Computational Structural Bioinformatics II: Molecular Models

Chandrajit Bajaj

October 25, 2010

Contents

Chapter 1

Introduction

intro to computational algorithms and data structures for modeling of molecular models and macromolecular complexes from PDB

Molecular Solvation Models and Minimal Surfaces

Molecular surfaces of proteins and other biomolecules, are often modeled as smooth analytic interfaces separating the molecule from solvent (an implicit solvation model).

These analytic solvation models are often of high genus with a myriad of interconnected tunnels and pockets with openings (mouths). All these interfaces are biochemically significant as pockets are often active sites for ligand binding or enzymatic reactions, and tunnels are often solvent ion conductance zones.

In this talk, we present a general characterization of these solvation interfaces and approximately model them as piecewise minimal surfaces, namely, the solution of non-linear elliptic or biharmonic partial differential Euler-Lagrange equations obtained from the minimization of high-order energy functionals.

Index to content of chapters:

chap 2 Spatial Occupancy :

Union of Balls (Weighted Points)– Voronoi-Dell Complexes (C^0) Dynamic Neighborhood Data Structures — Packing Grid, Octrees Hydrophobic

chap 3 Smooth Interfaces :

Moleculat-solvent Dielectric interfaces

Analytic volumetric models (Radial Basis Splines, C^inf) Mixed-Voronoi-Del complexes (C^1) A-patch complexes (mostly C^1); Algebraic Shell Splines (C^1) Variational B-spline Surfaces (C^2)

Molecular-molecular interfaces

Clustering/Multi-resolution

chap 4 Complementary Space Features:

Mouths, Tunnels, Pockets, Voids

chap 5 Shape Properties:

Areas, Volumes, Normals, Curvatures

chap 6 Energetics:

Hamiltonians, Lagrangians

Partial Charge Assignment

Lennard Jones Potential (vanderWaal)

Columbic

Dispersion

Generalized Born

Poisson Boltzmann

chap 7 Forces:

Area, Volume Derivatives

Energetic derivatives

chap 8 Docking:

Affinitiy Functions

Search & Scoring
Filters
Re-Ranking
chap 9 Molecular Machines:
viruses
ribosomes
chap 10 conclusion
Appendix
App1: X-ray structure determination
App2: Proteins, Nucleic Acids
App3: PDB, MMCIF

Chapter 2

Spatial Occupancy

Molecules are commonly modeled as a collection of atoms represented by spheres, with radii equal to their van der Waals radii. Three types of surfaces are defined based on this representation.

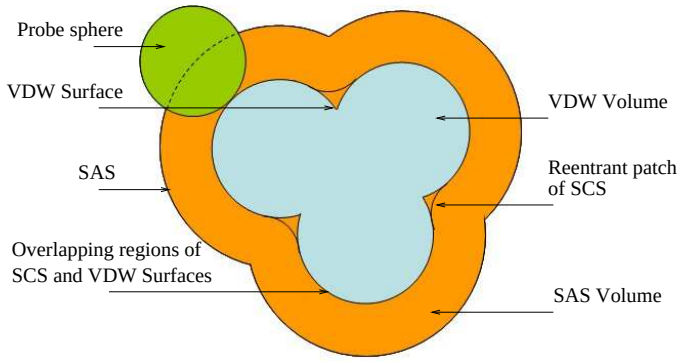


Figure 2.1: Different molecular surfaces and regions are shown for a 3 atom model in 2D. The SAS is the locus of the center of the rolling probe sphere. The VDW surface is the exposed union of spheres representing atoms with their van der Waals radii and contains the VDW volume. The lower side of the rolling probe defines the smooth SES which contains parts of the VDW surface and reentrant patches. The region between the SAS and the SES is defined as the SAS volume, and the region between the SAS and the VDW volume is refered to as the SES volume.

van der Waals (VDW) surface. The surface of the set of spheres is known as the *van der Waals surface*.

Solvent Accessible Surface (SAS). Proteins do not exist in isolation, but commonly found in solutions, especially water. The *Solvent Accessible Surface* (SAS) is defined as the locus of the center of a solvent proble (e.g., a water molecule) rolling along the VDW surface of a molecule [57]. If water molecules are modeled as spheres with radias 1.4 Å, then the SAS of a given molecule can be found by increasing the radius of each atom in that molecule by 1.4 Å, and taking the surface of the set of inflated atoms.

Solvent Contact Surface (SCS). The VDW surface contains too many internal atoms and patches which are not accessible by the solvent, and the SAS contains regions that should be occupied by the solvent. Thus both these surfaces contribute to large errors in biomolecular energy computation. In order to overcome this drawback, Richards [66] gave a definition for molecular surface as a set of contact and reentrant patches. A probe solvent sphere, rolling over the atoms of a protein defines a region in which none of its points pass through. The boundary of this volume is continuous and defines a new molecular surface. This surface is composed of convex patches where the probe touches the atom surfaces, concave spherical patches when the probe touches more than 2 atoms simultaneously and toroidal patches when the probe rolls between two atoms. This surface is commonly known as the *Solvent Contact Surface* (SCS), or *Solvent Excluded Surface* (SES), or *Lee-Richards* (LR) Surface, or simply the Molecular Surface. The major drawback of SCS is that cusps created by the self-intersection of the rolling probe, cause singularity during energy computation.

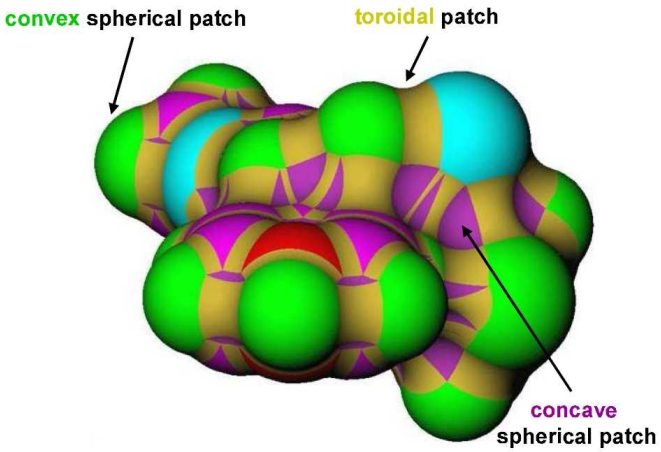


Figure 2.2: 3D image showing the decomposition of SCS into three different kinds of patches: convex spherical, toroidal and concave spherical.

Figure 2 shows the surfaces described above for a 3 atom example as a 2D cross section, and Figure 2 shows the different types of re-entrant patches on an SCS.

See Appendix A for further details about structure of biomolecules like proteins and RNA.

In Section 2.1 we introduce different ways to model the VDW, SAS and SCS surfaces of molecules. Sections 2.2 and 2.3 describe two different algorithms to produce such representations. Section 2.4 and 2.5 presents techniques to maintain the surfaces under dynamic change of radii and atom movements. And Section 2.6 discuss multiresolution models of molecules.

2.1 Surface Representations

2.2 Molecular Surface using Voronoi-Cell Complexes

2.2.1 Atom Boundary Patch as a Trimmed NURBS

The representation we use for molecule (property) surfaces is a boundary representation. Two classes of information are used: (a) geometric description of each patch, (b) topological relations amongst the patches. We maintain the following data structures related to the molecule.

1. The weighted Voronoi diagram [4, 53] (power diagram) $\mathscr{D}$ of the molecule atom centers (the weights are the squares of the atoms radii).
2. A regular triangulation $\mathscr{T}$ (dual of the power diagram) of the same set of weighted points as in [37].
3. A NURBS patch per molecule atom.

We have selected NURBS as basic modeling primitive [55]. For an appropriate choice of parameterization we obtain a single trimmed NURBS for each atom's external surface contribution. Each such patch is the intersection of one sphere (representing one atom) with the exterior of all its neighboring spheres. Consider the intersection $S \cap R$ of a spherical surface $S = \{x : \|x - x_0\| = r^0\}$ with the external of sphere $R = \{x : \|x - x_1\| \geq r^1\}$. There always exists an halfspace $\pi = \{x : (x \cdot l) \leq d\}$ such that:

$$S \cap R = S \cap \pi.$$

For each atom we can reduce our patch representation problem to the intersection of a sphere with a set of halfspaces. The union of balls model [35] provides the equation of each halfspace intersecting one atom[1]. Note that, since we use a parametric

[1]Given the Voronoi complex of the weighted centers of the molecule atoms, the halfspaces whose common intersection generates the Voronoi cell of the atom B are those with which $S = \partial B$ must be intersected.

representation $S = f(u,v)$, we need to compute the domain D in (u,v) space such that $f(D) = S \cap \pi$.

To have an efficient representation we want to obtain only one NURBS patch per atom. Moreover, since we will use this formulation to achieve a representation of the surface parametric in the radii of the atoms we need a formulation that maps continuous modifications of the radii into continuous modifications of the domain D. This is not achieved with the classical NURBS sphere representation as a rotational surface of a half circle [65] since there are two points (north and south pole) of the sphere that are the image of two lines in the parameter domain (say $u = 0$, $u = 1$ if the interval of the u domain is $[0,1]$). This implies that when the boundary plane of π crosses one of the poles the corresponding trimming curve in the (u,v) domain would have a discontinuous change in shape.

Without loss of generality we assume that S is the unitary sphere. The parameterization we adopt is the following (see [13]):

$$\begin{aligned} x &= \frac{2u}{u^2+v^2+1} \\ y &= \frac{2v}{u^2+v^2+1} \\ z &= \frac{u^2+v^2-1}{u^2+v^2+1} \end{aligned} \tag{2.1}$$

This parameterization maps the (infinite) rectangular domain

$$[-\infty,+\infty] \times [-\infty,+\infty]$$

to the unitary sphere. Note that in practice we do not deal with an infinite domain since we do not represent an entire sphere but only one spherical patch. In particular assume that we are considering the intersection S^* of the unit sphere S with the halfspace $z \leq d$ (with a rigid body transformation and a scaling we can always reduce the first intersection to this case). We determine a positive constant l such that $S^* \subset f(I)$, where I is the square domain $[-l,+l] \times [-l,+l]$. In the parameter domain this corresponds to the condition $D \subset I$. The minimum value of l that satisfies such condition is $l = \sqrt{\frac{1+d}{1-d}}$. Regarding the numerical stability of the method it is important to note that for $d = 0.999$ we get $l = 44.710\ldots$. Even when d is much larger than a realistic value, we still deal with a small domain region.

The next step is to determine the domain D. At this end we simply replace the parametric equations (2.1) of the of the sphere to the variables in the Cartesian inequality of π obtaining the Cartesian inequality defining D.

$$\frac{u^2+v^2-1}{u^2+v^2+1} \leq d \quad \Rightarrow \quad u^2+v^2 \leq l^2 \tag{2.2}$$

Thus the domain D is a disc with center in the origin and radius l. Note that a variation of d corresponds to a scaling of D, that can be performed by simply scaling its control polygon (once a NURBS representation is defined for the trimming curve of D). For any additional cutting halfspace $\bar{\pi} : ax+by+cz \leq d$ we have:

$$(c-d)u^2 + (c-d)v^2 + 2au + 2bv - (c+d) \leq 0 \tag{2.3}$$

If the plane $ax+by+cz = d$ contains the singular point of the parameterization $P = (0,0,1)$ then $c = d$. In this case the trimming curve is the straight line:

$$2au + 2bv - (c+d) = 0. \tag{2.4}$$

The domain D must be intersected with the half-plane $2au + 2bv - (c+d) \leq 0$.

If $c - d \neq 0$ the trimming curve derived from (2.3) has Cartesian equation:

$$(u + \frac{a}{c-d})^2 + (v + \frac{b}{c-d})^2 = \frac{a^2+b^2+c^2-d^2}{(c-d)^2} \tag{2.5}$$

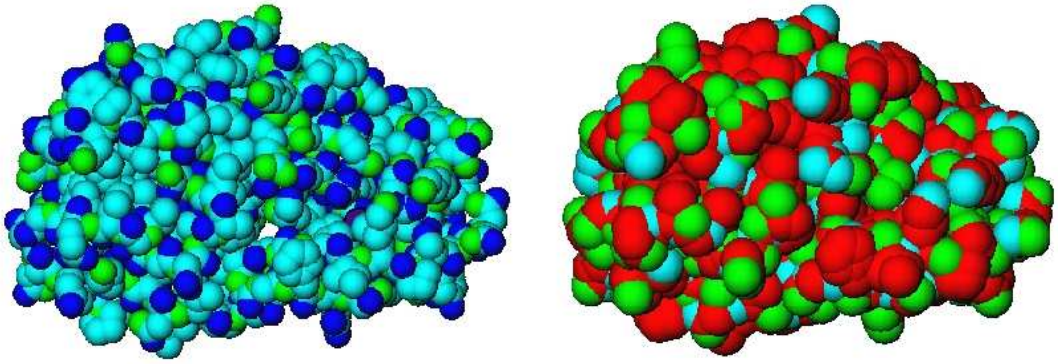
In general we note that all the trimming curves are circles (possibly with infinite radius) so that the region D can be modeled as progressive intersection/difference of a sequence of circles. Corresponding to the cutting halfspace $\bar{\pi}$ of normalized equation $ax+by+cz \leq d$, with $a^2+b^2+c^2$=1, we have in parameter space a circle C of center $(\frac{a}{c-d}, \frac{b}{c-d})$ and radius $\frac{\sqrt{1-d^2}}{c-d}$. The region defined by such circle (inside/outside) depends on the sign of the term $c-d$. For $c-d < 0$ $P = (0,0,1)$ is inside $\bar{\pi}$ and hence the points of the plane at infinity are included in the region corresponding to $\bar{\pi}$. That is $\bar{\pi}$ is mapped onto the outside of C. This requires C to be parameterized with a clockwise orientation. Symmetrically $c+d < 0$ implies that $\bar{\pi}$ corresponds to the region inside C and hence C must be parameterized with a counterclockwise orientation.

2.2.2 Solvent Accessible Surface

In this section we discuss the representation of the solvent accessible surface of a molecule. Since we are representing the molecule with a union of balls $\mathscr{B}$, in the following, with some abuse of terminology, we will call $\mathscr{B}$ both the molecule or the union of balls. Similarly each single ball B will be called either a ball or an atom.

Assume we have a ball B of radius r (a solvent atom) free to move in space without intersecting the union of balls $\mathscr{B}$ (a molecule). We say that B is in a *legal* position if its interior $\overset{\circ}{B}$ does not intersect $\mathscr{B}$.

Definition 2.2.1. *The* solvent accessible *surface S_a of the union of balls $\mathscr{B}$ relative to a solvent atom B of radius r, is the locus (envelope) of the centers of the spheres with radius r tangent to $\mathscr{B}$.*



(a) (b)

Figure 2.3: The *HIV-2 PROTEASE* (a) and one solvent accessible surface (b) for the same molecule.

From [35, 36] we know that S_a is the boundary surface of the union of balls $\mathscr{B}'$ that has the same set of atoms as $\mathscr{B}$ but with all the radii increased by r (see figure 2.3). On the basis of this property we can achieve a representation of S_a parametric in r. For $r = 0$ we obtain the van der Waals surface of the molecule $\partial\mathscr{B}$. Varying the value of r we get the accessible surfaces of different solvents.

Let $V' \in \mathscr{V}'$ be the convex cell corresponding to the ball $B' \in \mathscr{B}'$. V' is the intersection of a set of k halfspace $\pi_1 \cap \ldots \cap \pi_k$. The the contribution of B to the boundary of $\mathscr{B}$ (the surface S_a for $r = 0$) is given by $\partial B \cap \pi_1 \cap \ldots \cap \pi_k$.

Now assume $r > 0$ and consider the sphere B' in $\mathscr{B}'$ corresponding to B in $\mathscr{B}$. The contribution of B' to S_a is computed by intersecting $\partial B'$ with the same set of halfspaces $\pi'_1, \ldots, \pi'_{k'}$.

To compute the trimming curves in the parameter space (u,v) of the NURBS patch representing $\partial B' \cap \pi'_1 \cap \ldots \cap \pi'_{k'}$ we apply a mapping that transforms B' into the unitary ball B_u. Under this mapping the variation of r corresponds to have a fixed (unitary) radius ball B_u intersected with a set of varying halfspaces. Formally, if the ball B' and one halfspace π' have equations:

$$\begin{aligned} B' &: \quad x^2 + y^2 + z^2 \leq R^2 \\ \pi' &: \quad ax + by + cz + d \leq 0 \end{aligned}$$

we apply the coordinate transformation $x = Rx', y = Ry', z = Rz'$ to map B' to B_u:

$$\begin{aligned} B_u &: \quad x'^2 + y'^2 + z'^2 \leq 1 \\ \pi' &: \quad ax' + by' + cz' + \frac{d}{R} \leq 0 \end{aligned}$$

The change of the radius R of B' to $R+r$ is hence mapped in normalized coordinates (x', y', z') to the change of the parameter $\frac{d}{R}$ of the halfspace π' to $\frac{d}{R+r}$. This means that the equation of the trimming circles can be rewritten, including the parameter r, as:

$$(u + \frac{a}{c - \frac{d}{R+r}})^2 + (v + \frac{b}{c - \frac{d}{R+r}})^2 = \frac{a^2 + b^2 + c^2 - (\frac{d}{R+r})^2}{(c - \frac{d}{R+r})^2} \tag{2.6}$$

that is a circle of center $(\frac{a}{\frac{d}{R+r}-c}, \frac{b}{\frac{d}{R+r}-c})$ and radius $r = \frac{\sqrt{(R+r)^2-d^2}}{|(R+r)c-d|}$. To maintain the description of the domain D we have to maintain a 2D dynamic union of balls that is equivalent to maintain a weighted Voronoi diagram of moving points in the plane [43, 6].

Note also that the coefficient d of the plane equation is also function of r. In fact as the radius of each ball is increased by r the Voronoi plane that separates two balls moves toward the smaller one. An example is shown in figure 2.4. The distances l_1, l_2 of the Voronoi plane π from the centers of the two balls must be such that the power distances of π are equal, that is:

$$l_1^2 - r_1^2 = l_2^2 - r_2^2$$

Moreover the sum of two distances is constant (the two balls grow but do not move):

$$l_1 + l_2 = l$$

From these two equations we get for l_1 so:

$$l_1^2 - r_1^2 = (l - l_1)^2 - r_2^2 \quad \Rightarrow \quad l_1 = \frac{l^2 + r_1^2 - r_2^2}{2l}$$

When r_1 changes to $r_1 + r$ and r_2 changes to $r_2 + r$ we have:

$$l_1' = l_1 + \frac{r_1 - r_2}{2l} r$$

$$l_2' = l_2 + \frac{r_2 - r_1}{2l} r$$

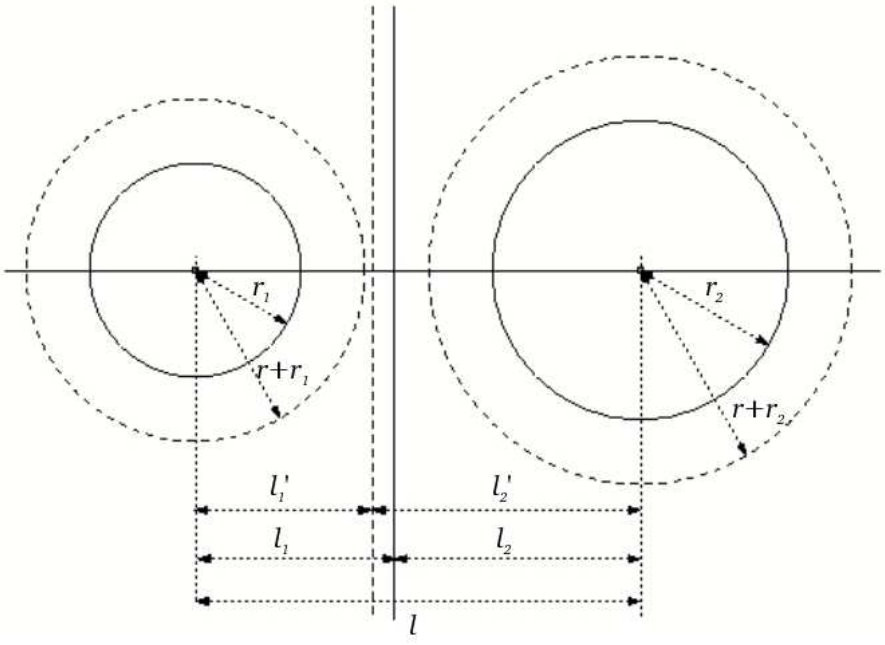


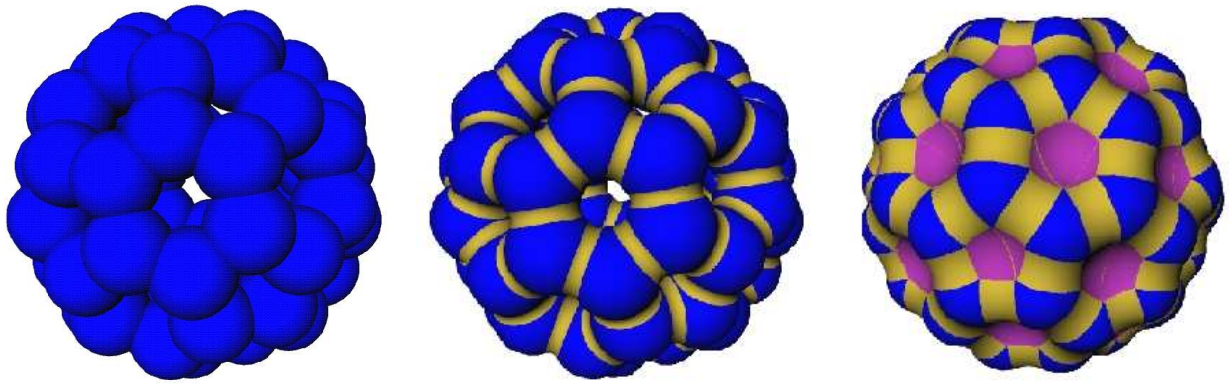
Figure 2.4: As the radius of the two balls is increased by 1 the Voronoi plane that separate them moves towards the smaller ball.

2.2.3 Rolling Ball Surface

In this section we extend the method to achieve an exact NURBS representation of the rolling ball surface S_r of a molecule $\mathscr{B}$. The goal is to achieve an intermediate stage toward to construction the solvent contact surface S_c defined in the next section.

In Figure 2.5 is shown the Fullerene molecule along with two solvent contact surfaces corresponding to two different solvent radii.

We assume to have a ball B of radius r (the solvent molecule) which is free to roll along the union of balls $\mathscr{B}$ (the molecule). The union of all the balls B (moving tangentially to $\mathscr{B}$ in all the possible directions) is a region whose outer envelope strictly contains (if $r > 0$) $\mathscr{B}$ and whose inner envelope is tangent to $\mathscr{B}$ (see [9]).



(a) (b) (c)

Figure 2.5: The Fullerene molecule (a) and two solvent contact surfaces (b),(c) corresponding to two different solvent radii.

Definition 2.2.2. *The* rolling ball *surface S_r of the molecule $\mathscr{B}$ with respect to a ball B of radius r is the inner envelope of the region described by B rolling on $\mathscr{B}$ in all possible directions.*

The close relationship between the solvent accessible surface and the rolling ball surface is evident from this definition.

Proposition 2.2.1. *(Necessary Condition) If a point p lies on the rolling ball surface S_r then it lies also on the boundary of a ball B with center on the solvent accessible surface S_a.*

PROOF. By Definition 2.2.2 when $p \in S_r$ there exists a ball B of radius r such that $p \in \partial B$, $B \cap \mathscr{B} \neq \emptyset$ and $\overset{\circ}{B} \cap \mathscr{B} = \emptyset$. But if the center q of B does not belong to S_a either

$$B \cap \mathscr{B} = \emptyset$$

or

$$\overset{\circ}{B} \cap \mathscr{B} \neq \emptyset$$

□

Using the regular triangulation $\mathscr{T}'$ associated with $\mathscr{B}'$ we can define the set of patches composing S_r. First, recall the relationship between $\partial\mathscr{T}'$ and $\partial\mathscr{B}'$:

- each vertex v of $\partial\mathscr{T}'$ corresponds to a spherical patch of $\partial\mathscr{B}'$;
- each edge e of $\partial\mathscr{T}'$ corresponds to the intersection line between two adjacent spherical patches of $\partial\mathscr{B}'$;
- each triangle t of $\partial\mathscr{T}'$ corresponds to the intersection point between three adjacent spherical patches of $\partial\mathscr{B}'$.

We base the construction of the rolling ball surface on these properties. Using Connolly's terminology [27] (as we will later see that the solvent contact surface is a subset of the rolling ball surface) we have (a) each vertex v of $\partial\mathscr{T}'$ corresponding to a "convex" spherical patch in S_r, (b) each edge e of $\partial\mathscr{T}'$ corresponding to a "saddle" toroidal patch in S_r, and (c) each triangle t of $\partial\mathscr{T}'$ corresponding to a "concave" spherical patch in S_r . The definitions of these three kinds of patches are reported in the following three sections.

"Convex" Spherical Patches Consider the spherical patch $\bar{v}$ with radius $r + r_1$ of $\partial\mathscr{B}'$ associated with the vertex v (see figure 2.6). It represents a moving solvent ball that maintains contact with $\partial\mathscr{B}$ at a point p. The surface described by the point

p is in turn a spherical patch of radius r (part of $\partial\mathscr{B}$). It can be computed from the power diagram of $\partial\mathscr{B}$. Call B the ball (of radius r_1) of $\mathscr{B}$ with center v. It contributes the patch $\partial B \cap \pi_1 \cap \ldots \cap \pi_k$ (that is the Voronoi cell of v is $\pi_1 \cap \ldots \cap \pi_k$) to $\partial\mathscr{B}$. The ball B contributes the patch $\partial B \cap \bar{\pi}_1 \cap \ldots \cap \bar{\pi}_k$ to S_c, where $\bar{\pi}_i$ is parallel to π_i but nearer to v. Without loss of generality we assume v to be the origin $(0,0,0)$ and π_1 to be orthogonal to the x axis (with a rigid body transformation we can always achieve this situation). The halfspace π_1 is $x \leq d$ and the halfspace $\bar{\pi}_1$ is $x \leq \bar{d}$ where:

$$\bar{d} = \frac{dr_1}{(r+r_1)}$$

.

We can so determine any halfspace $\bar{\pi}_i$ corresponding to π_i and hence $\partial B \cap \bar{\pi}_1 \cap \ldots \cap \bar{\pi}_k$.

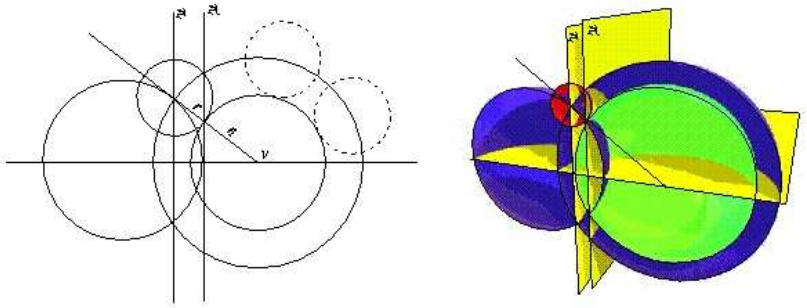


Figure 2.6: A solvent atom of radius r that rolls on the molecule surface $\mathscr{B}$ maintaining its center on the solvent accessible surface $\mathscr{B}'$. Its point of contact with $\mathscr{B}$ belongs to the solvent contact surface S_c.

"Saddle" toroidal patches

A similar argument holds for saddle toroidal patches. With reference to figure 2.7 we consider the edge e of $\partial\mathscr{T}'$ with extreme vertices v_1 and v_2. The edge e corresponds on $\partial\mathscr{B}'$ to a (portion of) circle $\bar{e}$ of intersection between two adjacent balls $\partial B_1 \cap \partial B_2$. Thus, it is possible to roll a solvent ball, moving its center along the arc $\bar{e}$.

If the edge e is not a facet of any triangle of $\partial\mathscr{T}'$ then $\bar{e}$ is an entire circle. The ball that rolls maintaining its center on $\bar{e}$ describes a torus E. We are interested in just a portion of ∂E. Consider the plane π of the Voronoi diagram on which e lies. Applying the procedure specified in the previous section we compute two planes π_1 and π_2 by translating π towards v_1 and v_2, respectively. The intersection of ∂E with the region within π_1 and π_2 generates two toroidal patches. The one nearest to the torus axis $\overline{v_1v_2}$ is the toroidal patch E^* that belongs to S_r.

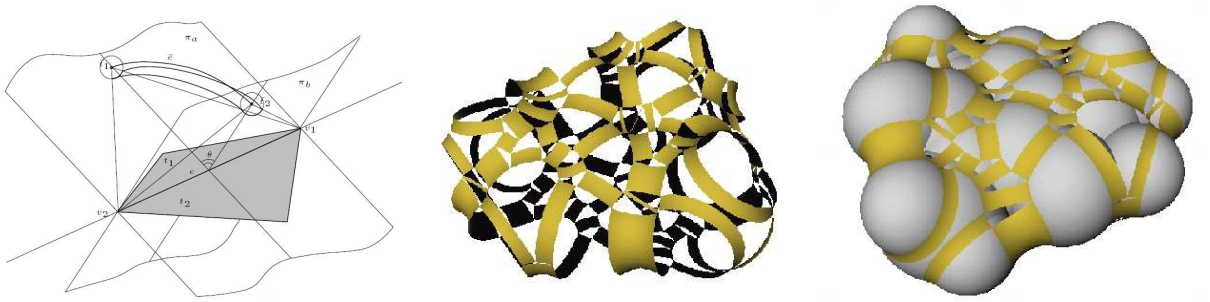


Figure 2.7: (Left) A solvent atom B of radius r that rolls on the molecule surface $\mathscr{B}$ maintaining its center on the solvent accessible surface $\mathscr{B}'$ and two points of contact with two molecule atoms. The portion of circle of ∂B that belongs to the triangle with the three vertices v_1,v_2 center of B, belongs to the rolling ball surface S_r. (Middle) The toroidal NURBS patches of the rolling ball surface S_r of the caffeine molecule. (Right) The toroidal NURBS patches of S_r shown together with the union of balls.

If the edge e is the arc from point $\bar{t}_1$ to point $\bar{t}_2$ then the toroidal patch associated with e is the portion of the patch E^* intersected with two more halfspaces. Call $\pi(v_1,v_2,v_3;v_4)$ the halfspace that contains v_1,v_2,v_3 in its boundary and v_4 in its

interior (with v_1, v_2, v_3, v_4 affinely independent). The toroidal patch corresponding to e is (see figure 2.7):

$$E^* \cap \pi(v_1, v_2, \bar{t_1}; \bar{t_2}) \cap \pi(v_1, v_2, \bar{t_2}; \bar{t_1}).$$

"concave" spherical patches

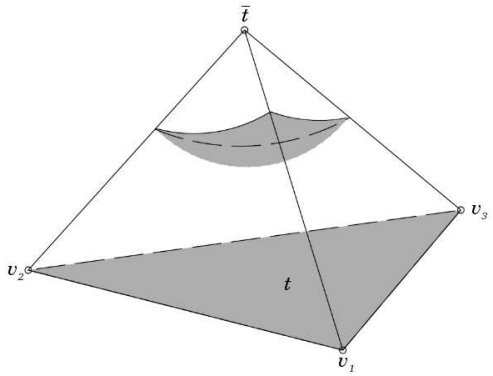


Figure 2.8: A solvent atom of radius r tangent to the molecule surface $\mathscr{B}$ maintaining its center on the solvent accessible surface $\mathscr{B}'$ and three points of contact with three molecule atoms. The portion of ∂B inside the tetrahedron with vertices v_1,v_2,v_2, center of B, belongs to the rolling ball surface S_c.

Finally, consider the triangle t of $\partial\mathscr{T}'$ with vertices v_1, v_2, and v_3. It corresponds to the point $\bar{t}$ in $\partial\mathscr{B}'$. In this case we have a solvent atom B with no degrees of freedom (it cannot roll since its center is fixed in $\bar{t}$). The contribution of B to S_c is thus given by:

$$\partial B \cap \pi(v_1, v_2, t; v_3) \cap \pi(v_1, v_3, t; v_2) \cap \pi(v_2, v_3, t; v_1).$$

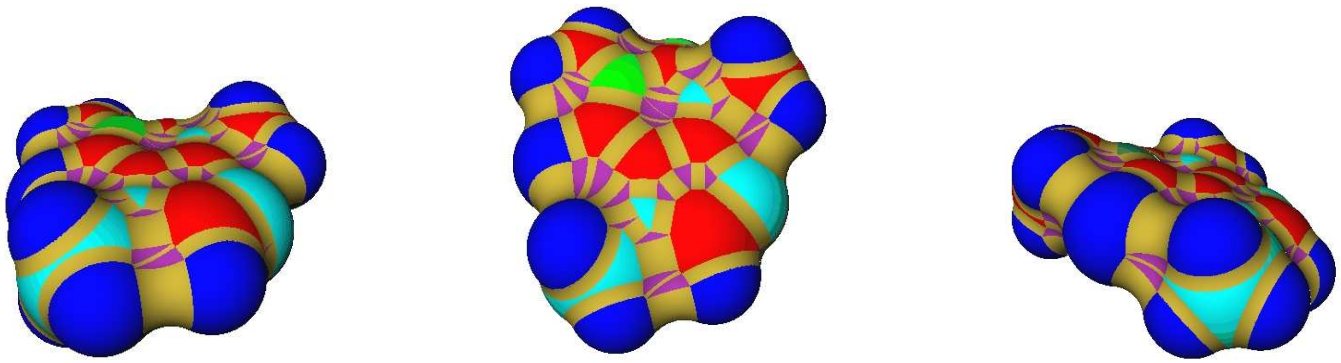


Figure 2.9: Complete Connolly surface of a caffeine molecule.

Figure 2.9 depicts a complete solvent contact surface (a superset of the rolling ball surface) of the caffeine molecule with the concave patches highlighted in purple.

2.2.4 Solvent contact surface

In this section we extend the method to achieve an exact NURBS representation of the solvent contact surface S_c (also known as the Connolly surface) of a molecule $\mathscr{B}$. The surface is defined as follows.

Definition 2.2.3. *A point p belongs to the* solvent contact *surface S_c of the molecule $\mathscr{B}$ with respect to a solvent with atoms of radius r iff:*

- *there exists a legal ball B_1 of radius r that contains p in its boundary:*

$$\exists B_1 \,|\, p \in \partial B_1 \text{ and } \overset{\circ}{B}_1 \cap \mathscr{B} = \emptyset \tag{2.7}$$

- *there is no legal ball B_2 of radius r that contains p in its interior:*

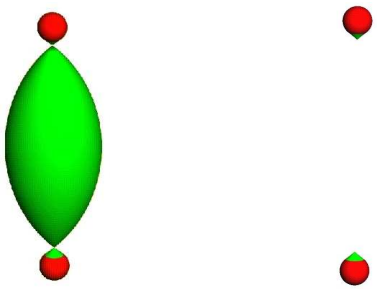
$$\overset{\circ}{B}_2 \cap \mathscr{B} = \emptyset \qquad \Rightarrow \qquad p \notin \overset{\circ}{B}_2 \tag{2.8}$$

The close relationship between the solvent contact surface and the rolling ball surface becomes clear from this definition.

