Enveloping Triangulation Method for Detecting Internal Cavities in Proteins and Algorithm for Computing Their Surface Areas and Volumes

JÁN BUŠA,[1,2] SHURA HAYRYAN,[1] CHIN-KUN HU,[1,3] JAROSLAV SKŘIVÁNEK,[1,4] MING-CHYA WU[1,5]
[1]*Institute of Physics, Academia Sinica, Nankang, Taipei 11529, Taiwan*
[2]*Department of Applied Mathematics, Technical University in Košice, Slovakia*
[3]*Center for Nonlinear and Complex Systems and Department of Physics, Chung Yuan Christian University, Chungli 32023, Taiwan*
[4]*John von Neumann Institute for Computing, Forschungszentrum Jülich, 52425 Jülich, Germany*
[5]*Research Center for Adaptive Data Analysis, National Central University, Chungli 32001, Taiwan*

Received 2 August 2007; Revised 25 April 2008; Accepted 27 May 2008
DOI 10.1002/jcc.21060
Published online 15 July 2008 in Wiley InterScience (www.interscience.wiley.com).

Abstract: Detection and quantitative characterization of the internal cavities in proteins remain an important topic in studying protein structure and function. Here we propose a new analytical method for detecting the existence of cavities in proteins. The method is based on constructing the special enveloping triangulation enclosing the cavities. Based on this method, we develop an algorithm and a fortran package, CAVE, for computing volumes and surface areas of cavities in proteins. We first test our method and algorithm in some artificial systems of spheres and find that the calculated results are consistent with exact results. Then we apply the package to compute volumes and surface areas of cavities for some protein structures in the Protein Data Bank. We compare our calculated results with those obtained by some other methods and find that our approach is reliable.

© 2008 Wiley Periodicals, Inc. J Comput Chem 30: 346–357, 2009

Key words: protein cavity; enveloping triangulation; CAVE; SMMP; surface area; volume

Introduction

Native folds of proteins represent densely packed structures, and this close packing is a major contributing factor to the stabilization of the native state.[1] However, despite the seemingly obvious advantage of optimal packing, native structures of many proteins have cavities.[2–5] Elucidating the role of cavities in function and energetics of the proteins has been a challenge in experimental and computational studies of proteins in the last two decades. Experiments on site-directed mutagenesis[6–10] have shown that changing the size and shape of the cavities influences considerably the stabilization energy of the protein structures. Some investigations[11] have showed that even the function of the protein can be changed with the sizes of the cavities.

Computational algorithms for detection and quantitative characterization of the cavities are based usually on the space-filling geometry model of the protein by Lee and Richards,[12] which interprets a protein as the union of overlapping spherical balls in three-dimensional space. Each ball represents an atom, and its size is defined by the van der Waals radius. The algorithms belong to the family of shape description algorithms and are closely related to those for calculating the surface area and the volume of molecules.[12–24] The first computational investigation of the cavities has been reported by Lee and Richards[12] who have modified for this purpose their own algorithm for surface and volume calculation. Rashin, et al.[25] have modified the algorithm by Shrake and Rupley[24] to develop a program for detection of the internal cavities and to predict the positions of buried water molecules. They investigated 12 high-resolution protein structures and predicted successfully the location of 80% of internal water molecules. Hubbard et al.[26,27] have analyzed internal cavities in 121 protein chains and observed that the overall cavity volume increases with protein size, though it remains only a small fraction of total protein volume. They examined

Correspondence to: C.-K. Hu; e-mail: huck@phys.sinica.edu.tw

Contract/grant sponsor: National Science Council of the Republic of China (Taiwan); contract/grant number: NSC 96-2911-M 001-003-MY3

Contract/grant sponsor: National Center for Theoretical Sciences (Taiwan)

Contract/grant sponsor: Academia Sinica (Taiwan); contract/grant numbers: AS-92-TP-A09, AS-95-TP-A07

the “empty” and water-containing cavities and found that the latter possesses a more polar surface. The cavities are nearly always present in proteins more than 100 residues in size. In their calculations they used algorithms from refs. 17, 18 and 28. They extended their research to investigate the cavities at protein interfaces. Zhang and Hermans[29] used the molecular surface (MS) calculation algorithm[17] to calculate energies and free energies of a water molecule in cavities and discuss the hydrophobicity of protein cavities. The energetics of empty cavities and internal mutations have been evaluated computationally as in ref. 30.

Based on the alpha shape theory,[31–33] which provides a quantitative method to describe and compute shapes at multilevels of detail in three-dimensional space, Liang et al.[34] proposed an analytically exact method for computing the metric properties of macromolecules. Later, they extended this method to study quantitatively the inaccessible cavities in proteins.[35] An analytical algorithm has been described in ref. 36. Procedures for identifying the buried water molecules in cavities are described in refs. 37 and 38. The surface triangulation method[39,40] has been applied by some authors to study the structure and interactions of proteins.[41] Currently, the cavity detection algorithms are included in some simulation packages mentioned in refs. 42–44. However, the source codes of these packages are not easily available. Thus, it is of interest to develop new algorithm and package to study cavities in proteins and other macromolecules, which could be easily available to researchers.

In the present paper, we propose a new analytical algorithm for detecting closed cavities in the interior of the single protein or at the interface of two or many molecules and computing their surface area and volume. The basic idea of the proposed method lies in the construction of a special enveloping triangulation, such that the conclusion if any point in the space belongs or does not belong to the cavity, depends only on the relation between the point and the triangulation. On the basis of this method, we develop a fortran package, CAVE, for computing volumes and surface areas of cavities in proteins. Our objective for this work is to develop our own cavity detection software for the protein simulation package SMMP,[45,46] which has been used widely to study structures and thermal properties of proteins.[47,48]

We first test our method and algorithm in some artificial systems of spheres and find that the calculated results are consistent with exact results. Then we apply the package to compute volumes and surface areas of cavities for some protein structures in the Protein Data Bank (PDB)[49] using different sets of van der Waals atomic radii. We compare our calculated results with those obtained by some other methods. The testings have shown that our algorithm is efficient, accurate, and reliable.

Methods

Basic Definitions and Concepts

1. *Molecule and its atoms.* $\mathcal{M} = \bigcup_i \mathcal{S}_i$ is the union of N overlapping spherical balls $\mathcal{S}_i$ $(1 \le i \le N)$ in three-dimensional (3D) space E_3; $\mathcal{S}_i$ represents the i-th atom and $\mathcal{M}$ can represent a molecule, e.g. a protein. We denote the surface of the atom $\mathcal{S}_i$ by $\delta\mathcal{S}_i$.
2. *Cavity.* A cavity $\mathcal{C}$ of the molecule $\mathcal{M}$ is a nonempty, bounded, and closed domain in E_3, whose full boundary consists of the parts of the molecular boundary, and whose interior, Int $\mathcal{C}$, has no common points with the molecule $\mathcal{M}$.

   Here we should make an important remark. When talking about the atoms we mean that their van der Waals radii are extended by the probe ball (usually the water molecule) radius. This means that the number, sizes, and shapes of cavities depend strongly on the probe radius (see Fig. 1). Different authors use different probe radius for their particular purposes. Also, however strange, currently there is not even a unique agreement about the van der Waals radii of protein atoms, and several sets are being used. This problem is discussed in ref. 50, which contains an extensive list of related references.
3. *Wall triangle.* Let C_1, C_2, C_3 be the center points of three atoms in $\mathcal{M}$. Then the triangle $\Delta C_1C_2C_3$ is called a wall triangle in $\mathcal{M}$ if and only if the intersection $\delta\mathcal{S}_1 \cap \delta\mathcal{S}_2 \cap \delta\mathcal{S}_3$ of the surfaces of the atoms is not empty and at least one of the intersection points does not belong to the interior of M, Int $\mathcal{M}$ (lies on the MS). See Figure 2 for an example.

