reentrant faces are those generated by singular free RS-edges, i.e., a probe intersecting an RS-edge and rotating 360° around that edge. It also was shown that an intersection between a probe in a fixed position and the associated RS-face is a

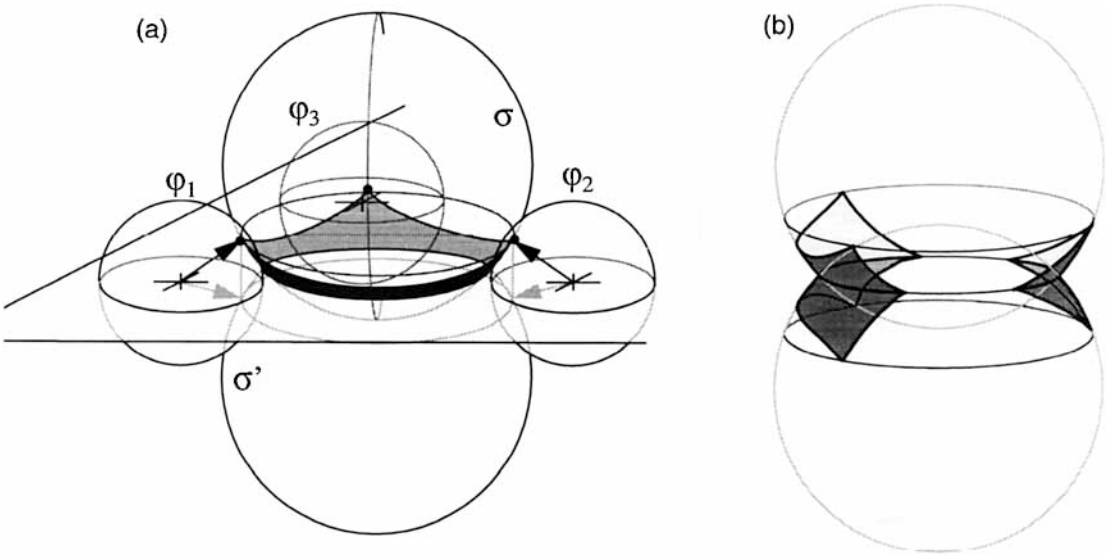


FIGURE 8 (a) The spheric reentrant face generated by the probe σ is partially eaten by a probe σ'. The intersection circle does not cut the face's edges. (b) The intersection circle cuts two edges of the face. Each reentrant face is split in two faces in this case.

necessary condition for singularities to occur. Such probes are detected and tagged during the generation of the reduced surface. Algorithm number 2 processed the radial singularities that involve only probes connected by an edge of the SAS. Nonradial singularities occur when a part of the surface generated by algorithm 2 is inside a probe in a fixed position, in which case we say that the probe "eats" a part of a face of the SES. The nonradial singularities are classified in three distinct categories that are treated separately. First, the *coincident* RS-faces generating singularities are treated. RS-faces F and F' are *coincident* if, for each vertex of F, there is a vertex of F' that has the same coordinates. These faces generate singularities when the probes they define intersect the plane of the RS-face (Figure 8). The intersection circle can cut 0, 1, 2, or 3 edges of the spheric reentrant faces. If no edge is cut (Figure 8a), a singular circle is created. The case where one edge is cut is ignored because this singularity falls into the next category. If the intersection circle cuts two edges (Figure 8b), each of both spheric reentrant faces is divided into a triangular and a quadrilateral face. Finally, the case where all three edges of the reentrant faces are cut is eliminated by deleting both RS-faces. Both spherical reentrant faces would be cut into three small spheric reentrant triangles, which can be disregarded.

To handle the second category of singularities, we go over the list of singular edges created by algorithm 2, and for each of them we check if a part of this edge is inside any probe. If not, this edge does not intersect the surface and it is kept as it is. If this edge, or part of it, is inside a probe, it is entirely or partially "eaten," and we compute the intersection points with the probes eating the biggest part of that edge. Figure 9 shows the most general case where a singular edge $a(s_1, s_2)$ is partially eaten by two probes σ_3 and σ_4. The edge a belongs to two spheric reentrant faces f_1 and f_2 lying respectively on the probes σ_1 and σ_2. At first, two singular vertices ns_1 and ns_2 are created (if there are not already vertices at those positions) and the edge $a(s_1, s_2)$ is split into two edges $a(s_1, ns_1)$ and $na(ns_2, s_2)$ that belong to the faces f_1 and f_2. Since σ_3 and σ_4 "ate" the biggest part of a, a and na cannot be inside any other probe and therefore they need no further testing. Since σ_3 is the probe that eats the greatest part of the initial edge a on s_1's side, we create two open-ended edges a_1 and a_2 starting at ns_1 and lying on the circles $\sigma_1 \cap \sigma_3$ and $\sigma_2 \cap \sigma_3$. In the same way we create two open-ended edges a_3 and a_4 starting at ns_2 and lying on the circles $\sigma_1 \cap \sigma_4$ and $\sigma_2 \cap \sigma_4$. Each of these open-ended edges are created only if the spheric reentrant faces that they would connect do not already share an edge. If the created edges a_1 and a_3 have an intersection, we create a singular vertex ns_3 that will end both edges at the intersection point; otherwise a_1 will stop at its intersection with the sphere σ_4 and a_3 at its intersection with the sphere σ_3. The same algorithm is applied to find ending vertices for a_2 and a_4. If the initial edge a was eaten entirely, ns_1 is equal to s_1 and ns_2 is equal to s_2 and both edges a and na are deleted. The edges a_1, a_2, a_3, and a_4 are added to the list of singular edges to be tested.