Proposition 2.2.2. *If a point p lies on the solvent accessible surface S_c then it lies also on the rolling ball surface S_r.*

PROOF. The proof can immediately be derived from the comparison of definition 2.2.2 with definition 2.2.3. Further, from this follows that lemma 2.2.1 holds not only for S_r, but also for S_c. □

The problem that remains to be solved is the removal of (possible) self intersections that the rolling ball surface might have, and that make it differ from the solvent contact surface (for a classification of the classes of self-intersection that may occur see [9], fig1). This problem can be geometrically highlighted even with a set of two small balls along which a large radius probe is rolled (see figure 2.10). In this case the blending surface is formed by a toroidal patch that is self-intersecting.



(a) (b)

Figure 2.10: (a) The rolling ball surface (in green) with a probe of radius 10 on two spheres (in red) of radius 1 is a self intersecting surface. (b) The corresponding solvent contact surface has no self intersection.

To show the same problem for the concave patches at least three spheres are needed. Figure 2.11 shows three possible configurations of the solvent contact surface for a set of three balls. From the picture it is clear how complex the shape can get (with sharp features, varying in genus and possibly disconnected) even for a simple configuration of three balls.

In the following sections we will show how the patches of the rolling ball surface can be trimmed to get the exact representation of the solvent contact surface. As for the previous case we will report a brief sketch of the proof of correctness.

Convex Patches The convex patches of the solvent contact surface are exactly the same of the rolling ball surface. This derives immediately from the following:

Proposition 2.2.3. *The solvent contact surface S_c of the molecule $\mathscr{B}$ is completely included within the region between $\partial\mathscr{B}$ and $\partial\mathscr{B}'$, where $\partial\mathscr{B}'$ is the corresponding solvent accessible surface.*

Since S_c does not intersect the interior of $\mathscr{B}$ there is no nee to further trim the convex patches since they belong to $\partial\mathscr{B}$.

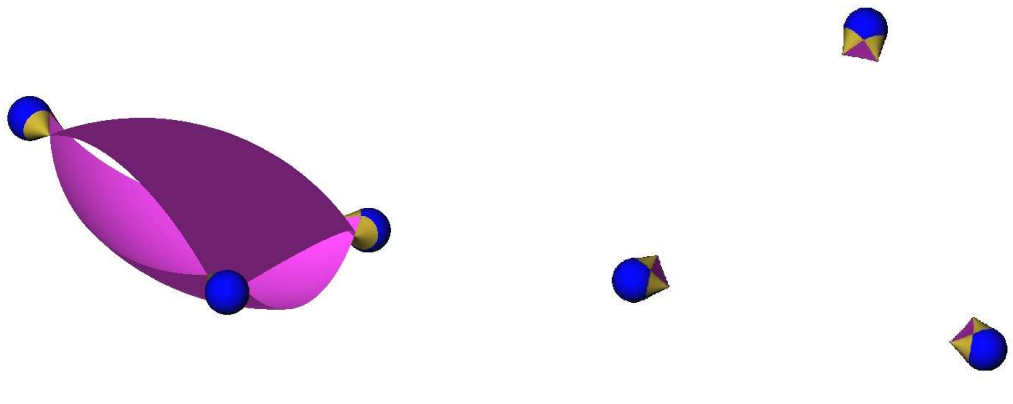
Toroidal Patches First of all, we exclude the possibilities of two toroidal patches intersecting each other and of a toroidal patch intersecting with a concave/convex patch.

Proposition 2.2.4. *Given two toroidal patches T_i,T_j (with $i \neq j$) their relative interiors are disjoint:*

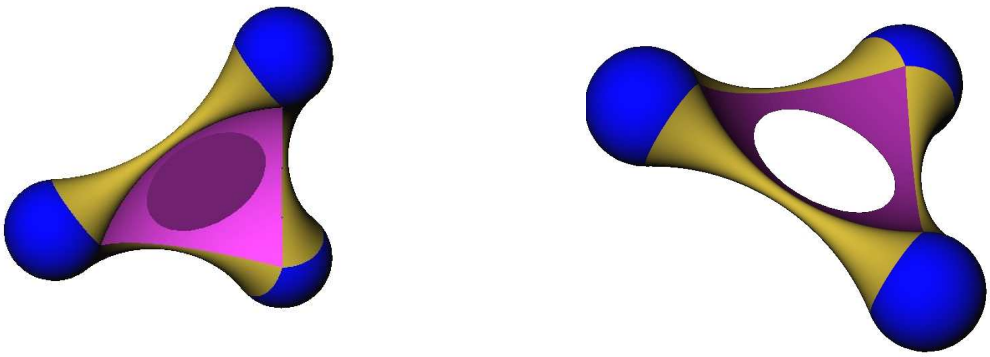
$$\overset{\circ}{T}_i \cap \overset{\circ}{T}_j = \emptyset$$

Proposition 2.2.5. *Given a toroidal patch T_i and a concave (convex) patch C_j, their relative interiors are disjoint:*

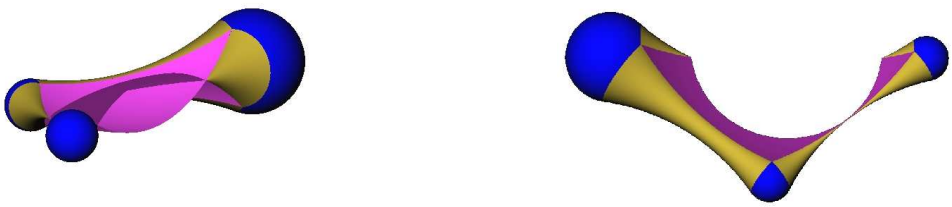
$$\overset{\circ}{T}_i \cap \overset{\circ}{C}_j = \emptyset$$



(a)



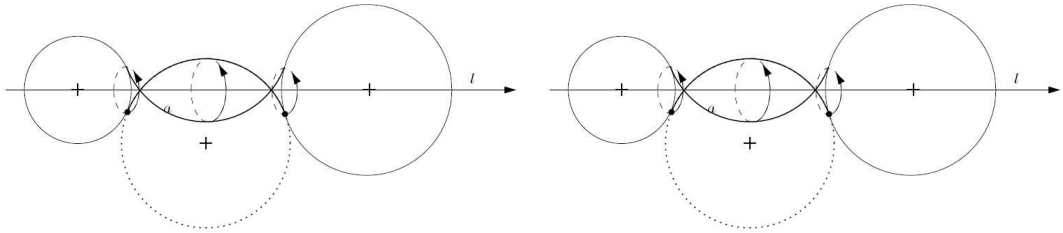
(b)



(c)

Figure 2.11: Three possible configurations of the solvent contact surfaces and rolling ball surfaces for different radii of the solvent and molecule atoms. On the left the self-intersecting rolling ball surfaces are shown. On the right the corresponding solvent contact surfaces are shown (without self-intersections).

From the two previous lemmas we derive that one toroidal patch can intersect only itself. This happens when it can be constructed as rotational surface of an arc of circle around an axes that intersect the arc (see figure 2.12). For each arc a rotating around an axis l intersecting a we must remove that portion of a lying on the "wrong" side of l. In this way we compute the arc a' (a disconnected subset of a) whose rotational surface around l has no self intersection as in figure 2.10.



(a) (b)

Figure 2.12: (a) The arc a rotating around the axes l describes a self intersection portion of torus. (b) The arc a' rotating around the axes l describes portion of torus with no self intersection.

Trimming the Concave Patches

First of all, we exclude the possibility of a concave patch intersecting either itself or a convex patch (we already know that it cannot intersect a toroidal patch).

Proposition 2.2.6. *Given a concave patch C_i and a convex patch C_j, their relative interiors are disjoint:*

$$\overset{\circ}{C}_i \cap \overset{\circ}{C}_j = \emptyset$$

Proposition 2.2.7. *One concave patch cannot intersect itself.*

As show in Figure 2.11 two distinct concave patches can intersect each other. Since each concave patch is a portion of sphere we have to deal again with a sphere-sphere intersection problem. Hence we can simply maintain the regular triangulation of the centers of the concave patches (in this case all the weights are equal) so that we have all the relation of reciprocal intersection between concave patches. It has been shown in section 2.2.1 that the intersection between each pair of spheres is mapped to the insertion of an additional trimming circle in the domain space. Taking into account the intersections between pairs of concave patches, we must add some trimming circles to the domains of each concave patch to obtain the result of Figure 2.11.

2.3 Molecular Surface Computation using Adaptive Grids

An algorithm to compute the molecules SES and other related properties is presented, which provides an accurate surface definition and efficient representation for operations required during docking.

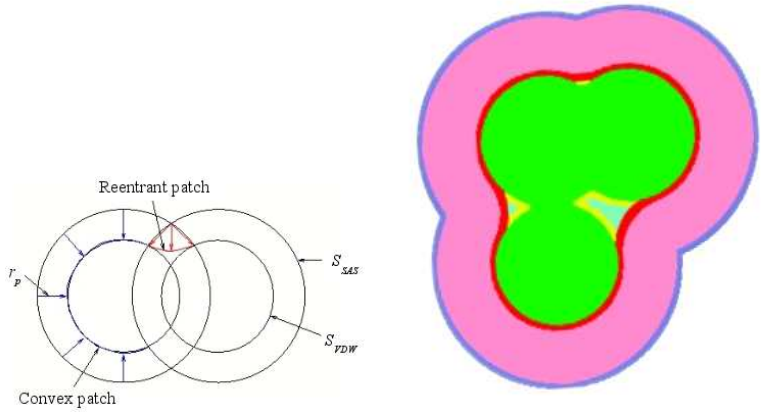
The algorithm uses an octree based subdivision scheme to adaptively improve the resolution of the representation near the surface. The surface itself is approximated as a level set of a signed distance function computed based on values assigned at the gridpoints.

2.3.1 Signed Distance Function based Family of Surfaces

We define a volume function Φ and use its contours to provide a family of molecular surfaces. Consider the union of atoms of the molecule $\cup B$. Inflate each atom b in this set by the probe radius (solvent radius) r_p to give the new complex $\cup B_{r_p}$. Let its boundary be Γ_B. Let Φ define the signed distance function of ΓB, such that the interior (closer to van der Waals) is given a positive sign. Let all regions within the atom (see [84] for definitions) be given a constant positive high value H.

Observations and lemmas:

- Isosurfaces S_I with isovalues $I : 0 \leq I \leq H$ form a family of surfaces.
- $\Gamma_B = S_0$, as defined by Lee and Richards, is the SAS of the molecule.



(a) (b)

Figure 2.13: (a)The Molecular Surface defined as the contour at a distance of r_p away from the S_{SAS}, towards the atom centers. (b)3 atoms showing different surfaces and regions. S_{SAS}: dark blue, V_{SAS}: pink, S_{SES}: red, V_{SES}: light blue, S_{VDW}: yellow and V_{VDW}: green

- S_{r_p} is the SES.
- $S_{x \to H^-}$ is the van der Waals surface.
- $\{x : 0 \leq \Phi(x) \leq H\}$ defines a volume exclusion function, which can be convenient to use in electrostatic computations.
- The region $\{x : -r_p \leq \Phi(x) \leq r_p\}$ has a high probability for the presence of surface atoms of a protein docked to the current one.

The above observations point to the obvious advantages in using such a definition for our molecular structure representation for docking. Let us further examine some of them in detail.

$\Gamma_B = S_0$ is the SAS, and S_{r_p} is the SES: By definition of the SAS, it is the locus of the center of the probe as it rolls over the spherical atoms of the protein. But it should be noted that the grid based definition also includes holes, which may be removed if necessary. The SES surface is always defined by points on the probe. It is in fact the boundary of the region accessible to any part of the probe radius. Hence, it is always at a constant distance of r_p away from the locus of the center. Therefore, our third observation follows. Again, holes are included in our definition and need to be removed if required.

$\{x : 0 \leq \Phi(x) \leq H\}$ provides a volume exclusion function: Volume exclusion functions are used in setting up dielectric constant for electrostatic computations. The twin requirements of smoothness at the boundary and accuracy in modeling the SES are not met by many of the definitions in practise today. Our definition is provides a 'sufficiently' smooth function around the SES (Φ is smooth in the radial direction), and contains the SES within it.

Isosurfaces S_I with isovalues $I : 0 \leq I \leq H$ form a family of surfaces

At the extremes isovalues, we have the SAS and the VDW surfaces, and the SES lies in between them at an isovalue of r_p.

Interface of docked ligand is in the region $\{x : -r_p \leq \Phi(x) \leq r_p\}$:

For good shape complementarity, as observed in docked complexes, atoms of the ligand must lie close to the surface of the protein. The above 'skin' definition provides a functional representation for such a region, as it defines the region where a probe sphere is in touch with the protein.

2.3.2 Notations

Let the molecule M is represented as a collection of atoms A_i and each atom is represented using a center $\mathbf{c}_i$ and radius r_i. Let the radius of the prove used for defining the Solvent accessible surface (SAS) and Solvent excluded surface (SES) be r_p.

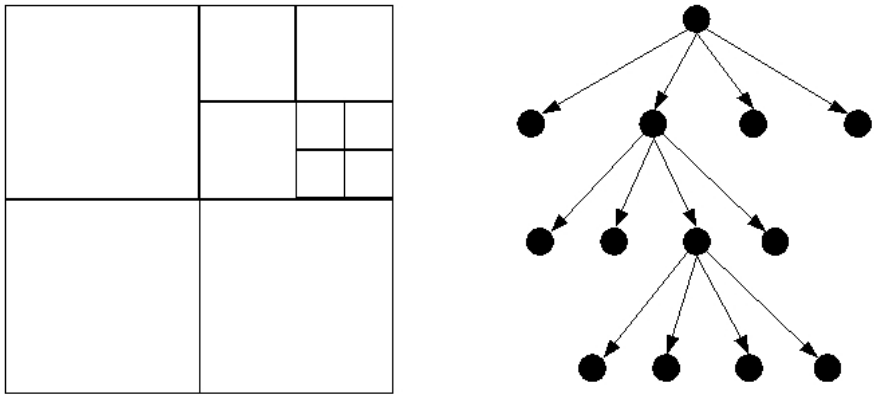


Figure 2.14: Quadtree, a 2D analog of Octree. The root represents the largest square. Children of a node represent the four sub-squares of the parent square (nodes are ordered left to right, corresponding to sub-squares ordered clockwise starting at top-left)

Let us consider a grid G in which the molecule is embedded to have a maximum and minimum grid spacing, h_{max}, h_{min}. Let, l be the length of the maximum side the molecule. We assume that, $h_{max} = 2^k h_{min}$ and $l = Nh_{min}$. For each grid-cell $g \in G$, let $dist(g, p)$ be the shortest distance from an arbitrary point p to any point on g. See notes at the end of this chapter for details.

Also, let S_{VDW} represent the van der Waals surface of the molecule and V_{VDW} be the volume enclosed by S_{VDW}. Similarly, we define S_{SAS}, V_{SAS} and S_{SES}, V_{SES}.

2.3.3 Adaptive grids octree

An octree is a spatial decomposition data structure for 3 dimensions. It is a special case of a k-d tree with $k = 8$. The entire volume (usually uniform cube) is represented by the root of the octree. Every node n_i of the octree has no children or exactly 8 children, corresponding to the 8 sub-cubes formed by bisecting the cube represented by n_i along X, Y and Z axes. If a node has no children, then it is called a leaf. See Figure 2.14 for a 2D example.

Hence, instead of having $O(N^3)$ grid-cells, we can only ncrease the resolution near regions of interest, namely near the surface of the molecule and have a coarse-grained grid elsewhere (outside the molecule and inside the VDW surface).

To facilitate the computation of S_{SES}, S_{VDW} etc., the adaptive grids is defined as an augmented octree, where each cell (node) and each gridpoint (corners of cells) contains some additional information as listed below.

- **Grid-cells:** The following are stored at each grid-cell $g \in G$-
  - Whether it belongs to S_{VDW} or S_{SAS}.
  - A list of atoms, $A_{VDW} | \forall A_i \in A_{SAS} : dist(g, \mathbf{c}_i) \leq r_i$
  - A list of atoms, $A_{SAS} | \forall A_i \in A_{SAS} : dist(g, \mathbf{c}_i) \leq (r_i + r_p)$
- **Grid-points** For each gridpoint **p**
  - Whether it belongs to V_{VDW}, V_{SAS} or lie outside V_{SAS}
  - A list atoms, $B_{VDW} | \forall A_i \in A_{VDW} : dist(\mathbf{p}, \mathbf{c}_i) \leq r_i$
  - A list atoms, $B_{SAS} | \forall A_i \in A_{SAS} : dist(\mathbf{p}, \mathbf{c}_i) \leq (r_i + r_p)$
  - A signed distance value δ denoting the shortest distance of **p** from the S_{SAS}

2.3.4 Algorithm sketch

The algorithm iteratively classifies the grid-cells and grid-points and then computes the signed distance values (δ).

- **Initialization:**
  1. Create a grid with uniform grid-size of h_{max}
  2. Create corresponding octree.
  3. Mark all cells as belonging to neither S_{VDW} nor S_{SAS}
  4. Mark all gridpoints as outside of V_{SAS}
  5. Assign $\delta = -\infty$ for all gridpoints.
- **Insertion:** For each atom $A_i \in M$
  1. Insert A_i into G by locally updating points $\mathbf{p} \in G$ as belonging to V_{SAS} or V_{VDW}. If $\mathbf{p}$ belongs to V_{VDW}, set $\delta = \infty$.
- **Refinement:**
  1. For each cell $g \in G$, determine whether it is a boundary cell.
  2. For all boundary cells intersected by more than three atoms, subdivide recursively and classify the subdivided vertices and cells.
- **Computing S_{SES}:** For each gridcell $g \in G$ classified as S_{VDW}
  1. For each point $\mathbf{p}$ around g belonging to V_{SAS}, find the closest distance δ of the point $\mathbf{p}$ from the S_{SAS} boundary.
  2. Use level sets of the signed distance function defined by the δ values. See Section 2.3.1 for details of the signed disance function based family of surfaces.

Each step of the algorithm is explained in greater detail in the following sections.

2.3.5 Algorithm details

Vertex classification

For any gridpoint $\mathbf{p}$, if $dist(\mathbf{p}, \mathbf{c}_i) \leq r_i$ then $\mathbf{p}$ is classified as belonging to the V_{VDW}, and if $r_i < dist(\mathbf{p}, \mathbf{c}_i) \leq (r_i + r_p)$, then $\mathbf{p}$ is classified as belonging to the V_{SAS}. The lists B_{VDW} and B_{SAS} are also updated. Note that, vertices classified as V_{VDW} are fixed while vertices marked V_{SAS} could become marked V_{VDW} with the insertion of new atoms.

We use the method described by [3] for sphere-cube intersection tests. The cost of this insertion is linear in the number of atoms and cubic in the resolution of the grid: $O(Mh_{max}^3)$.

Cell classification

We examine the classification of the eight corners of each cell of the grid. If some vertices are classified as V_{VDW} and others are not, then mark the cell (and the vertices) as belonging to S_{VDW}. Otherwise if some of the vertices are classified as V_{SAS} and some other vertices are classified as outside V_{SAS}, mark the gridcell (and the vertices) as belonging to S_{SAS}. Update the lists A_{VDW} and A_{SAS}. This operation is linear in the number of cells of the grid $O((N-1)^3)$.

Adaptive subdivision of S_{SAS}

Each boundary cell g which contains more than three atoms contributing to it is subdivided up to a user defined resolution. We classify each subdivided vertex using the atoms in A_{SAS} of g, as belonging to the interior of the V_{SAS} or not. Using a technique similar to obtain boundary cells, we generate a list of finer boundary cells in the subdivided cells. The maximum cost of this operation is $O((N-1)^3(h_{max}/h_{min})^3)$, although the average case cost should be much smaller as only the boundary cells are involved.

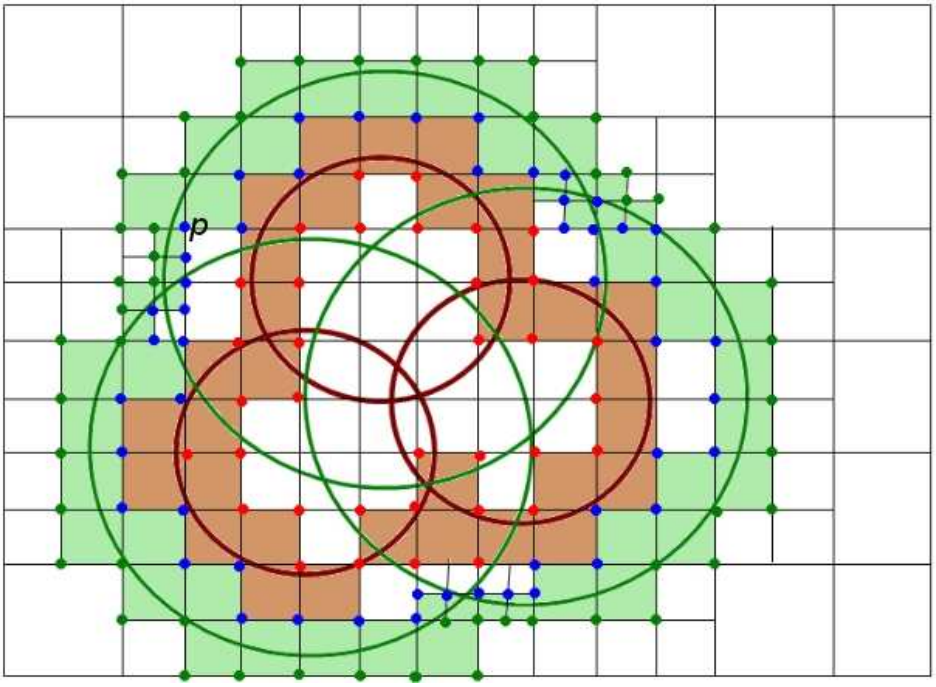


Figure 2.15: An adaptive subdivision of the input grid. In this 2D figure, we have subdivided quads with more than 1 SAS patch as an example. In the figure, brown and greed lines define the boundary of atoms without and with solvent enlarged radius. The cells belonging to S_{SAS} are colored light green and the cells belonging to S_{VDW} are colored light brown. Gridpoints belonging to V_{DVW} and S_{VDW} are colored red. Gridpoints belonging to V_{SAS} and S_{SAS} are colored blue (these are used for SES estimation). Gridpoints belonging to S_{SAS}, but outside of V_{SAS} are colored green

Computing δ

For each gridpoint $\mathbf{p}$ in V_{SAS} and S_{VDW}, search all cells belonging to S_{SAS} around $\mathbf{p}$ for the shortest distance δ from $\mathbf{p}$ to the S_{SAS}. For cells with only one intersecting atom, the exact distance from $\mathbf{p}$ to the spherical patch of S_{SAS} in that cell is computed (see Section 2.3.6 for details). On the other hand, if a cell contains more than one atom (and is hence subdivided), we just take the minimum distance from the center of all the subdivided cell to $\mathbf{p}$ as the distance of the spherical patch in the cell to $\mathbf{p}$. The cost of this search will vary as r_p^3, the number of boundary van der Waals cells in the volume and the accuracy desired (as provided by h_{min}).

2.3.6 Spherical Patch Intersection

Let us define a sphere as having a center $\mathbf{c} = \{c_x, c_y, c_z\}$ and radius r. Define a cube with points $\mathbf{a}_1, .., \mathbf{a}_8$. The following computations are applied for each gridpoint $\mathbf{a}_i$ iff $dist(\mathbf{a}_i, \mathbf{c}) \leq r$.

Intersection of sphere and face of cube

The intersection is always arc(s) of a circle. We will consider only a face parallel to the xy plane. Other cases should follow similarly. The point of projection of $\mathbf{c}$ to the plane containing the face is $\mathbf{c}' = \{c_x, c_y, \texttt{z coordinate of face}\}$. This point is the center of the circular arc. The radius using Pythagoras theorem is $\sqrt{r^2 - dist(\mathbf{p}', \mathbf{c})^2}$. The intersection points on the edges, if any, is now computed by intersecting this circle with the line containing the edge, and checking whether the points lie within the edge.

Shortest distance of point to a circular arc on the same plane

Lemma Given a circular arc with center $\mathbf{c}'$, radius $\mathbf{r}$ and endpoints $\mathbf{p}_1$, $\mathbf{p}_2$ and a point q on the same plane, let $d_0 = dist(\mathbf{q}, \mathbf{c}')$, $d_1 = dist(\mathbf{q}, \mathbf{p}_1)$ and $d_2 = dist(\mathbf{q}, \mathbf{p}_2)$. Then, the shortest distance of q to the arc is defined as follows-

- If the point is inside the infinite sector defined by the arc, then the shortest distance is : $\mathbf{r} - d_0$.
- Otherwise, the shortest distance = $min(d_1, d_2)$.

Shortest distance of point to a spherical patch inside a cube

The spherical patch is bounded by circular arcs on the faces of the cube. Consider the circle a boundary arc is part of. The center $\mathbf{c}$ of the sphere and this circle will form an infinite cone. Hence the collection of boundary arcs form a collection of infinite cones.

Lemma The shortest distance of a point $\mathbf{p}$ to a spherical patch in a cube is:

- Point is inside each of the infinite cones. The shortest distance is : $\mathbf{r} - dist(\mathbf{p}, \mathbf{c})$.
- Otherwise, the shortest distance is the minimum of the shortest distances of the point to each of the bounding arcs.

2.3.7 Complexity

For M atoms (including B boundary atoms), smallest grid spacing h, grid length N, VDW radius r and solvent radius r_p, the timing complexity is

- SDF initialization: $O(N^3)$
- Insertion of atoms: $O(M(\frac{2(r+r_p)}{h})^3)$
- Boundary atom detection:
  - Uniform grid traversal: $O(N^3)$
  - Sphere traversal: $O(M(\frac{2(r+r_p)}{h})^3)$

  - Octree traversal: $O(log(N^3)B) \leq O(log(N^3)M)$

- Patch voxel distance computation: $O(M(\frac{2(r+r_p)}{h})^6 C)$, C is cost(dist(patch, voxel))
- Isocontouring for visualization: $O(N^3)$

2.3.8 Self Intersections in Patch Complex Model

A patch complex (consisting of convex, concave and toroidal patches) can be derived using our adaptive grid structure and SAS sphere intersection enumeration. But the patch complex is known to have problems of bad intersections. According to lemma 3, 4, 5, 6 and 7 from Bajaj et al [15], there are only two possible self intersections that occur in the commonly used rolling ball model:

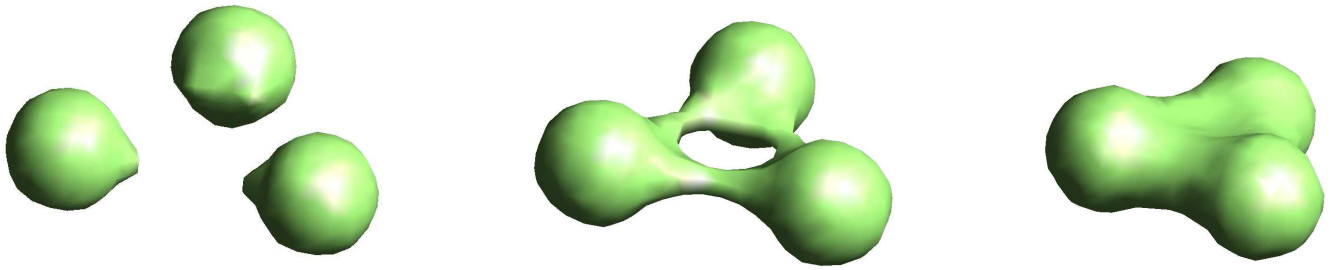
- A toroid can self intersect with itself (Figure 10(a) in [15]).
- A concave patch can intersect with another in the case of a 3 atom model (Figure 9 in [15]).

In figure 2.16, we show the surface computed when two atoms are present, and moved close till they form a single surface patch. In the case of surfaces computed from the rolling ball model, we would instead get a self intersecting toroidal patch when the gap between the atoms becomes smaller than the diameter of the probe radius. This can be computed by looking at all pairs of intersecting SAS spheres, which is already given in our adaptive grids. To examine the intersection of two concave patches, we look at the three atoms model as shown in figure 2.17. Again, we get similar results compared to [15]. This case occurs when there are three intersecting SAS atoms, and can be enumerated by our grid.

(a) The toroidal patch is disjoint and there is no self intersection.

(b) As the atoms come closer, a well defined toroidal patch is created.

Figure 2.16: The solvent excluded surfaces of two atoms which come closer.



(a) The 2 concave patches are disjoint and there is no wrong intersection.

(b) As the atoms come closer, a well defined patch, similar to the approximations in [15] is created.

(c) At mutually closer distances, the topology changes and the center hole disappears.

Figure 2.17: The solvent excluded surfaces of three atoms which come closer.

2.3.9 Operations Supported by the Adaptive Grid

1. **Surface atoms detection**

Surface atoms are defined as those within a certain distance from the Molecular Surface. To obtain these atoms, we first compute the Molecular Surface. Next we search locally around each atom to find the distance of the atom from the surface. This operation is linear in the number of atoms and cubic in the resolution of the grid.

2. **Population of skin region**

   We define the skin region of one molecule as the region belonging to the probe as it rolls on the surface, and defined as Solvent Accessible Surface 2 Volume (V_{SAS2}). We define the skin implicitly as a set of spheres packing the region. The packing density is itself chosen to approximately equal the packing of the atoms belonging to the molecular surface. The region is defined over a trilinear grid in which the molecule is embedded. The grid spacing h is chosen to preserve the features of the molecule. Assuming that the interatomic distance is ~ 1, we can use $h = 0.5$. By finding the boundary vertices of the *SAS*, we can obtain potential centers for the skin spheres. A packing algorithm decides, based on the packing density required, if a potential center should contain an atom or not.

3. **Area volume computations**

   We use primal contouring to define the surface and volumes. The area under the surface is approximated by piecewise linear elements of the isocontour. The volume is approximated by the volume enclosed by that piecewise linear approximation. This cost is linear in the size of the grid.

4. **Curvature and normal computations**

   These differential properties are computed using a two step process. Initially, when we propagate the distance from the S_{SAS}, we also store whether the nearest patch is the intersection of one, two or more spheres. In each case, we can analytically provide the answer to the curvatures. For example, for a sphere with radius r, the Mean and Gaussian curvatures are $-1/r$ and $1/r^2$ respectively. In the second step, we compute the derivatives from the isocontour. At points where the two vary significantly, we choose to keep the value provided by the differencing scheme as the signed-distance algorithm used is only an approximation.

2.3.10 Sum of Gaussians (SoG) based approximation

The adaptive grid is also used to compute a sum of Gaussians approximation to the Molecular Surface. A base uniform grid is used to compute the Fourier coefficients of the atom centers and the kernel function using a non-equispaced fast Fourier Transform. The summation is evaluated at points around the surface chosen from the adaptive grid. For details of this algorithm, kindly refer to the technical report [14]. The cost of the algorithm, for M atoms, N output points, n Fourier coefficients and a accuracy requirement ε is:

Lemma For tensor product kernels with Fourier coefficients K_ω, the number of coefficients n needed is at most: $n = min(\hat{n}) : \sum_{\omega \in I_{\hat{n}}} (K_\omega)^2 \geq \frac{V}{2\pi} - \frac{Mmin_j(|c_j|^2)V}{(||c||_1)^2}(\frac{\varepsilon}{3})^2$, where V is the integral of the kernel from $(-0.5..0.5]^3$.

Lemma For tensor product kernels whose Fourier coefficients decay at least inversely with frequency, the number of coefficients n needed is $O(M^{1/3}\varepsilon^{3/2})$.

Lemma The fourier coefficients of a Gaussian function e^{-Bx^2} decay as the inverse of the frequency ω: $G_\omega \leq max(\frac{1}{2\pi\sqrt{\pi}}, \frac{1}{2\pi}erf(\frac{\pi}{\sqrt{B}}), \frac{3\sqrt{B}}{e\pi^{3/2}}, 4\sqrt{\frac{2}{\pi e}}, \frac{Be^{-(1+\pi^2/B)}}{\pi^4})\frac{1}{\omega}, \quad (\omega \geq 2)$. The truncation of the Gaussian can be expressed as convolution with a sync function in Fourier space. Hence the Fourier series coefficients of the truncated Gaussian function can be now written as $\int_{-\infty}^{\infty} \sqrt{\frac{\pi}{B}} e^{-\pi^2 t^2/B} sin(2\pi\omega)/(2\pi\omega - t)dt$. We then bound the sync function with a polynomial and integrate by parts to obtain the result.

2.3.11 Results

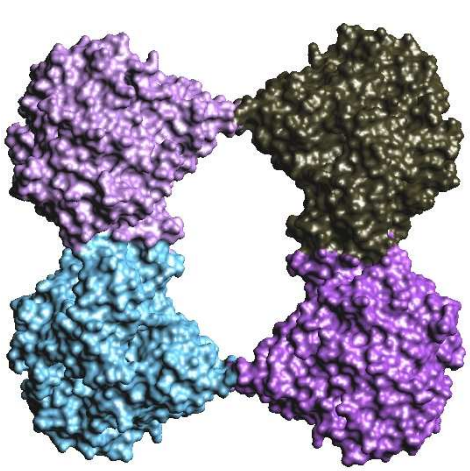
Region classification and construction of molecular surfaces

Before we provide timing, geometric and functional properties and skin, surface regions, we present the results of our classification and signed distance function on a 3 atom model in figure **??**. Using a relatively high resolution grid of 128^3, we

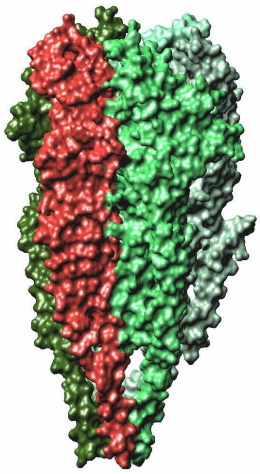
classify grid points depending on the volume and surface it is part of, giving priorities of surface class over volumes and SES class over other surfaces. The figure is a 2D cross section of a volume rendering of the classified volume.

The Solvent Excluded Surface

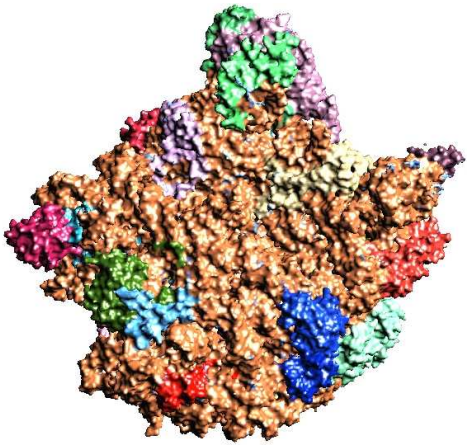
The solvent excluded surface is obtained as an isocontour with value equal to probe radius. In figure 2.18, we show colored visualizations of four different molecules.