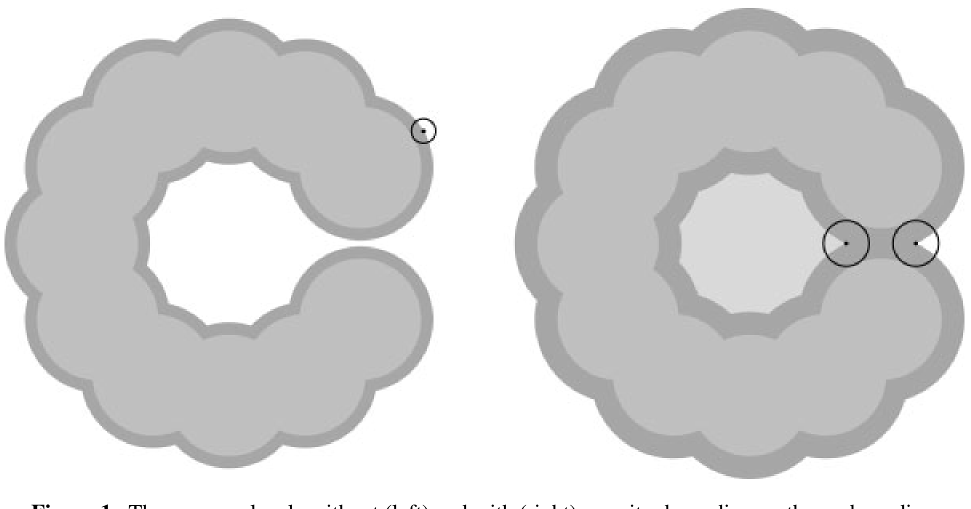

Figure 1. The same molecule without (left) and with (right) a cavity depending on the probe radius.

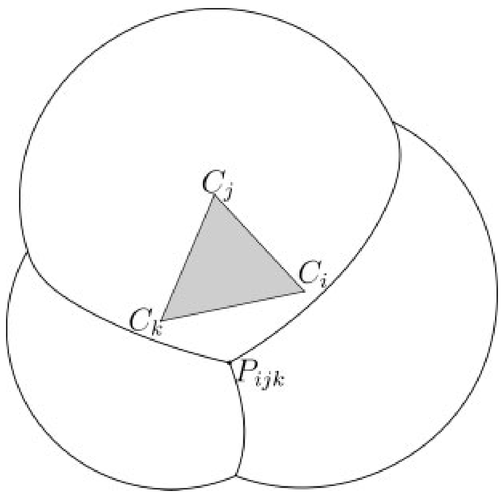

Figure 2. A wall triangle $\Delta C_iC_jC_k$. $\mathcal{P}_{ijk}$ is the intersection point of three spheres, and it lies on the surface of the molecule. Collecting wall triangles to enclose all cavities in a protein is an important step in enveloping triangulation method for detecting internal cavities in that protein.

4. *Neighboring wall triangles.* Two triangles are called neighbors if they share a common edge. The planes of the neighbor triangles form a dihedral angle. Two vertices of the neighbor triangles coincide, and the third one is called free vertex.
5. *Triangulation.* Triangulation means a special set of polyhedrons with triangular faces that enclose all cavities of the protein. The triangulation of the molecule $\mathcal{M}$ is denoted by $\Delta\mathcal{M}$.

   Triangulation $\Delta\mathcal{M}$ is a set of triangles which can be nonempty even if no cavity exists in $\mathcal{M}$. However, we can construct a minimal covering triangulation such that it is empty if there exists no cavity.

   Note: It is obvious that the empty triangulation will imply the lack of cavities.
6. *Oriented triangle.* Suppose the vertexes of all triangles are enumerated. If we view perpendicularly to the triangle's plane and find that the order numbers of vertexes increase in counterclockwise direction, then we say that the triangle is positively oriented; otherwise, the triangle is negatively oriented. Obviously, a positively oriented triangle will turn into an negatively oriented one when viewed from the opposite side of the plane. See Figure 3 for an example.
7. *Enveloping triangulation.* The enveloping triangulation $\Delta\mathcal{M}$ for cavities (briefly envelope triangulation) of a protein (or a system of balls) $\mathcal{M}$ is either an empty set or it is such a set of wall triangles in $\mathcal{M}$ with coincident inward orientation that
   i. $\Delta\mathcal{M}$ consists of a family of closed polyhedrons $\mathcal{P}_i$ whose facets represent inwardly oriented wall triangles such that $\text{Int}\,\mathcal{P}_i \cap \text{Int}\,\mathcal{P}_j = \emptyset$, for all $i \neq j$,
   ii. any cavity point lies inside some of these polyhedrons,
   iii. if any point from $E_3 - \mathcal{M}$ is not in any cavity of $\mathcal{M}$ then it lies outside the triangulation $\Delta\mathcal{M}$.

   Note: The term inwardly oriented in "(i)" mentioned earlier means that the triangle is positively oriented when viewed from the interior of the polyhedron.
8. *Cavity volume and surface area.* There are two basic definitions of the cavity volume and surface area. When the probe is being rolled over the van der Waals surface of the atoms, exposed to the interior of the cavity, its center creates a solvent accessible (SA) surface. The volume of the corresponding body is called SA volume. At the same time, the front of the probe sphere which is in touch with the van der Waals surface of the atoms, creates a MS, and the volume of the corresponding body is called MS volume of the cavity. We can use CAVE to calculate the SA surface area and volume of a cavity based on analytic equations, and calculate numerically the MS volume of the cavity.

Proposition 1. *If $\Delta C_1C_2C_3$ is a wall triangle in $\mathcal{M}$ with corresponding spheres $\delta\mathcal{S}_1$, $\delta\mathcal{S}_2$, $\delta\mathcal{S}_3$, then the planes of intersections of all pairs of these spheres intersect with the plane of the centers $C_1(x_1, y_1, z_1)$, $C_2(x_2, y_2, z_2)$, $C_3(x_3, y_3, z_3)$ in the single point (x, y, z), which is, consequently, the unique solution to the following system of linear equations*

$$(x_2 - x_1)x + (y_2 - y_1)y + (z_2 - z_1)z = \frac{1}{2}\left(r_1^2 - r_2^2 - x_1^2 + x_2^2 - y_1^2 + y_2^2 - z_1^2 + z_2^2\right)$$

$$(x_3 - x_1)x + (y_3 - y_1)y + (z_3 - z_1)z = \frac{1}{2}\left(r_1^2 - r_3^2 - x_1^2 + x_3^2 - y_1^2 + y_3^2 - z_1^2 + z_3^2\right)$$

$$\begin{vmatrix} x - x_1 & y - y_1 & z - z_1 \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{vmatrix} = 0, \tag{1}$$

where $r_i, i = 1, 2, 3$ are the radii of the spheres.

The first two equations in the system of equations mentioned above represent the planes of intersections of spheres, and the third is the equation of the plane $C_1C_2C_3$. Moreover, this solution falls into each ball of the triplet.

The 2D Analogue of the Enveloping Triangulation

Some aspects of triangulation can be visualized by introducing the two-dimensional (2D) analogue of the molecule. Here the balls are replaced by disks, spheres by circles, and the triangles are reduced to the segments of straight line, and the edges of the triangles are reduced to points (Fig. 4).