Once all singular edges have been tested, the only remaining singularities are those where two spheric reentrant faces F and F' lying respectively on the probes σ and σ' have an intersection circle $C = \sigma \cap \sigma'$, which does not intersect the edges of the face F or F', and the probes are not lying on the same three atoms. This situation generates a singular circle connecting the two faces F and F'.

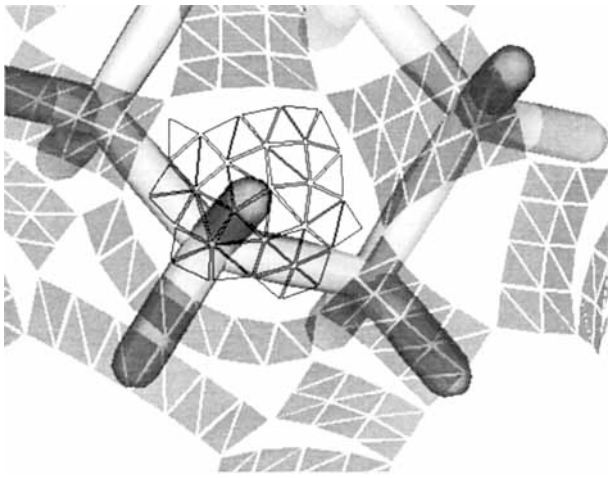


FIGURE 10 Close up on the triangulation of a contact face. A pretriangulated template sphere is placed on the atom. The triangles and the vertices inside the SES contact face are selected. The toric reentrant faces are shown as filled triangles.

Computational Complexity

The first step handles the RS-faces having the same vertices that are found during the computation of the reduced surface. Therefore, the computational cost for this algorithm is a linear function of the number of such coincident faces in the reduced surface.

In the second step, each singular edge of the SES is checked for intersections with probes in fixed positions. The number of probes to consider is constrained by the fact that only a fraction of them are capable of generating singularities, and we only have to consider the ones that are close enough to intersect the edge. These probes are selected using a BSD-Tree structure.

For the last step, we have to find intersections between probes able to generate singularities and lying on different triples of atoms. These spheres are also found using a BSD-Tree. The experimental results show that the time spent in this part of the program is negligible and behaves as a linear function of the number of atoms (Figure 15).

Triangulation of the Solvent-Excluded Surface (Algorithm 4)

The analytical description of the SES produced by the previous algorithm can be triangulated with a user-specified density of vertices. This algorithm first triangulates the toric reentrant faces. After this operation all nonsingular edges of the solvent-excluded surface have a discrete representation as a set of segments of comparable size. The singular edges are then partitioned into sets of segments. The spheric reentrant and contact faces are triangulated using pretriangulated template spheres of the same radius as the sphere holding the face. For each atomic radius and for the probe radius a template sphere is triangulated at the specified vertex density. All triangles on each template belong to the convex hull of the set