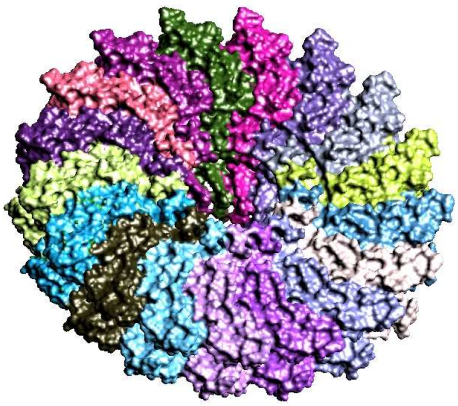
(a) An acetylcholine esterase (1C2B.PDB). It is shown in its tetramer form. Each unit, containing 4172 atoms each, is colored with a different color.



(b) The nicotinic acetylcholine receptor with over 14,000 atoms (2BG9.PDB). It has 5 chains, shown in different colors.



(c) The large ribosomal subunit (1JJ2.PDB) has almost 100,000 atoms. The main RNA chain (in brown) and other chains are shown.

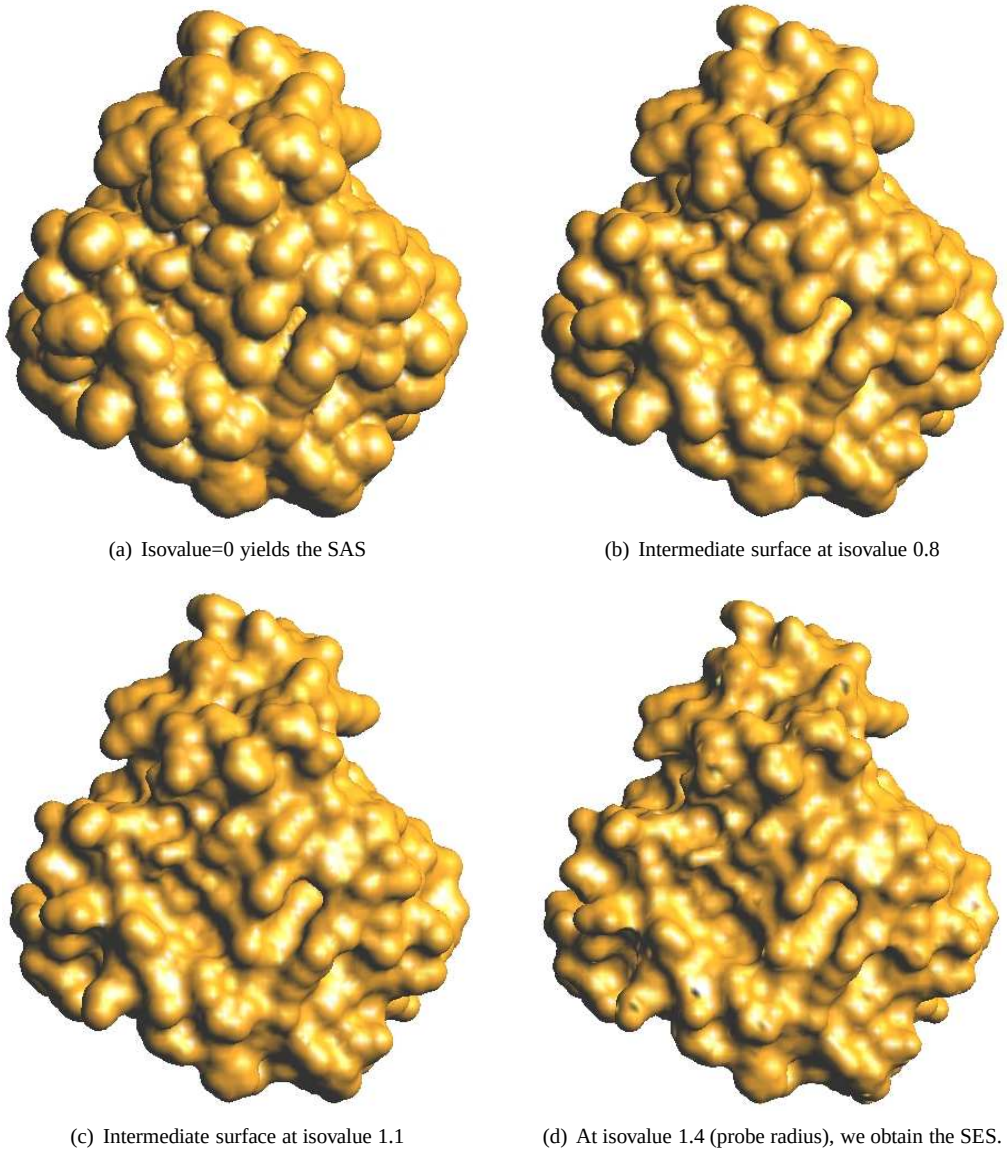


(d) The tobacco mosaic virus, a helical virus (1EI7.PDB). The repeating subunits, each containing 2806 atoms, are shown.

Figure 2.18: The solvent excluded surfaces of four different molecules.

Family of surfaces

In figure 2.19, we show four different surfaces computed from the adaptive grid, at four different isovalues. The myoglobin molecule, 101m.pdb, is used as a test case. At a distance of 0, we get the SAS surface, which is the union of spheres model, with each atom represented as a sphere with radius equal to the sum of its radius and a probe radius. In this example, we used a probe radius of 1.4Å. As we go further away, we get a smooth deformation of the SAS surface to the SES surface, as shown in the different figures. Since we are interested in the SES, we do not compute further in practise, but in theory, higher isovalues will take us closer to the van der Waals surface. This example shows the utility of our method as a volume exclusion function for computing electrostatics, which needs a smooth decay at the SES boundary.



(a) Isovalue=0 yields the SAS

(b) Intermediate surface at isovalue 0.8

(c) Intermediate surface at isovalue 1.1

(d) At isovalue 1.4 (probe radius), we obtain the SES.

Figure 2.19: Our signed distance function based definition yields a family of surfaces which we can extract using a novel adaptive grid based algorithm.

Timing

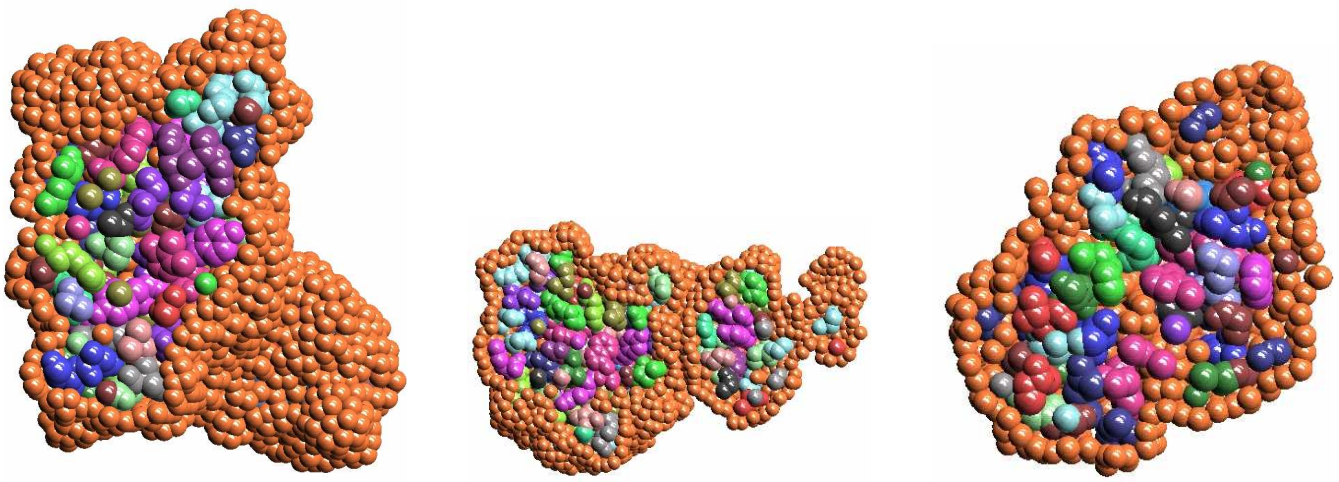
The cost of the algorithm depends on the depth of the adaptive grid, the resolution of the initial base grid and the size of the molecule. In table 2.1, we provide the time taken to compute the properties on the grid, including surfaces and demarking volumetric regions for different molecules and grid sizes. As the number of atoms increase, the time taken increases, but the fixed output grid size reduces the number of relevant search points within the SAS and VDW regions. Hence there is no direct correlation seen between the two. If the grid resolution can be chosen depending on the molecule size, then the time would increase monotonically with the number of atoms for molecules with similar distribution of atoms (say for a set of globular proteins).

Surface atoms detection

The surface atoms of three proteins from the complexes, anti-idiotypic fab (1iai.pdb), hemagglutinin (2vir.pdb) and bob-white quail lysosyme (1bql.pdb) are visualized in figure 2.20. The interior atoms are colored by the residue they belong to while the outer surface atoms all have an orange color. We show a cutoff of the three molecules to reveal the surface and interior.

PDB Id	Number of atoms	time (64^3)	time (128^3)
3sgb	1912	11	85
1brc	2197	6	58
2ptc	2243	6	53
2kai	2267	7	74
3tpi	2313	6	54
1tab	2387	9	72
1ppf	2520	7	63
4cpa	2739	10	85
1mkw	4844	8	60

Table 2.1: Times (in seconds) taken to compute the adaptive grid based surfaces and volume regions for different initial grids which are adaptively subdivided to a depth of 3.



(a) Hemagglutinin (b) Anti-Idiotypic Fab (c) Bob-white quail lysosyme

Figure 2.20: Surface atoms of three complexes shown in orange over the interior atoms which are colored by their residue type.

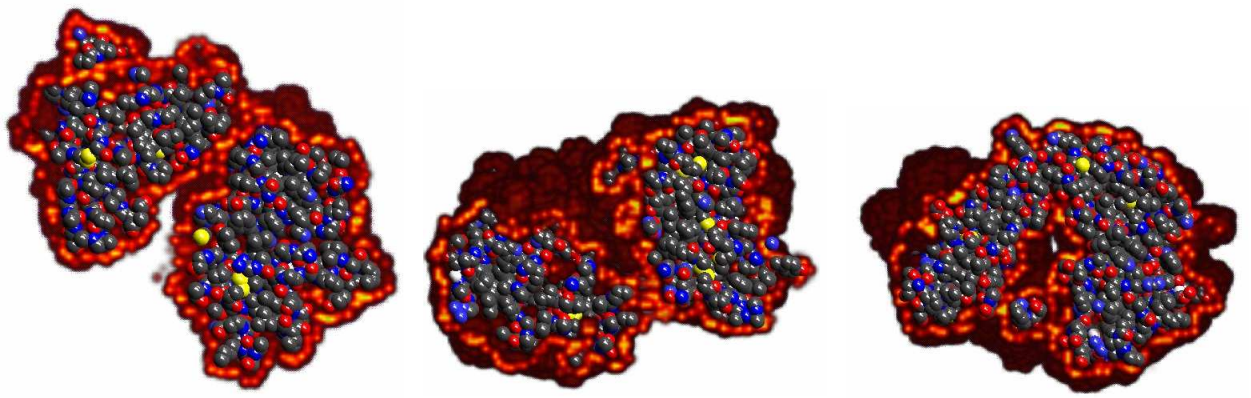
$\mathbf{SAS}^2$ skin region construction

From the same above three complexes (1iai,2vir and 1bql), we extract the second protein and compute the skin regions (see figure 2.21) defined by the volume where the probe is present and touching the molecule. This region is used later in docking as it represents a volume where the interface atoms from the docking protein have a high probability of being present.

2.4 Dynamic Update of Molecular Surface Under Change in Radii

We analyze the complexity of two main classes of updates that yield a family of all the molecular surfaces obtained for different solvent radii: (1) updates that keep the Power Diagram fixed (quadratic growing of the radius of the solvent ball); (2) updates that modify the Power Diagram (linear growing of the radius of the solvent ball).

In both cases efficiency is achieved trough the introduction of a novel geometric construction. In case (1) we use a new constructive approach to duality that generalizes the standard "lifting" scheme [35], showing that the Power Diagram of a molecule (3D union of balls) constitutes a compact representation of the collection of all the Power Diagrams of the trimming circles of all the patches in a molecular surface. In particular the convex cell of the 3D Power Diagram relative to the ball B is the dual of the 2D Power Diagram of the trimming circles of B. As a first approximation (with the bonus of being simpler



(a) Immunoglobulin from influenza virus (b) Idiotypic fab of virus neutralizing antibody (c) Hylel 5 Fab

Figure 2.21: Surface atoms of three complexes shown in orange over the interior atoms which are colored by their residue type.

and more efficient) we consider the molecular surfaces obtained by disproportionally increasing the solvent radius so that the associated Power Diagram remains unchanged. We show how we can keep track of the topological changes that occur in the trimming curves of the patches that form the molecular surface so that its boundary representation can be updated efficiently. Furthermore, we compute and dynamically update an exact boundary representation of the molecular surface so that the same dynamic data structure is also suitable for molecular modeling operations such as those supporting synthetic drug design [55].

In the case (2) setting, where the 3D Power Diagram is subject to flips, we use the same construction as in [41] based on the definition of a 4D complex of convex polytopes $\mathscr{C}$ whose "horizontal" slices are all the possible 3D Power Diagrams of the growing balls for any growth factor r. Hence we apply a simple hyperplane sweep algorithm to optimally maintain the dynamic Power Diagram of the linearly growing balls. Thus in this case we compute exactly the offset of the union of balls (so that its topology can be precisely determined), even when it requires a change in the nearest neighbor (under power distance) relations among the atoms corresponding to flips in the associated Regular Triangulation. More generally, for a set of balls in d-dimensional space this requires the construction of a complex of convex polytopes in $(d+1)$-dimensional space whose "horizontal" slices are all the possible Power Diagrams.

In Section 2.4.1 we introduce the fundamental equations that form the basis of the presented approach for molecular modeling. For a more extensive discussion of the conditions under which the present approach can be extended to a more general case unifying geometries other than spheres, the interested reader is referred to [18]. While for our purposes we deal with $d = 3$, the results are easily extended to arbitrary dimension.

See Section 2.2.1 for a discussion on trimmed NURBS representation of Molecular surfaces. And the Relavant Math Section for details about power diagrams.

2.4.1 Preliminaries

Balls in $\Re^3$ and Halfspaces in $\Re^4$

In this section we introduce the fundamental equations that form the basis of the presented approach for molecular modeling. For a more extensive discussion of the conditions under which the present approach can be extended to a more general case unifying geometries other than spheres, the interested reader is referred to [18]. While for our purposes we deal with $d = 3$, the results are easily extended to arbitrary dimension.

Consider in $\Re^4$ the implicit equation of the unit ball:

$$\xi_1^2 + \xi_2^2 + \xi_3^2 + \xi_4^2 - 1 \leq 0 \ . \tag{4.9}$$

Its boundary has parametric equations which are:

$$\xi_i = \frac{2x_i}{x_1^2 + x_2^2 + x_3^2 + 1}, \quad i = 1,2,3 \qquad\qquad \xi_4 = \frac{x_1^2 + x_2^2 + x_3^2 - 1}{x_1^2 + x_2^2 + x_3^2 + 1} \ . \tag{4.10}$$

The boundary of the ball (4.9) is the closure of the image of $\Re^3$ in $\Re^4$ under the mapping (4.10). The inverse map of (4.10) is given by

$$x_i = \frac{\xi_i}{1-\xi_4}\ , \quad i = 1,2,3 \tag{4.11}$$

for $(\xi_1,\xi_2,\xi_3,\xi_4)$ on the unit sphere $\xi_1^2+\xi_2^2+\xi_3^2+\xi_4^2 = 1$. The point $(0,0,0,1)$ in $\Re^4$ is the image of the point at infinity of $\Re^3$.

Now consider the linear halfspace:

$$h: \quad a_0 + a_1\xi_1 + a_2\xi_2 + a_3\xi_3 + a_4\xi_4 \le 0\ , \tag{4.12}$$

where not all of $\{a_1,a_2,a_3,a_4\}$ are zero. Its pre-image in $\Re^3$, given by the mapping (4.10), is

$$b: \quad a_0(x_1^2+x_2^2+x_3^2+1) + a_1\,2x_1 + a_2\,2x_2 + a_3\,2x_3 + a_4(x_1^2+x_2^2+x_3^2-1) \le 0\ . \tag{4.13}$$

If $a_1^2+a_2^2+a_3^2+a_4^2-a_0^2 \ge 0$ and $a_0+a_4 > 0$, this is the ball of center $-(a_1,a_2,a_3)/(a_0+a_4)$ and radius $(a_1^2+a_2^2+a_3^2+a_4^2-a_0^2)^{1/2}/(a_0+a_4)$. If $a_1^2+a_2^2+a_3^2+a_4^2-a_0^2 \ge 0$ and $a_0+a_4 < 0$, this is the union of the sphere of center $-(a_1,a_2,a_3)/(a_0+a_4)$ and radius $(a_0^2-a_1^2-a_2^2-a_3^2-a_4^2)/(-a_0-a_4)$ and its exterior. When $a_0+a_4 = 0$, this is a halfspace, and when $a_1^2+a_2^2+a_3^2+a_4^2-a_0^2 < 0$ and $a_0+a_4 \ne 0$, this is a ball of imaginary radius, and contains no real points.

A fundamental relationship is that spheres that contain a point (c_1,c_2,c_3) in $\Re^3$ map to hyperplanes that pass through the point $(2c_1,2c_2,2c_3,c_1^2+c_2^2+c_3^2-1)/(c_1^2+c_2^2+c_3^2+1)$ in $\Re^4$. This is a result of the relation

$$(c_1^2+c_2^2+c_3^2+1)a_0 + 2(c_1a_1+c_2a_2+c_3a_3) + (c_1^2+c_2^2+c_3^2-1)a_4 = 0\ .$$

A consequence of this relationship is that a set of spheres passing through two distinct points in $\Re^3$ correspond to a set of hyperplanes in $\Re^4$ that contain a certain line. Since the actual points of intersection in $\Re^3$ are mapped to points on B, the line in $\Re^4$ must intersect B in two points. A set of spheres in $\Re^3$ which intersect at one point are mapped to into hyperplanes whose line of intersection is tangent to B. A set of spheres whose combined intersection is empty are mapped to hyperplanes whose line of intersection, if any, does not intersect B. This situation is illustrated for $d = 2$ in Figure 2.22. Let l be the line of intersection of the boundaries $\partial h' \cap \partial h''$ corresponding to two distinct intersecting balls b' and b''. We have that $\partial b'$ intersects $\partial b''$ if and only if l intersects B, that is, if the distance from l to the origin O is smaller than 1:

$$\partial b' \cap \partial b'' = 1 \text{ or } 2 \text{ points} \quad \Longleftrightarrow \quad l \cap B \ne \emptyset \quad \Longleftrightarrow \quad \text{dist}(l,O) \le 1\ .$$

$$\dim(b' \cap b'') = 0 \quad \Longleftrightarrow \quad l \cap B = 1 \text{ point} \quad \Longleftrightarrow \quad \text{dist}(l,O) = 1\ .$$

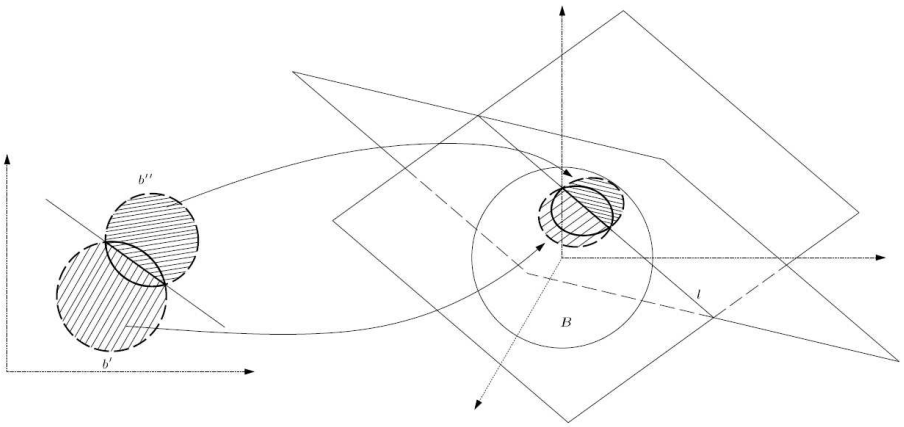


Figure 2.22: The intersection between the boundaries of two disks b', b'' in $\Re^2$ corresponds to a line l intersecting the sphere B.

Similarly we can consider three distinct disks b', b'', and b'''. If their intersection is a region bounded by three circular arcs, one from each disk, then the three boundary circles correspond to three planes $\partial h'$, $\partial h''$, and $\partial h'''$ that intersect in a point p

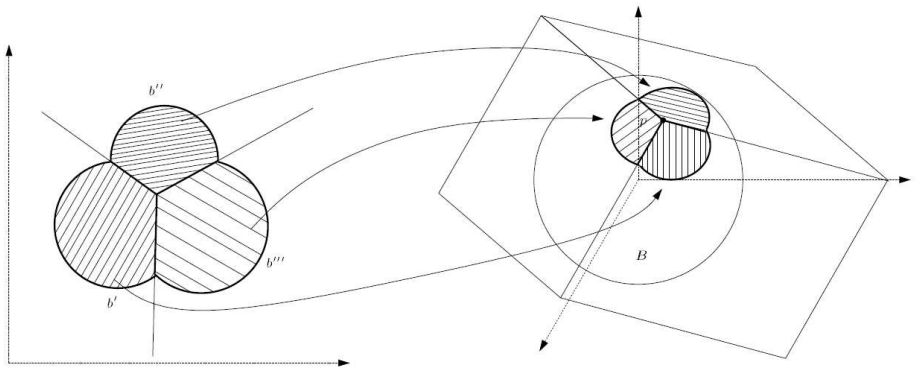


Figure 2.23: The non-empty intersection, when bounded by three circular arcs, between three disks b', b'', and b''' in $\Re^2$, corresponds to a point p contained in the ball B.

contained in B. This is illustrated in Figure 2.23. If the three circular boundaries intersect in one or two points, then the planes intersect in a point on ∂B (or possibly in a line that intersects B).

$$b' \cap b'' \cap b''' = \text{ region bounded by 3 arcs (or points)} \quad \Longleftrightarrow \quad p \in B \tag{4.14}$$

$$\dim\,(\partial b' \cap \partial b'' \cap \partial b''') = 0 \quad \Longleftrightarrow \quad p \in \partial B\ . \tag{4.15}$$

Proof of equation (4.14).

Let the three circles be $(x-x_i)^2+(y-y_i)^2 = r_i^2$, $i = 1,2,3$. Then the three corresponding planes are $(1+x_i^2+y_i^2-r_i^2)-2x_i\xi_1-2y_i\xi_2+(1-x_i^2-y_i^2+r_i^2)\xi_3 = 0$. Their point of intersection, if unique and finite, is given by $(\xi_1,\xi_2,\xi_3) = (D_1/D_4, D_2/D_4, D_3/D_4)$, where

$$D_1 = \begin{vmatrix} -1-x_1^2-y_1^2+r_1^2 & -2y_1 & 1-x_1^2-y_1^2+r_1^2 \\ -1-x_2^2-y_2^2+r_2^2 & -2y_2 & 1-x_2^2-y_2^2+r_2^2 \\ -1-x_3^2-y_3^2+r_3^2 & -2y_3 & 1-x_3^2-y_3^2+r_3^2 \end{vmatrix}\ ,$$

$$D_2 = \begin{vmatrix} -2x_1 & -1-x_1^2-y_1^2+r_1^2 & 1-x_1^2-y_1^2+r_1^2 \\ -2x_2 & -1-x_2^2-y_2^2+r_2^2 & 1-x_2^2-y_2^2+r_2^2 \\ -2x_3 & -1-x_3^2-y_3^2+r_3^2 & 1-x_3^2-y_3^2+r_3^2 \end{vmatrix}\ ,$$

$$D_3 = \begin{vmatrix} -2x_1 & -2y_1 & -1-x_1^2-y_1^2+r_1^2 \\ -2x_2 & -2y_2 & -1-x_2^2-y_2^2+r_2^2 \\ -2x_3 & -2y_3 & -1-x_3^2-y_3^2+r_3^2 \end{vmatrix}\ ,\ D_4 = \begin{vmatrix} -2x_1 & -2y_1 & 1-x_1^2-y_1^2+r_1^2 \\ -2x_2 & -2y_2 & 1-x_2^2-y_2^2+r_2^2 \\ -2x_3 & -2y_3 & 1-x_3^2-y_3^2+r_3^2 \end{vmatrix}\ .$$

The condition that this point of intersection lies in the interior of B is

$$D_1^2+D_2^2+D_3^2-D_4^2 < 0\ . \tag{4.16}$$

If $D_4 = 0$, then the point of intersection is at infinity, and the inequality (4.16) cannot be satisfied. (If $D_1 = D_2 = D_3 = D_4 = 0$, then the three planes have a line in common which intersects B, and it can be shown that the centers of the three circles are collinear and the circles intersect in two points.)

The intersection of three disks is bounded by three circular arcs exactly when each disk contains exactly one of the two points of intersection of the other two circles. In order for the first two circles to intersect in two points, we need that the distance between their centers is strictly between $|r_1-r+2|$ and r_1+r_2. This can be expressed algebraically as

$$A_1 = [(x_1-x_2)^2+(y_1-y_2)^2-(r_1-r_2)^2][(x_1-x_2)^2+(y_1-y_2)^2-(r_1+r_2)^2] < 0\ . \tag{4.17}$$

Next, we need that r_3 is between the distance from (x_3, y_3) to the two points of intersection of the first two circles. This condition turns out to be expressible as

$$\frac{\begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{vmatrix}^2 A_1 + A_2^2}{[(x_1-x_2)^2+(y_1-y_2)^2]^2} < 0 \tag{4.18}$$

where

$$\begin{aligned} A_2 \;=\; & [(x_2-x_1)(x_2-x_3)+(y_2-y_1)(y_2-y_3)]r_1^2 \\ & +[(x_2-x_1)(x_3-x_1)+(y_2-y_1)(y_3-y_1)]r_2^2 \\ & +[(x_2-x_1)^2+(y_2-y_1)^2][(x_3-x_1)(x_3-x_2)+(y_3-y_1)(y_3-y_2)-r_3^2] \ . \end{aligned}$$

Remarkably,

$$D_1^2+D_2^2+D_3^2-D_4^2 = \begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{vmatrix}^2 A_1 + A_2^2 \ .$$

Therefore, if the intersection of the three disks is bounded by three circular arcs ((4.17) and (4.18) hold), then the intersection point of the three planes is within B ((4.16) holds). If the intersection point of the three planes is a point within B ((4.16) holds), then (4.18) holds. Since (4.18) holds, we must have $A_1 < 0$, so that (4.17) holds as well, and then the three circles intersect pairwise in two points, and each disk contains exactly one of the two points of intersection of the other two circles.

Convex Hulls and Boolean Combination of Balls

Consider the intersection of n balls or their complements, such as $b_1 \cap \bar{b}_2 \cap \bar{b}_3 \cap \cdots \cap b_n$. We can map each of the b_i or $\bar{b}_i$ to a halfspace h in $\Re^{d+1}$ so that the computation of the intersection is reduced to a convex hull computation. Note that if all the balls are complemented we get the complement of the union of balls as in [35]. In general, for the computation of the topological structure of a non-linear, non-convex, possibly disconnected region in $\Re^d$, the intersection of inequalities of the type (4.13) is reduced to the computation of the boundary of the convex polytope CP, intersection of halfspaces (4.12), and intersecting this boundary with the unit sphere (4.9).

This mapping generalizes the "lifting" scheme [33] so that it can represent both the interior and the exterior of balls and so that one can compute any boolean combination of balls instead of just their union. In the present formulation we also represent the balls by their implicit inequality (4.12) instead of just a center and a radius, so that one can deal with infinite radius spheres (note that such cases arise in practice in the computation of trimming curves).

An additional advantage of the present mapping with respect to the "lifting" scheme is the compact representation of several collections of curve arrangements in the special case of the collection of trimming circles of patches that form a molecular surface. In fact, in this case we need only to observe that the convex polytope CP, that is dual to each arrangement of trimming curves of each patch, is indeed the cell of that patch in the 3-dimensional Power Diagram. This implies we need not represent a separate polytope for each arrangement of trimming curves since the 3-dimensional Power Diagram contains them all. The advantage in storage comes from representing only once any lower-dimensional face shared by more than one polytope. This sharing of faces also provides savings in storage of explicit adjacency information for each boundary curve of each patch.

2.4.2 Maintaining the Molecular Surface Under Quadratic Growth

We call quadratic growth the scheme of growing balls which keeps the Power Diagram unaltered and thus the topology of the union of balls is given by the corresponding α-shape. Under this growth of the balls we only need to maintain the set of trimming curves of each patch in the surface. In particular we need to efficiently detect any topological change (new intersections between curves, creation/deletion of connected components) that occur in the trimming curves (circles and lines) in the domain plane.

This goal can be achieved by looking at each patch separately (actually the computation can be performed in parallel for all patches) and classifying the faces of its associated polytope CP with respect to the relative ball B at the current size. This is achieved by using the relations stated in Section 2.4.1 as follows:

- Each facet of CP that intersects ∂B corresponds to a circle that is effectively involved in the set of trimming curves.

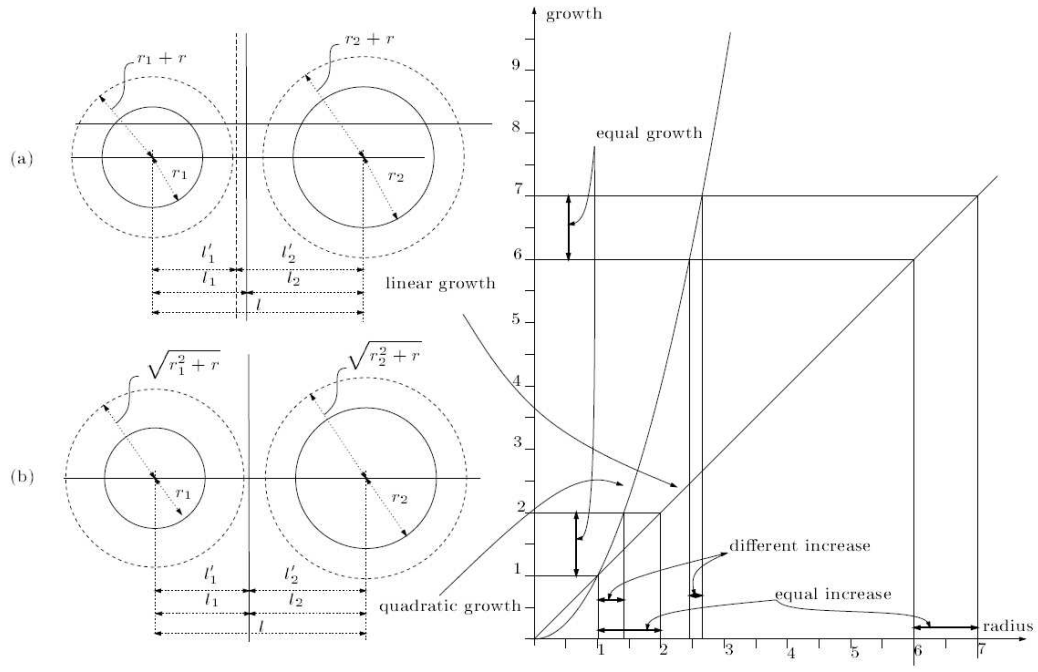


Figure 2.24: (a) If the radii of two balls are incremented by the same amount, then their Voronoi separator moves towards the smaller one. (b) If the squares of the radii of two balls are incremented by the same amount, then their Voronoi separator remains the same.

- Each edge of CP that intersects ∂B corresponds to two circles that intersect each other.

This leads to the following algorithm for maintaining trimming curves. For each face f of CP we determine its minimum distance d_m and its maximum distance d_M from the origin (the center of B). This tells us when the circle associated with f is involved in the boundary of the trimming circles. We organize the ranges of all the faces in an Interval tree so that we can efficiently perform range queries, optimal in space and time. While growing the ball B we look at the faces of CP which range $[d_m, d_M]$ contains the current radius r of B to directly determine the topology of the trimming circles. For example, if the range of a facet of CP contains f but none of its boundary edges implies that an entire circle forms a separate component in the boundary of the trimming curves.

At the same time this tells us that in the growing process the values of d_m, d_M of the faces of CP constitute the set of "event points" at which the growth of r produces some topological change in the trimming circles. Hence we can efficiently maintain the dynamic arrangement of circles in the plane.

The topological structure of the molecule is given by the Regular Triangulation and its dual, the Power Diagram. We examine the family of triangulations that yield the topological structure of the molecular surfaces (solvent contact or solvent excluded surfaces) while the solvent radius grows.

The determination of the topological structure of such molecular surfaces is an important problem addressed by several papers [30]. The family of shapes obtained from a weighted α-shape [37, 35] is based on a quadratic growth of the radii of the balls and therefore not directly related to the family based on the growth of the solvent ball radius. In fact the fundamental property on which the α-shape construction is based on is that for any α, the Power Diagram/Regular Triangulation remains the same. This is achieved by growing each sphere by a different amount, namely the radius of each sphere is augmented by a quantity such that the square of each radius is increased by the same quantity (see Figure 2.24). This implies that smaller spheres are grown more than the larger ones. As a consequence the resulting surface does not reflect exactly the required molecular surface (see Figure 2.25). When this level of approximation (possibly incorrect both in geometry and in topology) is not satisfactory, one needs to resort to the method introduced in the following section.

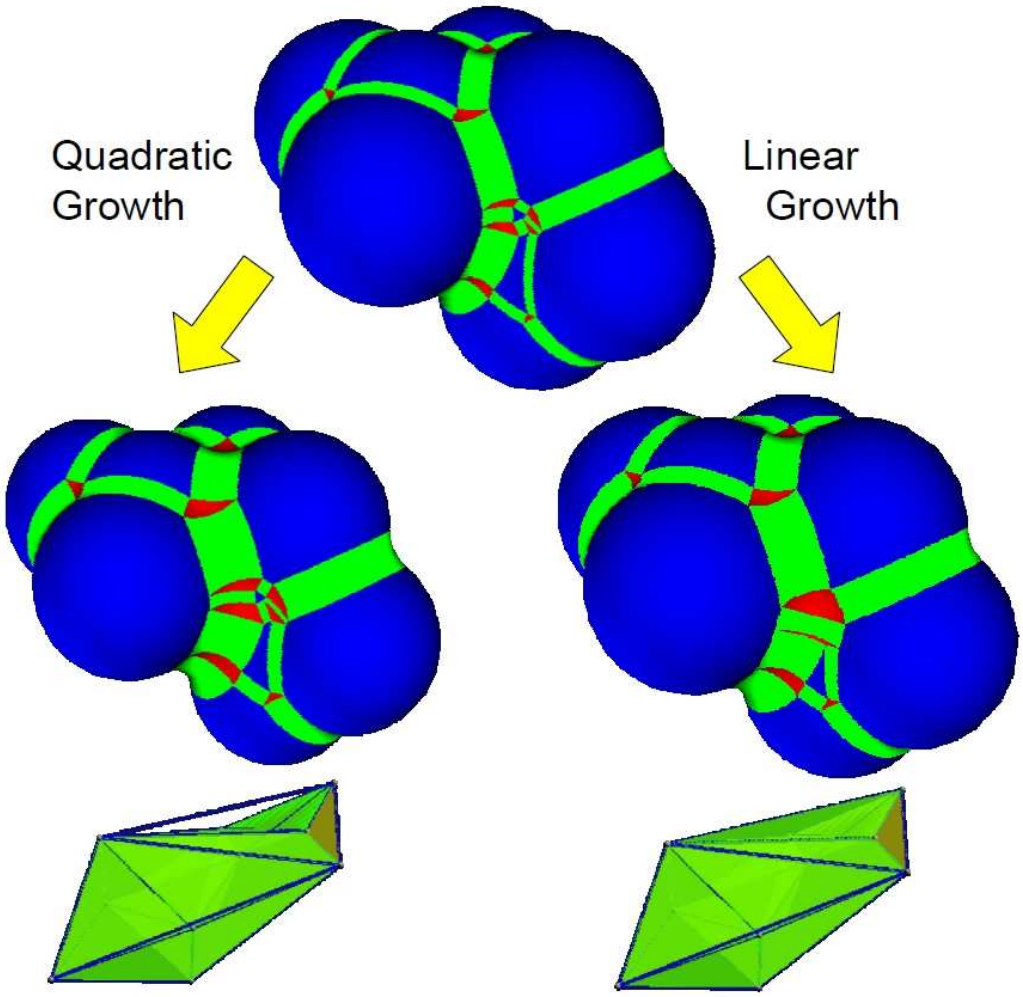


Figure 2.25: The difference between a quadratic and a linear growth of the molecule for a given probe radius. The molecular surface (top) is grown quadratically (middle left), hence maintaining the topology of the set of patches, giving an approximation to the real molecular surface computed by linear growth (middle right). The topology differences can be seen in the weighted zero alpha shapes (bottom) from a different view point.