The centers of two intersecting disks are connected by a straight line segment if one of the intersection points lies on the outside

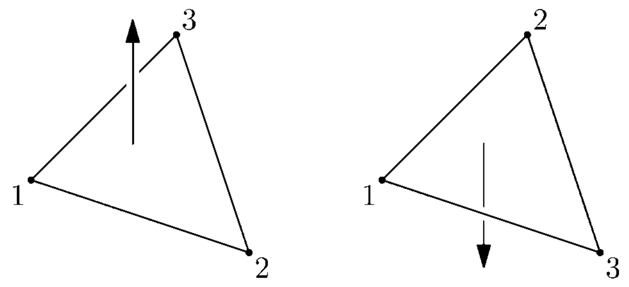

Figure 3. Positively (arrow up) and negatively (arrow down) oriented triangles when viewed in the direction, perpendicular to the triangle's plane from above.

boundary of two disks. In this way we obtain some special 2D net. There is always a closed route (envelope) in this net that encompasses the internal cavity (the bold polygon on Fig. 4) if the cavity exists. The circles which intersect with only one other circle can be safely removed without destroying the envelope. We shall call such circle (a sphere in 3D case) a "lug." Removing one lug may give rise to a new lug which must be removed again. Our goal is to construct a minimal closed cycle (envelope) of connections which will enclose all arcs which are parts of the boundary of some cavity. By minimal it means that removing any connection would destroy the cycle.

The Cavity Test for the Arcs

The boundary of the molecule consists of two types of surfaces. The outer boundary is the van der Waals envelope of the molecule, and the inner boundary includes the surfaces of atoms forming cavities. Both boundaries consist of circular arcs (or spherical segments in 3D case). The complete list of the circular arcs of the boundary is assembled by the algorithm for calculating the SA surface area.[23,51] In order to detect the cavity the algorithm has to distinguish between the arcs (the caps) which constitute the outer boundary and the ones that belong to the boundary of the cavity. In other words, one has to make a decision whether the given arc (cap) is external or internal with respect to the envelope. This is equivalent to the decision whether the central point of the arc lies inside the envelope or not. Such decision can be made by computing the sum of the oriented angles under which each straight line segment of "triangulation" is seen from a given point. For inner points this sum equals to 2π (point P_1 on Fig. 4) while for the outer ones it is 0 (point P_2). Similar criterion is applied for the 3D case by using the notion of spherical angle. The method to calculate the spherical angles will be given in Appendix.

Construction of the Outermost Enveloping Cycle

Our goal is to construct an "envelope triangulation," based on the net of connections between the centers of the intersecting circles. To assemble the outermost cycle, we can start with one outer "triangle" (say the southernmost), and then go on by adding at each step one new outer "triangle" (see Figs. 4 and 5).

The Algorithm for Constructing the Envelope Triangulation

We assume that all spheres are enumerated by assigning numbers $1, 2, \dots, N$ to their central points. The algorithm for constructing the envelope triangulation consists of the following steps.

1. *Preparing the preliminary list of the wall triangles.* The i-th sphere and the j-th sphere with radius r_i and r_j, respectively, are neighbors if their separation r_{ij} satisfies the relation: $r_{ij} \leq r_i + r_j$. We first list all possible wall triangles in $\mathcal{M}$ by using neighborhood relations for all pairs of spheres. Using the neighborhood list we check possible triplets, if they satisfy the system of equations (1) just in one point. Triplets with unique solution may be of two kinds: (a) if this solution lies inside of all three spheres, and if at least one of the intersection points does not lie inside of any sphere of $\mathcal{M}$ then we get a wall triangle and we add it to the preliminary list; (b) otherwise, the triplet is not a wall triangle. See Figure 5 for an example of the preliminary set of "wall triangles" in two dimensions.

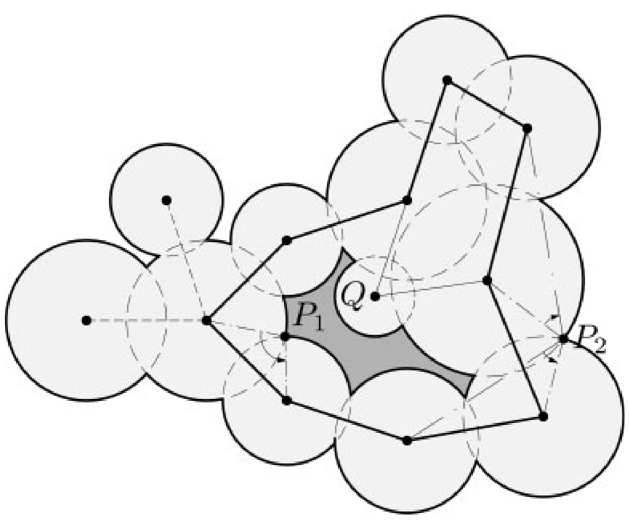

Figure 4. Definition of triangulation in $2D$ space.

2. *Constructing dynamic data matrix* `WTRMAT` *for wall triangles.* Each row in the matrix represents one triangle and has the following structure: (a) columns 1–3: vertices of the given triangle in increasing order; (b) several columns starting from the 4th contain the order numbers of the neighbors at each edge: the first neighbor at the 1st side, the second neighbor at the 1st side, so on, the *maxtsn*-th neighbor at the 3rd side; (c) next three columns contain the total number of the neighbors at each side; (d) the last column contains a special status of the triangle: *status* $= 0$ means that the given triangle has been removed from the list during the lug-cutting procedure, *status* $= 1$ means that the triangle has not been processed yet. Other values of the given variable will be discussed later. The initial value is set equal to 1. The variable *maxtsn* mentioned earlier denotes the maximum possible number of neighbors at one side of the triangle.
3. *Lugs cutting.* Any wall triangle from the list which has a free side (no neighbors at that side) is removed from the list by setting its *status* to 0. The order numbers of the vertices of such triangle are deleted from all rows of its neighbors, and the corresponding total numbers of neighbors are corrected. After this operation, new triangles with free side may appear and the whole process must be repeated iteratively until each of the remaining triangles has neighbors on each side.
4. *Determination of a unique outer neighbor for each side.* Suppose the first triangle in triangulation is picked up. This may be, for example, the "southernmost" triangle in the preliminary list. Since this triangle may have more than one neighbors at the given edge, then one needs a criterion to determine which neighbor to choose as the next member of the actual triangulation. The proper choice is the neighbor that forms a maximal dihedral angle with the given one from the point of view of inward orientation. Here the problem is that in the beginning of the procedure it is not known which orientation of the triangles must be considered as inward orientation. So, we need to define the neighboring triangles for each side for both orientations. This information is stored

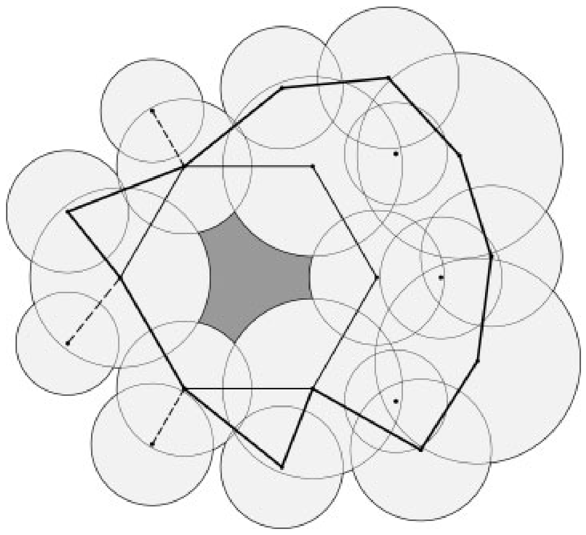
Figure 5. Preliminary set of “wall triangles” in two dimensions.

in the matrix NTOUT, whose rows describe triangles; the possible neighbors for one orientation are written to the columns 1–3 (orientation of the triangle given by the increasing order numbers of its vertices) and the columns 4–6 contain possible neighbors for another orientation.