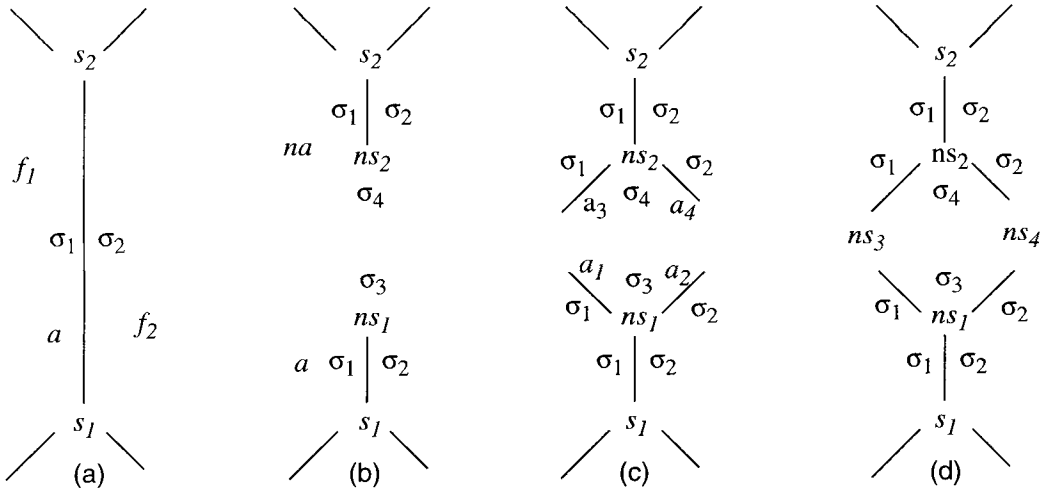


FIGURE 9 General case of nonradial singularity handling. (a) The singular edge (s_1, s_2) belongs to the spheric reentrant faces f_1 and f_2 lying on probe spheres σ_1 and σ_2. (b) This edge is partially eaten by two probes σ_3 and σ_4. Two singular vertices ns_1 and ns_2 are created and the edge a is split in two (s_1, ns_1) and (ns_2, s_2). (c) Open-ended singular edges are added on the circles $\sigma_i \cap \sigma_j$ if the faces f_i and f_j are not already neighbor faces. (d) The vertices ns_3 and ns_4 ending the newly created edges are added. The edges (s_1, ns_1) and (ns_2, s_2) are correct and need no more testing. The edges connecting the new vertices are added to the list of singular edges to be tested.

of vertices on that template. The pretriangulated template is placed on the sphere holding the face. The template vertices and triangles inside the face of the SES are selected (Figure 10). Two oriented contours are created, the first one made of the list of segments representing the face's edges, the second one being the perimeter of the selected triangles of the template sphere. These two contours are then connected using triangles that belong to the convex hull of the vertices of that face (Figure 11).

BSD Tree. The complexity of computing molecular surfaces depends on the number of operations required to retrieve the atoms that can be touched by the probe as it rolls over a pair of atoms. To find these "neighbor" atoms, we use a data structure called a BSD-Tree. This tree is obtained by recursively dividing the set of atom centers into two subsets of comparable size. The division is made using a plane orthogonal to the largest axis of the bounding box. For each node of the tree we store the dimension along which the box has been subdivided and the coordinate value used to divide the set of points. The subdivision stops when each box contains less than a given number of atom centers called the *granularity* of the tree. For a set of n atoms the structure obtained is a binary tree of height $O[\log(n)]$. When we compute such a tree for a molecule, the atom centers are stored in an array ATOM of size n. To subdivide a box, one has to rearrange the atom centers inside the box around the median element in such a way that all atoms in the subboxes are contiguous in the array ATOM. The complexity of that operation is $O[p]$ for p points, and since every atom belongs only to one box, at each level of the tree each atom will be considered once. Since the height of the tree is $O[\log(n)]$, this tree can be computed in $O[n \log(n)]$ operations (Figure 12). A granularity of 10 atoms per box was chosen for our application. The space requirement for such a structure is of order n.

The number of atoms in a sphere of given radius has an upper bound that is independent on the molecule size.[21] Figure 13 shows the distribution of the minimal, average, and maximal number of atoms that can be touched by the probe sphere when it rolls over two atoms generating surface. These data were obtained from a set of 709 proteins without hydrogen atoms and for a probe sphere radius of 1.5 Å. The maximum number of neighbors found is 86 and the average ranges from 20 to 39 with an average over the different molecules of 34. We only considered atom pairs involved in the outer component of the solvent-excluded surface. These numbers are larger for atoms buried in the core of the protein. The time to retrieve these neighbor atoms as a function of the size of the molecule is shown in Figure 12. To retrieve the atoms within a distance d from a given point P one must enter the tree at the root and, at each node, test if the cube of center P and of edge length $2 \cdot d$ overlaps with one or two of the subboxes of that node. If we assume that d is much smaller than R, which is the radius of the smallest sphere containing all the points, one can expect that for $\log(R/d)$ nodes, only one subbox will be selected. In the remaining binary subtree, the number of nodes that will be visited is $2 \cdot k - 1$, where k is the number of neighbor atoms found in a sphere of radius d. So, the neighbors of an atom can be found in $O[\log(R/d) + k]$ operations. If d is equal to R, all atoms are selected and the complexity is $O[n]$. But when the molecule gets larger, R increases, and since d is fixed (typically below 7 Å), with a fixed number of neighbors k, one can expect the logarithmic behavior we observe in Figure 12.

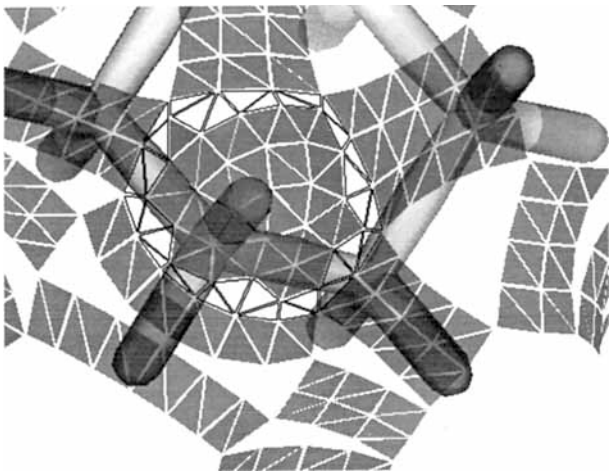


FIGURE 11 The faces selected on the template are sewn to the edge defined by the convex arcs of the toroidal faces using triangles that belong to the convex hull of the set of vertices lying on that atom.