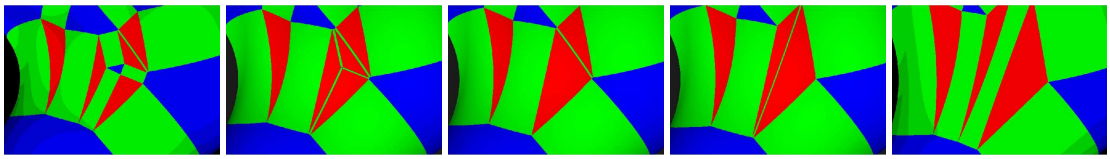


Figure 2.26: Examples of several topological changes in the set of NURBS patches, while growing the probe radius linearly.

2.4.3 Maintaining the Molecular Surface under Linear Growth

The fundamental dynamic setting we consider is the case of a global linear growth of all the atoms of a molecule, corresponding to a linear growth of the solvent atom radius r. In this case the Voronoi Diagram (or more exactly Power Diagram) plane that separates the two balls moves as a function of r resulting in topological changes of the triangulations and in the set of NURBS patches defining the molecular surface (see Figure 2.26). In fact, as the radius of each ball is increased by r, the Voronoi plane that separates the two balls moves towards the smaller ball. For example in Figure 2.24, the distances l_1, l_2 of the Voronoi plane π from the centers of the two balls must be such that the power distances of π are equal, that is:

$$l_1^2 - r_1^2 = l_2^2 - r_2^2$$

Moreover, the distance between the two balls is constant (the two balls grow but do not move):

$$l_1 + l_2 = l$$

>From these two equations we obtain for l_1:

$$l_1^2 - r_1^2 = (l - l_1)^2 - r_2^2 = l^2 + l_1^2 - 2l_1 l - r_2^2$$

$$l_1 = \frac{l^2 + r_1^2 - r_2^2}{2l}$$

When r_1 changes to $r_1 + r$ and r_2 changes to $r_2 + r$ we have:

$$\begin{aligned} l_1 &= \frac{l^2 + (r_1 + r)^2 - (r_2 + r)^2}{2l} \\ &= \frac{l^2 + r_1^2 + r^2 + 2r_1 r - r_2^2 - r^2 - 2r_2 r}{2l} \\ &= \frac{l^2 + r_1^2 - r_2^2 + 2r(r_1 - r_2)}{2l} \end{aligned}$$

In general, consider two balls B_1, B_2 (of radii r_1 and r_2 respectively) in $\Re^d$ and assume, without loss of generality, a coordinate system with the origin in the center of B_1 and the center of B_2 on the positive part of the x_1 axis (the center of B_2 is the point $(l, 0, \ldots, 0)$). The hyperplane of the Power Diagram that separates B_1 from B_2 has the equation:

$$\pi : \qquad x_1 = \frac{l^2 + r_1^2 - r_2^2}{2l} + r\frac{2(r_1 - r_2)}{2l} \tag{4.19}$$

which is linear in r. Hence this is also a hyperplane in the $(d+1)$-dimensional space $(x_1, \ldots, x_d, r)$. Figure 2.27 shows the 1-dimensional case of two balls (segments) that grow quadratically (a) or linearly (b). In the first case the hyperplane of the Power Diagram that separates B_1 from B_2 remains the same for all values of r. In the second case, the hyperplane of the Power Diagram that separates B_1 from B_2 moves linearly with r with a slope towards the center of B_1.

This fundamental observation leads to the construction of the Power Diagram of a set of growing balls as the intersection of a hyperplane $r = const$ with a complex $\mathscr{C}$ of convex polytopes in the $(d+1)$-dimensional space $(x_1, \ldots, x_d, r)$. If the molecule $\mathscr{B}$ is composed of n balls $\{B_1, \ldots, B_n\}$ then the complex $\mathscr{C}$ is a collection of n convex polytopes $\{C_1, \ldots, C_n\}$ one per ball. In particular the cell C_i associated with the ball B_i is the intersection of all the halfspaces of points "nearer" to B_i then to B_j (with $j = 1, \ldots, i-1, i+1, \ldots, n$). The boundary hyperplane of such halfspaces is given by equation (4.19). Note that cell C_i is defined as the intersection of all possible $n+1$ halfspaces since by linear growing many flips can occur in the Regular Triangulation. A flip occurs when an edge connecting two opposite vertices of a quadrilateral comprising two triangles in the triangulation is replaced by the edge connecting the other two vertices, as illustrated in Figure 2.28. The brute force application of the technique as described here requires the computation of n convex hulls [22] in four-dimensional space, which leads to an $O(n^3)$ time worst case complexity. For our purposes this is just a preprocessing step needed to construct the data structure used for animating the molecular surface, so we do not report in the present paper the details of an efficient computation of this complex $\mathscr{C}$. Note however, that in the case of a molecule in three dimensions $(d = 3)$ we have to compute a set of 4-dimensional convex hulls that can be computed more efficiently, in an output sensitive sense, by using the algorithm given in [21]. The use of this algorithm would indeed be beneficial because the overall number of faces in $\mathscr{C}$ is indeed $O(n^2)$. This is proved by a

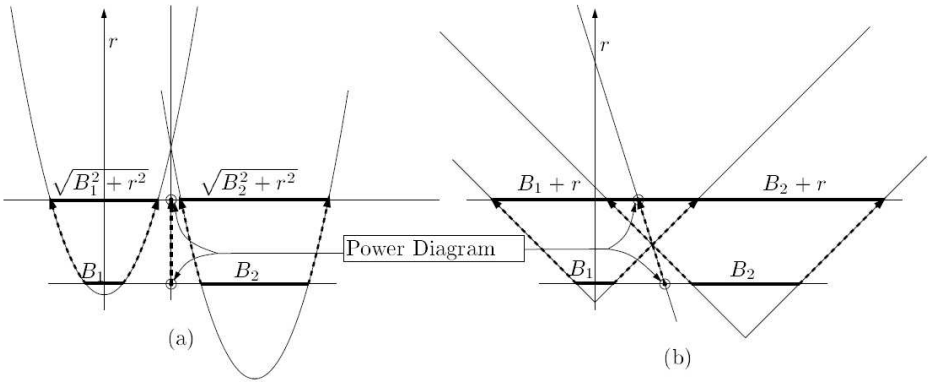


Figure 2.27: The 1-dimensional case of ball growth. The quadratic growth (a) keeps the Power Diagram hyperplane (a point) still. The linear growth (b) moves the Power Diagram hyperplane linearly with r.

technique introduced in [18] that generalizes the "lifting" scheme for the computation of Power Diagrams [33] and maps the construction of the complex $\mathscr{C}$ to a convex hull computation (intersection of halfspaces) in one dimension higher (that is in dimension $d+2$). In the case of a molecule in three dimensions, this leads to the computation of the convex hull in dimension five that can be computed optimally [22] in $O(n^2)$ time. This is certainly optimal in odd dimension (and in particular in the case of molecules where $d=3$) since a single Power Diagram (and $\mathscr{C}$ contains many of them) already has the same number of faces as a $(d+1)$-dimensional convex polytope.

In the previous section we introduced the construction of a complex of convex polytopes $\mathscr{C}$ embedded in the $(d+1)$-dimensional space $(x_1,\ldots,x_d,r)$ whose "horizontal" slices (that is an intersection with the hyperplane $r=const$) are the Power Diagrams of the balls $\mathscr{B}$ with radii uniformly increased by r. This data structure allows us to animate (update) efficiently the representation of a molecular surface (solvent accessible or solvent contact) with respect to a change in the solvent radius.

In particular we can achieve simple and efficient updates on the Power diagram localized in regions where the topological changes actually occur. In this way we can then in turn directly apply the method described in Section 2.4.2.

Being that the Power Diagram is the intersection of a horizontal hyperplane $H: r=const$ with the complex $\mathscr{C}$, in the dynamic setting the linear growth of the radii is simply a sweep of such horizontal hyperplanes H along the r-axis. Hence the "event points" at which we have to update the topological structure of the Power Diagram are the vertices of $\mathscr{C}$. In particular to compute these hyperplane sections of $\mathscr{C}$ we apply the robust approach in [17] which is based on the robust "above or "below" classification of the vertices of $\mathscr{C}$ with respect to H. We sort the vertices of $\mathscr{C}$ by their r coordinates so that their classification is obtained in logarithmic time by locating the current height value of H in such a sorted list of vertices. This approach is also suitable for the dynamic growth setting in which we will be continuously moving the hyperplane H. In fact in such a scenario, each time we cross a vertex of $\mathscr{C}$, we will need to update only the cells incident to this vertex. Moreover in general, if we suddenly change our solvent radius from a value r_1 to a value r_2, we will be able to detect the vertices whose r coordinate is in the range $[r_1,r_2]$, change their above/below classification and consequently update all the incident faces of $H\cap\mathscr{C}$.

We reach the conclusion that when spheres grow linearly, some flips can occur in the Regular Triangulation, unlike the quadratic growth, so that the usual α-shape construction is invalid (see Figure 2.28).

2.4.4 Examples

Example 1

Here we choose a coordinate system so that two of the balls have centers on the ξ_1-axis in $\Re^3$. Specifically, consider three balls B', B'', and B'''. Choose a coordinate system so that their centers are located at $(0,0,0)$, $(l_{12},0,0)$, and $(l_{13}\cos\beta, l_{13}\sin\beta, 0)$, where l_{12} and l_{13} are the distances between the centers of B' and B'', and between the centers of B' and B''', respectively, and β is the angle made by the three centers, with B' at the vertex. We can assume $0<\beta<\pi$. Let the solvent ball have radius r.

We consider the two planes π_1,π_2 relative to two trimming curves c_1,c_2. The position of the line $l=\pi_1\cap\pi_2$ of intersection is used to track the intersection between c_1 and c_2 and to give their 2D NURBS representation.

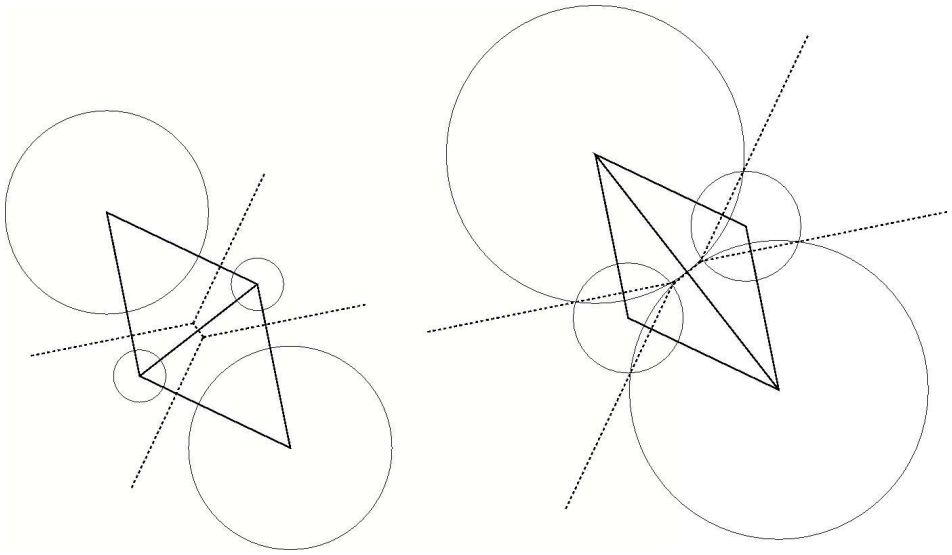


Figure 2.28: A simple case of a Regular Triangulation for which the topology changes in a simple linear growth of the radius of the balls.

With the above coordinate system, the two planes have equations (see Figure 2.29):

$$\pi_1 \quad : \quad \xi_1 = a_1 + r a_2 \tag{4.20}$$

$$\pi_2 \quad : \quad (\cos\beta)\xi_1 + (\sin\beta)\xi_2 = a_3 + r a_4 \tag{4.21}$$

where

$$a_1 = \frac{l_{12}^2 + r_1^2 - r_2^2}{2 l_{12}} \qquad a_2 = \frac{r_1 - r_2}{l_{12}}$$

$$a_3 = \frac{l_{13}^2 + r_1^2 - r_3^2}{2 l_{13}} \qquad a_4 = \frac{r_1 - r_3}{l_{13}} \ .$$

in accordance with (4.19).

The image of the trimming curve is the intersection of the spherical surfaces of the balls $B'(r)$ and $B''(r)$, which we define as the balls of radii $r_1 + r$ and $r_2 + r$ centered at $(0,0,0)$ and $(l_{12},0,0)$, respectively. The implicit equation of the spherical surface of $B'(r)$ is then $\xi_1^2 + \xi_2^2 + \xi_3^2 = (r_1 + r)^2$, and one finds that the ξ_3 coordinate of the two points of intersection between this sphere and the line l is

$$\xi_3 \quad = \quad \pm\sqrt{(r_1 + r)^2 - \xi_1^2 - \xi_2^2} \tag{4.22}$$

The segment of the line $l = \pi_1 \cap \pi_2$ within $B'(r)$ then has the parametrization:

$$\begin{aligned} \xi_1 &= a_1 + r a_2 \\ \xi_2 &= a_5 + r a_6 \\ \xi_3 &= \sqrt{(r_1 + r)^2 - (a_1 + r a_2)^2 - (a_5 + r a_6)^2}\ u \ , \end{aligned} \tag{4.23}$$

$$-1 \le u \le 1 \ ,$$

where

$$a_5 = \frac{a_3 - a_1 \cos\beta}{\sin\beta} \qquad a_6 = \frac{a_4 - a_2 \cos\beta}{\sin\beta} \ .$$

For brevity, these quantities which will appear frequently in the sequel will be named as follows. Keep in mind that all of these b_i are functions of r.

$$\begin{aligned} b_1 &= r_1 + r \\ b_2 &= a_1 + r a_2 \\ b_3 &= a_3 + r a_4 \\ b_4 &= a_5 + r a_6 \\ b_5 &= \sqrt{b_1^2 - b_2^2 - b_4^2} \\ b_6 &= \sqrt{b_1^2 - b_2^2} \\ b_7 &= \sqrt{b_2^2 + b_4^2} \end{aligned}$$

To map the surface of the ball $B'(r)$ to a plane, we use an inverse mapping similar to (4.11) but for a sphere of radius $r_1 + r$ instead of 1 and specifically $d = 2$:

$$\begin{aligned} x_1 &= \frac{\xi_1}{r_1 + r - \xi_3} \\ x_2 &= \frac{\xi_2}{r_1 + r - \xi_3} \ . \end{aligned}$$

>From this one obtains the intersection points q_1 and q_7 (see Figure 2.29(b); these points lie on a line through the origin) in the $(x_1(r), x_2(r))$ parameter space as

$$\begin{aligned} q_1 &= \left(\frac{b_2}{b_1 + b_5}, \frac{b_4}{b_1 + b_5} \right) \\ q_7 &= \left(\frac{b_2}{b_1 - b_5}, \frac{b_4}{b_1 - b_5} \right) \end{aligned}$$

and the trimming curve is an arc of the circle with center

$$q_0 = \left(\frac{b_1}{b_2}, 0 \right)$$

and radius

$$\frac{b_6}{b_2} \ .$$

One next needs to find suitable break points q_3 and q_5 (see Figure 2.29). Ideally we want none of the arcs $q_1\widehat{q}_3$, $q_3\widehat{q}_5$, $q_5\widehat{q}_7$ to be close to 180°. We can make sure that none of these arcs exceeds 120° as follows. Let q_8 be the midpoint of segment $\overline{q_1 q_7}$, and let q_9 be the intersection of the perpendicular bisector of $\overline{q_1 q_7}$ with the arc $q_3\widehat{q}_5$. Now choose q_3 and q_5 to be on the line perpendicular to $\overleftrightarrow{q_8 q_9}$ that intersects the $\overleftrightarrow{q_8 q_9}$ at a point 3/4 of the way from q_8 towards q_9. In the limiting case when q_1 and q_7 coincide, which occurs when $(r_1 + r)^2 = (a_1 + r a_2)^2 + (a_5 + r a_6)^2$, the three arcs $q_1\widehat{q}_3$, $q_3\widehat{q}_5$, $q_5\widehat{q}_7$ are all 120°, and they all shrink as the arc $q_1 - q_3 - q_5 - q_7$ shrinks.

In the x_1x_2-plane, line $\overleftrightarrow{q_1 q_7}$ has the equation $(a_5 + r a_6)x - (a_1 + r a_2)y = 0$. We also have

$$q_8 = \left(\frac{b_1 b_2}{b_7^2}, \frac{b_1 b_4}{b_7^2} \right)$$

and

$$q_9 = \left(\frac{b_1}{b_2} + \frac{b_4 b_6}{b_2 b_7}, -\frac{b_6}{b_7} \right) \ .$$

>From this we get

$$\begin{aligned} q_3 &= \left(\frac{4b_1b_7^2 - b_1b_4^2 + 3b_4b_6b_7}{4b_2b_7^2} - \frac{\sqrt{6b_1^2b_4^2 + 7b_2^2b_5^2 + 6b_1b_4b_6b_7}}{4b_7^2}, \right. \\ &\qquad \left. \frac{b_1b_4 - 3b_6b_7}{4b_7^2} - \frac{b_4\sqrt{6b_1^2b_4^2 + 7b_2^2b_5^2 + 6b_1b_4b_6b_7}}{4b_2b_7^2} \right) \\ q_5 &= \left(\frac{4b_1b_7^2 - b_1b_4^2 + 3b_4b_6b_7}{4b_2b_7^2} + \frac{\sqrt{6b_1^2b_4^2 + 7b_2^2b_5^2 + 6b_1b_4b_6b_7}}{4b_7^2}, \right. \\ &\qquad \left. \frac{b_1b_4 - 3b_6b_7}{4b_7^2} + \frac{b_4\sqrt{6b_1^2b_4^2 + 7b_2^2b_5^2 + 6b_1b_4b_6b_7}}{4b_2b_7^2} \right) . \end{aligned}$$

We now determine q_2, q_4, and q_6 as the points of intersection of the tangents lines through q_1, q_3, q_5, and q_7. We get

$$q_4 = \left(\frac{-7b_1^2b_4^3 + 4b_1^2b_4b_7^2 - 12b_1b_6b_7^3 - 7b_2^2b_4b_5^2 - 9b_4b_6^2b_7^2}{4b_2b_7^2(b_1b_4 - 3b_6b_7)}, \frac{7b_1^2b_4^2 + 7b_2^2b + 5^2 + 9b_6^2b_7^2}{4b_2b_7^2(b_1b_4 - 3b_6b_7)} \right) .$$

Also

$$\begin{aligned} q_2 &= (1/d_1)\Big(b_2[7b_1^3b_2^2b_4^3 + 7b_1^3b_4^5 - 4b_1^3b_4^3b_7^2 + 7b_1^2b_2^2b_4^3b_5 + 4b_1^2b_2^2b_4b_5b_7^2 + 7b_1^2b_4^5b_5 \\ &\quad - 4b_1^2b_4^3b_5b_7^2 + 12b_1^2b_4^2b_6b_7^3 + 7b_1b_2^4b_4b_5^2 + 7b_1b_2^2b_4^3b_5^2 - 4b_1b_2^2b_4^3b_7^2 + 13b_1b_2^2b_4b_6^2b_7^2 \\ &\quad - 12b_1b_2^2b_5b_6b_7^3 + 9b_1b_4^3b_6^2b_7^2 + 12b_1b_4^2b_5b_6b_7^3 + 7b_2^4b_4b_5^3 + 7b_2^2b_4^3b_5^3 + 12b_2^2b_4^2b_6b_7^3 \\ &\quad + 9b_2^2b_4b_5b_6^2b_7^2 - 12b_2^2b_6^3b_7^3 + 9b_4^3b_5b_6^2b_7^2 \\ &\quad + (-4b_1^2b_2b_4b_7^2 - 8b_1b_2b_4b_5b_7^2 + 4b_2b_4^3b_7^2 - 4b_2b_4b_6^2b_7^2)c_1], \\ &\quad 7b_1b_2^2b_4^2b_5^4 + 7b_1b_2^4b_5^2b_6^2 + 7b_1b_2^2b_4^2b_5^2b_6^2 + 7b_1^2b_2^2b_4^2b_5^3 + 7b_2^2b_4^2b_5^3b_6^2 + 12b_1^2b_4b_6^3b_7^3 \\ &\quad + 9b_1b_4^2b_6^4b_7^2 + 7b_1^2b_2^2b_4^2b_5b_6^2 - 12b_2^2b_4b_6^3b_7^3 + 7b_1^3b_2^2b_4^2b_6^2 + 9b_1^2b_2^2b_5b_6^2b_7^2 - 12b_1b_2^2b_4b_5b_6b_7^3 \\ &\quad - 4b_1b_2^2b_4^4b_7^2 + 7b_1^4b_2^2b_4^2b_5 - 4b_1^4b_4^2b_5b_7^2 + 4b_1^2b_2^2b_4^2b_5b_7^2 + 9b_1b_2^2b_6^4b_7^2 + 7b_1^2b_4^4b_5b_6^2 \\ &\quad - 4b_1^3b_4^2b_6^2b_7^2 + 4b_1b_2^2b_4^2b_6^2b_7^2 - 4b_1^3b_4^2b_5^2b_7^2 + 9b_1b_2^2b_5^2b_6^2b_7^2 + 12b_1^2b_4b_5^2b_6b_7^3 + 9b_1b_4^2b_5^2b_6^2b_7^2 \\ &\quad + 12b_1^3b_4b_5b_6b_7^3 + 7b_1^3b_2^2b_4^2b_5^2 + 5b_1^2b_4^2b_5b_6^2b_7^2 + 9b_2^2b_5b_6^4b_7^2 + 12b_1b_4b_5b_6^3b_7^3 + 9b_4^2b_5b_6^4b_7^2 \\ &\quad + 7b_1^3b_4^4b_6^2 + 7b_1^4b_4^4b_5 + 7b_1^3b_4^4b_5^2 + 7b_1b_2^4b_5^4 + 7b_2^4b_5^3b_6^2 + 7b_1^2b_2^4b_5^3 + 12b_2^2b_4^3b_6b_7^3 \\ &\quad + (-4b_1^3b_2b_5b_7^2 - 4b_1^2b_2b_5^2b_7^2 - 4b_1^2b_2b_6^2b_7^2 + 4b_1b_2^3b_5b_7^2 - 4b_1b_2b_5b_6^2b_7^2 - 4b_2^3b_4^2b_7^2 + 4b_2^3b_6^2b_7^2)c_1\Big) \end{aligned}$$

and

$$\begin{aligned} q_6 &= \frac{1}{d_2}\Big(b_2[4b_1^2b_4^3b_5b_7^2 - 9b_2^2b_4b_5b_6^2b_7^2 - 12b_1b_4^2b_5b_6b_7^3 - 9b_4^3b_5b_6^2b_7^2 + 7b_1^3b_4^5 + 12b_1^2b_4^2b_6b_7^3 \\ &\quad + 9b_1b_4^3b_6^2b_7^2 - 7b_1^2b_2^2b_4^3b_5 + 12b_2^2b_4^2b_6b_7^3 - 12b_2^2b_6^3b_7^3 + 12b_1b_2^2b_5b_6b_7^3 - 4b_1^2b_2^2b_4b_5b_7^2 \\ &\quad + 13b_1b_2^2b_4b_6^2b_7^2 + 7b_1^3b_2^2b_4^3 - 7b_1^2b_4^5b_5 - 4b_1^3b_4^3b_7^2 - 4b_1b_2^2b_4^3b_7^2 - 7b_2^4b_4b_5^3 \\ &\quad - 7b_2^2b_4^3b_5^3 + 7b_1b_2^2b_4^3b_5^2 + 7b_1b_2^4b_4b_5^2 \\ &\quad + (4b_1^2b_2b_4b_7^2 + 4b_2b_4b_6^2b_7^2 - 8b_1b_2b_4b_5b_7^2 - 4b_2b_4^3b_7^2)c_1], \\ &\quad - 7b_1b_2^2b_4^2b_5^4 - 7b_1b_2^4b_5^2b_6^2 - 7b_1b_2^2b_4^2b_5^2b_6^2 + 7b_1^2b_2^2b_4^2b_5^3 + 7b_2^2b_4^2b_5^3b_6^2 - 12b_1^2b_4b_6^3b_7^3 \\ &\quad - 9b_1b_4^2b_6^4b_7^2 + 7b_1^2b_2^2b_4^2b_5b_6^2 + 12b_2^2b_4b_6^3b_7^3 - 7b_1^3b_2^2b_4^2b_6^2 + 9b_1^2b_2^2b_5b_6^2b_7^2 - 12b_1b_2^2b_4b_5b_6b_7^3 \\ &\quad + 4b_1b_2^2b_4^4b_7^2 + 7b_1^4b_2^2b_4^2b_5 - 4b_1^4b_2^2b_4b_5b_7^2 + 4b_1^2b_2^2b_4^2b_5b_7^2 - 9b_1b_2^2b_6^4b_7^2 + 7b_1^2b_4^4b_5b_6^2 \\ &\quad + 4b_1^3b_2^2b_4^2b_6^2b_7^2 - 4b_1b_2^2b_4^2b_6^2b_7^2 + 4b_1^3b_2^2b_4^2b_5^2b_7^2 - 9b_1b_2^2b_5^2b_6^2b_7^2 - 12b_1^2b_4b_5^2b_6b_7^3 - 9b_1b_2^2b_4^2b_5^2b_6^2b_7^2 \\ &\quad + 12b_1^3b_4b_5b_6b_7^3 - 7b_1^3b_2^2b_4^2b_5^2 + 5b_1^2b_4^2b_5b_6^2b_7^2 + 9b_2^2b_5b_6^4b_7^2 + 12b_1b_4b_5b_6^3b_7^3 + 9b_4^2b_5b_6^4b_7^2 \\ &\quad - 7b_1^3b_4^4b_6^2 + 7b_1^4b_4^4b_5 - 7b_1^3b_4^4b_5^2 - 7b_1b_2^4b_5^4 + 7b_2^4b_5^3b_6^2 + 7b_1^2b_2^4b_5^3 - 12b_2^2b_4^3b_6b_7^3 \\ &\quad + (4b_2^3b_6^2b_7^2 - 4b_1^2b_2b_5^2b_7^2 + 4b_1b_2b_5b_6^2b_7^2 + 4b_1^3b_2b_5b_7^2 - 4b_1^2b_2b_6^2b_7^2 - 4b_1b_2^3b_5b_7^2 - 4b_2^3b_4^2b_7^2)c_1\Big) \end{aligned}$$

where

$$\begin{aligned} c_1 &= \sqrt{6b_1^2b_4^2+7b_2^2b_5^2+6b_1b_4b_6b_7} \\ d_1 &= b_7^2(b_1+b_5)[(-b_1b_4b_5-b_2^2b_4-b_4b_6^2)c_1 \\ &\quad +b_1^2b_2b_4b_5-b_1b_2b_4^3+b_1b_2b_4b_6^2-3b_1b_2b_5b_6b_7+3b_2b_4^2b_6b_7-3b_2b_6^3b_7] \\ d_2 &= b_7^2(b_1-b_5)[(b_1b_4b_5-b_2^2b_4-b_4b_6^2)c_1 \\ &\quad +b_1^2b_2b_4b_5+b_1b_2b_4^3-b_1b_2b_4b_6^2-3b_1b_2b_5b_6b_7-3b_2b_4^2b_6b_7+3b_2b_6^3b_7] \end{aligned}$$

We now need rational parametrizations of the circular arcs. The parametrization for arc $q_1-q_2-q_3$ is provided by

$$(x_1,x_2)=\frac{(1-t)^2q_3+2t(1-t)w_1q_2+t^2q_1}{(1-t)^2+2t(1-t)w_1+t^2}\ ,\ 0\le t\le 1\ ,$$

for a particular value for the weight w_1, which turns out to be the cosine of half the angle $\angle q_1q_0q_3$, or $\cos q_1q_0q_2$. This can be computed as

$$w_1=\frac{(q_1-q_0)\cdot(q_2-q_0)}{\|q_1-q_0\|\|q_2-q_0\|}\ .$$

Analogous parametrizations hold for arcs $q_3-q_4-q_5$ and $q_5-q_6-q_7$.

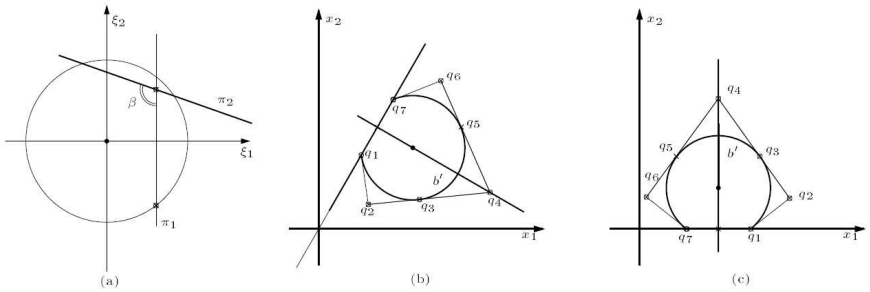


Figure 2.29: (a) (ξ_1,ξ_2) section of the (ξ_1,ξ_2,ξ_3) space. The circle is a cross section of ball $B'(r)$ of radius r_1+r. Line l, which is parallel to the ξ_3 axis, is the intersection of the planes π_1 and π_2, which in turn are the Voronoi planes separating $B'(r)$ and $B''(r)$ and separating $B'(r)$ and $B'''(r)$. (b) control points of the trimming curve that is part of the boundary of $b'(r)$ for Example 1. (c) the same control points in Example 2.

Example 2

Here we place the balls in $\Re^3$ so that the line through the endpoints of a trimming arc is parallel to the x_1-axis in x_1x_2-space. Consider three balls B', B'', and B'''. Choose a coordinate system so that their centers are located at $(0,0,0)$, $(l_{12}\cos\alpha,-l_{12}\sin\alpha,0)$, and $(l_{13}\cos(\beta-\alpha),l_{13}\sin(\beta-\alpha),0)$ where l_{12} and l_{13} are the distances between the centers of B' and B'', and between the centers of B' and B''', respectively, β is the angle made by the three centers, with B' at the vertex ($0<\beta<\pi$), and

$$\alpha=\tan^{-1}\left[\frac{(a_3+ra_4)-(a_1+ra_2)\cos\beta}{(a_1+ra_2)\sin\beta}\right]\ .$$

With this definition we have that α is the angle between the ray through the centers of B' and B'', and the ξ_1-axis, and

$$\begin{aligned} \cos\alpha &= \frac{b_2\sin\beta}{(b_2^2-2b_2b_3\cos\beta+b_3^2)^{1/2}} \\ \sin\alpha &= \frac{b_3-b_2\cos\beta}{(b_2^2-2b_2b_3\cos\beta+b_3^2)^{1/2}}\ . \end{aligned}$$

Note that α is a function of r. This coordinate system is chosen so that the Voronoi planes defined below intersect at the ξ_1-axis.

Let the solvent ball have radius r. We consider the two planes π_1, π_2 relative to two trimming curves c_1, c_2. The position of the line $l = \pi_1 \cap \pi_2$ of intersection is used to track the intersection between c_1 and c_2 and to give their 2D NURBS representation.

With the above coordinate system, the two planes have equations:

$$\pi_1 \quad : \quad (\cos\alpha)\xi_1 - (\sin\alpha)\xi_2 = a_1 + ra_2 \tag{4.24}$$

$$\pi_2 \quad : \quad [\cos(\beta-\alpha)]\xi_1 + [\sin(\beta-\alpha)]\xi_2 = a_3 + ra_4 \tag{4.25}$$

where the a_i are the same as in Example 1:

$$a_1 = \frac{l_{12}^2 + r_1^2 - r_2^2}{2l_{12}} \qquad a_2 = \frac{r_1 - r_2}{l_{12}}$$
$$a_3 = \frac{l_{13}^2 + r_1^2 - r_3^2}{2l_{13}} \qquad a_4 = \frac{r_1 - r_3}{l_{13}} \ ,$$

in accordance with (4.19).