At this point, it is checked whether there are more than one triangles forming a maximum angle with the actual triangle. Such multiple triangles are removed and only one is left. This special situation with multiple neighbors arises when the centers of more than three balls produce a set of triangles for which the edges form a complete graph. The triangle to be deleted is chosen arbitrarily. Deletion of some triangle activates the deletion of some of its neighbors.

5. *Starting a new envelope.* We start with some wall triangle from the preliminary list of wall triangles. A convenient choice is the southernmost triangle (the one for which the south poles of its vertex spheres include the southernmost point in $\mathcal{M}$).

   For this purpose, we first find all triangles with the corresponding southernmost South pole vertex. If there are more than one such triangles, then we choose the one whose normal vector with nonnegative third component forms the minimum angle with the vector $(0, 0, 1)$ (note that the triangle whose plane is vertical to the (x, y)-plane, forms a maximum angle $\pi/2$ with the vector $(0, 0, 1)$). If there are again more than one such triangles, then the choice is arbitrary. When the choice of the first triangle is done, the inward positive orientation of this triangle is defined. If the increasing order numbers of its vertex spheres is corresponding to the positive inward orientation (see Fig. 3 left), then the orientation is set to 0; otherwise it is equal to 1.
6. *Looking for a new neighbor.* The list of triangles of the next envelope triangulation component is made in successive steps. The first triangle of this list is the southernmost triangle, described earlier. It will be the first actual triangle. In the next step, one new neighbor triangle from the preliminary wall triangles list with corresponding orientation is drawn by using the matrix NTOUT, and it is added to the list. When a new neighbor is added to the actual list of the component, it is deleted from the preliminary list of the wall triangles (its status is set to 2). For this newly added triangle, all side bindings with other triangles from the actual component's list are checked. This information is stored to the triangles list. When all sides of the actual triangle are occupied by neighbor triangles a new actual triangle in the component's list, with at least one free side is chosen.
7. *Completion of the envelope triangulation component.* At some step all triangles in the list of the actual component will have all sides occupied. Now the new envelope triangulation component is completed. All triangles of the component are written to the output list and acquire status 3 in the preliminary list of wall triangles. All other triangles from the preliminary list are checked if some vertex lies inside the actual envelope triangulation component. If yes, then such a triangle is removed from the list (its status is set to 0).

   When the component is collected, we can check, if it contains a cavity, or not. According to the definition, at least one of the intersection points of three spheres of each wall triangle must be external to $\mathcal{M}$. If any of these points lies inside the triangulation polyhedral component, then a cavity belonging to this triangulation component exist (let us call these Z-type points). Otherwise, there is no cavity inside.

   So, after each cycle of assembling the envelope triangulation component the number of triangles from the preliminary list with status equal to 1 becomes smaller and smaller. When less than four such triangles are left, the outer envelope triangulation is completed.

Determination of the Minimal Covering Triangulation

The set of all Z-type points defines the list of all wall triangles and the corresponding spheres which are partially exposed into some cavity. These spheres form a set of closed domains each of which contains exactly one cavity inside, and hence, defines a minimal covering polyhedron of wall triangle. A 2D example of the minimal covering triangulation with marked internal vertices is shown in Figure 6. It can be compared with the outer triangulation in Figure 4 or in Figure 5 (the internal cycle).

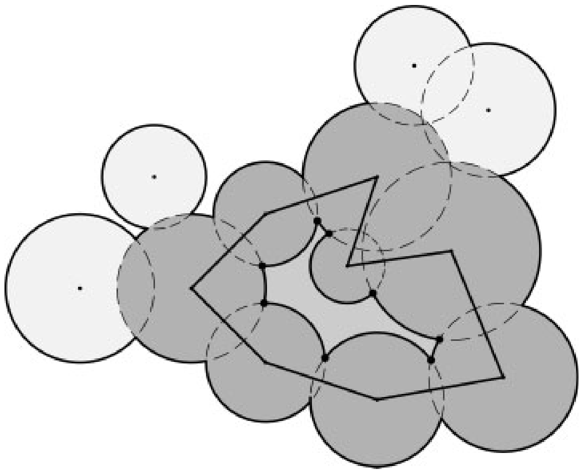
Figure 6. Minimal cavity triangulation.

The minimal covering triangulation may contain internal lugs. In this situation, the 2D and 3D cases are essentially different. In the 2D case, all boundary atoms are vertices for some "wall triangle," while in the 3D case, there may exist internal lugs or even internal links composed of atoms that are not vertices of any wall triangle. Detection of such atoms is important when the volume and surface area of the cavities are to be calculated. So, at the end of the construction process, it is necessary to check whether the centers of some atoms fall into the cavity bulk.

For this purpose, following steps are implemented:

1. Once more we fill the `NTOUT` list. For each triangle
   - Orientation of the triangles is defined using the internal triangle's point—the first three columns of each row are the triangle vertices and their order defines the inward orientation of the triangle;
   - Positions from 4 to 6 are the neighbor triangles (forming the minimum inner angle with the actual oriented triangle in the case when the side has more than one neighbor) for each side of the actual triangle.
2. The boundary triangles of the cavity are written to `ITTT`. The neighborhood relations are being checked. When the boundary is closed, the triangles are written to the list `IIT`. Information about the number of triangles for the given cavity and the starting index of the triangles in the list `IIT` are stored to the lists `NC` and `ISC`, respectively.
3. All atoms inside the cavity triangulation segment are determined.

Testing the Method in Artificial Systems of Spheres

In this section, we describe some tests for the accuracy of the algorithm. For this we construct an artificial systems of spheres that certainly contain/do not contain cavities. Unfortunately, there are no known analytical formulas for detection and quantitative calculation of the cavities. This means that one is not able to check the accuracy of the program by calculations "by hand." So, we have to devise some other kind of tests.

Checking the Position of the Space Points with Respect to the System

One of the robust tests is to check whether the program is able to detect the existence of the cavity and to tell the position of a given space point with respect to the cavity. We did this test with different numbers of the spheres and with different sets of space points. Here we bring the result of such a test for the system of six spheres. In our tests, we shall distinguish four possible positions of an arbitrary point with respect to the system:

1. The point is outside of the molecular envelope formed by the outer boundary.
2. The point belongs to the interior space of some atom.
3. The point is outside the molecule but within a distance less or equal to the solvent radius.
4. The point belongs to the interior of some cavity.

Our system consists of six spheres, labeled by $i = 1, 2, 3, 4, 5,$ and 6; every sphere has radius 10, i.e. $r_i = 10$, $i = 1, 2, \dots, 6$. The centers of the spheres are positioned on the six half-axes x, y, z at distances $d = 11$ from the origin (0,0,0). It is obvious that there is a cavity around the point $(0, 0, 0)$. Now we input the coordinates of five points and run the program. Here is the result.

For the solvent radius $r_w = 0$

```
1      .000000    .000000    .000000  4
2      .000000    .000000    .800000  4
3      .000000    .000000   5.000000  2
4    21.200000    .000000    .000000  1
5    22.000000    .000000    .000000  1
```

And for $r_w = 0.4$

```
1      .000000    .000000    .000000  4
2      .000000    .000000    .800000  3
3      .000000    .000000   5.000000  2
4    21.200000    .000000    .000000  3
5    22.000000    .000000    .000000  1
```

Here the first column is the order number of the point, the next three columns contain the coordinates, and the last one shows the position of the point with respect to the system according to the definition given earlier.