APPLICATION

The solvent-excluded surfaces of a set of 709 molecules have been computed using MSMS and PQMS for comparison. Figure 14 shows the SES computed by MSMS for some of these proteins along with the SES of some nucleic acids. These molecules have been selected from the Brookhaven Protein Data Bank to provide a wide range of protein sizes including variations of the same structure and all x-ray structures having more than 5000 atoms. The size of the molecules ranges from 33 atoms for a thermolysin substrate (7tmn.pdb) to 43632 atoms for glutamine synthetase (2gls.pdb). The surfaces were computed for a probe radius of 1.5 Å. MSMS has been tested on the following platforms: DEC Alpha 3000/500, HP 9000/735, SGI Challenge, and Indigo 2 with the density parameter set to 1.0 vertex/Å^2. PQMS was run on a DEC 5000/240 with the angle parameter set to 1.5. For

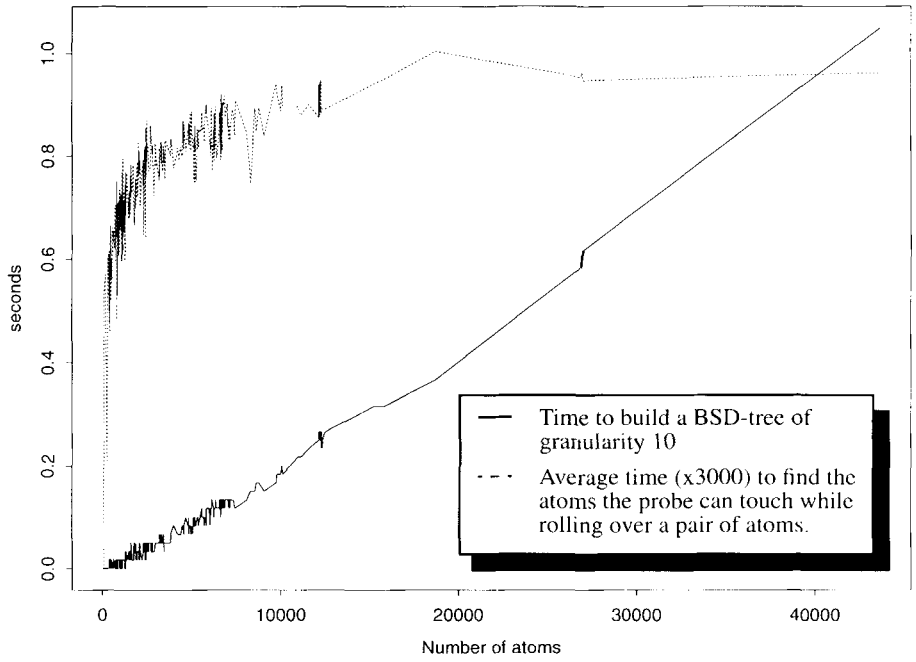


FIGURE 12 The solid line shows the time to build a BSD-Tree of granularity 10 as a function of the size of the molecules. The dashed line shows the experimental average time (multiplied by 3000) to retrieve the atoms that can be touched by a probe of radius 1.5 Å that rolls over pairs of solvent-accessible atoms. These data have been measured while computing the outer components of the reduced surface of our set of 709 proteins. The logarithmic behavior is in agreement with the theoretical complexity.

the external component of the solvent-excluded surface, simple tests were performed on the triangulation to verify that every vertex belongs to, at least, two edges and that each edge belongs to exactly two triangles. In Table I we report under "Good surfaces" the cases where the programs found a surface that passed the diagnostic tests. PQMS generated 603 "good surfaces" out of 709 molecules (85%) whereas MSMS computed all of them correctly. The two rows "Problem in SES computation" and "Problem in triangulation" in Table I show the number of molecules for which the programs stopped without generating a triangulated surface. This happened with PQMS (computation of the analytical representation of the SES) for 10.7% of the molecules, and with TRB (Connolly's program to triangulate the analytical SES computed with PQMS) for 4.2% of the structures.

When MSMS and PQMS detect errors or inconsistencies that they cannot correct, they decide to modify atomic radii, typically by adding 0.1 Å to the radius of the atoms generating the problem, and restart the computation with this new set of spheres. It is best to minimize the use of this procedure for the following two reasons: it substantially increases the computation time and it modifies the initial input data. In Table II we report the number of times each program restarted the computation 1, 2, or 3 times on the same molecule. MSMS restarted the computation for 84 molecules with a maximum of 2 restarts on the same one. Although these numbers are smaller for MSMS than for PQMS, they are still quite large. We anticipate them to decrease with further refinement of the singularity handling algorithm. Table II also shows the number of atoms (over all molecules) with modified radii. PQMS changes fewer atoms (486) than MSMS (542) but the changes are more significant, since in PQMS 10 atoms have their radii increased by 0.3 and 50 by 0.2 Å while MSMS only adds 0.2 Å to the radii of 24 atoms.