The image of the trimming curve is the intersection of the spherical surfaces of the balls $B'(r)$ and $B''(r)$, which we define as the balls of radii $r_1 + r$ and $r_2 + r$ centered at $(0,0,0)$ and $(l_{12}\cos\alpha, -l_{12}\sin\alpha, 0)$, respectively. The implicit equation of the spherical surface of $B'(r)$ is then $\xi_1^2 + \xi_2^2 + \xi_3^2 = (r_1 + r)^2$, and one finds that the ξ_3 coordinate of the two points of intersection between this sphere and the line l is

$$\xi_3 \quad = \quad \pm\sqrt{(r_1+r)^2 - \xi_1^2 - \xi_2^2} \tag{4.26}$$

The segment of the line $l = \pi_1 \cap \pi_2$ within $B'(r)$ then has the parametrization:

$$\begin{aligned} \xi_1 &= (a_1 + ra_2)/\cos\alpha \\ \xi_2 &= 0 \\ \xi_3 &= \sqrt{(r_1+r)^2 - (a_1+ra_2)^2/\cos^2\alpha}\ u \ , \end{aligned} \tag{4.27}$$

$$-1 \le u \le 1 \ .$$

To map the surface of the ball $B'(r)$ to a plane, we use an inverse mapping similar to (4.11) but for a sphere of radius $r_1 + r$ instead of 1 and specifically $d = 2$:

$$\begin{aligned} x_1 &= \frac{\xi_1}{r_1 + r - \xi_3} \\ x_2 &= \frac{\xi_2}{r_1 + r - \xi_3} \ . \end{aligned}$$

>From this one obtains the intersection points q_1 and q_7 (see Figure 2.29(c)) in the $(x_1(r), x_2(r))$ parameter space as

$$q_1 = \left(\frac{b_1\cos\alpha - \sqrt{b_1^2\cos^2\alpha - b_2^2}}{b_2}, 0 \right)$$
$$q_7 = \left(\frac{b_1\cos\alpha + \sqrt{b_1^2\cos^2\alpha - b_2^2}}{b_2}, 0 \right)$$

and the trimming curve is an arc of the circle with center

$$q_0 = \left(\frac{b_1\cos\alpha}{b_2}, -\frac{b_1\sin\alpha}{b_2} \right)$$

and radius

$$\frac{b_6}{b_2} \ .$$

In the x_1x_2-plane, line $\overleftrightarrow{q_1q_7}$ is just the x_1-axis, and line $\overleftrightarrow{q_8q_9}$ is $x_1 = b_1\cos\alpha/b_2$. We also have

$$q_8 = \left(\frac{b_1\cos\alpha}{b_2}, 0\right)$$

and

$$q_9 = \left(\frac{b_1\cos\alpha}{b_2}, -\frac{b_1\sin\alpha + b_6}{b_2}\right) \ .$$

>From this we find that the break points q_3 and q_5 are

$$\begin{aligned} q_3 &= \left(\frac{b_1\cos\alpha}{b_2} - \frac{(7b_6^2 - b_1^2\sin^2\alpha + 6b_1b_6\sin\alpha)^{1/2}}{4b_2}, -\frac{3}{4}\frac{b_1\sin\alpha + b_6}{b_2}\right) \\ q_5 &= \left(\frac{b_1\cos\alpha}{b_2} + \frac{(7b_6^2 - b_1^2\sin^2\alpha + 6b_1b_6\sin\alpha)^{1/2}}{4b_2}, -\frac{3}{4}\frac{b_1\sin\alpha + b_6}{b_2}\right) \ . \end{aligned}$$

We now determine q_2, q_4, and q_6 as the points of intersection of the tangents lines through q_1, q_3, q_5, and q_7. We get

$$q_4 = \left(\frac{b_1\cos\alpha}{b_2}, -\frac{b_1\sin\alpha}{b_2} + \frac{4b_6^2}{b_2(b_1\sin\alpha - 3b_6)}\right) \ .$$

Also

$$q_2 = \left(\frac{b_1\cos\alpha}{b_2} - \frac{3b_6^2(b_1\sin\alpha + b_6)}{b_2[b_1(c_2 - c_1)\sin\alpha + 3b_6c_1]}, -\frac{b_1\sin\alpha}{b_2} + \frac{b_6^2(c_2 - 4c_1)}{b_2[b_1(c_2 - c_1)\sin\alpha + 3b_6c_1]}\right)$$

and

$$q_6 = \left(\frac{b_1\cos\alpha}{b_2} + \frac{3b_6^2(b_1\sin\alpha + b_6)}{b_2[b_1(c_2 - c_1)\sin\alpha + 3b_6c_1]}, -\frac{b_1\sin\alpha}{b_2} + \frac{b_6^2(c_2 - 4c_1)}{b_2[b_1(c_2 - c_1)\sin\alpha + 3b_6c_1]}\right)$$

where

$$\begin{aligned} c_1 &= \sqrt{b_1^2\cos^2\alpha - b_2^2} \\ c_2 &= \sqrt{7b_6^2 - b_1^2\sin^2\alpha + 6b_1b_6\sin\alpha} \ . \end{aligned}$$

2.5 Maintaining Union of Balls Under Atom Movements

We describe the *packing grid data structure* [7, 8] for maintaining a set M of balls in 3-space efficiently under the following set of queries and updates. By $B = (c, r)$ we denote a ball withcenter c and radius r.

Queries.

- INTERSECT(c, r): Return all balls in M that intersect the given ball $B = (c, r)$. The given ball may or may not belong to the set M.
- RANGE(p, δ): Return all balls in M with centers within distance δ of point p. We assume that δ is at most a constant multiple of the radius of the largest ball in M.
- EXPOSED(c, r): Returns *true* if the ball $B = (c, r)$ contributes to the outer boundary of the union of the balls in M. The given ball must belong to M.

	TIME COMPLEXITY	
OPERATIONS	ASSUMING $t_q = \mathcal{O}(\log\log w)$, $t_u = \mathcal{O}(\log w)$	ASSUMING $t_q = \mathcal{O}(\log\log n)$, $t_u = \mathcal{O}\left(\frac{\log n}{\log\log n}\right)$
RANGE(p, δ) | INTERSECT(c, r) | EXPOSED(c, r) ($\delta = \mathcal{O}(r_{max})$)	$\mathcal{O}(\log\log w)$ (w.h.p.)	$\mathcal{O}(\log\log n)$ (w.h.p.)
SURFACE( )	$\mathcal{O}$(#balls on surface) (worst-case)	
ADD(c, r) | REMOVE(c, r) | MOVE(c_1, c_2, r)	$\mathcal{O}(\log w)$ (w.h.p.)	$\mathcal{O}\left(\frac{\log n}{\log\log n}\right)$ (w.h.p.)
ASSUMPTIONS: (i) RAM with w-bit Words, (ii) Collection of n Balls, and (iii) $r_{max} = \mathcal{O}$(minimum distance between two balls)		

Table 2.2: Time complexities of the operations supported by the packing grid data structure.

- SURFACE( ): Returns the outer boundary of the union of the balls in M. If there are multiple disjoint outer boundary surfaces defined by M, the routine returns any one of them.

Updates.

- ADD(c, r): Add a new ball $B = (c,r)$ to the set M.
- REMOVE(c, r): Remove the ball $B = (c,r)$ from M.
- MOVE(c_1, c_2, r): Move the ball with center c_1 and radius r to a new center c_2.

We assume that at all times during the lifetime of the data structure the following holds.

Assumption 2.5.1. *If r_{max} is the radius of the largest ball in M, and d_{min} is the minimum Euclidean distance between the centers of any two balls in M, then $r_{max} = \mathcal{O}(d_{min})$.*

In general, a ball in a collection of n balls in 3-space can intersect $\Theta(n)$ other balls in the worst case, and it has been shown in [25] that the boundary defined by the union of these balls has a worst-case combinatorial complexity of $\Theta(n^2)$. However, if M is a "union of balls" representation of the atoms in a molecule, then assumption 2.5.1 holds naturally [49, 76], and as proved in [49], in that case, both complexities improve by a factor of n. The following theorem states the consequences of the assumption.

Theorem 2.5.1. *(Theorem 2.1 in [49], slightly modified) Let $M = \{B_1, \ldots, B_n\}$ be a collection of n balls in 3-space with radii $r_1, \ldots, r_n$ and centers at $c_1, \ldots, c_n$. Let $r_{max} = \max_i\{r_i\}$ and let $d_{min} = \min_{i,j}\{d(c_i,c_j)\}$, where $d(c_i,c_j)$ is the Euclidean distance between c_i and c_j. Also let $\delta M = \{\delta B_1, \ldots, \delta B_n\}$ be the collection of spheres such that δB_i is the boundary surface of B_i. If $r_{max} = \mathcal{O}(d_{min})$ (i.e., Assumption 2.5.1 holds), then:*

(i) *Each $B_i \in M$ intersects at most $216 \cdot (r_{max}/d_{min})^3 = \mathcal{O}(1)$ other balls in M.*

(ii) *The maximum combinatorial complexity of the boundary of the union of the balls in M is $\mathcal{O}\left((r_{max}/d_{min})^3 \cdot n\right)$ $= \mathcal{O}(n)$.*

PROOF. Similar to the proof of Theorem 2.1 in [49]. □

Therefore, as Theorem 2.5.1 suggests, for intersection queries and boundary construction, one should be able to handle M more efficiently if assumption 2.5.1 holds. The efficiency of our data structure, too, partly depends on this assumption.

2.5.1 Preliminaries

Before we describe our data structure we present several definitions in order to simplify the exposition.

Definition 2.5.1 (r-grid and grid-cell)**.** *An* r-grid *is an axis-parallel infinite grid structure in 3-space consisting of cells of size* $r \times r \times r$ *(*$r \in \mathbb{R}$*) with the* root *(i.e., the corner with the smallest x, y, z coordinates) of one of the cells coinciding with origin of the (Cartesian) coordinate axes. The grid cell that has its root at Cartesian coordinates* (ar, br, cr) *(where* $a, b, c \in \mathbb{Z}$*) is referred to as the* (a,b,c,r)*-cell or simply as the* (a,b,c)*-cell when* r *is clear from the context.*

Definition 2.5.2 (grid-line)**.** *The* (b,c,r)-line *(where* $b, c \in \mathbb{Z}$*) in an* r*-grid consists of all* (x,y,z,r)*-cells with* y *and* z *fixed to* b *and* c*, respectively. When* r *is clear from the context the* (b,c,r)*-line will simply be called the* (b,c)*-line.*

Observe that each cell on the (b,c,r)-line can be identified with a unique integer, e.g., the cell at index $a \in \mathbb{Z}$ on the given line corresponds to the (a,b,c,r)-cell in the r-grid.

Definition 2.5.3 (grid-plane)**.** *The* (c,r)-plane *(where* $c \in \mathbb{Z}$*) in an* r*-grid consists of all* (x,y,z,r)*-cells with* z *fixed to* c*. The* (c,r)*-plane will be referred to as the* c*-plane when* r *is clear from the context.*

The (c,r)*-plane* can be decomposed into an infinite number of lines each identifiable with a unique integer. For example, index $b \in \mathbb{Z}$ uniquely identifies the (b,c,r)-line on the given plane. Also each grid-plane in the r-grid can be identified with a unique integer, e.g., the (c,r)-plane is identified by c. The proof of the following lemma is straight-forward.

Lemma 2.5.1. *Let* $M = \{B_1, \ldots, B_n\}$ *be a collection of* n *balls in 3-space with radii* $r_1, \ldots, r_n$ *and centers at* $c_1, \ldots, c_n$*. Let* $r_{max} = \max_i \{r_i\}$ *and let* $d_{min} = \min_{i,j} \{d(c_i, c_j)\}$*, where* $d(c_i, c_j)$ *is the Euclidean distance between* c_i *and* c_j*. Suppose* M *is stored in the* $2r_{max}$*-grid* G*. Then*

(*i*) *If* $r_{max} = \mathcal{O}(d_{min})$ *(i.e., Assumption 2.5.1 holds) then each grid-cell in* G *contains the centers of at most* $64 \cdot (r_{max}/d_{min})^3 = \mathcal{O}(1)$ *balls in* M*.*

(*ii*) *Each ball in* M *intersects at most 8 grid-cells in* G*.*

(*iii*) *For a given ball* $B \in M$ *with center in grid-cell* C*, the center of each ball intersecting* B *lies either in* C *or in one of the 26 grid-cells adjacent to* C*.*

(*iv*) *The number of* non-empty *(i.e., containing the center of at least one ball in* M*) grid-cells in* G *is at most* n*, and the same bound holds for grid-lines and grid-planes.*

At the heart of our data structure is a fully dynamic one dimensional integer range reporting data structure for word RAM described in [62]. The data structure in [62] maintains a set S of integers under updates (i.e., insertions and deletions), and answers queries of the form: report any or all points in S in a given interval. The following theorem summarizes the performance bounds of the data structure which are of interest to us.

Theorem 2.5.2. *(proved in [62]) On a RAM with* w*-bit words the fully dynamic one dimensional integer range reporting problem can be solved in linear space, and w.h.p. bounds of* $\mathcal{O}(t_u)$ *and* $\mathcal{O}(t_q + k)$ *on update time and query time, respectively, where* k *is the number of items reported, and*

(*i*) $t_u = \mathcal{O}(\log w)$ *and* $t_q = \mathcal{O}(\log\log w)$ *using the data structure in [62]; and*

(*ii*) $t_u = \mathcal{O}(\log n / \log\log n)$ *and* $t_q = \mathcal{O}(\log\log n)$ *using the data structure in [62] for small* w *and a fusion tree [42] for large* w*.*

The data structure can be augmented to store satellite information of size $\mathcal{O}(1)$ with each integer without degrading its asymptotic performance bounds. Therefore, it supports the following three functions:

1. INSERT(i, s): Insert an integer i with satellite information s.
2. DELETE(i): Delete integer i from the data structure.
3. QUERY(l, h): Return the set of all $\langle\, i,\ s \,\rangle$ tuples with $i \in [l,h]$ stored in the data structure.

2.5.2 Description (Layout) of the Packing Grid Data Structure

We are now in a position to present our data structure. Let DPG be the data structure. We represent the entire 3-space as a $2r_{max}$-grid (see Definition 2.5.1), and maintain the non-empty grid-planes (see Definition 2.5.3), grid-lines (see Definition 2.5.2) and grid-cells (see Definition 2.5.1) in DPG. A grid component (i.e., cell, line or plane) is non-empty if it contains the center of at least one ball in M. The data structure can be described hierarchically. It has a tree structure with 5 levels: 4 internal levels (levels 3, 2, 1 and 0) and an external level of leaves (see Figure 2.30). The description of each level follows.

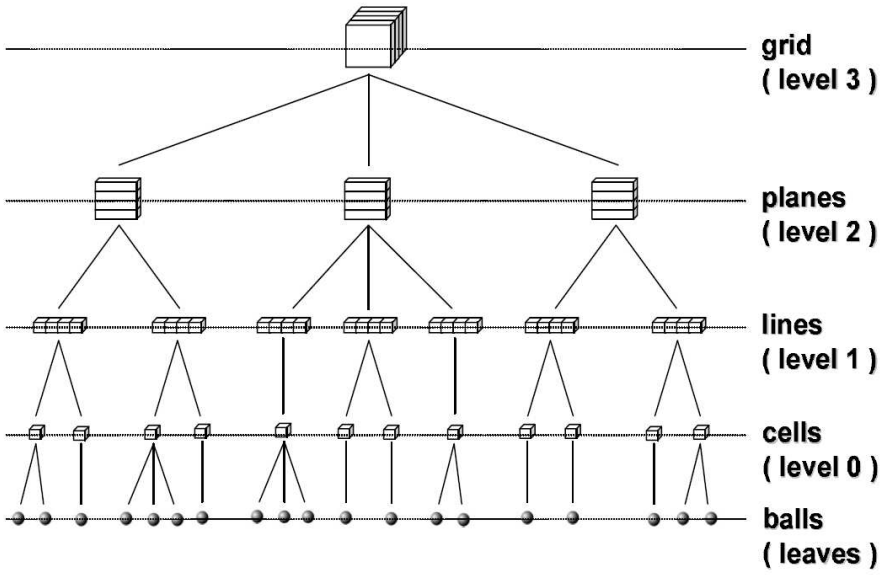


Figure 2.30: Hierarchical structure of DPG

The Leaf Level "Ball" Data Structure (DPG_{-1}). The data structure stores the center $c = (c_x, c_y, c_z)$ and the radius r of the given ball B. It also includes a Boolean flag *exposed* which is set to *true* if B contributes to the outer boundary of the union of the balls in M, and *false* otherwise. If another ball B' intersects B, it does so on a circle which divides the boundary δB of B into two parts: one part is buried inside B' and hence cannot contribute to the union boundary, and the other part is exposed w.r.t. B' and hence might appear on the union boundary. The circular intersections of all balls intersecting B define a 2D arrangement A on δB which according to Theorem 2.5.1 has $\mathcal{O}(1)$ combinatorial complexity. A face of A is exposed, i.e., contributes to the union boundary, provided it is not buried inside any other ball. Observe that if at least one other ball intersects B, and A has an exposed face f, then each edge of f separates f from another exposed face f' which belongs to the arrangement A' of a ball intersecting B. We store all exposed faces (if any) of A in a set F of size $\mathcal{O}(1)$, and with each face f we store pointers to the data structures of $\mathcal{O}(1)$ other balls that share edges with f and also the identifier of the corresponding face on each ball. Observe that if B does not intersect any other balls then F will contain only a single face and no pointers to any other balls.

The Level 0 "Grid-Cell" Data Structure (DPG_0). The "grid-cell" data structure stores the root (see Definition 2.5.1) (a,b,c) of the grid-cell it corresponds to. A grid-cell can contain the centers of at most $\mathcal{O}(1)$ balls in M (see Lemma 2.5.1). Pointers to data structures of all such balls are stored in a set S of size $\mathcal{O}(1)$. Since we create "grid-cell" data structures only for non-empty grid-cells, there will be at most n (and possibly $\ll n$) such data structures, where n is the current number of balls in M.

The Level 1 "Grid-Line" Data Structure (DPG_1). We create a "grid-line" data structure for a (b,c)-line provided it contains at least one non-empty grid-cell. The data structure stores the values of b and c. Each (a,b,c)-cell lying on this line is identified with the unique integer a, and the identifier of each such non-empty grid-cell is stored in an integer range search data structure RR as described in Section 2.5.1 (see Theorem 2.5.2). We augment RR to store the pointer to the corresponding "grid-cell" data structure with each identifier it stores. The total number of "grid-line" data structure created is upper bounded by n and possibly much less than n.

The Level 2 "Grid-Plane" Data Structure (DPG_2). A "grid-plane" data structure is created for a c-plane provided it contains at least one non-empty grid-line. Similar to the "grid-line" data structure it identifies each non-empty (b,c)-line lying on the c-plane with the unique integer b, and stores the identifiers in a range reporting data structure RR described in Section 2.5.1. A pointer to the corresponding "grid-line" data structure is also stored with each identifier. The data structure also stores c. The total number of "grid-plane" data structures created cannot exceed n, and will possibly be much less than n.

The Level 3 "Grid" Data Structure (DPG_3). This data structure maintains the non-empty grid-planes of the $2r_{max}$-grid in an integer range reporting data structure RR (see Section 2.5.1). Each c-plane is identified by the unique integer c, and each such integer stored in RR is also accompanied by a pointer to the corresponding "grid-plane" data structure. The "grid" data structure also stores a *surface-root* pointer which points to the "Ball" data structure of an arbitrary exposed ball in M.

We have the following lemma on the space usage of the data structure.

Lemma 2.5.2. *Let M be a collection of n balls as defined in Theorem 2.5.1, and let Assumption 2.5.1 holds. Then the packing grid data structure storing M uses $\mathscr{O}(n)$ space.*

PROOF. The space usage of the data structure is dominated by the space used by the range reporting data structures, the grid-cells and the "ball" data structures. Since the range reporting data structures use linear space (see Theorem 2.5.2) and total number of non-empty grid components (i.e., planes, lines and cells) is $\mathscr{O}(n)$ (see Lemma 2.5.1), total space used by all such data structures is $\mathscr{O}(n)$. The grid cells contain pointers to "ball" data structures, and since no two grid-cells point to the same "ball" data structure, total space used by all grid-cells is also $\mathscr{O}(n)$. Each "ball" data structure contains the arrangement A and the face decomposition F of the exposed (if any) faces of the ball. The total space needed to store all such arrangements and decompositions is $\mathscr{O}\left((r_{max}/d_{min})^3 \cdot n\right)$ (see Theorem 2.5.1) which reduces to $\mathscr{O}(n)$ under Assumption 2.5.1. Thus the total space used by the data structure is $\mathscr{O}(n)$. □

2.5.3 Queries and Updates

The queries and updates supported by the data structure are implemented as follows.

Queries.

(1) RANGE$(\ p,\ \delta\ )$: Let $p = (p_x, p_y, p_z)$. We perform the following steps.

i. **Level 3 Range Query:** We invoke the function QUERY$(\ l,\ h\ )$ of the range reporting data structure RR under DPG_3 (i.e., the level 3 "grid" data structure) with $l = \lfloor (p_z - \delta)/(2r_{max}) \rfloor$ and $h = \lfloor (p_z + \delta)/(2r_{max}) \rfloor$. This query returns a set S_2 of tuples, where each tuple $\langle\ c,\ P_c\ \rangle \in S_2$ refers to a non-empty c-plane with a pointer P_c to its level 2 "grid-plane" data structure.

ii. **Level 2 Range Query:** For each $\langle\ c,\ P_c\ \rangle \in S_2$, we call the range query function under the corresponding level 2 data structure with $l = \left\lfloor (p_y - \delta')/(2r_{max}) \right\rfloor$ and $h = \left\lfloor (p_y + \delta')/(2r_{max}) \right\rfloor$, where $(\delta')^2 = \delta^2 - (c - p_z)^2$ if $c - p_z < \delta$, and $\delta' = r_{max}$ otherwise. This query returns a set $S_{1,c}$ of triples, where each triple $\langle\ b,\ c,\ P_{b,c}\ \rangle \in S_{1,c}$ refers to a non-empty $(\ b,\ c\ )$-line with a pointer $P_{b,c}$ to its level 1 "grid-line" data structure. We obtain the set S_1 by merging all $S_{1,c}$ sets.

iii. **Level 1 Range Query:** For each $\langle\ b,\ c,\ P_{b,c}\ \rangle \in S_1$, we call the integer range query function under the corresponding level 1 "grid-line" data structure with $l = \lfloor (p_x - \delta'')/(2r_{max}) \rfloor$ and $h = \lfloor (p_x + \delta'')/(2r_{max}) \rfloor$, where $(\delta'')^2 = \delta^2 - (b - p_y)^2 - (c - p_z)^2$ if $\delta^2 > (b - p_y)^2 + (c - p_z)^2$, and $\delta'' = r_{max}$ otherwise. This query returns a set $S_{0,b,c}$ of quadruples, where each quadruples $\langle\ a,\ b,\ c,\ P_{a,b,c}\ \rangle \in S_{0,b,c}$ refers to a non-empty $(\ a,\ b,\ c\ )$-cell with a pointer $P_{a,b,c}$ to its level 0 "grid-cell" data structure. We obtain the set S_0 by merging all $S_{0,b,c}$ sets.

iv. **Ball Collection:** For each $\langle\ a,\ b,\ c,\ P_{a,b,c}\ \rangle \in S_0$, we collect from the level 0 data structure of the corresponding $(\ a,\ b,\ c\ )$-cell each ball whose center lies within distance δ from p. We collect the pointer to the leaf level "ball" data structure of each such ball in a set S, and return this set.

The correctness of the function follows trivially since it queries a region in 3-space which includes the region covered by a ball of radius δ centered at p. It is straight-forward to see that the function makes at most $\mathscr{O}\left(\pi \cdot (\lceil \delta/r_{max} \rceil + 1)^2\right)$ calls to

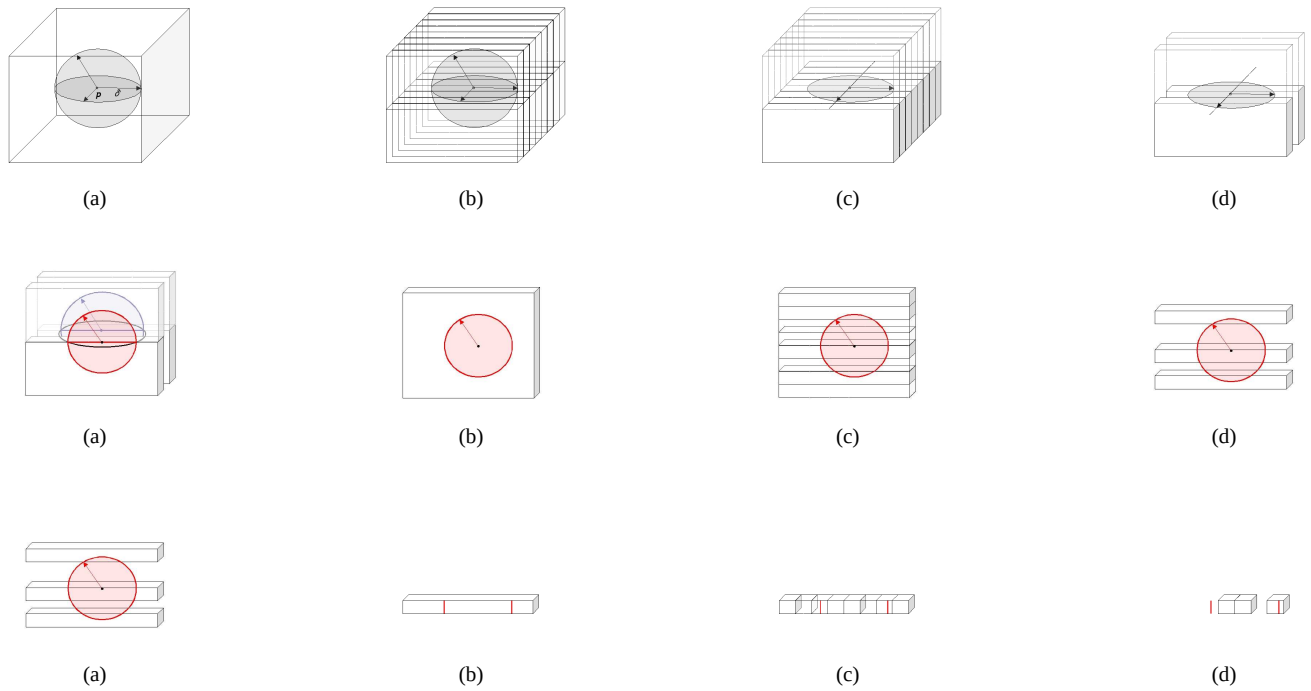


Figure 2.31: **Top row:** Level 3 Range Query. (a) Query region defined by the sphere of radius δ at point p inside the level 3 grid, (b) The level 3 grid as a collection of level 2 grid-planes, (c) and (d) Range reporting query returns the set of non-empty grid-planes within the query region. **Middle row:** Level 2 Range Query. (a) On each grid-plane, query region is defined by a circular slice of the sphere of radius δ at point p, (b), (c) and (d) Range reporting query on such a grid-plane returns the set of non-empty grid-lines within the query region. **Bottom row:** Level 1 Range Query. (a) and (b) Query region in each grid-line is defined as an interval, (c) and (d) For each grid-line, range reporting query returns the set of non-empty grid-cells

a range reporting data structure, and collects balls from at most $\mathcal{O}\left(\frac{4}{3}\pi\cdot(\lceil\delta/r_{max}\rceil+1)^3\right)$ grid-cells. Using Lemma 2.5.1 and Theorem 2.5.2, we conclude that w.h.p. the function terminates in $\mathcal{O}\left((\delta/r_{max})^2\cdot t_q+((\delta+r_{max})/d_{min})^3\right)$ time. Assuming $r_{max}=\mathcal{O}(d_{min})$ (i.e., Assumption 2.5.1) and $\delta=\mathcal{O}(r_{max})$, the complexity reduces to $\mathcal{O}(t_q)$ (w.h.p.).

(2) INTERSECT(c, r)**:** Let $B=(c,r)$ be the given ball. We perform the following two steps.

i. **Ball Collection:** We call RANGE(c, $r+r_{max}$) and collect the output in set S which contains pointers to the data structure of each ball in M with its center within distance $r+r_{max}$ from c.

ii. **Identifying Intersecting Balls:** From S we remove the data structure of each ball that does not intersect B, and return the resulting (possibly reduced) set.

We know from elementary geometry that two balls of radii r_1 and r_2 cannot intersect unless their centers lie within distance r_1+r_2 of each other. Therefore, step (i) correctly identifies all balls that can possibly intersect B, and step (ii) completes the identification. Step (i) takes
$\mathcal{O}\left(t_q+(r_{max}/d_{min})^3\right)$ time w.h.p., and step (ii) terminates in $\mathcal{O}\left((r_{max}/d_{min})^3\right)$ time in the worst case. Therefore, under Assumption 2.5.1 w.h.p. this function runs in $\mathcal{O}(t_q)$ time.

(3) EXPOSED(c, r)**:** Let $B=(c,r)$ be the given ball. We locate B's data structure by calling RANGE(c, 0), and return the value stored in its *exposed* field. Clearly, the function takes $\mathcal{O}\left(t_q+(r_{max}/d_{min})^3\right)$ time (w.h.p.) which reduces to $\mathcal{O}(t_q)$ (w.h.p.) under Assumption 2.5.1.

(4) SURFACE( ): The *surface-root* pointer under the level 3 "grid" data structure points to the "ball" data structure of a ball B on the union boundary of M. We scan the set F of exposed faces of B, and using the pointers to other exposed balls stored in F we perform a depth-first traversal of all exposed balls in M and return the exposed faces on each such ball. Let m be the number of balls contributing to the union boundary of M. Then according to Theorem 2.5.1 the depth-first search takes $\mathcal{O}\left((r_{max}/d_{min})^3 \cdot m\right)$ time in the worst case which reduces to $\mathcal{O}(m)$ under Assumption 2.5.1.

Updates.

(1) ADD(c, r): Let $c = (c_x, c_y, c_z)$ and let $c'_u = \left\lfloor \frac{c_u}{2r_{max}} \right\rfloor$, where $u \in \{x, y, z\}$. We perform the following steps.

i. If $M \neq \emptyset$, let G be the grid data structure, otherwise create and initialize G. Add input ball to M.

ii. Query the range reporting data structure $G.RR$ to locate the data structure P for the c'_z-plane. If P does not exist create and initialize P, and insert c'_z along with a pointer to P into $G.RR$.

iii. Query $P.RR$ and locate the data structure L for the (c'_y, c'_z)-line. If L does not exist then create and initialize L, and insert c'_y along with a pointer to L into $P.RR$.

iv. Locate the data structure C for the (c'_x, c'_y, c'_z)-cell by querying $L.RR$. Create and initialize C if it does not already exist, and insert c'_x and a pointer to C into $L.RR$.

v. Create and initialize a data structure B for the input ball and add it to the set $C.S$.

vi. Call INTERSECT(c, r) and find the set I of the "ball" data structures of all balls that intersect the input ball. Create the arrangement $B.A$ using the balls in I. The new ball may partly or fully bury some of the balls it intersects, and hence we need to update the arrangement $B'.A$, the set $B'.F$ and the flag $B'.exposed$ of each $B' \in I$. The set $B.F$ is created and $B.exposed$ is initialized using the information in the updated data structures in I. If the *surface-root* pointer was pointing to a ball in I that got completely buried by the new ball, we update it to point to B instead.

Observe that the introduction of a new ball may affect the surface exposure of only the balls it intersects (i.e., bury some/all of them partly or completely), and no other balls. Hence, the updates performed in step (vi) (in addition to those in earlier steps) are sufficient to maintain the correctness of the entire data structure. Steps (i) and (v) take $\mathcal{O}(1)$ time in the worst case, and w.h.p. each of steps (ii), (iii) and (iv) takes $\mathcal{O}(t_q + t_u)$ time. Finding the intersecting balls in step (vi) takes

$\mathcal{O}\left(t_q + (r_{max}/d_{min})^3\right)$ time w.h.p., according to Theorem 2.5.1 creating and updating the arrangements and faces will take $\mathcal{O}\left((r_{max}/d_{min})^3 \times (r_{max}/d_{min})^3\right) = \mathcal{O}\left((r_{max}/d_{min})^6\right)$ time (w.h.p.). Thus the ADD function terminates in

$\mathcal{O}\left(t_q + t_u + (r_{max}/d_{min})^6\right)$ time w.h.p., which reduces to $\mathcal{O}(t_u)$ (w.h.p.) assuming $r_{max} = \mathcal{O}(d_{min})$ (i.e., Assumption 2.5.1).

(2) REMOVE(c, r): This function is symmetric to the ADD function, and has exactly the same asymptotic time complexity. Hence, we do not describe it here.

(3) MOVE(c_1, c_2, r): This function is implemented in the obvious way by calling **REMOVE(c_1, r)** followed by **ADD(c_2, r)**. It has the same asymptotic complexity as the two functions above.

Therefore, we have the following theorem.

Theorem 2.5.3. *Let M be a collection of n balls in 3-space as defined in Theorem 2.5.1, and let Assumption 2.5.1 holds. Let t_q and t_u be as defined in Theorem 2.5.2. Then the packing grid data structure storing M on a word RAM:*

(i) *uses $\mathcal{O}(n)$ space;*

(ii) *supports updates (i.e., insertion/deletion/movement of a ball) in $\mathcal{O}(t_u)$ time w.h.p.;*

(iii) *reports all balls intersecting a given ball or within $\mathcal{O}(r_{max})$ distance from a given point in $\mathcal{O}(t_q)$ time w.h.p., where r_{max} is the radius of the largest ball in M; and*

(*iv*) *reports whether a given ball is exposed or buried in* $\mathcal{O}(t_q)$ *time w.h.p., and returns the entire outer union boundary of M in* $\mathcal{O}(m)$ *worst-case time, where m is the number of balls on the boundary.*

In Table 2.2 we list the time complexities of the operations supported by our data structure.

2.5.4 Molecular Surface Maintenace Using DPG

In this section, we briefly describe applications of the packing grid data structure for efficient maintenance of molecular surfaces.

Maintaining van der Waals Surface of Molecules

For dynamic maintenance of the van der Waals surface of a molecule we can use the packing grid data structure directly. Each atom is treated as a ball with a radius equal to the van der Waals radius of the atom (see [19] for a list of van der Waals radius of different atoms).

Maintaining Lee-Richards (SCS/SES) Surface

We can use the packing grid data structure for the efficient maintenance of the Lee-Richards surface of a molecule under insertion/deletion/movement of atoms. The performance bounds given in Table 2.2 remain unchanged. We maintain two packing grid data structures: DPG and DPG'. The DPG data structure keeps track of the patches on the Lee-Richards surface, and DPG' is used for detecting intersections among concave patches.

Before adding an atom to DPG, we increase its radius r_s, where r_s is the radius of the rolling solvent atom. The DPG data structure keeps track of all solvent exposed atoms, i.e., all atoms that contribute to the outer boundary of the union of these enlarged atoms. Theorem 2.5.1 implies that each atom in DPG contributes $\mathcal{O}(1)$ patches to the Lee-Richards surface, and the insertion/deletion/movement of an atom results in local changes of only $\mathcal{O}(1)$ patches. We can modify DPG to always keep track of where two or three of the solvent exposed atoms intersect, and once we know the atoms contributing to a patch we can easily compute the patch in $\mathcal{O}(1)$ time [10].

The Lee-Richards surface can self-intersect in two ways: (i) a toroidal patch can intersect itself, and (ii) two different concave patches may intersect [10]. The self-intersections of toroidal patches can be easily detected from DPG. In order to detect the intersections among concave patches, we maintain the centers of all current concave patches in DPG', and use the INTERSECT query to find the concave patch (if any) that intersects a given concave patch.

2.6 Clustering and Decimation of Molecular Surfaces

In this section we discuss a multiresolution representation scheme for molecular shapes using the object's skeletal structure (i.e. zero-shape). This scheme is coupled with error estimates that takes into account the actual boundary surface of the shape. The boundary representation is derived from the topological structure underlying the representation of the molecular body (see Figure 2.40). Specifically, we consider the following three different boundary surface models: the Solvent Accessible Surface (SAS), the Lee-Richards Solvent Contact Surface, and the molecular skin. These three surfaces all have an underlying topological structure based on the regular (weighted Delaunay) triangulation and power diagram of the input set of balls. In case of actual molecules the input is a set of atoms each represented as a ball with its van der Waals radius. The corresponding weighted-point representation is the center of the atom associated with a weight equal to the square of the van der Waals radius. See section 2.2.1 for details about this representation.

A particular kind of vertex clustering as is used as decimation primitive. The clustering replaces two balls (atoms) with one. The weight of the new ball is chosen in order to preserve some covering relation between the coarse and fine levels of resolution. This covering property is important to guarantee a conservative estimate of the location where the molecule lies and can be used in several application domains such as collision detection and ray casting. The Delaunay property is preserved after the clustering by applying a sequence of flips in the triangulation.