One can see that the first point is inside an internal cavity, the second point is inside the cavity, but in a neighborhood of the system (not farther than the solvent radius), the third lies inside an atom, the fourth point is, again, outside the system but in its neighborhood, and the fifth one is outside the system at all.

Detection and Quantitative Calculation of the Known Cavity

The next test on accuracy is to ask the algorithm to detect the known cavity or to report on its absence, with further calculation of quantitative parameters. Here we bring an example with 26 spheres and two different values of their radii. First we take $r_1 = 0.8$, and $r_2 = 0.7$

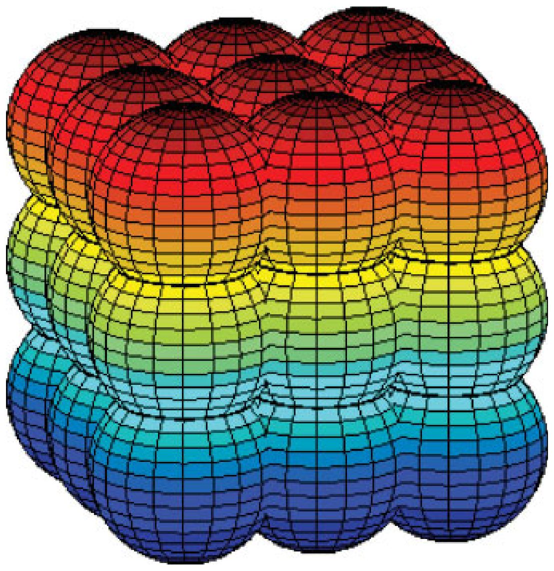

Figure 7. Spheres arranged in the form of cubic lattice.

The centers of the spheres are positioned at points $(i, j, k) \neq (0, 0, 0)$, $i, j, k \in \{-1, 0, 1\}$ (see Fig. 7).

Case 1. Let us first set all radii of the spheres to $r = 0.8$. This systems certainly contains a cavity. The output of the program is

```
Number of spheres:  26
Solvent radius:      0.000000000000000 E +000
Total/maximal number of neighbors: 216/20000
Triangulation size: 48
There exists a cavity!
Inaccessible volume:     37.036248686213050
Accessible area:         66.748085985837720
Cavity volume:       1.183073021905514 E -001
Surface area of cavity:   3.035763953402949
```

Case 2. Now, if we put a new sphere at the point (0, 0, 0) with radius $0.9445 \leq r_0 \leq 1.5455$, the cavity must disappear. The real output is

```
Number of spheres:   27
Solvent radius:      0.000000000000000 E +000
Total/maximal number of neighbors: 268/20000
Triangulation size:  48
There is no cavity!
Inaccessible volume:     37.036248686213050
Accessible area:         66.748085985837720
```

Case 3. Now we return to case 1, but set the radii of the spheres to $r = 0.7$; the results are

```
Number of spheres:  26
Solvent radius:      0.000000000000000 E +000
Total/maximal number of neighbors: 96/20000
Triangulation size:  0
There is no cavity!
Inaccessible volume:     29.715277712754660
Accessible area:         75.649551098442220
```

One can see that when the radii are small, the "cube" becomes perforated like a cheese and there are no closed cavities anymore.

Results

On the basis of the algorithm presented earlier, we have developed the fortran code, CAVE, and used it to calculate the number, volumes and surface areas of the cavities in several protein structures from PDB.[49] Calculations of van der Waals surface areas and volumes have been done for 11 molecules using CAVE and some other packages. The comparative data for areas and volumes are summarized in Tables 1 and 2, respectively.

Table 1 lists the van der Waals surface areas computed with CAVE, ARVO,[51] alpha shape-based analytical method VOLBL/CAST by Liang et al.,[34] analytical method MSDOT by Connolly,[15,17,52] and GEPOL method by Pascual-Ahuir and Coworkers.[53,54] Table 2 lists the van der Waals volumes computed with CAVE, ARVO, GEPOL, VOLUME,[42] and VOLBL. Please note that ARVO[51] was developed by us; we can use it to calculate volume and surface area of a protein, but we cannot use it to detect the existence of cavity.

Table 1 shows that our results coincide with the high accuracy with the values, obtained by well-known packages. The small differences may be due to slightly different values of the van der Walls radii. Different authors sometimes modify the values of radii for different reasons depending on certain problem under consideration. Table 2 shows that results obtained by using CAVE, ARVO, GEPOL, and VOLBL are also quite consistent, but the results obtained by using VOLUME are much larger than those obtained by using other methods. It is valuable to check carefully the computing algorithm or program of VOLUME.[42]

After this test, we calculate the location, surface area, and volumes for cavities in proteins. In Table 3 we show the results of these calculations and compare them with those obtained by using two other methods.[25,35] The columns of the table contain the following information: (1) the identification code of the protein in PDB; (2) the number of cavities detected by our program CAVE using the Richard's set of the atomic radii and the SA model; (3) the same obtained by using the radii set from ref. 25; (4) the same, obtained by the VOLBL package;[42] (5) the same, obtained by Rashin et al.;[25] (6-9) the surface areas of the cavities; (10-12) the volumes. The probe radius in this calculations is 1.2 Å.

We think that the results obtained by CAVE can be considered as comparable with two others. The differences between our and Rashin's results may be due to the different natures of algorithms.

Table 1. Computed van der Waals Surface (VW) Area (in Å^2) of Selected Proteins.

Protein	Atom	VOLBL	MSDOT	GEPOL	ARVO	CAVE	Number of cavities
1eca	1049	13852.0	13842.9	13909.4	13761.8	13761.8	0
1nxb	472	5673.7	5672.1	5683.3	5676.4	5676.4	0
2act	1657	21252.3	21260.6	21162.0	21111.9	21111.9	4
2cha	1736	22551.2	22521.6	22505.6	22429.7	22429.7	3
2lyz	1001	12720.4	12700.7	12704.8	12676.6	12676.6	3
2ptn	1629	21433.1	21439.9	21480.5	21304.8	21304.8	0
2sn3	495	6453.6	6449.8	6432.4	6454.2	6454.2	2
3cyt	1606	21158.1	21170.3	21125.1	21015.1	21015.1	4
3rn3	957	12332.2	12328.1	12287.9	12333.9	12333.9	0
4pti	454	5939.1	5939.4	5919.6	5921.0	5921.0	0
5mbn	1217	16201.3	16192.0	16247.8	16133.6	16133.6	0

Table 2. van der Waals (VW) Volume (in Å^3) of Selected Proteins.

Protein	VOLBL	VOLUME	GEPOL	ARVO	CAVE
1eca	13402.0	18740.1	13383.1	13722.7	13722.7
1nxb	5841.0	7522.3	5471.0	5943.2	5943.2
2act	20930.5	28303.9	21055.6	21552.0	21552.0
2cha	22431.3	31631.7	22460.5	22820.3	22820.3
2lyz	12663.4	17228.2	12594.1	12933.9	12933.9
2ptn	21031.0	29298.8	21305.9	21380.7	21380.7
2sn3	6256.3	8410.9	6529.1	6426.0	6426.0
3cyt	20667.6	29345.1	20606.5	21150.8	21150.8
3rn3	12110.8	16279.2	12196.0	12418.4	12418.4
4pti	5836.5	7640.8	5830.2	5978.2	5978.2
5mbn	15816.5	22147.4	15944.2	16159.1	16159.1

Our algorithm is analytical while Rashin's approach is based on numerical integration. At the same time, one might expect that since our program and VOLBL use analytical methods, the results must coincide with higher accuracy than is shown in the table. One reason for this may be the aforementioned uncertainty in atomic radii. We used for the both sets of radii the values which we found in the literature. Yet another source of discrepancy may be in differences in the PDB file. There are always uncertainties in atomic coordinates, which different researchers handle in different ways. Our experience tells us that cavities are extremely sensible to both factors.