For visualization purposes it is important to avoid very thin triangles, which are poorly rendered. Some of them are inherent to the surface definition, but the number of such triangles generated during the triangulation process should be minimized. The percentage of triangles for which the length of the normal vector, obtained by cross product of the two vectors defined by two triangle

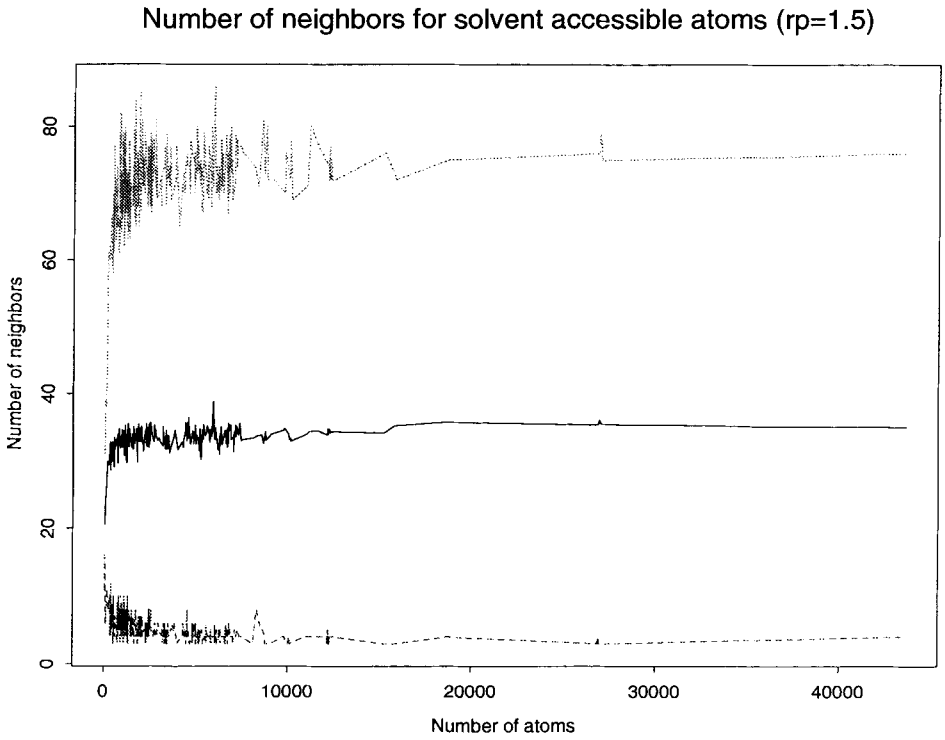


FIGURE 13 Minimum, average, and maximum number of atoms inside a sphere of radius $r_t + (2 \cdot r_p) + r_m$ and of center c_t, where r_t is the radius of the torus described by the probe when it rolls over two atoms and c_t is the center of this torus, r_m is the largest atomic radius in the molecule, and r_p the probe radius (in this case 1.5 Å). These data are the number of neighbors found while computing the outer components of the reduced surface of our set of 709 proteins.

edges, was less than 10^{-8} Å, is reported in the last row of Table II. As one can see, MSMS produces significantly fewer thin triangles than PQMS.

Figure 15 shows the central processing unit time in seconds used by the different algorithms comprising MSMS for all molecules in our testing set. The time to compute the reduced surface, the solvent-excluded surface, to handle the singularities and to triangulate the solvent-excluded surface were measured once even when the program restarted the computation. This was done to show that the measured complexity of the algorithms fits the theoretical one. The total time includes the time spent to recompute the surfaces in the case of a restart. This explains why the total time is, for some molecules, much larger than the sum of the timings of the individual algorithms. Nevertheless, MSMS is shown to be very fast since it computes a triangulation of the solvent-excluded surface of 2gls (46632 atoms, 476656 triangles) in 51.48 + 13.36 + 5.39 + 34.3 = 104.53 s. These times were obtained on an HP 9000/735 with 80 Mbytes of memory and running at 99 MHz. The program had no hardware-dependent optimization.

DISCUSSION

MSMS computes the solvent-excluded surface from a triangulation of the reduced surface that is found by rolling a probe over the set of sphere representing a molecule. The idea of rolling the probe over the molecule rather than computing all of its possible fixed positions was used previously by Perrot et al.[19] Their program, MSEED, computes an analytical representation of the SAS. Because MSEED rolls the probe from a fixed position to the next one, it cannot detect free RS-edges (SAS edges without vertices; Figure 3). Therefore it is known that MSEED can miss a part of the surface when two RS-components are connected by a free RS-edge. But this can also happen when two closed surfaces of the reduced surface are connected by only one vertex (Figure 4). In our set of proteins, and for a probe radius of 1.5 Å, we have found 1 case of a free RS-edge bridging two closed surfaces of the reduced surface for a total number of free RS-edges of 370, and 1 case of closed surfaces sharing only one RS-vertex, but these numbers would increase