For fast traversal, the multiresolution data structure created using this ball clustering primitive is a Directed Acyclic Graph (DAG) of nodes, where each node represents a clustering operation and the edges denote dependencies between nodes. A cut in this graph is a collection of edges which intersect all paths from the root to the leaves once and only once. Any such cut represents a valid multi-resolution approximation of the model [29, 58]. A more adaptive and space-efficient model is a forest of binary trees storing the cluster ball as the parent of the two replaced balls. This model depends on run-time updates of the triangulation (flips), but supports a much larger space of possible triangulations due to the reduced number of dependencies. The hierarchy is built bottom up by a sequence of decimation stages until a maximum error tolerance is reached or there are no more balls to be removed.

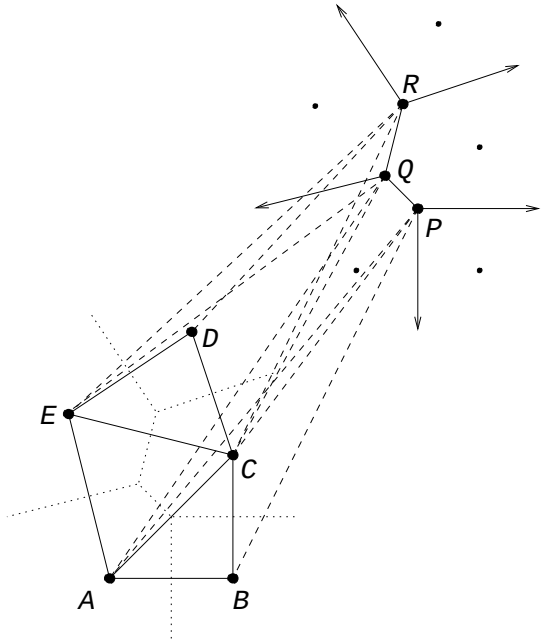


Figure 2.32: Scaffolding model. Points A, B, C, D, E, in the plane $z = 0$, are the atom centers, and are the vertices of a Delaunay triangulation. Points P, Q, R, in the plane $z = 1$, are the vertices of the corresponding Voronoi diagram.

Several error norms are used to evaluate the quality of any adaptive level of detail including support for the estimation of conservative bounds of the exact Hausdorff distance. While the decimation scheme, the hierarchical structure and the error estimates are defined in any dimension, we show practical results for a 2D implementation.

2.6.1 Preliminary- Mixed Cell Complex

The mixed cell complex consists of the weighted Delaunay triangulation, or regular triangulation (see the Related Math Section for definitions), at the lowest level, say the plane $z = 0$, and the corresponding weighted Voronoi diagram, or power diagram, at the highest level, say $z = 1$. Then each vertex of the power diagram is connected by line segments to the three vertices of the triangle to which it corresponds in the regular triangulation. (see Figure 2.32). Thus tetrahedra are formed by this construction; one example is tetrahedron *PABC*. Furthermore, two points connected by an edge in the power diagram are connected to two triangles that share an edge in the regular triangulation. Thus another set of tetrahedra is formed by the four endpoints of two such corresponding edges. For example, P and Q are connected to triangles $\triangle ABC$ and $\triangle ACE$, and the edges $\overline{PQ}$ and $\overline{AC}$ form tetrahedron *PQAC*.

For any value of z between 0 and 1 we can take a cross section of the structure defined above and obtain intermediate tessellation of the space into convex cells. In each non empty tile we have a portion of a quadratic surface (curve) that match in C^1 continuity with the patches defined in the neighboring tiles. The whole surface is called the *molecular skin* and is used as a representation for molecular boundaries.

At each level we can connect appropriate segments with A-splines or a molecular skin to represent the molecular surface at varying degrees of resolution. Figure 2.33 shows the power diagram and regular triangulation at $z = 1/3$ and $z = 2/3$, respectively.

2.6.2 Decimation of Molecular Shapes

In this subsection we focus on the problem of decimating molecular shapes. We consider the problem from the viewpoint of decimating the set of weighted points that induces the molecular shape (the centers of the atoms) rather then decimating some triangulation of its boundary. This approach has two main advantages: (i) one has to deal with a set of smaller cardinality because a high quality representation of the boundary would require a dense sampling with many points per atom, (ii) the same

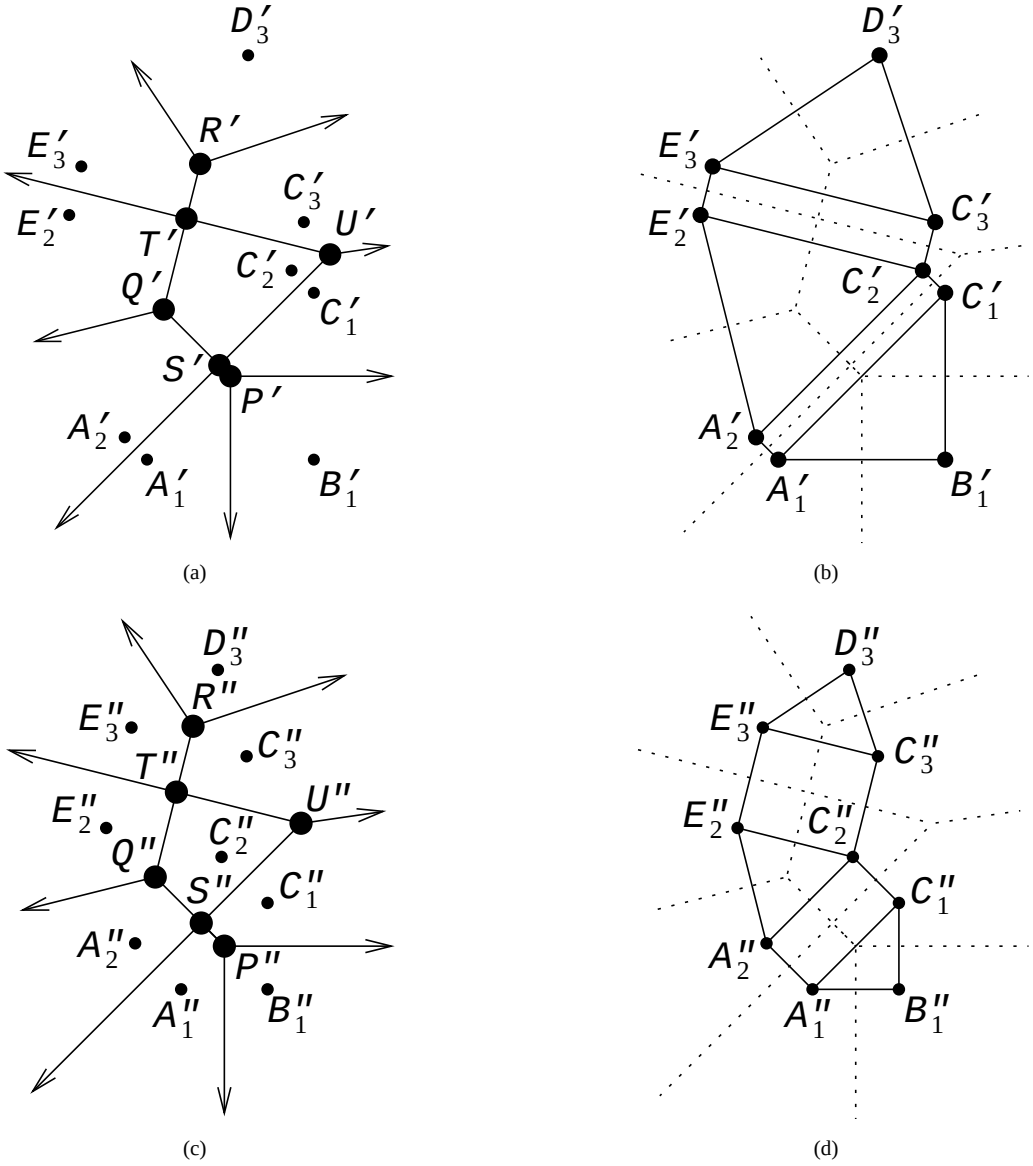


Figure 2.33: (a) Power diagram and (b) regular triangulation for the cross section of the mixed cell at $z = 1/3$, and (c) power diagram and (d) regular triangulation at $z = 2/3$.

multiresolution data structure induces a hierarchical representation for several types of molecular shapes (e.g. SAS, SCS or skin) instead of having a distinct multiresolution representation for each of them. Roughly speaking the goal of decimating a molecular shape is to produce a *coarse* but *simpler* representation of the original model that is too large for the available computational resources.

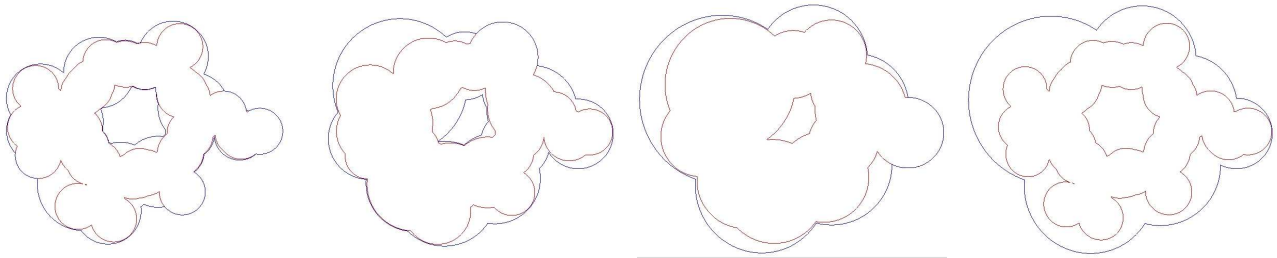
Definition 2.6.1. (Coarsening). *Given a model M (molecular shape) of cardinality k (number of balls) any* coarse representation M' *of M (written* $M' \succ M$*) is a model of cardinality* $k' < k$ *and such that:* $p \in M \Rightarrow p \in M'$.

We say that M' is a coarsening of M. For example Figure 2.34(a-d) shows three coarse representations M_1, M_2, M_3 of the molecular shape M in Figure 2.40. The four representations are in the following relationship: $M_3 \succ M_2 \succ M_1 \succ M$

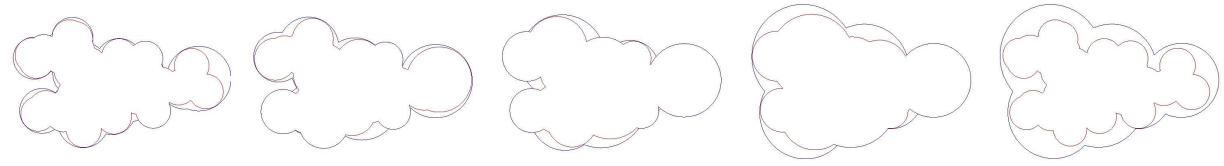
More generally it is easy to show that:

Property 1. *The relation "* $\succ$*" is transitive.*

The transitive property of "$\succ$" suggests a simple and efficient way to build a multiresolution representation of a molecular model by successive application of local coarsening primitives.



(a) $M_1 \succ M$ (b) $M_2 \succ M_1$ (c) $M_3 \succ M_2$ (d) $M_3 \succ M$



(e) $M_1 \succ M$ (f) $M_2 \succ M_1$ (g) $M_3 \succ M_2$ (h) $M_4 \succ M_3$ (i) $M_4 \succ M$

Figure 2.34: **Top:** Coarsening relationships between three representations M_1, M_2, M_3 of the molecular shape M in Figure 2.40. **Bottom:** Coarsening relationships between four representations M_1, M_2, M_3, M_4 of a second molecular shape M.

Vertex Clustering

General decimation schemes like edge-contraction do not preserve the Delaunay property which is the basis for all our molecular models. The known schemes like [28] for decimation that guarantee the Delaunay property while building a multiresolution hierarchy also do not seem appropriate in our case. This is because we do not use the triangulation as a direct shape representation of the molecular body. The triangulation is instead used to describe the skeletal structure of the molecule. Hence in the decimation process we have to take into account more than the modifications that occur in the triangulation itself the modifications that are induced to the corresponding molecular shape (union of balls, SAS, SCS, ...).

Let $p_u = (u, w_u), p_v = (v, w_v)$ be two weighted vertices. $e = (u, v)$ is a part of the zero shape if: $\|u - v\|^2 - w_u - w_v < 0$ (see [34] for the complete condition). This means that the two balls representing p_u and p_v overlap, and therefore they are good candidates for clustering. In order not to be dependent on the radius of the atoms, we actually use the Euclidean distance between the vertices to represent priority for clustering. Hence, we are looking on all atoms which overlap and cluster the closest ones first.

Clustering is done by removing the two vertices from the triangulation and inserting a new one. We choose the weight and position of the new vertex such that its ball will enclose the two replaced vertices balls. The fact is, that if we choose a ball which is ε greater than the enclosing ball, and correct to preserve a regular triangulation, then the two old vertices will be redundant and will not need to be removed[2]. We use two variation of the Vertex Clustering primitive depending on the tradeoff between efficiency and accuracy.

Definition 2.6.2. (Coarse Representation Ball). *Let $M = \{b_i\}_{i=0}^{k}$ be the set of balls representing the original atoms of a given molecular shape, and let $M' \subset M$. Then b is a* coarse representation *of M' if for each $b_j \in M$', $b \succ b_j$.*

LVC (Local Vertex Clustering) is the simpler and more efficient version of vertex clustering (see Figure 2.35(a)):

Definition 2.6.3. (LVC). *Let b_1, b_2 be be a coarse representation of M_1, M_2, $(M_i \subset M)$. The Local Vertex Clustering (LVC) coarse representation of $M_1 \cup M_2$ is a single ball b of radius r such that:*

$$b = \min_r \{b : b \succ b_1 \cup b_2\} \ .$$

Figure 2.35a shows the case where the balls a, b, c, d, e and f are in the following relation: $e = LVC\{a, b\}$, $f = LVC\{c, d\}$ and $g = LVC\{e, f\}$. It is easy to see that the *LVC* can be computed in constant time. In particular consider two balls b_1, b_2 of radii r_1, r_2 and which centers c_1, c_2 have distance $d = \|c_1 - c_2\|$. The ball $b = LVC\{b_1, b_2\}$ has radius r and center c given by:

$$r = \frac{d + r_1 + r_2}{2}, \qquad c = \frac{c_1 + c_2}{2} + \frac{r_1 - r_2}{2d}(c_1 - c_2).$$

A more tight coarsening procedure is based on the following clustering scheme (see Figure 2.35(b)):

Definition 2.6.4. (MVC). *Let b_1, b_2 be be a coarse representation of M_1, M_2, $(M_i \subset M)$. The Minimum Vertex Clustering (MVC) coarse representation of $M_1 \cup M_2$ is a single ball b of radius r such that:*

$$b = \min_r \{b : b \succ M_1 \cup M_2\} \ .$$

Clearly, the computation of the MVC coarsening is more expensive than the LVC. We determine the MVC using the Smallest Enclosing Ball Library by Dave White [79] which implements the optimal algorithm by Emo Welzl [78] generalized from the case of a set of points to the case of a set of balls.

(a) (b)

Figure 2.35: Two cascaded step of Vertex Clustering coarsening. (a) Local Vertex Clustering (LVC). (b) Minimal Vertex Clustering (MVC).

[2]Note that the points do not always become redundant in the sense that a flip will remove them from the triangulation. They are redundant only in the sense the points of their ball are also points of at least one other ball, hence their removal would not alter the molecular body itself (the set of boundary points does not change and the set of interior points remains the same).

2.6.3 Multiresolution Hierarchy

Construction

The basic decimation operation used for building the hierarchy is vertex clustering of edges which lie on the zero-shape.

Definition 2.6.5. (independent clustering). *Let $e_v = (v_0, v_1), e_u = (u_0, u_1)$ be two edges in the triangulation. and v_2, u_2 the two new vertices introduced if e_v, e_u were clustered. e_v and e_u can be* independently clustered *if and only if $\forall i, v_i \neq u_i$.*

The construction algorithm proceeds by creating consecutive levels of coarser approximations of the triangulation. Each level is constructed by using a priority queue (heap) for the zero-shape edges according to the error norm used (see Section 2.6.4). For fast traversal, The decimation operations are collected in a DAG similar to [28]. This structure requires that only non-dependent vertices be clustered in each level.

Definition 2.6.6. (Dependent Vertex). *A vertex v is considered* dependent *at level k if one of the following is true:*
1) v has been clustered at any level $i < k$.
2) v has been introduced (as a cluster of two others) at level k.
3) v is a neighbor of u, where u satisfies either 1 or 2 above.

In order to gain larger adaptiveness in the space of possible triangulations, and reduce considerably the storage size, our scheme introduces a cluster-forest of binary trees instead of the DAG. Each node in this forest represents a new ball as the parent of the two balls being clustered. The triangulation is updated during runtime while traversing the trees and inserting/removing balls. In this scheme, a vertex is considered dependent only if it satisfies the first two conditions of the above definition, hence the number of dependencies are much smaller and the space of possible triangulations is larger.

In both hierarchies, to guarantee a broad structure, edges outside the zero-shape are considered if a predefined minimum percent of the vertices are not removed (very rare in practice). An outline of the algorithm for building the hierarchy is as follows:

```
let T be the triangulation
let Z be the zero-shape
let H be a min heap of zero edges
loop until coarse enough:
   insert all Z edges to H
   while H is not empty do:
      remove minimal e=(u,v) from H
         check that u and v
            are non-dependent
         cluster u and v to w:
         remove u from T and correct
         remove v from T and correct
         insert w to T and correct
         update Z
         update H
```

The supporting structures for this algorithm are the triangulation, the zero-shape, and the heap of zero edges. After each decimation step, the triangulation is changed, which induces a change in the zero shape. Some zero edges could be gone, and new zero edges can be created. This means the heap needs to be updated after each decimation step. Also, in order to maintain an independent set in each level, vertices are marked as dependent (and in the case of the DAG, also their neighbors).

2.6.4 Error Estimates

While we use the zero-shape to guide the decimation process, it is important to have a bound on the geometric error while decimating, and during traversal of the hierarchy. The first type of error metric that we consider is just the length of the edge. In order not to be dependent on the radii of the atoms, we actually use the Euclidean distance between the vertices to represent the priority. Hence, we are looking at all atoms which overlap and cluster the closest ones first. However, since we are interested in the union of balls and not the actual zero shape, the second error metric we use involves the difference in area between the new ball and the two old balls. The larger this area is the more likely the shape will change drastically if this clustering is used. The last error metric actually computes the exact Hausdorff distance between the boundary of the two old balls and the new one. The Hausdorff distance function is defined as:

Definition 2.6.7. (One Way Hausdorff Distance). *Given two molecular shapes M_1, M_2 and a point-point distance function (norm) $d(p,q)$, the one-way Hausdorff distance $h(M_1,M_2)$ is the maximum of the minimum point-point distance function $d(p,q)$ for all p in M_1 and q in M_2:*

$$h(M_1,M_2) = \max_{p\in M_1}\{\min_{q\in M_2} d(p,q)\} .$$

Definition 2.6.8. (Hausdorff Distance). *The Hausdorff distance between two molecular shapes $H(M_1,M_2)$ is the maximum between the two one-way Hausdorff distance functions:*

$$H(M_1,M_2) = \max\{h(M_1,M_2), h(M_2,M_1)\} .$$

By definition 2.6.1 of the relation "$\succ$" we have immediately that:

Property 2. *M_1 is a coarse representation of M_2 if and only if the one-way Hausdorff distance of M_2 from M_1 is zero:*

$$M_1 \succ M_2 \Leftrightarrow h(M_2,M_1) = 0 .$$

Property 3. *If M_1 is a coarse representation of M_2, then the Hausdorff distance between M_1 and M_2 is equal to the one-way Hausdorff distance of M_1 from M_2:*

$$M_1 \succ M_2 \Rightarrow H(M_1,M_2) \equiv h(M_1,M_2) .$$

We determine a conservative estimate of the Hausdorff distance $H(M_1,M_2)$ by computing the one-way Hausdorff distance between two adjacent levels in a Local Vertex Clustering step as follows.

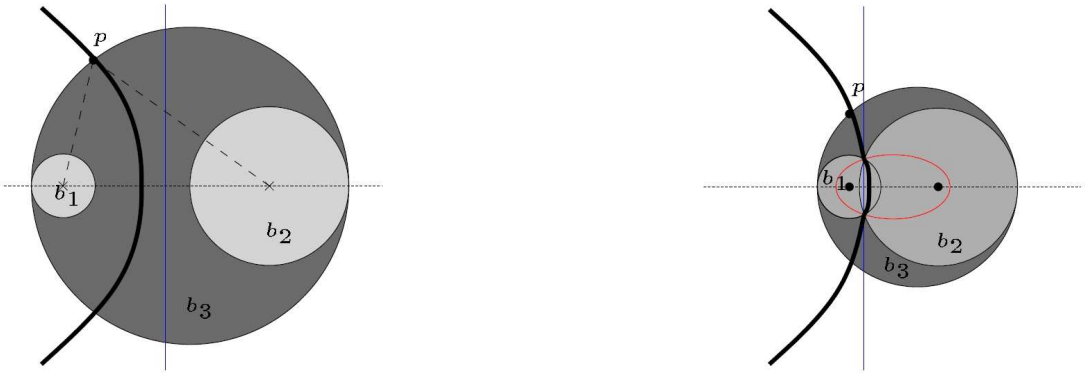


Figure 2.36: The one-way Hausdorff distance between b_3 and $b_1 \cup b_2$ is determined at the point p of intersubsection between ∂b_3 and the Voronoi separator between b_1 and b_2. (a) Configuration of $b_1 \cap b_2 \neq \emptyset$ where $l > 0$ (note that in such 2D subsection of the balls, the Voronoi separator has a secondary closed curve inside the two circles that is not of our interest). (b) Configuration for $b_1 \cap b_2 = \emptyset$ where $l < 0$.

Consider two balls b_1, b_2 of radii $r_1, r_2 > 0$ and whose centers c_1, c_2 are distance $\|c_1 - c_2\| = r_1 + r_2 - 2l$ apart (see Figure 2.36). We assume that neither $b_1 \subseteq b_2$ nor $b_2 \subseteq b_1$, which implies $l < \min\{r_1, r_2\}$ or $l > \max\{r_1, r_2\}$. Without loss of generality we place the centers of the two circles around the origin along the x axis so that their centers have coordinates $c_1 = (-r_1 + l, 0)$ and $c_2 = (r_2 - l, 0)$. The LVC ball b_3 then has center $c_3 = (r_2 - r_1, 0)$ and radius $r_1 + r_2 - l$. The point p where we can evaluate the distance $H(b_1 \cup b_2, b_3)$ is the intersubsection between the boundary of b_3 and the Voronoi separator of b_1 and b_2. It is hence given by the solution of the following system:

$$\begin{cases} (x - r_2 + r_1)^2 + y^2 = (r_1 + r_2 - l)^2 \\ \\ \sqrt{(x + r_1 - l)^2 + y^2} - r_1 = \\ \qquad\qquad \sqrt{(x - r_2 + l)^2 + y^2} - r_2 \end{cases}$$

This has a closed form solution that can be readily derived from:

$$\begin{cases} (r_1+r_2-2l)^2x^2 \\ \quad +2(r_1-r_2)[(r_1-l)^2+(r_2-l)^2]x \\ \quad -(2r_1-l)(2r_2-l)(r_1-r_2)^2=0 \\ \\ y=\pm[-x^2+2(r_2-r_1)x \\ \qquad +(r_1+r_2-l)^2-(r_1-r_2)^2]^{1/2}\,. \end{cases}$$

Error of MVC Using the Minimal Vertex Clustering decimation we can achieve a tighter encapsulation between each level of resolution and the input model. This makes it more difficult to compute the error bound as we cannot simply accumulate the error bounds from level to level. This is because each level of resolution contains the input model but not the immediately finer approximation. This means that we have to compare each ball at the current level of approximation with the contained balls in the finest level of resolution, requiring the evaluation of the error function at several vertices of the actual Voronoi diagram (the real Voronoi diagram and not the Power diagram) and at the intersubsection between the Voronoi diagram and the cluster ball. Fortunately it can be shown (see Appendix 2.6.6 for details) that the square root of the norm of the Power distance can be used as an upper bound of the actual Euclidean distance so that we can use the Power diagram in place of the real Voronoi diagram. This makes such computation viable in practice because we have to compute the Power diagram and hence this error estimate does not substantially increase the complexity of the computation.

2.6.5 Analysis

Hierarchy Construction.

We first consider the case of LVC where the entire hierarchy is constructed by a sequence of vertex insertion steps. Each insertion can make at least two vertices in the previous level of resolution redundant or irrelevant. Redundant means they are removed from the triangulation by the flipping sequence. Irrelevant means they do not contribute to the boundary of the molecular shape any more and hence are not considered in the following decimation steps. Therefore, the complexity of the molecular shape decreases by at least one ball per decimation step so that for an initial shape based on n balls, the hierarchy is constructed with a sequence of less than n LVC steps. In other words the complexity of constructing the initial fine resolution mesh and the complexity of constructing the entire hierarchy are the same. If d is the dimension of the embedding space, then the complexity is $O(n\log n+n^{\lfloor d/2\rfloor})$ (or $O(n\log n+n^{\lceil d/2\rceil})$ if appropriate randomization applies) [38] since it is at most the triangulation time for $2n$ points. Note that the logarithmic factor introduced by the using a priority queue in the decimation does not increase the overall complexity. If MVC decimation is used the only difference is that instead of just inserting a vertex, each decimation step involves the insertion of one ball and the removal of two, which means a factor of three is added.

Traversal

As mentioned previously we consider two possible options: (a) explicit storage in the hierarchy of the sequence of flips performed during the decimation or (b) reduction of the hierarchy to a tree of balls.

In the first case the storage size is $O(n^{\lfloor d/2\rfloor})$ (which reduces to expected $O(n^{\lceil d/2\rceil})$ in the randomized case), this being the order of the total number of d-simplices in the triangulation as well as the total number of flip operations performed during the construction of the hierarchy. In the second case, the storage size remains linear in any dimension, since it is only a balanced tree of $2n$ nodes in the worst case.

The complexity of the traversal needed to transform the mesh from one cut of the hierarchy to another is proportional to the number of flips it takes to perform the transition, which is proportional to the number of simplices that are being replaced in the initial and final triangulations. In particular if k d-simplices are being created in the new cut then the transition time is $O(k)$. The constant that is hidden in the $O(k)$ depends on the kind of hierarchy that is used. For a full DAG representation the constant is very small since the new triangulation is just read from the DAG. In the case of a tree hierarchy the constant is large since each flip operation involves the determination of Delaunay conditions that are equivalent to computation of $(d+1)\times(d+1)$ determinants. Using for example Gaussian elimination for the determinant evaluation, it would make the overall complexity $O(d^3k)$. Note that in this case the space of possible adaptive triangulations corresponding to cuts of the tree hierarchy is much larger than in the case of the DAG hierarchy because one is not constrained by neighboring dependencies between the pre-recorded sequences of flips.

2.6.6 Examples

The method was tested on both artificial shapes and 2D projections of real molecules (see Figure 2.37). The hierarchies built would be different mainly in the order of decimation steps, and differences would be mostly local in nature. As can be seen in the examples, the desired behavior of preserving as much as possible the structure of the shape while decimating is met.

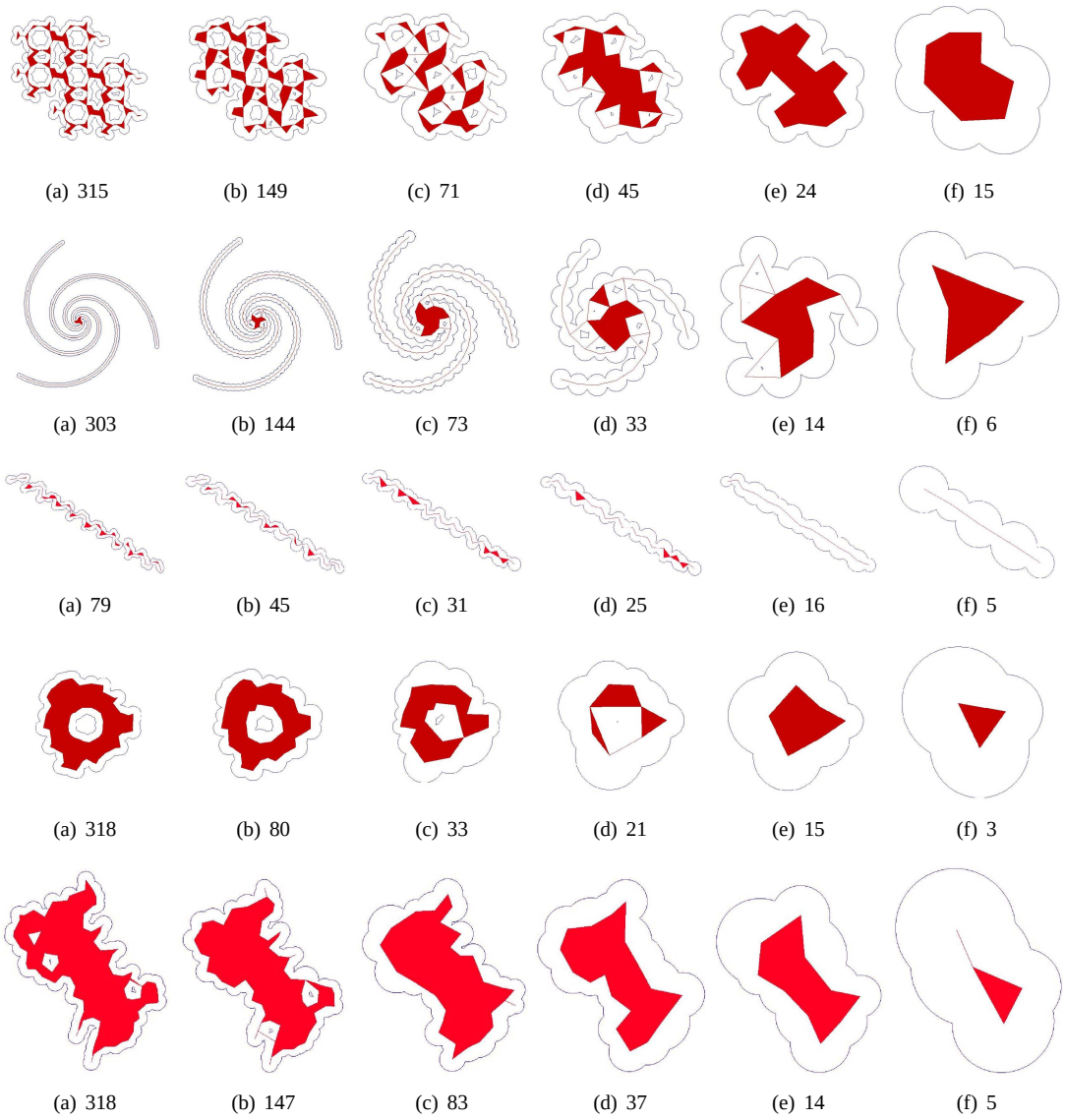


Figure 2.37: Multiresolution molecular shapes. The images show the boundary of the union of balls and the zero α-shape. The numbers denote the number of balls in each resolution. Rows 1-3 are artificial examples demonstrating how the topological structure is followed during decimation. The asymmetry in the spirals shape is a result of the pair-wise clustering. Rows 4 and 5 are two different parallel projections of the gramicidin molecule. There is a large amount of overlapping balls due to the projection, which accounts for the rapid drop in the number of balls at the first stages of decimation with small error.

Related Work

Union of Balls using Voronoi-Cell Complexes

Several different approaches have been developed to achieve this efficiency for molecular surface computations [27, 70, 71, 72, 75, 77]. Other work on surface representations features the use of metaballs, molecular surfaces, and blobby models [1, 20, 80, 31, 44, 47, 52, 60, 63, 64, 81, 82, 83].

In previous work on dynamic triangulations the focus has been mostly on the simpler Delaunay/Voronoi structures (unweighted case) [6, 53, 24, 43, 48, 2, 67, 68]. Little has been done on the more general case of dynamic Regular Triangulation/Power Diagrams and for dimensions greater than two. Moreover, the kinds of dynamic operations developed are usually just the insertion/deletion of a single point. Such local operations become inefficient when we need to perform even a simple but global modification.

Molecular Surface Computation using Adaptive Grids

Since Richards introduced the SES definition, a number of techniques have been devised to compute the surface, both static and dynamic, implicit and explicit. Connolly introduced two algorithms to compute the surface. First, a dot based numerical surface construction and second, an enumeration of the patches that make up the analytical surface (See [27], [26] and his PhD thesis). In [77], the authors describe a distance function grid for computing surfaces of varying probe radii. Our data structure contains approaches similar to their idea. A number of algorithms were presented using the intersection information given by voronoi diagrams and the alpha shapes introduced by Edelsbrunner [37], including parallel algorithms in [75] and a triangulation scheme in [1]. Fast computations of SES is described in [71] and [70], using Reduced sets, which contains points where the probe is in contact with three atoms, and faces and edges connecting such points. Non Uniform Rational BSplines (NURBs) descriptions for the patches of the molecular surfaces are given in [11], [10] and [12]. You and Bashford in [84] defined a grid based algorithm to compute a set of volume elements which make up the Solvent Accessible Region.

Maintaining Union of Balls Under Atom Movements

Though a number of techniques have been devised for the static construction of molecular surfaces (e.g., [27, 26, 77, 37, 75, 1, 71, 70, 84, 46, 11, 10, 85, 16]), not much work has been done on neighborhood data structures for the dynamic maintenance of molecular surfaces as needed in MD. In [12] Bajaj et al. considered limited dynamic maintenance of molecular surfaces based on Non Uniform Rational BSplines (NURBS) descriptions for the patches. Eyal and Halperin [39, 40] presented an algorithm based on dynamic graph connectivity that updates the union of balls molecular surface after a conformational change in $\mathcal{O}\left(\log^2 n\right)$ amortized time per affected (by this change) atom.

Clustering and Decimation of Molecular Surfaces

Using multiresolution models for molecules can substantially improve rendering speed and interactive response rates in molecular interaction tools. Similar improvements in performance would be achieved when a set of balls is used as an approximate representation of a generic object either for modeling (meta-balls [47, 64], blobby models [83]) or for collision detection [52]. Direct application of previous approaches for the decimation and multiresolution representation of the surfaces themselves [71, 56] can have serious embedding and self-intersection problems and are specific to the surface definition. A possible solution if this problem has been addressed in [74] but limited to the case of the boundary surface of tetrahedral meshes. Our multiresolution scheme updates the underlying structure of the molecule, maintaining at any level of detail a regular triangulation of the current weighted point-set. In this way we explicitly track the topology of the molecular body at any adaptive level of resolution. Moreover this guarantees correct embedding in all resolutions and creates an approximation from which the surface boundary can be computed in any of the previous schemes.

There are many approaches for creating multiresolution representations of geometric data for graphics and visualization [69, 59, 54]. They vary in both the simplification scheme like vertex removal [28], edge contraction [50], triangle contraction [45], vertex clustering [73], wavelet analysis [32], and also in the structure used to organize the levels of detail (either a linear order or a using a DAG).