Let us demonstrate this graphically. While the molecular graphics programs might show a more vivid picture on the computer screen, we use just a simple 2D scheme. Figure 8 shows a system of atoms that clearly contains a cavity when the atoms are of the shown sizes. Now suppose that in order to fit some experimental data one uses a little smaller radius for the atom A. Then, obviously, the gap will appear between A and its right neighbor, and the cavity will disappear (open up). Unfortunately, one should mention that currently there is no well standardized set of atomic radii. Very often, even if the authors cite a known set, they use some modified (even a little) values or introduce radii for atomic groups instead of single atoms. For example in CH group one author may just neglect the H atom and use the published value for carbon radius while the another may again neglect the H atom but use a modified value for the carbon atom even if the carbon is of the same type (aromatic, aliphatic, etc.), only to show the difference between "stripped" carbon and CH group. Of course, this is not just a fancy of researchers but is an objective situation arising from the fact that only a few types for every kind of atoms are defined so far. Hopefully, the situation will be improved in the near future when more types of atoms are introduced. One such attempt has been done recently in an earlier cited article.[50] Similarly, the cavity can be destroyed if some structure definition program gives a little different position (coordinates) for the atom A.

Membrane Protein Bacteriorhodopsin

We have carried out a detailed calculations for the membrane protein bacteriorhodopsin (BR) (2brd.pdb) and compared the results with the data from Table X in ref. 35. The data are represented in Table 4. The structure of BR has been obtained by Henderson and coworkers from electron microscopy studies.[55,56] The coordinates of atoms are given in the file `2brd.pdb`.

We use 1.4 Å as the probe size for a water molecule, all heteroatoms are included, like in ref. 35. In ref. 35 no information is given about the van der Waals radii.

Table 3. Cavities Computed for the Listed Proteins Using Triangulation Method: Comparison with Results from Liang et al.[35] and Rashin et al.[25]

	Number of cavities				Area				Volume		
Protein	C^{SA}_{Ri}	C^{SA}_{Ra}	VOLBL	Rashin	C^{SA}_{Ri}	C^{SA}_{Ra}	VOLBL	Rashin	C^{SA}_{Ri}	C^{SA}_{Ra}	VOLBL
1eca	8	12	10	9	59.1	35.7	46.4	69	7.8	4.0	6.4
1nxb	3	4	3	0	0.15	0.29	0.4	0	0.0	0.0	0.0
2act	18	19	20	21	127.3	143.7	140.4	130	31.4	36.7	35.0
2cha	21	26	23	26	119.6	143.0	132.2	120	16.3	20.8	20.4
2lyz	11	12	12	8	50.3	58.0	58.7	53	7.4	8.6	9.0
3ptn	16	17	19	13	159.3	181.7	175.4	168	27.3	32.5	31.5
2sn3	1	1	2	2	3.74	5.1	6.0	6	0.3	0.4	0.4
3cyt	5	5	8	5	1.8	2.7	3.1	2	0.1	0.1	0.1
3rn3	1	1	4	5	1.1	0.0	1.3	3	0.1	0.0	0.1
4pti	2	2	2	2	21.8	26.5	23.4	20	3.2	4.8	3.7
5mbn	12	16	17	23	56.9	69.3	95.8	85	8.4	10.7	14.1
8tln	30	42	42	30	127.0	166.0	163.4	117	16.5	22.9	22.2

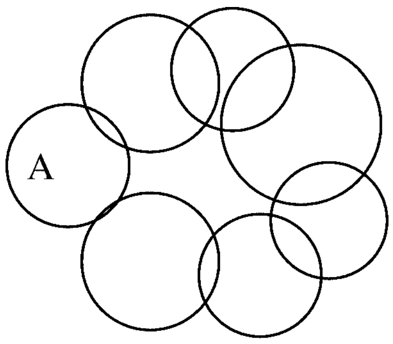

Figure 8. Sensibility of the cavity against atomic radii and coordinates.

First the Richard's radii set[13] was used as in ref. 34. Authors of ref. 35 report 18 cavities. We obtain only 16. But for selected amino acids groups of three cavities reported in Table X,[35] our calculations show that such cavities do not exist (for Richard's parameter set).

With radii used by Rashin et al.,[25] the correspondence was much better. The number of cavities was the same −18. Table 4 compares our results to those from Table X of ref. 35. Areas are in Å^2 and volumes are in Å^3, respectively.

The differences between residues contributing to the cavities are also listed in the last column.

The first two columns contain the order numbers of the cavities detected by VOLBL[35] and our program, respectively. The results given in ref. 35 have subscript "V," our data are denoted by subscript "C." The last column contains the residues reported in ref. 35 not detected by our program, except the G125*, which has been detected by our program, but not mentioned in Table X of ref. 35. The situation is interesting with the cavity No. 13 detected in our program. Its boundary is formed by four atoms: 640 LEU87, 672 PRO91, 845 GLY116, and 869 ILE119. Its SA volume is 8.4×10^{-7}, and the SA cavity surface area is 6.6×10^{-4}. So, such cavity may appear/disappear as the result of different numerical algorithms, used in the calculations (even if the algorithms are analytical). This cavity is not shown in the table. On the other hand, authors of ref. 35 report cavity no. 17, whose boundary should be created by the atoms L152, F171, L174, and R175. This cavity has not been detected by our algorithm and is marked by a "×" sign. To clarify the situation, we have selected all 38 atoms of these residues and run our program with these selected data. The result is presented as follow:

```
Protein from file 2brdc17.pdb
Protein: 2brdc17
Solvent atom radius: 1.400
Rashin van der Waals radii set
Atoms number:    38
...
Segments number:
Total:  1 Deleted:  1 Cavity enveloping: 0
The total number of cavities is 0.
The total number of spheres is 0.
The total number of triangles is 0.
There is no cavity!
Inaccessible volume:   1589.430136533074000
Accessible surface area:  853.614508447047100
```

The similar result has been obtained by using the atomic radii set from Richards. The output of the program shows that such a cavity cannot be created by given atoms. Again, the reason may be in the difference in radii and/or PDB data.

Interestingly, when we exclude the retinal heteroatoms from the calculations, in both (Rashin's resp. Richard's sets) a large cavity

Table 4. Inaccessible Cavities Computed by VOLBL and CAVE for Membrane Protein Bacteriorhodopsin from `2brd.pdb`.

Cavity order no.		Surface area		Volume				
VOLBL	CAVE	SA$_V$	SA$_C$	SA$_V$	SA$_C$	MS$_V$	MS$_C$	Residue differences
1	15	40.3	37.6	10.5	8.6	134.0	113.5	
2	2	13.9	12.4	2.42	2.15	71.2	54.2	I78
3	14	10.6	10.3	1.31	1.25	71.0	58.7	
4	12	9.29	12.2	1.25	1.86	51.9	53.5	
5	5	5.79	5.84	0.80	0.83	39.9	35.8	
6	4	3.70	4.64	0.40	0.54	32.4	31.7	
7	3	6.13	7.07	0.56	0.75	48.6	44.5	A53, V213
8	8	4.14	5.27	0.49	0.65	34.8	34.9	
9	10	3.71	5.41	0.21	0.34	35.8	37.9	
10	17	1.81	0.50	0.09	0.02	28.5	13.6	W137, Ret(C3), G125*
11	7	2.38	2.85	0.17	0.21	28.1	26.4	
12	16	2.21	0.75	0.13	0.03	28.9	16.5	
13	6	0.31	0.13	0.008	0.002	16.6	11.5	D96
14	9	0.82	0.12	0.03	0.002	22.4	12.4	Y57, W86
15	18	0.84	0.07	0.019	0.001	21.9	0.0	L207
16	1	0.62	0.22	0.025	0.005	19.0	12.1	
17	×	0.02		0.000		12.7		L152, F171, L174, R175
18	11	0.01	0.14	0.000	0.001	12.7	11.5	

appears, whose boundary is created by parts of surfaces of 67 resp. 63 atoms.