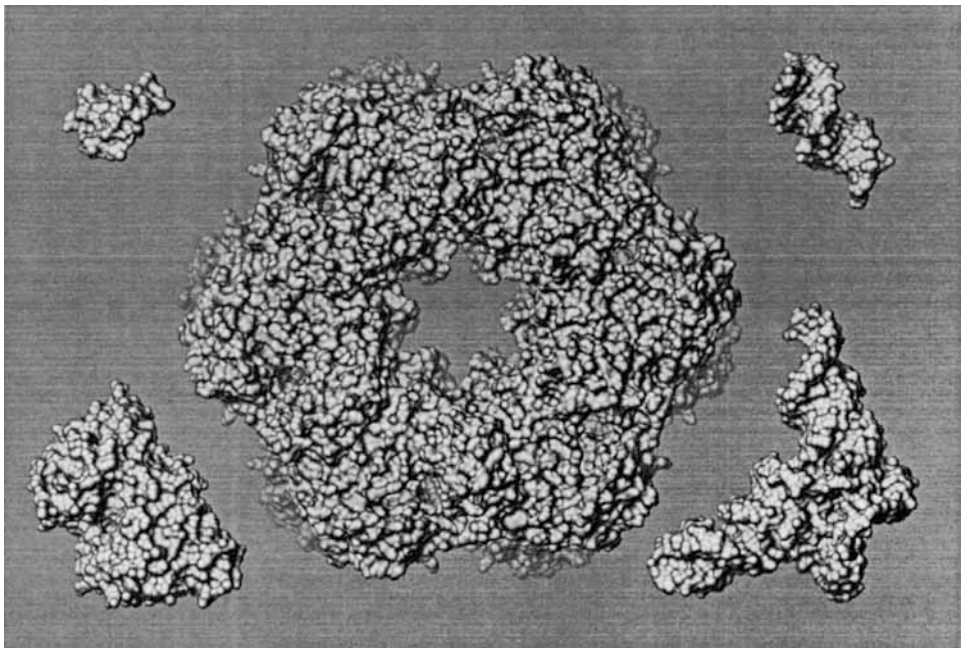


FIGURE 14 Some examples of molecular surfaces computed by MSMS. The upper left molecule is crambin (1crn, 46 residues, 327 atoms). Triangulated with a density of 6.0 vertices/Å^2, the surface is made of 24,590 triangles. The lower left molecule is thermolysin (3tln, 316 residues, 2437 atoms, density 3.0, 67,786 triangles). The center molecule is glutamine synthetase (2gls, 5676 residues, 43,632 atoms, density 1.0, 494,586 triangles). This is the largest entry in the Brookhaven Protein Data Bank. This three molecules belong to the set of 709 molecules used to test MSMS. The upper right molecule is a deoxyribonucleic acid (8bna, 12 base pairs, 486 atoms, density 3.0, 19,682 triangles). The lower right molecule is a transfer ribonucleic acid (5tra, 76 bases, 1821 atoms, density 3.0, 70,926 triangles).

with a smaller probe that is more likely to roll over atom pairs without touching a third atom.

The introduction of the reduced surface and the study of its properties[16] has been very useful in the implementation of MSMS and the analysis of its complexity. Moreover, the reduced surface is a very compact way to store all the information required to build either the SES or the SAS since its vertices are atom centers, whose coordinates have to be stored anyway, and its edges and faces can be described by pairs and triples of integer atom indices.

By default, MSMS computes the outer component of the molecular surfaces (reduced, solvent-excluded, solvent-accessible), but it can also compute a particular component if the user specifies 1, 2, or 3 atoms the probe should touch initially, or even compute automatically all surface compo-

Table 1 Comparison of the Results of PQMS vs MSMS for the Computation of a Set of 709 Molecules Taken from the Brookhaven Protein Data Base

	PQMS: Dec 5000/240	MSMS		
		Dec Alpha 3000/500	HP 9000/735	SGI Challenge
Good surfaces	603	709	709	709
Wrong surfaces	0	0	0	0
Problem in SES computation	76	0	0	0
Problem in triangulation	30	0	0	0

Table II Comparison of the Number of Cases Where PQMS and MSMS Restarted the Computation

	PQMS: Dec 5000/240	MSMS		
		Dec Alpha 3000/500	HP 9000/735	SGI Challenge
Restarts				
1	75	79	81	81
2	16	6	6	6
3	4	0	0	0
Atom radii modifications				
+0.1	426	518	519	519
+0.2	50	23	24	24
+0.3	10	0	0	0
Total number of restarts/molecules	118/95	90/83	93/87	93/87
Small triangles[a]	0.29%	0.084%	0.081%	0.081%

[a] Triangles such that the length of the normal vector obtained by cross-product of two of its edges is smaller than 10^{-8} Å.

nents, i.e., the outer component and one for each internal cavity. However, in the latter case there is a performance penalty because, in the current implementation, no special technic is used to find the initial position of the probe for internal cavities. This process can be speeded up by a smart selection of triple of atoms to be considered for the initial positioning of the probe. Once the RS-components have been computed, each of them is used to build an initial analytical model of the corresponding SES component. In this first model only radial singularities are handled. Next the singularity-handling algorithm takes care of the nonradial singularities that are the most troublesome part of the