Maintaining the regular triangulation at all resolutions rules out the possibility of using decimation techniques like edge or triangle contraction, which do not guarantee the (weighted) Delaunay property. Other known decimation schemes that can

guarantee this property such as vertex removal, do not seem appropriate in this case since they do not preserve the molecule features as a subset of the whole triangulation. Techniques which preserve features in the triangulation by tagging specific edges or vertices [23] are more suitable for preserving specific edges or regions. We are more interested in applying the decimation on a subset of the triangulation while this subset can change during the decimation.

Sphere trees have also been used in [51] for the purpose of fast collision detection. In this work, Sphere hierarchies are built around a given object either by replacing special octree regions or by placing balls on the medial-axis surfaces approximated using voronoi edges of some point sampling of the object. The basic approach of building the hierarchy by clustering pairs of balls for collision detection [52] is similar to ours. However in this scheme the simplification process does not update the underlying triangulation and hence does not track the topological changes induced by the decimation process. This make also the scheme unable to cluster balls that get in contact only after some simplification steps.

Relevant Mathematics

Voronoi-Delaunay Diagram

For a finite set of points P in $\mathbb{R}^3$, the Voronoi cell of $p \in P$ is

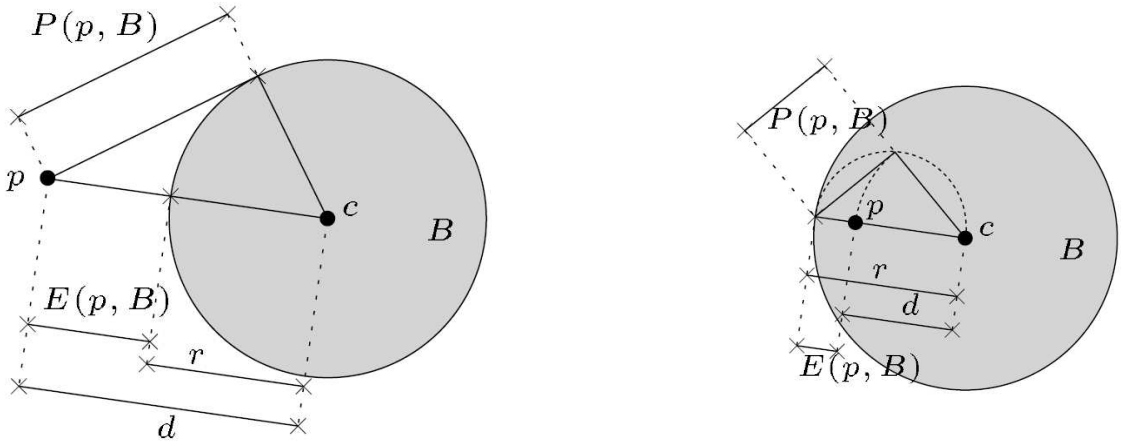
$$V_p = \{x \in \mathbb{R}^3 : \forall q \in P - \{p\}, \|x-p\| \leq \|x-q\|)\}.$$

If the points are in general position, two Voronoi cells with non-empty intersection meet along a planar, convex Voronoi facet, three Voronoi cells with non-empty intersection meet along a common Voronoi edge and four Voronoi cells with non-empty intersection meet at a Voronoi vertex. A cell decomposition consisting of the *Voronoi objects*, that is, Voronoi cells, facets, edges and vertices is the Voronoi diagram $\mathrm{Vor}\,P$ of the point set P.

The dual of $\mathrm{Vor}\,P$ is the Delaunay diagram $\mathrm{Del}\,P$ of P which is a simplicial complex when the points are in general position. The tetrahedra are dual to the Voronoi vertices, the triangles are dual to the Voronoi edges, the edges are dual to the Voronoi facets and the vertices (sample points from P) are dual to the Voronoi cells. We also refer to the Delaunay simplices as *Delaunay objects*.

Euclidean VS Power distance.

For MVC the choice of using the Power distance in place of the Euclidean distance is motivated by the the efficiency and simplicity of the construction of the power diagram together with the fact that the power distance can be proven to be an upper bound of the Euclidean distance.



(a) (b)

Figure 2.38: Relationship between the Euclidean distance $E(p,B)$ between the point p and the ball B and their Power distance $P(p,B)$, (a) Configuration for $d > r$. (b) Configuration for $d < r$.

Consider a point p at distance d from the center c a ball B of radius r as in Figure 2.38. We define:

$$E(p,B) = |d-r|\,,\ \ P(p,B) = \sqrt{|d^2-r^2|}\,.$$

Then we have the following chain of inequalities (where r and d are positive numbers):

$$\begin{aligned} 0 \leq 4dr(d-r)^2 &= 4d^3r - 8d^2r^2 + 4dr^3 \\ (d-r)^4 &= d^4 - 4d^3r + 6d^2r^2 - 4dr^3 + r^4 \\ &\leq d^4 - 2d^2r^2 + r^4 = (d^2-r^2)^2 \\ E(p,B) = |d-r| &\leq \sqrt{|d^2-r^2|} = P(p,B)\,. \end{aligned}$$

In conclusion we have that for any given ball B and point p, the function $P(p,B)$ provides an upper bound on the distance $E(p,b)$:

$$E(p,B) \leq P(p,B)\ , \tag{6.28}$$

with equality holding only when $d = r$, i. e. the point is on the surface of the ball (and in trivial cases where d or r is zero).

For a collection of n balls $\mathscr{B} = \{B_1, \ldots, B_n\}$ the distance functions are extended as follows:

$$E(p,\mathscr{B}) = \min_{1\leq i\leq n} |d_i - r_i| \tag{6.29}$$

$$P(p,\mathscr{B}) = \sqrt{\min_{1\leq i\leq n} |d_i^2 - r_i^2|} \tag{6.30}$$

The problem in comparing $E(p,\mathscr{B})$ with $P(p,\mathscr{B})$ is that they may achieve their minimum for different values of i because in general the Power diagram is not coincident with the Voronoi diagram. Figure 2.6.6 shows an example of comparison between the Voronoi diagram of two circles (in red) with the corresponding Power diagram (in blue). In this example the minimum distance of the point p from the set $\mathscr{B} = \{B_1, B_2\}$ is achieved at $i = 1$ for $P(p,\mathscr{B})$ and at $i = 2$ for $E(p,\mathscr{B})$:

$$P(p,\mathscr{B}) = P(p,B_1)$$

$$E(p,\mathscr{B}) = E(p,B_2)\ .$$

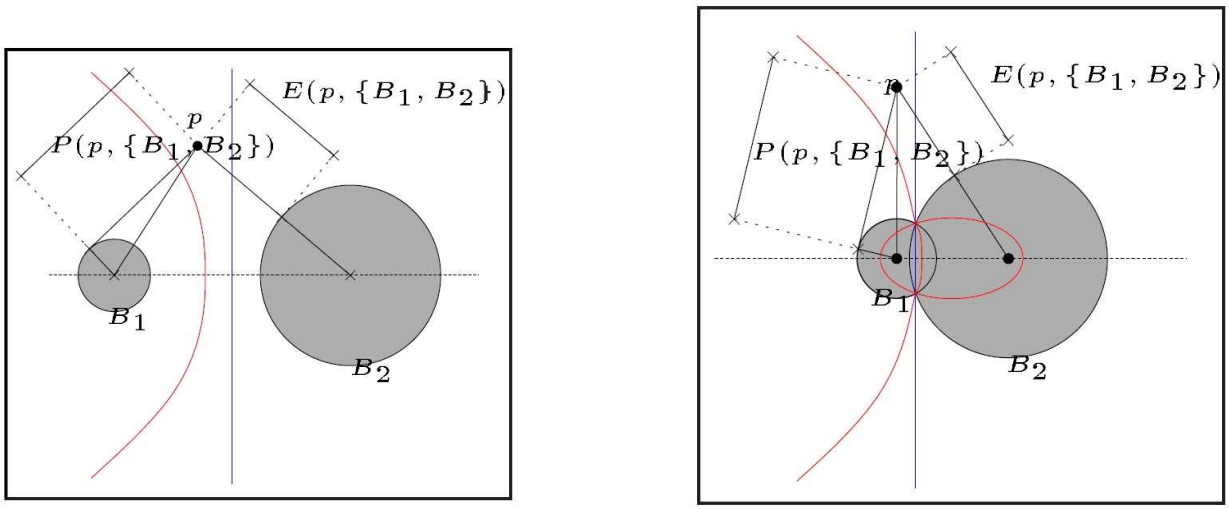
In general for a given point p we call i_P, i_E the two indices such that:

$$P(p,\mathscr{B}) = P(p,B_{i_P})$$

$$E(p,\mathscr{B}) = E(p,B_{i_E})\ .$$

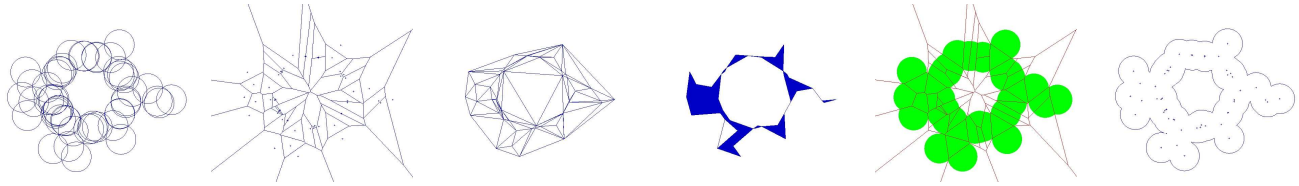
From equations (6.29) and (6.28) we have that:

$$\begin{aligned} E(p,\mathscr{B}) &= E(p,B_{i_E}) \leq E(p,B_{i_P}) \\ &\leq P(p,B_{i_P}) = P(p,\mathscr{B})\ . \end{aligned}$$



(a) (b)

Figure 2.39: Power diagram (in blue) and Voronoi diagram (in red) of two circles. (a) Case of nonintersecting circles. (b) Case of intersecting circles.



(a) (b) (c) (d) (e) (f)

Figure 2.40: The combinatorial and geometric structures underlying a molecular shape: (a) The collection of balls (weighted points). (b) Power diagram of a set of the points. (c) Regular triangulation. (d) The α-shape (with $\alpha = 0$) of the points. (e) Partitioning of the molecular body induced by the power diagram. (f) The boundary of the molecular body.

Power Diagram

Given a weighted point $P = (p, w_p)$ where $p \in \mathbb{R}^n$ and $w \in \mathbb{R}$, the *power distance* from a point $x \in \mathbb{R}^n$ to P is defined as

$$\pi_P(x) = \sqrt{\|p - x\|^2 - w_p} \ ,$$

where $\|p - x\|$ is the ordinary Euclidean distance between p and x.

In molecule context, we define the weight of an atom B with center at p and radius r to be $w_B = r^2$. The *power distance* of x to B is

$$\pi_B(x) = \sqrt{\|p - x\|^2 - r^2} \ .$$

Given a set $\{P_i\}$ of weighted vertices (each vertex has a weight w_i associated with it), the Power Diagram is a tiling of the space into convex regions where the ith tile is the set of points nearest to the vertex P_i, in the power distance metric [4]. The power diagram is similar to the Voronoi diagram using the power distance instead of Euclidean distance.

The weighted Voronoi cell of a ball B in a molecule $\mathscr{B}$ is the set of points in space whose weighted distance to B is less than or equal to their weighted distance to any other ball in $\mathscr{B}$ [36]:

$$V_B = \{x \in \mathbb{R}^2 | \pi_B(x) \leq \pi_C(x) \ \forall C \in \mathscr{B}\} \ .$$

The *power diagram* of a molecule is the union of the weighted Voronoi cells for each of its atoms (Figure 2.40(b)).

Regular Triangulation

The *regular triangulation*, or *weighted Delaunay triangulation*, is the dual (face adjacency graph) of the power diagram, just as the Delaunay triangulation is the dual shape of the Voronoi diagram. Vertices in the triangulation are connected if and only if their corresponding weighted Voronoi cells have a common face (Figure 2.40(c)). This implies that two vertices are connected if and only if they have a nearest neighbor relation measured in power distance metric

Given a set of n 2D points with weights, it has been shown [38], that their regular triangulation can be computed in $O(n \log n)$ time, by incrementally inserting new points to the existing triangulation and correcting it using edge flips.

Weighted Alpha Shapes

A simplex s in the regular triangulation of $\{P_i\}$ belongs to the α-shape of $\{P_i\}$ only if the orthogonal center of (the weighted point orthogonal to the vertices of) s is smaller than α (see [34] for the complete condition). The alpha shape where $\alpha = 0$, called the zero-shape, is the topological structure of molecules [41]. For example, an edge $e = (u, v)$ is a part of the zero-shape only if $\|u - v\|^2 - w_u - w_v < 0$, which means that the two balls centered at u and v intersect (Figure 2.40(d)).

Adaptive grid, Oct-tree and k-d tree

B-splines and B-patches

A-splines and A-patches

An A-patch of degree n over the tetrahedron $[\partial_1\partial_2\partial_3\partial_4]$ is defined by

$$G_n(x,y,z) := F_n(\alpha) = F_n(\alpha_1,\alpha_2,\alpha_3,\alpha_4) = 0, \tag{6.31}$$

where

$$F_n(\alpha_1,\alpha_2,\alpha_3,\alpha_4) = \sum_{i+j+k+l=n} a_{ijkl} B^n_{ijkl}(\alpha_1,\alpha_2,\alpha_3,\alpha_4), \tag{6.32}$$

$$B^n_{ijkl}(\alpha_1,\alpha_2,\alpha_3,\alpha_4) = \frac{n!}{i!\,j!k!l!}\alpha_1^i\alpha_2^j\alpha_3^k\alpha_4^l,$$

and $(x,y,z)^T$ and $(\alpha_1,\alpha_2,\alpha_3,\alpha_4)^T$ are related by

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \partial_1 & \partial_2 & \partial_3 & \partial_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix}. \tag{6.33}$$

Control points computation for trimmed NURBS patches

In this appendix we explain the computation of the NURBS control points. The approach we take is to compute the control points once for all molecule atoms. That is each atom will be represented by its specific domain D in (u,v) space and the same set of normalized control points that represent the unitary sphere with center in the origin. Then we apply an affine transformation to map the unitary sphere to the position taken by the atom. To have a unique base set of control points (defining a portion of the normalized sphere) that can represent any atom we need to be sure that for each ball B in $\mathscr{B}$ there is at least a neighbor ball $\bar{B}$, that intersects B for the smallest portion. This is because we wish to compute the control points of a portion of sphere which is a (bounded) rectangular domain and a minimum superset of any domain D of any atom.

Fortunately this condition is satisfied for all molecules. For example in the ball and stick representation used in Raster3D [5, 61] a bond (stick) is drawn between to atoms of radii r_1, r_2 if the distance r between the centers of the two atoms is less than $0.6(r_1+r_2)$. Since in a molecule there is at least one bond per atom we have that for each atom there is at least a neighbor atom for which $r < 0.6(r_1+r_2)$. If we also consider that minimum atom size in a molecule is 1.3 $\mathring{\text{A}}$ and the maximum is 2.18 $\mathring{\text{A}}$ we have that each atom is intersected by a neighbor atom for at least 0.15477% of its radius. This means that, with reference to equation (2.2) we can always assume to have $d \leq 0.84523$ that is $l \leq 3.45288299571568$. For this fixed value of l we apply a change of polynomial basis to get the coordinates (x,y,z) of one quarter of the control points (and relative weight w) as in the table below.

The other control points are just computed mirroring these twice with respect to the x and y axis. The knots vectors are $u : [-1-1-100111]$ $v : [-1-1-100111]$.

	x	y	z	w
P_1	0	0	−1	1
P_2	0	1	−1	1
P_3	0	2	0	2
P_4	1	0	−1	1
P_5	1	1	−1	1
P_6	1	2	0	2
P_7	2	0	0	2
P_8	2	1	0	2
P_9	2	2	1	3

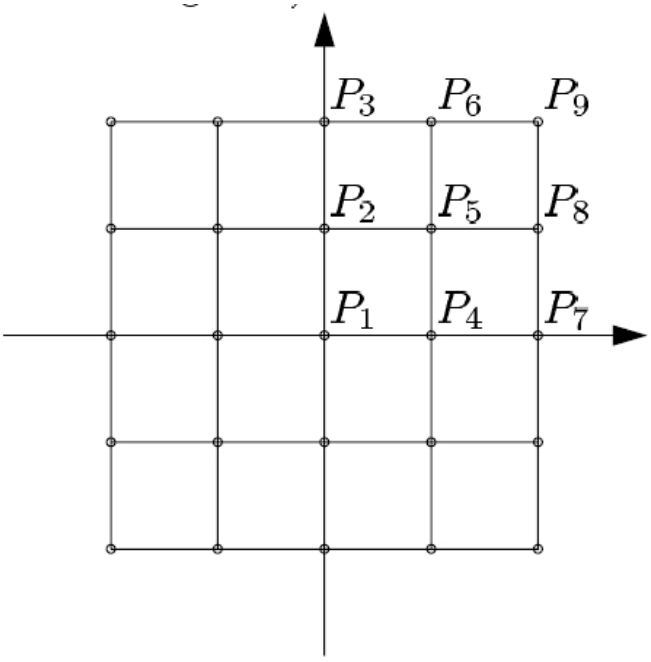


Figure 2.41: Control point computation

Bibliography

[1] N. Akkiraju and H. Edelsbrunner. Triangulating the surface of a molecule. *Discrete Applied Mathematics*, 71(1-3):5–22, 1996.

[2] G. Albers and T. Roos. Voronoi diagrams of moving points in higher dimensional spaces. In *Proc. 3rd Scand. Workshop Algorithm Theory*, volume 621 of *Lecture Notes in Computer Science*, pages 399–409. Springer-Verlag, 1992.

[3] J. Arvo. A simple method for box-sphere intersection testing. pages 335–339, 1990.

[4] F. Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23:345–405, 1991.

[5] D. J. Bacon and W. F. Anderson. A fast algorithm for rendering space-filling molecule pictures. *Journal of Molecular Graphics*, 6:219–220, 1988.

[6] C. Bajaj and W. J. Bouma. Dynamic Voronoi diagrams and Delaunay triangulations. In *Proceedings of the 2nd Canadian Conference on Computational Geometry*, pages 273–277, 1990.

[7] C. Bajaj, R. Chowdhury, and M. Rasheed. A dynamic data structure for flexible molecular maintenance and informatics. Technical Report TR-10-31, ICES, Univ. of Texas at Austin, July 2010.

[8] C. Bajaj, R. A. Chowdhury, and M. Rasheed. A dynamic data structure for flexible molecular maintenance and informatics. In *SPM '09: 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*, pages 259–270, New York, NY, USA, 2009. ACM.

[9] C. Bajaj and M. Kim. Generation of configuration space obstacles: The case of moving algebraic curves. *Algoritmica*, 4:157–172, 1989.

[10] C. Bajaj, H. Y. Lee, R. Merkert, and V. Pascucci. NURBS based B-rep models for macromolecules and their properties. In *Proceedings of the 4th ACM Symposium on Solid Modeling and Applications*, pages 217–228, New York, NY, USA, 1997. ACM.

[11] C. Bajaj, V. Pascucci, A. Shamir, R. Holt, and A. Netravali. Multiresolution molecular shapes. Technical report, TICAM, Univ. of Texas at Austin, Dec. 1999.

[12] C. Bajaj, V. Pascucci, A. Shamir, R. Holt, and A. Netravali. Dynamic maintenance and visualization of molecular surfaces. *Discrete Applied Mathematics*, 127(1):23–51, 2003.

[13] C. Bajaj and A. Royappa. Finite representations of real parametric curves and surfaces. *International Journal of Computational Geometry Applications*, 5(3):313–326, 1995.

[14] C. Bajaj and V. Siddavanahalli. Fast error-bounded surfaces and derivatives computation for volumetric particle data. ICES Tech Report 06-03, U.T. Austin, 2006.

[15] C. Bajaj, G. Xu, R. Holt, and A. Netravali. NURBS approximation of a-splines and a-patches. *International Journal of Computational Geometry and Applications*, 13(5):359–389, November 2003.

[16] C. Bajaj, G. Xu, and Q. Zhang. A fast variational method for the construction of resolution adaptive c^2-smooth molecular surfaces. *Computer Methods in Applied Mechanics and Engineering*, 198(21-26):1684–1690, 2009.

[17] C. L. Bajaj and V. Pascucci. Splitting a complex of convex polytopes in any dimension. In *Proceedings of the Twelfth Annual Symposium On Computational Geometry (ISG '96)*, pages 88–97, May 1996.

[18] C. L. Bajaj and V. Pascucci. Wrapping the Voronoi diagram: A constructive approach to duality. Technical report, University of Texas at Austin, 1997.

[19] S. Batsanov. Van der Waals radii of elements. *Inorganic Materials*, 37:871–885(15), September 2001.

[20] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, July 1982.

[21] T. M. Y. Chan, J. Snoeyink, and C. K. Yap. Output-sensitive construction of polytopes in four dimensions and clipped Voronoi diagrams in three. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 282–291, San Francisco, California, January 1995.

[22] B. Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete & Computational Geometry*, 10:377–409, 1993.

[23] L. P. Chew. Constrained Delaunay triangulations. In *Proc. 3rd Annual ACM Symposium Computational Geometry*, pages 215–222, 1987.

[24] P. Chew. Near-quadratic bounds for the L_1 Voronoi diagram of moving points. *Computational Geometry Theory and Applications*, 7, 1997.

[25] K. L. Clarkson, H. Edelsbrunner, L. J. Guibas, M. Sharir, and M. Welzl. Combinatorial complexity bounds for arrangements of curves and spheres. *Discrete Computational Geometry*, 5(2):99–160, 1990.

[26] M. Connolly. Solvent-accessible surfaces of proteins and nucleic acids. *Science*, 221(4612):709–713, 19 August 1983.

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

[28] M. de Berg and K. T. G. Dobrindt. On levels of detail in terrains. *Graphical Models and Image Processing*, 60:1–12, 1998.

[29] L. De Floriani. A pyramidal data structure for triangle-based surface description. *IEEE Computer Graphics and Applications*, 9(2):67–78, March 1989.

[30] C. J. A. Delfinado and H. Edelsbrunner. An incremental algorithm for Betti numbers of simplicial complexes on the 3-sphere. *Computer Aided Geometric Design*, 12(7):771–784, 1995.

[31] M. Desbrun and M. Gascuel. Animating soft substances with implicit surfaces. In R. Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 287–290, Los Angeles, August 1995. Addison Wesley.

[32] M. Eck, T. DeRose, T. Duchamp, T. Hoppe, H. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *ACM Computer Graphics Proceedings, SIGGRAPH'95*, pages 173–180, 1995.

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

[34] H. Edelsbrunner. Weighted alpha shapes. Technical Report 1760, University of Illinois at Urbana-Champaign, 1992.

[35] H. Edelsbrunner. The union of balls and its dual shape. *Discrete and Computational Geometry*, 13(3-4):415–440, 1995.

[36] H. Edelsbrunner, M. Facello, and J. Liang. On the definition and the construction of pockets in macromolecules. Tech Report UIUCDCS-R-95-1935, University of Illinois Urbana-Champaign, 1995.

[37] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[38] H. Edelsbrunner and N. R. Shah. Incremental topological flipping works for regular triangulations. *Algorithmica*, 15:223–241, 1996.

[39] E. Eyal and D. Halperin. Dynamic maintenance of molecular surfaces under conformational changes. In *SCG '05: Proceedings of the 21st Annual Symposium on Computational Geometry*, pages 45–54, 2005.

[40] E. Eyal and D. Halperin. Improved maintenance of molecular surfaces using dynamic graph connectivity. *Algorithms in Bioinformatics*, pages 401–413, 2005.

[41] M. A. Facello. *Geometric Techniques for Molecular Shape Analysis*. PhD thesis, University of Illinois, 1996. Department of Computer Science Technical Report # 1967.

[42] M. L. Fredman and D. E. Willard. Surpassing the information theoretic bound with fusion trees. *Journal of Computer and System Sciences*, 47(3):424–436, 1993.

[43] J. J. Fu and R. C. T. Lee. Voronoi diagrams of moving points in the plane. *International Journal of Computational Geometry Applications*, 1(1):23–32, 1991.

[44] T. Fujita, K. Hirota, and K. Murakami. Representation of splashing water using metaball model. *Fujitsu*, 41(2):159–165, 1990. in Japanese.

[45] T. S. Gieng, B. Hamann, K. I. Joy, G. L. Schussman, and I. J. Trotts. Constructing hierarchies for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):145–161, 1998.

[46] J. Grant and B. Pickup. A gaussian description of molecular shape. *Journal of Physical Chemistry*, 99:3503–3510, 1995.

[47] G. Graves. The magic of metaballs. *Computer Graphics World*, 16(5):27–32, 1993.

[48] L. Guibas, J. S. B. Mitchell, and T. Roos. Voronoi diagrams of moving points in the plane. In *Proceedings of the 17th Internatational Workshop on Graph-Theoretical Concepts in Computer Science*, volume 570 of *Lecture Notes in Computer Science*, pages 113–125. Springer-Verlag, 1991.

[49] D. Halperin and M. H. Overmars. Spheres, molecules, and hidden surface removal. In *SCG '94: Proceedings of the 10th Annual Symposium on Computational Geometry*, pages 113–122, 1994.

[50] H. Hoppe. Progressive meshes. In *ACM Computer Graphics Proceedings, SIGGRAPH'96*, pages 99–108, 1996.

[51] P. H. Hubbard. *Collision Detection for Interactive Graphics Applications*. PhD thesis, Department of Computer Science, Brown University, Providence, Rhode Island, April 1995.

[52] P. M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15(3), 1996.

[53] H. Imai, M. Iri, and K. Murota. Voronoi diagram in the laguerre geometry and its applications. *SIAM J. Comput.*, 14:93–10, 1985.

[54] R. Klein and J. Kramer. Multiresolution representations for surface meshes. In *Proceedings of the SCCG*, 1997.

[55] S. Krishnan, D. Manocha, and A. Narkhede. Representation and evaluation of boolean combinations of NURBS solids. In *Fifth MSI-Stony Brook Workshop on Computational Geometry, Stony Brook*, October 1995.

[56] S. Kumar, D. Manocha, and A. Lastra. Interactive display of large-scale NURBS models. *IEEE Transactions on Visualization and Computer Graphics*, 2(4):323–336, 1996.

[57] B. Lee and F. Richards. The interpretation of protein structures: estimation of static accessibility. *Journal of Molecular Biology*, 55(3):379–400, February 1971.

[58] P. Magillo. *Spatial Operations for Multiresolution Cell Complexes*. PhD thesis, Universitá di Genova, 1998. Dipartimento di Informatica e Scienze dell'Informazione.

[59] A. Maheshwari, P. Morin, and J. R. Sack. Progressive tins: Algorithms and applications. In *Proceedings 5th ACM Workshop on Advances in Geographic Information Systems*, Las Vegas, 1997.

[60] N. L. Max. Computer representation of molecular surfaces. *IEEE Computer Graphics and Applications*, 3(5):21–29, August 1983.

[61] E. A. Merritt and M. E. P. Murphy. Raster3d version 2.0: A program for photorealistic molecular graphics. *Acta Cryst.*, D50:869–873, 1994.

[62] C. W. Mortensen, R. Pagh, and M. Pătraçcu. On dynamic range reporting in one dimension. In *STOC '05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 104–111, 2005.

[63] H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa, and K. Omura. Object modeling by distribution function and a method of image generation. *Transactions IECE Japan, Part D*, J68-D(4):718–725, 1985.

[64] T. Nishita and E. Nakamae. A method for displaying metaballs by using Bézier clipping. In *Computer Graphics Forum*, volume 13, pages 271–280. Eurographics, Basil Blackwell Ltd, 1994. Eurographics '94 Conference issue.

[65] L. Piegl and W. Tiller. Curve and surface construction using rational b-splines. *Computer-Aided Design*, 19(9):485–498, 1987.

[66] F. Richards. Areas, volumes, packing, and protein structure. *Annual Review of Biophysics and Bioengineering*, 6:151–176, June 1977.

[67] T. Roos. Voronoĭ diagrams over dynamic scenes. *Discrete Applied Mathematics. Combinatorial Algorithms, Optimization and Computer Science*, 43(3):243–259, 1993.

[68] T. Roos. New upper bounds on Voronoi diagrams of moving points. *Nordic Journal of Computing*, 4(2):167–171, 1997.

[69] J. Rossignac and P. Borrel. Multi-resolution 3d approximation for rendering complex scenes. In B. Falcidieno and T. Kunii, editors, *Geometric Modeling in Computer Graphics*, pages 455–465. Springer-Verlag, 1993.

[70] M. Sanner, A. Olson, and J. Spehner. Fast and robust computation of molecular surfaces. In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 406–407. ACM Press, 1995.

[71] M. Sanner, A. Olson, and J. Spehner. Reduced surface: an efficient way to compute molecular surfaces. *Biopolymers*, 38(3):305–320, March 1996.

[72] M. F. Sanner and A. J. Olson. Real time surface reconstruction for moving molecular fragments. In *Proceedings of the Pacific Symposium on Biocomputing '97*, Maui, Hawaii, January 1997.

[73] W. J. Schroeder. A topology modifying progressive decimation algorithm. In R. Yagel and H. Hagen, editors, *IEEE Visualization '97*, pages 205–212. IEEE, nov 1997.

[74] O. G. Staadt and M. H. Gross. Progressive tetrahedralizations. In *Proceddings of the IEEE Visualization Conference*, pages 397–402, 1998.

[75] A. Varshney and F. Brooks. Fast analytical computation of richards's smooth molecular surface. In *VIS '93: Proceedings of the 4th conference on Visualization '93*, pages 300–307, 1993.

[76] A. Varshney, F. P. Brooks Jr., and W. V. Wright. Computing smooth molecular surfaces. *IEEE Computer Graphics Applications*, 14(5):19–25, 1994.

[77] R. Voorintholt, M. T. Kosters, G. Vegter, G. Vriend, and W. G. Hol. A very fast program for visualizing protein surfaces, channels and cavities. *Journal of Molecular Graphics*, 7(4):243–245, December 1989.

[78] E. Welzl. Smallest enclosing disks (balls and ellipsoids). In H. Maurer, editor, *New Results and New Trends in Computer Science*, number 555 in Lecture Notes in Computer Science, pages 359–370. Springer, 1991.

[79] D. White. Enclosing ball software library. Source code available at http://vision.ucsd.edu/~dwhite/ball.html.

[80] J. B. with C. Bajaj, J. Blinn, M.-P. Cani-Gascuel, A. Rockwood, B. Wyvill, and G. Wyvill, editors. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers, San Francisco, 1997.

[81] B. Wyvill, C. McPheeters, and G. Wyvill. Animating soft objects. *The Visual Computer*, 2(4):235–242, 1986.

[82] B. Wyvill, C. McPheeters, and G. Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, 1986.

[83] S. Yoshimoto. Ballerinas generated by a personal computer. *Journal of Visualization and Computer Animation*, 3(2):55–90, 1992.

[84] T. You and D. Bashford. An analytical algorithm for the rapid determination of the solvent accessibility of points in a three-dimensional lattice around a solute molecule. *Journal of Computational Chemistry*, 16(6):743–757, 1995.

[85] W. Zhao, G. Xu, and C. Bajaj. An algebraic spline model of molecular surfaces. In *SPM '07: Proceedings of the 2007*

ACM symposium on Solid and physical modeling, pages 297–302, 2007.

Chapter 3

Smooth Surfaces

The computation of electrostatic solvation energy (also known as polarization energy) for biomolecules plays an important role in the molecular dynamics simulation [63], the analysis of stability in protein structure prediction [110], and the protein-ligand binding energy calculation [66]. The explicit model of the solvent provides the most rigorous solvation energy calculation [87]. However, due to the large amount of solvent molecules, most of the computation time is spent on the trajectories of the solvent molecules, which severely increases the computation cost of this method [96]. An alternative method is to represent the solvent implicitly as a dielectric continuum [102], then the electrostatic potential is known by solving the Poisson-Boltzmann (PB) equations [26][80]. A more efficient method is to approximate the PB electrostatic solvation energy by the generalized Born (GB) model [111][28][72], which computes the electrostatic solvation energy ΔG_{elec} as

$$G_{\text{pol}} = -\frac{\tau}{2}\sum_{i,j}\frac{q_i q_j}{[r_{ij}^2 + R_i R_j \exp(-\frac{r_{ij}^2}{F R_i R_j})]^{\frac{1}{2}}}, \tag{0.1}$$

where $\tau = \frac{1}{\varepsilon_p} - \frac{1}{\varepsilon_w}$, ε_p is the solute (low) dielectric constant, ε_w is the solvent (high) dielectric constant, q_i is the atomic charge of atom i, r_{ij} is the distance between atom i and j, F is an empirical factor (could be 4 [111] or 8 [72]), and R_i is the effective Born radius of atom i. The effective Born radius reflects how deep an atom is buried in the molecule and consequently determines the importance to the polarization. The formulation of the effective Born radii is derived in [55]:

$$R_i^{-1} = \frac{1}{4\pi}\int_\Gamma \frac{(\mathbf{r}-\mathbf{x}_i)\cdot\mathbf{n}(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_i|^4}\, dS, \tag{0.2}$$

where Γ is the molecular surface of the solute, $\mathbf{x}_i$ is the center of atom i, and $\mathbf{n}(\mathbf{r})$ is the unit normal of the surface at $\mathbf{r}$. The details of the derivation of (6.13) and a fast evaluation algorithm based on the fast Fourier transform (FFT) for (6.13) is discussed in [20]. Since the numerical integrations are done on the molecular surface Γ, an accurate and analytic representation of Γ is needed.

Three well-known molecular surfaces are shown in Figure 3.1 in 2D. The van der Waals surface (VWS) is the union of a set of spheres with atomic van der Waals radii. The solvent accessible surface (SAS) is the union of augmented van der Waals spheres with each radius enlarged by the solvent probe radius (normally taken as 1.4) [69]. The solvent excluded surface (SES, also called molecular surface or Connolly surface) is the boundary of the union of all possible solvent probes that do not intersect with the interior of the VWS [40][93]. As described in [40], the SES consists of the convex spherical patches which are parts of the VWS as well, the toroidal pathces and the concave spherical patches, which are generated by the probes rolling along the intersections of neighboring atoms. The VWS causes an overestimation of the electrostatic solvation energy, while the SAS leads to an underestimation [72]. The SES is the most accurate when it is applied in the energetic calculation and therefore it is most often used to model the molecular surface. However the SES still has one significant drawback: it contains cusps when the rolling probe self-intersects, which may cause singularity in the Born radii and the force calculations.

In the energetic computation, knowing the patch complexes of the molecular surface is not enough. For convenience, an analytical representation of the molecular surface is needed and the singularity should also be avoided.

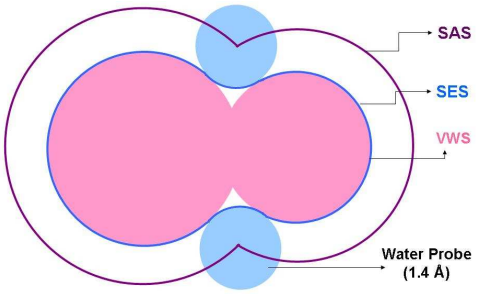


Figure 3.1: Three molecular surfaces are shown for two atoms in two dimension. The boundary of the union of balls (pink) with the van der Waals radii is the VWS. The SAS (purple) is the union of augmented van der Waals spheres with each radius enlarged by the radius of a solvent probe (light blue). The SES (the blue curve) is boundary of all possible solvent probes that do not intersect with the interior of the VWS.