Discussion

The merits of the proposed algorithm are its simplicity and strictly analytical nature. There are no approximations made. The algorithm implements a search process among the objects and systematizes them according to certain rules. Calculations of the quantitative parameters are done through analytical formulas. The program CAVE written on the basis of the algorithm has been used for detection and calculation of cavities for many different systems including artificial system of spheres and real protein molecules from PDB databank. The results obtained by CAVE have been compared with other similar programs available in the literature and in internet. Here we would not like to declare that our algorithm is more efficient than the others. But certainly we can say that it is competitive enough. It should be mentioned that the calculated data do not always coincide with those found in literature, but the differences are not larger than the data obtained by any other two known methods. Unfortunately, we do not have any of the cited programs at hand that creates certain problems for exact comparison.

Here we want to discuss the probable sources of discrepancies. In our opinion, the biggest problem is the different treatment of the atomic radii. As we mentioned earlier, the real values of the atomic radii used by some groups are often quite different from the basic sets and are not published anywhere. During the testings of our program we have communicated with the authors of one package and requested to tell us the real values of the atomic radii which they used. They have been very kind to send us these values and we discovered that, for example, in Richard's set of radii they use 25 types of atoms and/or atomic groups while the basic set published by Richard contains only six types. We tried to use these values but some other questions have arisen which required more information from the authors. Unfortunately, our communications have been interrupted by other party and we could not use these values adequately. So, we use in our calculations the atomic radii as published in the quoted papers. Some time ago we encountered a similar problem when testing our algorithm for calculation of the SA area.[23] But then we have been fortunate to find in the internet the program GETAREA[57] from W. Brauns's group,[58] which appeared to be very convenient for comparison of two algorithms because it suggests a very simplified set of the atomic radii: the radius of the hydrogen is set to 0 and all other atoms are assigned the value 1. Although this may not be the best choice for real molecular calculations, it provides an easy way for exact comparison of two algorithms. Another source of differences may be the data in the PDB file. Some researchers may slightly modify the atomic coordinates for certain reasons. This happens less frequently, but we are not completely sure that we are using exactly the same atomic coordinates.

The algorithm presented earlier allows to detect the cavities and to study their properties such as localization, the boundary "atoms," the volume, and the surface area.

The program implementation of the triangulation algorithm, together with several auxiliary programs could be used for solving different problems: determination of the position status of a given point from the point of view of the cavities, localization of the cavities, and calculation of their volume and surface area. The results, presented here, are in good agreement with the results given by other authors.[25,59]

Appendix: Calculation of the Spherical Angles and the Cavity Index

In the 2D case, the sum of oriented angles can be used to make a decision if a point belongs to the interior or to the exterior of some closed curve. Similarly, in the 3D case, the sum of spherical angles can be used to check if a point lies inside or outside the closed polyhedron. Here we describe the algorithm for computing the spherical angle of a point with respect to some spherical triangle; a spherical triangle is a figure formed on the surface of a sphere by three great circular arcs intersecting pairwise at three vertices. It is sometimes called an Euler triangle.

Let a spherical triangle ΔABC on the unit sphere have angles A, B, and C measured in radians (Fig A1). Then the area of this triangle is

$$\sigma(\Delta ABC) = A + B + C - \pi. \tag{A1}$$

The angular arc lengths α, β, γ and the vertex angles A, B, C are related by the cosine formula[60–62]

$$\begin{aligned}\cos\alpha &= \cos\beta\cos\gamma + \sin\beta\sin\gamma\cos A;\\ \cos\beta &= \cos\gamma\cos\alpha + \sin\gamma\sin\alpha\cos B;\\ \cos\gamma &= \cos\alpha\cos\beta + \sin\alpha\sin\beta\cos C.\end{aligned} \tag{A2}$$

Calculation of the Spherical Angle

An oriented triangle $\Delta V_1V_2V_3$ in E_3 is being viewed from a point P (other than vertices) under the spherical angle that is equal to the surface area of the projection of the triangle onto the unit sphere with the center P (Fig. A2). Denote this oriented spherical angle by $\sigma_P(\Delta V_1V_2V_3)$. Let $\sigma_P(\Delta V_1V_2V_3)$ be negative if the point P is not on the inward side of the triangle.

First the angles α, β, γ are computed. Then we compute $\cos A$, $\cos B$, and $\cos C$ using Eq. (A2). By substituting the values for A, B, and C into Eq. (A1) we get the absolute value of $\sigma_P(\Delta V_1V_2V_3)$.

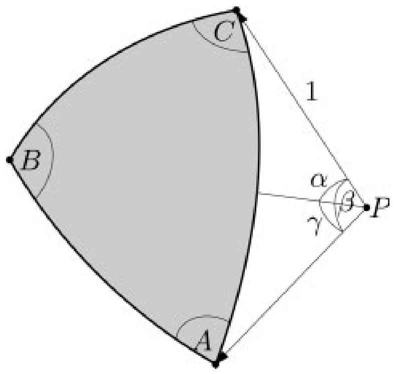

Figure A1. The surface area of the spherical triangle.

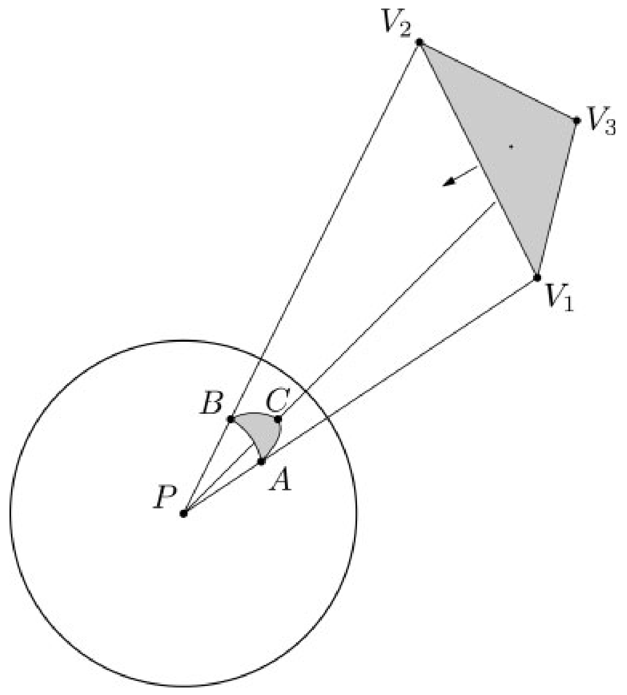

Figure A2. Cavity indexing.

The Cavity Index

Now we consider some point *P*, which does not lie on the triangulation surface. All vertices of cavity triangulation are projected onto the unit sphere with the center in *P* (Fig. A2).