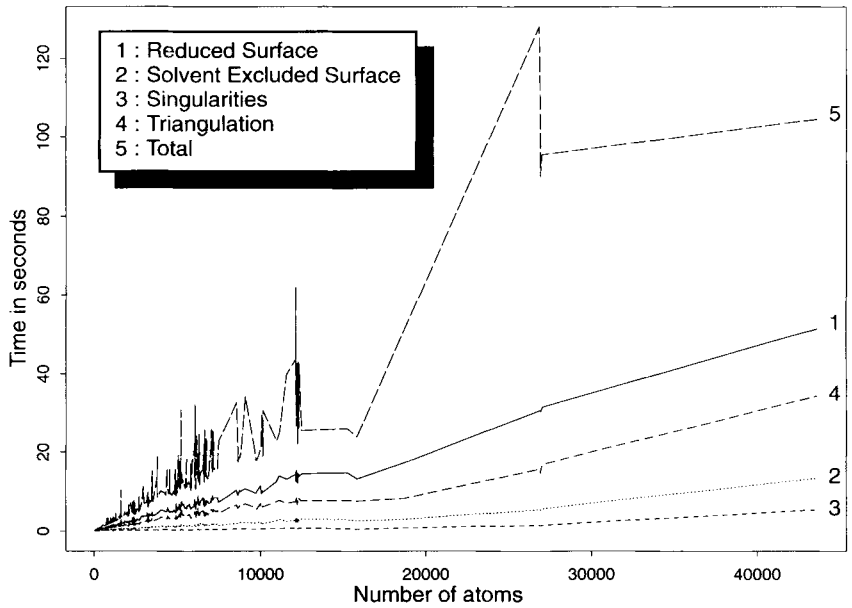


FIGURE 15 Time to compute triangulated solvent-excluded surfaces with MSMS for a density of 1.0 vertex/Å^2 and a probe radius of 1.5 Å. These times have been measured on a HP 9000/735 with 80 Mbytes of memory and running at 99 MHz. The times for the computation of the reduced surface (1), solvent-excluded surface (2), singularities (3), and triangulation (4) are reported for 709 molecules. The total time (5) includes the time spent to recompute the surfaces in the case of a restart. This explains why it is sometimes significantly larger than the sum of the four basic algorithms.

computation and are the major cause of restarts. Our algorithm to handle self-intersecting parts of the SES resolves all of these singularities in only one pass, unlike the one implemented in PQMS, which needs several passes and prints a message if more passes might be required. Moreover, in MSMS the time spent to handle these singularities is almost negligible [Figure 15(3)]. The SES computed by the two programs are also different when two SES components intersect. PQMS merges them into one component even though they correspond to nonconnected SAS components, in other words, the probe cannot move from one cavity to the other. To be consistent with our definition of the surface components, MSMS does not do this. The triangulation algorithm uses pretriangulated spherical templates for the spherical faces. The parts of the template that are found to be inside a spherical face are selected and connected to the edges of that face using triangles that belong to the convex hull of all the vertices of that face. In this way we avoid creating very thin triangles that are badly rendered. Although MSMS produces fairly regular triangulations, there are a number of small triangles caused by our assumption that no probes can contact more than three atoms at a time. These very thin faces could be avoided by using an SAS multiple vertex approach as proposed by Eisenhaber.[23] Each component of the triangulated SES can be stored separately. For each component one file contains the vertex coordinates along with the analytically computed normal vector and the index of the closest atom. Another file contains the indices of the triangle's vertices as well as, for each triangle, the number and the type of the analytical SES face it belongs to. MSMS also computes analytically the solvent-excluded and solvent-accessible surface areas using equations that are the same as, or similar to, the equations given by Connolly.[4] These areas are reported for each atom. The computation of the genus of the analytical solvent-excluded surface and of its triangulated representation turned out to be a very inexpensive and powerful test for surface correctness.

Further developments of MSMS include options to save the reduced surface as well as the analytical representation of the SES and to produce a triangulation of the SAS. We also consider modifying the program in order to correct only the parts of the reduced surface that are concerned by the modifications of atomic radii in the case of a restart. The number of such cases should decrease once all computations are done in double precision and the singularity handling algorithm will be further refined. MSMS is written in the C programming language. It is implemented as an AVS[24] module and as a stand-alone program that accepts input from either files or Unix socket connections. The stand-alone version of MSMS can also be operated with a Tcl/Tk interface.