Definition 1. *Let P be a point in E_3 not belonging to the cavity triangulation $\Delta\mathcal{M}$. The index of P with respect to the cavity triangulation $\Delta\mathcal{M}$ is the number*

$$\chi_P(\Delta\mathcal{M}) = \frac{1}{4\pi} \sum_{\delta \in \Delta\mathcal{M}} \sigma_P(\delta).$$

Proposition 2. *An envelope triangulation $\Delta\mathcal{M}$ is setting bounds to the point P (different from vertices of triangulation) if and only if $\chi_P(\Delta\mathcal{M}) = 1$.*

So, if any intersection point mentioned at the end of the previous section has the index 1, the corresponding triangulation component contains a cavity inside.

References

1. Kauzman, W. Adv Protein Chem 1959, 14, 1.
2. Schoenborn, B. P. J Mol Biol 1969, 45, 297.
3. Tilton, R. F.; Kuntz, I. D.; Petsko, G. A. Biochemistry 1984, 23, 2849.
4. Wierenga, R. K.; Noble, M. E. M.; Davenport, R. C. J Mol Biol 1992, 224, 1115.
5. Sreenivasan, U.; Axelsen, P. H. Biochemistry 1992, 31, 12785.
6. Kellis, J. T., Jr.; Nyberg, K.; Sali, D.; Fersht, A. R. Nature 1988, 333, 784.
7. Kellis, J. T., Jr.; Nyberg, K.; Fersht, A. R. Biochemistry 1989, 28, 4914.
8. Eijsink, V. G. H.; Gijkstra, B. W.; Vriend, G.; van der Zee, J. R.; Veltman, O. R.; van der Vinne, B.; van der Burg, B.; Kempe, S.; Venema, G. Protein Eng 1992, 5, 421.
9. Eriksson, A. E.; Baase, W. A.; Wozniak, J. A.; Mattheus, B. W. Nature 1992, 355, 371.
10. Eriksson, A. E.; Baase, W. A.; Mattheus, B. W. J Mol Biol 1993, 229, 747.
11. Lambright, D. G.; Balasubramanian, S.; Decatur, S. M. Biochemistry 1994, 33, 5518.
12. Lee, B.; Richards, F. M. J Mol Biol 1971, 55, 379.
13. Richards, F. M. Annu Rev Biophys Bioeng 1977, 6, 151.
14. Chothia, C. Nature 1974, 248, 338.
15. Connolly, M. L. J Appl Crystallogr 1983, 16, 548.
16. Richmond, T. J. J Mol Biol 1984, 178, 63.
17. Connolly, M. L. Science 1983, 221, 709.
18. Connolly, M. L. J Am Chem Soc 1985, 107, 1118.
19. Gibson, K. D.; Scheraga, H. A. Mol Phys 1987, 62, 1247.
20. Gibson, K. D.; Scheraga, H. A. J Phys Chem 1987, 91, 4121.
21. Petitjean, M. J Comput Chem 1994, 15, 507.
22. Totrov, M.; Abagyan, R. J Struct Biol 1996, 116, 138.
23. Hayryan, S.; Hu, C.-K.; Skřivánek, J.; Hayryan, E.; Pokorny, I. J Comput Chem 2005, 26, 334.
24. Shrake, A.; Rupley, J. A. J Mol Biol 1973, 79, 351.
25. Rashin, A. A.; Iofin, M.; Honig, B. Biochemistry 1986, 25, 3619.
26. Hubbard, S. J.; Gross, K.-H.; Argos, P. Protein Eng 1994, 7, 613.
27. Hubbard, S. J.; Argos, P. Protein Sci 1994, 3, 2194.
28. Connolly, M. L. J Mol Graphics 1993, 11, 139.
29. Zhang, L.; Hermans, J. Proteins 1996, 24, 433.
30. Rashin, A. A.; Rashin, B. H.; Rashin, A.; Abagyan, R. Protein Sci 1997, 6, 2143.
31. Edelsbrunner, H.; Kirkpatrick, D. G.; Seidel, R. IEEE Trans Inform Theor 1983, 29, 551.
32. Edelsbrunner, H. Discrete Comput Geom 1995, 13, 415.
33. Edelsbrunner, H.; Mucke, E. P. ACM Trans Graph 1994, 13, 43.
34. Liang, J.; Edelsbrunner, H.; Fu, P.; Sudhakar, P. V.; Subramaniam, S. Prot Struct Func Genet 1998, 33, 1.
35. Liang, J.; Edelsbrunner, H.; Fu, P.; Sudhakar, P. V.; Subramaniam, S. Prot Struct Func Genet 1998, 33, 18.
36. Chakravarty, S.; Bhinge, A.; Varadarajan, R. J Biol Chem 2002, 277, 31345.
37. Williams, M. A.; Goodfellow, J. M.; Thornton, J. M. Prot Sci 1994, 3, 1224.
38. Bakowies, D.; van Gunsteren, W. F. Proteins 2002, 47, 534.
39. Connolly, M. L. J Appl Crystallogr 1985, 18, 499.
40. Akkiraju, N.; Edelsbrunner, H. Discrete Appl Math 1996, 71, 5.
41. Goncalves, P. F. B.; Stassen, H. J Chem Phys 2005, 123, 214109.
42. Willard, L.; Ranjan, A.; Zhang, H.; Monzavi, H.; Boyko, R. F.; Sykes, B. D.; Wishart, D. S. Nucleic Acids Res 2003, 31, 3316.
43. Laskowski, R. A. J Mol Graph 1995, 13, 323.
44. Liang, J.; Edelsbrunner, H.; Woodward, C. Protein Sci 1998, 7, 1884.
45. Eisenmenger, F.; Hansmann, U. H. E.; Hayryan, S.; Hu, C.-K. Comput Phys Commun 2001, 138, 192.
46. Eisenmenger, F.; Hansmann, U. H. E.; Hayryan, Sh; Hu, C.-K. Comput Phys Commun 2006, 174, 422.
47. Lin, C.-Y.; Hu, C.-K.; Hansmann, U. H. E. Proteins 2003, 52, 436.
48. Ghulghazaryan, R. G.; Hayryan, S.; Hu, C.-K. J Comput Chem 2007, 28, 715.
49. http://www.rcsb.org/pdb/home.
50. Li, A.-J.; Nussinov, R. Proteins 1998, 32, 111.
51. Buša, J.; Džurina, J.; Hayryan, E.; Hayryan, S.; Hu, C.-K.; Plavka, J.; Pokorný, I.; Skřivánek, J.; Wu, M.-C. Comput Phys Commun 2005, 165, 59.
52. Pacios, L. ARVOMOL, Quantum Chemistry Program Exchange 132 (1993).
53. Silla, E.; Villar, F.; Nilsson, O.; Pascual-Ahuir, J. L.; Tapia, O. J Mol Graph 1990, 8, 168.

54. Silla, E.; Tun, I.; Pascual-Ahuir, J. L. J Comput Chem 1991, 12, 1077.
55. Henderson, R.; Baldwin, J. M.; Ceska, T. A.; Zemlin, F.; Beckmann, E.; Downing, K. H. J Mol Biol 1990, 213, 899.
56. Grigorieff, N.; Ceska, T. A.; Downing, K. A.; Baldwin, J. A.; Henderson, R. J Mol Biol 1996, 259, 393.
57. http://www.scsb.utmb.edu/cgi-bin/get_a_form.tcl.
58. Fraczkiewicz, R.; Braun, W. J Comput Chem 1998, 19, 319.
59. Alard, P.; Wodak, S. J. J Comput Chem 1991, 12, 918.
60. mathworld.wolfram.com/SphericalTriangle (August 5, 2003).
61. mathworld.wolfram.com/SphericalTrigonometry (August 5, 2003).
62. www.shef.ac.uk/∼phys/people/vdhillon/Teaching/TheCourse/SphericalGeometry/-/SphericalTrigonometry (August 5, 2003).