CONCLUSION

The reduced surface provides a compact way to store geometric information that can be used to build several molecular surfaces: van der Waals, solvent-excluded, and solvent-accessible surface. It enables the checking of atom or residue exposure to the solvent, and computes accurate surface areas and volumes. This surface contains all the information needed to decide what is inside or outside a specific molecular surface, so that it can be used to check the position of water molecules relative to a protein or to select lattice points inside or outside a molecule. We have shown how to compute this surface with an algorithm whose actual performance fits the theoretical complexity analysis. This was made possible by the use of the binary spatial division tree, which is a very general data structure enabling the efficient retrieval of atoms in the neighborhood of any point. BSD-Trees have applications in several fields such as clustering techniques and density control, and can be used in any dimension. Our algorithm to handle the singularities has a higher reliability than that used in PQMS. The entire MSMS package has been shown to be efficient and reliable since it invariably succeeded in producing a topologically correct surface. The speed of this program enables interactive recomputation of dynamic molecular surfaces for small proteins undergoing molecular dynamics or normal mode motion. Higher speed could likely be achieved by optimizing the code, since to this point only automated compiler optimization has been done. The program behaves remarkably well with large molecules where the efficiency of the implementation of the algorithm is crucial. With the growing number of molecular surface computation programs, we felt it was important to have a program that reliably computes mathematically well-defined molecular surfaces. Moreover, because of the growing size of the molecules and molecular complexes of known three-dimensional structure, the program should work well on large molecules. The program MSMS will be made available by the authors upon request (olson@scripps.edu or spehner@univ-mulhouse.fr).

This work was started while Michel F. Sanner was employed by Sandoz Pharmaceutical in Basel, Switzerland, as a graduate student in the molecular modeling group and finished while employed as a Research Associate in the Molecular Graphics Laboratory of The Scripps Research Institute in La Jolla, California. We would like to thank Armin Widmer (Molecular Modeling group, Sandoz A.G.) for providing an efficient implementation of the binary spatial division tree, and Bruce Duncan (Molecular Graphics Lab., TSRI) who provided the program used to check the correctness of the triangulated surfaces. This work was supported, in part, by NIH grant number P01GM38794. This is publication number 8982-MB from The Scripps Research Institute.

REFERENCES

1. Lee, B. & Richards, F. M. (1971) *J. Mol. Biol.* **55,** 379–400.
2. Richards, F. M. (1977) *Ann. Rev. Biophys. Bioeng.* **6,** 151–176.
3. Greer, J. & Bush, B. L. (1978) *Proc. Natl. Acad. Sci. USA* **75,** 303–307.
4. Connolly, M. L. (1983) *J. Appl. Cryst.* **16,** 548–558.
5. Vila, J., Williams, R. L., Vásquez, M. & Scheraga, H. A. (1991) *Proteins Struct. Funct. Genet.* **10,** 199–218.
6. Palmer, K. A. & Scheraga, H. A. (1991) *J. Comp. Chem.* **12,** 505.
7. Still, W. C., Tempczyk, A., Hawley, R. C. & Hendrickson, T. (1990) *J. Am. Chem. Soc.* **112,** 6127.
8. Zheng, C., Wong, C. F. & McCammon, J. A. (1990) *Biopolymers* **29,** 1877–1883.
9. Tũñon, I., Silla, E. & Pascual-Ahuir, J. L. (1992) *Protein Eng.* **5,** 715–716.
10. Jackson, R. M. & Sternberg, M. J. E. (1995) *J. of Mol. Biol.* **250,** 258–275.
11. Duncan, B. S. & Olson, A. J. (1994) personal communication.
12. Connolly, M. L. (1993) *J. Mol. Graph.* **11,** 139–141.
13. Silla, E., Villar, F., Nilsson, O., Pascual-Ahuir, J. L. & Tapia, O. (1990) *J. Mol. Graph.* **8,** 168–172.
14. Varshney, A., Wright, W. V. & Brooks, F. P., Jr. (1994) *IEEE Comput. Graph. Appl.* **15,** 19–25.
15. Connolly, M. L. (1985) *J. Appl. Cryst.* **18,** 499–505.
16. Sanner, M. F. (1992) Ph.D. dissertation thesis, Université de Haute-Alsace, France.
17. Edelsbrunner, H. & Mucke, E. P. (1992) NCSA Report No UIUCDCS-R-92-1734, Computer Science Department, University of Illinois, Urbana.
18. Zauhar, R. J. & Morgan, R. S. (1990) *J. Comput. Chem.* **11,** 603–622.
19. Perrot, G., Cheng, B., Gibson, K. D., Villa, J., Palmer, K. A., Nayeem, A., Maigret, B. & Scheraga, H. A. (1992) *J. Comp. Chem.* **13,** 1–11.
20. Halperin, D. & Overmars, M. H. (1994) *Proc. 10th ACM Symp. Comput. Geom.* 113–122.
21. Varshney, A. (1994) Ph.D. dissertation thesis, University of North Carolina, Chapel Hill, NC 27599-3175.
22. Barnes, J. & Hut, P. (1986) *Nature* **324,** 446–449.
23. Eisenhaber, F. & Argos, P. (1993) *J. Comput. Chem.* **14,** 1272–1280.
24. Upson, C., Faulhaber, T., Jr., Kamins, D., Laidlaw, D., Schlegel, D., Vroom, J., Gurwitz, R. & vanDam, A. (1989) *IEEE Comput. Graph. Appl.* **9,** 30–42.