3.1 Implicit Solvation Surface from volumetric Density Maps (Radial Basis Splines, C^inf)

We extract an implicit solvation surface (molecular surface) as a level set (isocontour) of the volumetric electron density maps [15]. The implicit solvation surface is chosen to be a good approximation of the Lee-Richards molecular surface [69] by choosing an appropriate weighting parameter of the summation of Gaussian kernel functions.

3.1.1 Gaussian Density Map

The molecular surface has been approximated in the past [29, 52, 58] by an isocontour:

$$M := \left\{x \in \mathbb{R}^3 : G(x) = 1\right\} \qquad with \qquad G(x) = \sum_{i=1}^{N} e^{B_i\left(\frac{\|x-x_i\|^2}{r_i^2}-1\right)}, \tag{1.3}$$

where (x_i, r_i) are the center and radius of the ith atom in the biomolecule, and $B_i < 0$ is called the 'decay rate', which controls the rate of decay of each atom's Gaussian kernel. Note that B_i must be negative to ensure that the density function goes to zero as $\| x - x_i \|$ goes to infinity. In order to make the distance between M and M_0 as uniform as possible, we take $C = B_i/r_i^2$, where $C < 0$ is a given constant. Now $G(x)$ becomes

$$G(x,C) = \sum_{i=1}^{N} e^{C(\|x-x_i\|^2 - r_i^2)}. \tag{1.4}$$

In the following for the molecular surface $M(C_i) = \{x \in \mathbb{R}^3 : G(x,C_i) = 1\}$, we consider $C = C_1, \ldots, C_l$. As shown in Fig. 3.2, the different effects of C and constant $B_i (= B)$ are shown for a two-sphere system, one is centered at (0, 0, 0) with radius of 1.0, the other one is at (2.8, 0, 0) with radius of 2.0. It can be observed that

Table 1: C (1/Angstrom^2) /B_i (constant) and Implicit Solvation Models in Fig. 3.2

	Red	Green	Magenta	Blue
Fig.3.2(a)	C = -0.125	C = -0.25	C = -0.5	C = -1.0
Fig.3.2(b)	B_i= -0.125	B_i= -0.25	B_i= -0.5	B_i= -1.0

- C leads to more uniform inflation than B_i.
- Small balls have more inflation than big ones.
- Large error occurs around the intersection region, and the error reduces gradually away from it.

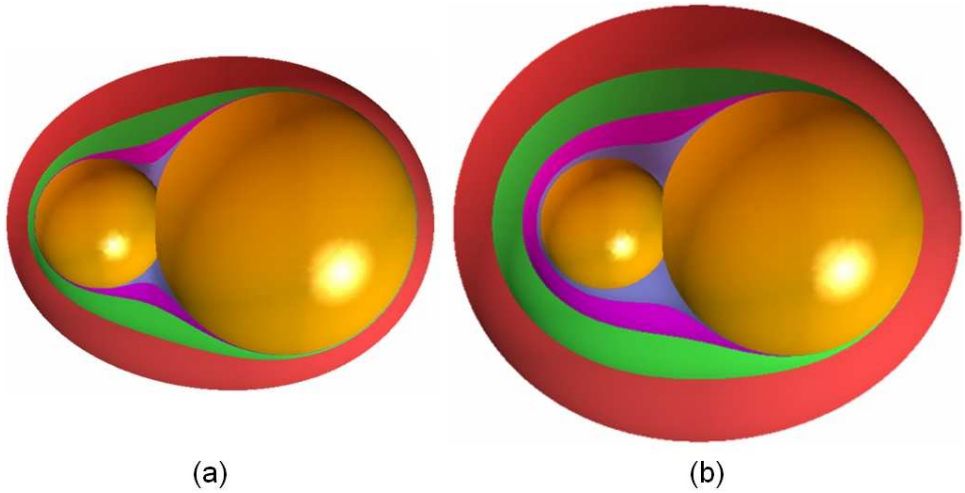


Figure 3.2: Implicit Solvation models by choosing various C in (a) and B_i in (b). Yellow balls are two input atoms. The correspondence between C/B_i values and these models are shown in Table 1.

- Larger C and B_i lead to greater inflation. For the same C and B_i value, e.g., -0.125, B_i tends to introduce more inflation.
- Inflation of the molecular surfaces distorts the polar solvation energies and hence to be seriously avoided.

Fig. 3.3 shows implicit solvation models of Ribosome 30S. Compared with Fig. 3.3(a), proteins inflate much more seriously in Fig. 3.3(e). rRNA in Fig. 3.3(c) and (f) looks similar, but proteins in Fig. 3.3(f) look a little more inflated than Fig. 3.3(b). rRNA in Fig. 3.3(d) and (g) looks similar too, but proteins in Fig. 3.3(g) are close to proteins in Fig. 3.3(c).

3.1.2 Multi-Level Gaussian Density Map

In order to reduce the inflation caused by Gaussian summation as well as to model molecular surfaces with varying resolution on the implicit solvation surface, we introduce a multi-level Gaussian map. First, we introduce some notation as shown in Fig. 3.4. Let $N_0 = \{N_0^{(0)}, \cdots, N_0^{(n)}\}$ denote the index set of all the atoms with $N_0^{(i)} = \{i\}$. Suppose N_0 is grouped into several subsets $N_1^{(i)}$, $i = 1, 2, \cdots, n_1$, such that

$$\bigcup_{i=1}^{n_1} N_1^{(i)} = N_0, \quad N_1^{(i)} \bigcap_{1 \le i \ne j \le n_1} N_1^{(j)} = \phi. \tag{1.5}$$

The set $N_1 := \{N_1^{(i)}\}_{i=1}^{n_1}$, whose elements are also sets, may be further grouped into some subsets $N_2^{(i)}$, $i = 1, 2, \cdots, n_2$, such that

$$\bigcup_{i=1}^{n_2} N_2^{(i)} = N_1, \quad N_2^{(i)} \bigcap_{1 \le i \ne j \le n_2} N_2^{(j)} = \phi. \tag{1.6}$$

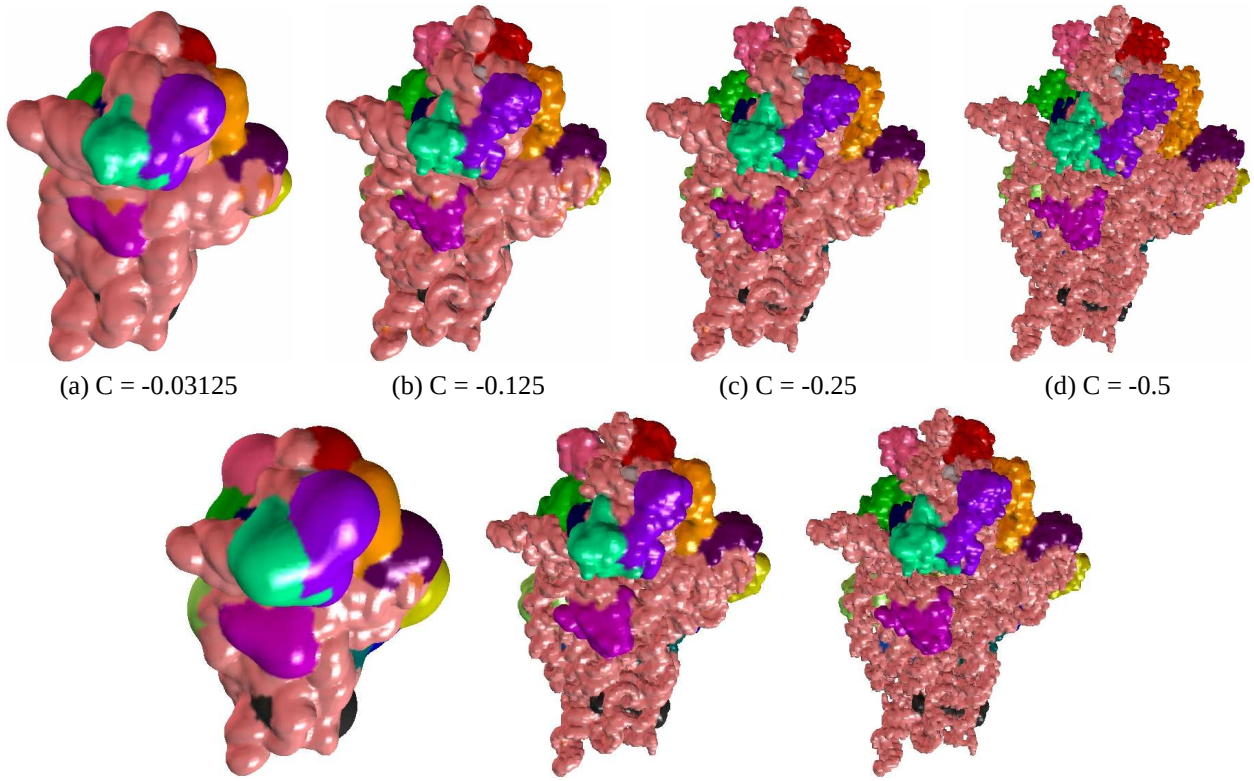
The collection of $\{N_2^{(i)}\}_{i=1}^{n_2}$ is denoted by N_2. This hierarchical grouping process could be repeated several times according to the molecular complex considered. In practice, two or three iterations suffice. By using these sets $N_k^{(i)}$ and a given sequence $\{p_k\}$ of p with $p_k > 0$, the k-level Gaussian map are defined recursively as

$$G_{N_k^{(i)}}(x) = \sum_{N \in N_k^{(i)}} [G_N(x)]^{p_k}, \quad N_k^{(i)} \in N_k,$$

where 0-level Gaussian map is defined by Eqn. 1.4 (C = 1.0) or

$$G_{N_0^{(i)}}(x) = K(\|x - x_i\|)/K(r_i), \qquad K(x) = e^{-x^2}.$$

The atom group format depends on what kind of structure we wish to model and mesh. For a protein, atoms may be grouped by residues, meaning that atoms in the same residue are classified into one group. Then the residues are grouped according to their neighborhood along the protein backbone.



(a) C = -0.03125 (b) C = -0.125 (c) C = -0.25 (d) C = -0.5

(e) B = -0.03125 (f) B = -0.125 (g) B = -0.5

Figure 3.3: Implicit solvation models of Thermus Thermophilus small Ribosome 30S (1J5E) crystal subunit for various Gaussian kernel parameters. The pink color shows 16S rRNA and the remaining colors are proteins.

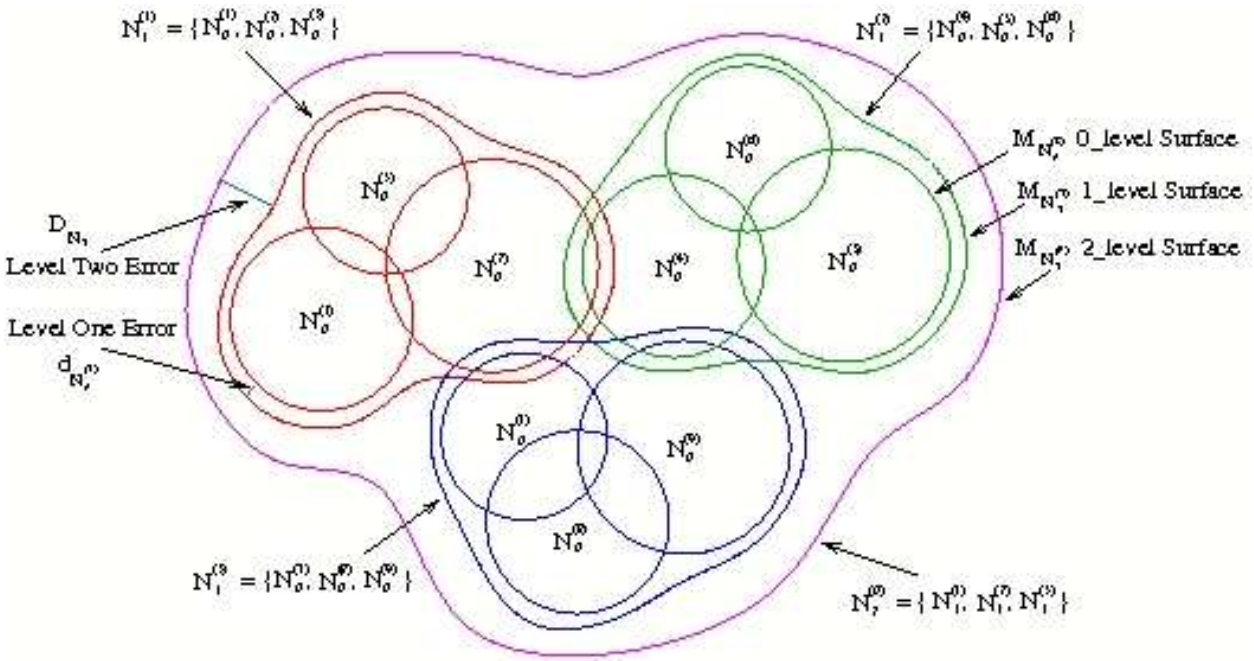
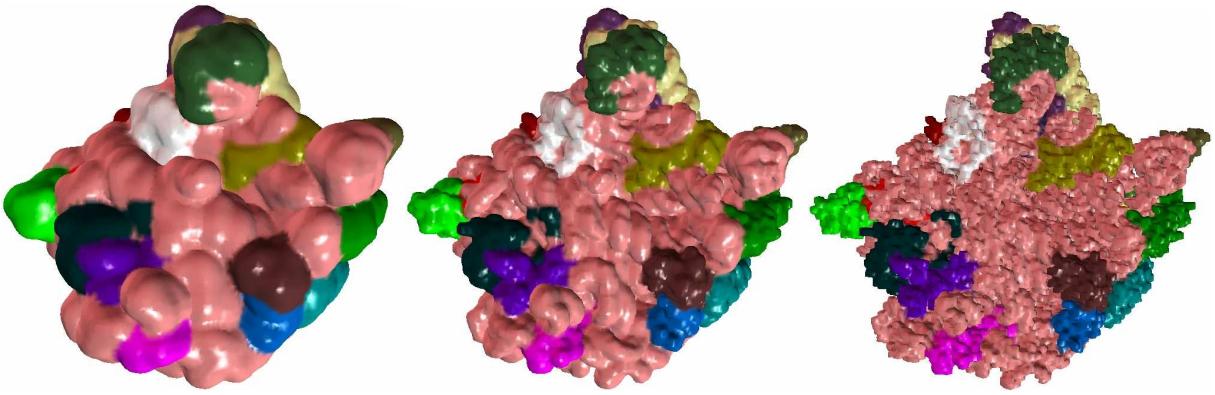


Figure 3.4: The definition of multi-level surfaces.



(a) low resolution (b) residue-level resolution (c) atomic level resolution

Figure 3.5: Implicit solvation models of Haloarcula Marismortui large Ribosome 50S (1JJ2) crystal subunit. (a) $p_1 = 0.03125$; (b) $p_1 = 0.125$; (c) $p_1 = 0.5$. $p_2 = 1.0$. The light yellow and the pink color show 5S and 23S rRNA respectively, the remaining colors are proteins.

For each k-level Gaussian Map $G_{N_k^{(i)}}(x)$, a k-level surface is defined by

$$M_{N_k^{(i)}} := \{x \in I\!R^3 : G_{N_k^{(i)}}(x) = 1\}.$$

This surface encloses the surface M_N for $N \in N_k^{(i)}$. Hence, all these $N_k^{(i)}$ define a hierarchical surface family. We call the surface M_N as the child of $M_{N_k^{(i)}}$, and $M_{N_k^{(i)}}$ the parent of M_N. The enclosing relation of this hierarchical surface family is strict, meaning that the minimal distance from M_N to $M_{N_k^{(i)}}$ is greater than zero for any $N \in N_k^{(i)}$. We further define the B-surface of M_N for all $N \in N_k^{(i)}$ as

$$S_{N_k^{(i)}} = \mathrm{Bd}\Big(\bigcup_{N \in N_k^{(i)}} \{x \in I\!R^3 : G_N(x) \le 1\}\Big),$$

where Bd() denotes the boundary of a region in $I\!R^3$. Note that $S_{N_k^{(i)}}$ is enclosed strictly by $M_{N_k^{(i)}}$.

The purpose of introducing a multi-level Gaussian map is to address the structure of molecules at a certain level. For instance, at the residue level of a protein, we dealt with each residue as one unit and therefore the protein is considered at the residue level resolution. The sub-structures of the residue (atoms), are not individually identifiable. Similarly, at the next higher level, a group of residues is dealt as one unit and therefore the protein is considered at an even coarser feature resolution. The goal of addressing certain level structure and un-addressing the higher level ones is achieved by the properly selection of the parameter p_k in the multi-level Gaussian map. Basically, larger p_k should be chosen to address the k-level structure and a smaller p_{k-1} is used to un-address the $(k-1)$-level structures.

Considering three levels of structures, including the atomic, the residue and the next level of grouping, we can construct a three level Gaussian map with given p_1, p_2 and p_3. To address the second level structure, we need to choose p_3 larger and p_2 smaller, while p_1 has less influence than the second level structure. Quite often it also suffices to consider only a two-level Gaussian map instead of three: level one is at the protein residue level, while level two is at a coarser resolution level. Henceforth in this paper, we provide details for only two-level Gaussian maps.

In computing implicit solvation molecular surfaces, various models are constructed by choosing different $p_1 \in (0,\infty)$ and $p_2 \in (0,\infty)$ in the Gaussian map. To make the constructed model correspond to a certain level, p_1 and p_2 need to be selected properly. For a fixed level, the structure at this level should be distinguishable. For instance, at the residue level, the individual residues should be observed, while atoms may not be distinguished clearly. Fig. 3.5 shows constructed models of Ribosome 50S at low resolution, residue and atomic level resolutions.

3.1.3 Approximation Computation

In order to obtain a good approximation to the molecular surface from the multi-level Gaussian map, we bound the error at each level. To bound the approximation for the first level, we need to compute the minimal distance from $M_N,\ N \in N_1^{(i)}$ to its parent surface $M_{N_1^{(i)}}$. On the other hand, in order to have an error controlled approximation of the second level surface, we need to compute the maximal error from $M_N,\ N \in N_2^{(i)}$ to its parent surface $M_{N_2^{(i)}}$. Hence, we need to consider the error computation for both levels of surfaces. The error computations are based on a point projection algorithm.

Given the surface M_N, a point $q \notin M_N$ and a unit direction n, the point projection algorithm in the following computes a nearby intersection point p of the line $q+tn$ ($t \in (-\infty,\infty)$) with the surface M_N.

Algorithm 3.1.3.1 (*Point Projection*).

1. *Compute an interval $[a,b]$ for t, on which $G_N(q+tn)-1$ changes sign once. This is achieved by a linear search step in a certain range $[A,B]$. If $(\nabla G_N(q))^T n[G_N(q)-1] < 0$, search in n direction, otherwise in $-n$ direction. If such an interval could not be found, the project point does not exist and return a failure signal. After the interval is determined, set $t_0 = \frac{a+b}{2}$ and $k = 0$.*

2. *Compute t_{k+1} by the Newton iteration method*

$$t_{k+1} = t_k - \frac{G_N(q+t_k n)}{n^T \nabla G_N(q+t_k n)}. \tag{1.7}$$

   If $t_{k+1} \notin (a,b)$, replace t_{k+1} by $\frac{a+b}{2}$.

3. *Replace the interval $[a,b]$ by $[a,t_{k+1}]$ if $G_N(q+tn)-1$ changes sign over $[a,t_{k+1}]$, and replace $[a,b]$ by $[t_{k+1},b]$ otherwise.*

4. *If $|b-a| < \varepsilon$ (ε is a given error tolerance, we usually take it to be 10^{-4}), stop the iteration and $p = q+t_{k+1}n$ is the projection point, otherwise, set $k = k+1$ and go back to step 2.*

We specify the searching range $[A,B]$ in step 1 of the algorithm to be $[-4,4]$, since the atom diameters are around 4. Errors beyond that are not considered here. If the projection exists, then the projection point p of point q on the surface M_N in the direction n is denoted by $P_{M_N}(q,n)$.

Minimal Error of Level One Surface

Now we assume $k = 1$, then the child surfaces are atoms. Let $N = \{j\} \in N_1^{(i)}$, the minimal error from $M_N = S_N$ to $M_{N_1^{(i)}}$ is defined by

$$d_N := \min_{p \in M_{N_1^{(i)},N}} \|p - x_j\| - r_j, \quad j \in N.$$

Let $q = x_j + r_j \frac{p-x_j}{\|p-x_j\|}$, then q is on the sphere S_N and p is the projection of q to the surface $M_{N_1^{(i)}}$ in the spherical normal direction $n(q)$. That is, $p = P_{M_{N_1^{(i)}}}(q,n(q))$. Hence in order to compute d_N, we need to compute $P_{M_{N_1^{(i)}}}(q,n(q))$ for $q \in S_N$.

Next we consider the computation of the minimal distance from M_N to $M_{N_1^{(i)}}$, where $N \in N_1^{(i)}$. First we assume that each atom (sphere) is uniformly sampled with m vertices. This sampling is achieved by translating a triangulated unit sphere to each of the atomic centers and re-scaling it to the atom's van der Waal radius. We obtain the unit sphere triangulation from [116]. For each vertex q on the triangulated atom surface M_N, $P_{M_{N_1^{(i)}}}(q,n(q))$ is computed using the *point projection* algorithm, where $n(q)$ is the spherical normal at q.

Algorithm 3.1.3.2 (*Minimal Error computation*).

Set $d_N = 4$. for each triangle vertex $q \in S_N \cap S_{N_0}$ do \{ compute $P_{M_{N_1}}(q,n(q))$, and then compute

$$\begin{aligned} &d_N = \min\{d_N, \|P_{M_{N_1}}(q,n(q)) - x_j\| - r_j\}, \\ &\text{if } P_{M_{N_1}}(q,n(q)) \in M_{N_1,N}. \end{aligned} \tag{1.8}$$

}

Table 2 shows the minimal error of our level one surface for a residue and a chain from Ribosome 30S, where $e(M)$ is defined as $e(M) := \max_{N \in N_1^{(i)}} d_N$. It can be observed that the error decreases as p increases. The algorithm for computing minimal error can also be used to compute the maximal error by changing the min to max in (1.8). Maximal errors for Ribosome 30s are also listed in Table 2 for different p_1 (see the second row).

Table 2: Minimal Error and Maximal Error of First Level Surfaces of Ribosome 30S (1J5E) (Angstrom)

p_1	0.25	0.5	1.0	2.0	4.0	8.0	16.0
Min Error (atomic)	8.338e-02	2.829e-03	6.287e-06	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$
Max Error (atomic)	1.634e+00	8.656e-01	4.121e-01	2.038e-01	8.893e-02	3.940e-02	1.842e-02

Maximal Error of Level Two Surface

The maximal error from M_N to $M_{N_2^{(i)}}$, $N \in N_2^{(i)}$ is defined as

$$d_N := \max_{q \in M_N, P_{M_{N_2^{(i)}}}(q,n) \in M_{N_2^{(i)},N}} \|q - P_{M_{N_2^{(i)}}}(q,n)\|,$$

where $q \in M_N$, $P_{M_{N_2^{(i)}}}(q,n)$ is the normal direction projection of q to the surface $M_{N_2^{(i)}}$. This error is computed as follows. Let $N_1 \in N_2^{(i)}$. **Algorithm 3.1.3.3** (*Maximal Error computation*).

Set $d_{N_1} = 0$.
for each $N \in N_1$ *do* {
  for each triangle vertex $q \in S_N \cap S_{N_0}$ *do* {
    compute $\tilde{q} := P_{M_{N_1}}(q, n(q))$, *and*
    compute $P_{M_{N_2^{(i)}}}(\tilde{q}, n(\tilde{q}))$ *if* $\tilde{q} \in M_{N_1,N}$
    and then compute
    $d_{N_1} = \max\{d_{N_1}, \|P_{M_{N_1}}(q,n(q)) - P_{M_{N_2^{(i)}}}(\tilde{q}, n(\tilde{q}))\|$
    if $P_{M_{N_2^{(i)}}}(\tilde{q}, n(\tilde{q})) \in M_{N_2^{(i)},N_1}$.
  }
}

Again, the projection points $\tilde{q} = P_{M_{N_1}}(q, n(q))$ and $P_{M_{N_2^{(i)}}}(\tilde{q}, n(\tilde{q}))$ are computed by the point projection algorithm, where the searching range $[A,B]$ is set to be $[0,4]$, since we know $M_{N_2^{(i)}}$ enclosing M_N and we are not interested in the errors that are larger than 4.

The first row of Table 3 shows the maximal errors of the second level (residue level) surfaces for ribosome 30s, where p_1 is chosen to be 0.5, $p_2 = 0.25, 0.5, 1.0, \cdots, 16$. The second row lists the maximal errors of the second level (low level) surfaces for the same p_1 and p_2. The results show that the errors decrease approximately as a linear rate as p_2 increases.

Table 3: Maximal Error of Second Level Surfaces of Ribosome 30S (1J5E) (Angstrom)

p_2	0.25	0.5	1.0	2.0	4.0	8.0	16.0
Max Error (residue)	3.923e+00	2.124e+00	6.832e-01	3.240e-01	1.550e-01	7.794e-02	3.278e-02
Max Error (low)	9.899e+00	7.695e+00	8.045e-01	2.365e-01	1.390e-01	6.113e-02	2.653e-02

3.1.4 Good Approximations of Molecular Surfaces

We have discussed that it is often sufficient to consider a two-level Gaussian map to approximate molecular surfaces. To address certain structures, p_1 is taken to be a small value to blur the higher level details, p_2 is chosen to be larger to enhance the feature

of the current level structure. As we have shown in the last section, a smaller p_1 leads to a larger error for the level one surface, and a larger p_2 leads to a smaller error for the second level surface. Therefore, our strategy for obtaining a tight enclosing surface approximation is to remove the level one error and ignore the error of the second level.

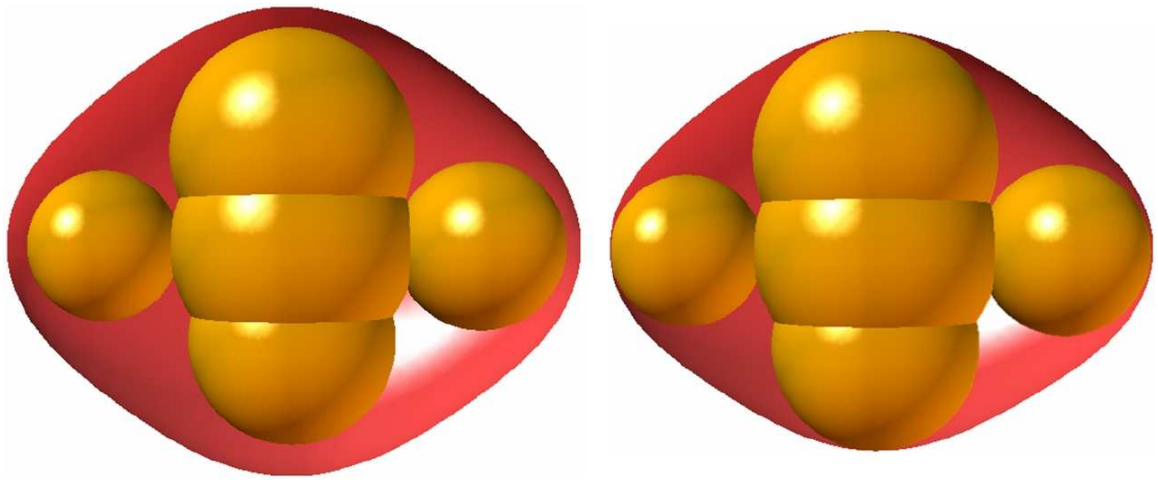


Figure 3.6: The left picture shows the inflation effect by the Gaussian map. The right one shows the tight enclosure of atoms. The centers of the five atoms are $(-2,0,0)$, $(2,0,0)$, $(0,-1,0)$, $(0,1,0)$ and $(0,0,0)$. The corresponding radii are 0.8, 0.9, 1.1, 1.3 and 1.3. The parameter p in the Gaussian map is chosen to be 0.4. The tight approximation on the right figure is obtained by shrinking the five radii into 0.55644, 0.72525, 0.60476, 1.04567 and 0.0 respectively. The unit is Angstrom.

The main idea to obtain a tight level one enclosing surface $M_{N_1^{(1)}}$ is to reduce the radii of the atoms, such that $M_{N_1^{(i)}}$ touches the original atoms (see Fig. 3.6). Suppose $y \in M_{N_1^{(i)}}$ is the nearest point to the j-th atom, $j \in N_1^{(i)}$, and the distance from y to the atom is d_j. Then we have

$$\sum_{l \in N_1^{(i)}, l \neq j} [K(\|y - x_l\|)/K(r_l)]^{p_1} + [K(\|y - x_j\|)/K(r_j)]^{p_1} = 1. \tag{1.9}$$

where $K(x) = e^{-x^2}$. Now we adjust the radius r_j to $\tilde{r}_j$, such that the new nearest point y is on the j-th sphere. Since the dominating part of (1.9) is the second term of the left hand side, we therefore require $\tilde{r}_j$ satisfying

$$0 \leq \tilde{r}_j \leq r_j, \tag{1.10}$$

$$K(r_j + d_j)/K(r_j) = K(r_j)/K(\tilde{r}_j). \tag{1.11}$$

From this we obtain

$$\tilde{r}_j = \begin{cases} K^{-1}\left[\frac{K(r_j)^2}{K(r_j+d_j)}\right], & \text{if } \frac{K(r_j)^2}{K(r_j+d_j)} \in \text{Range}(K), \\ 0, & \text{otherwise,} \end{cases}$$

where K^{-1} denotes the inverse function of $K(x)$, $\text{Range}(K) := \{y \in I\!R : y = K(x),\ x \in (0,\infty)\}$. Based on this analysis, we build the following iterative algorithm for computing $\tilde{r}_j$.

Algorithm 3.1.4.1 (*Sphere Shrinking*).

For $i = 1, 2, \cdots, n_1$ do the following steps:

1. *Set $l = 0$, $r_j^{(l)} = r_j$, $d_j^{(l)} = \infty$, $\forall j \in N_1^{(i)}$.*
2. *Compute the minimal distance $d_j^{(l+1)}$, $\forall j \in N_1^{(i)}$ from the j-th atom to the iso-surface defined by the multi-level Gaussian map $G^{(l)}_{N_1^{(i)}}(x) = \sum_{j \in N_1^{(i)}} [K(\|x - x_j\|)/K(r_j^{(l)})]^{p_1}$, using Algorithm 4.2.*

3. *Compute*

$$r_j^{(l+1)} = \begin{cases} K^{-1}\left[\frac{K(r_j)K(r_j^{(l)})}{K(r_j+d_j^{(l)})}\right], & \text{if } \frac{K(r_j)K(r_j^{(l)})}{K(r_j+d_j^{(l)})} \in \text{Range}(K), \\ 0, & \text{otherwise.} \end{cases}$$

4. *If* $\max_{j\in N_1^{(i)}} |d_j^{(l)} - d_j^{(l+1)}| < \varepsilon$ *(we take* $\varepsilon = 10^{-4}$*), terminate the* l *loop and* $r_j^{(l+1)}$ *are the required results. Otherwise, set* $l = l+1$ *and go back to step 2.*

Remark: The condition $\frac{K(r_j)K(r_j^{(l)})}{K(r_j+d_j^{(l)})} \in \text{Range}(K)$ may lead to some of the atoms located in the interior of the molecule to become untouchable. Figure 3.6 shows that the circle at the origin is not touched.

The experiments show the sphere shrinking algorithm converges at a linear rate. Table 4 lists the error $e_{max}^{(l)} = \max_{j\in N_1^{(i)}} |d_j^{(l)}|$ for 20 amino acids with $p_1 = 0.4$.

Table 4: Errors $e_{max}^{(l)}$ for 20 amino acids and $p_1 = 0.4$

l	ALA	ARG	ASN	ASP	CYS	GLN	GLU	GLY	HSD	ILE
0	5.13e-01	6.97e-01	5.99e-01	6.23e-01	5.36e-01	6.26e-01	7.06e-01	4.34e-01	7.36e-01	6.00e-01
2	6.22e-02	1.37e-01	2.66e-01	6.75e-02	5.86e-02	1.16e-01	7.78e-02	5.33e-02	7.20e-02	5.62e-02
4	2.80e-03	3.79e-02	5.83e-02	1.50e-03	6.82e-04	1.76e-03	4.57e-04	1.90e-02	1.45e-02	2.73e-03
6	5.76e-04	2.30e-02	1.83e-04	4.93e-04	1.81e-04	4.51e-04	1.38e-04	8.62e-05	5.30e-03	5.60e-04
8	1.30e-04	6.95e-04	6.06e-05	1.64e-04	4.97e-05	1.74e-04	4.26e-05	6.31e-06	2.20e-03	1.25e-04
10	3.14e-05	2.18e-04	2.22e-05	5.59e-05	1.39e-05	7.84e-05	1.32e-05	7.16e-07	9.94e-04	3.11e-05

l	LEU	LYS	MET	PHE	PRO	SER	THR	TRP	TYR	VAL
0	8.48e-01	8.62e-01	6.08e-01	6.14e-01	7.98e-01	9.63e-01	1.06e-00	6.01e-01	6.10e-01	7.07e-01
2	6.51e-02	3.96e-01	1.13e-01	8.94e-02	2.06e-03	8.81e-02	3.06e-02	9.17e-02	6.03e-02	2.86e-02
4	5.72e-03	1.54e-03	7.78e-03	6.50e-03	3.62e-04	5.28e-04	6.63e-03	1.49e-02	4.25e-02	5.76e-03
6	1.27e-03	5.18e-04	2.25e-03	1.90e-03	9.12e-05	1.19e-04	1.68e-03	6.42e-03	1.69e-03	1.36e-03
8	3.03e-04	1.77e-04	6.77e-04	7.13e-04	2.35e-05	2.66e-05	4.67e-04	2.90e-03	6.93e-04	3.56e-04
10	7.52e-05	6.23e-05	2.09e-04	3.02e-04	6.26e-06	5.88e-06	1.36e-04	1.56e-03	3.52e-04	9.78e-05

Fig. 3.7 shows multi-resolution implicit solvation surface approximations of an ASN-THR-TYR peptide with various p_1 and p_2. Fig. 3.7(a) shows an atomic level model, Fig. 3.7(a∼g) are residue level models. It can be observed that when the same p_1 is selected, smaller p_2 leads to fatter surfaces. Compared with Fig. 3.7(g), Fig. 3.7(f) is more tight.

Fig. 3.8 shows multi-resolution implicit solvation surface approximation of Ribosome 30S. Fig. 3.8(a) is a low level model, the pink color shows 16S rRNA and the remaining colors are proteins. One protein (Chain B) is separated from the whole structure. The residue level model can be constructed by selecting small p_1 and large p_2 as shown in Fig. 3.8(b), and the atomic level model is constructed by selecting large p_1 and small p_2 as shown in Fig. 3.8(c).

3.2 Mixed-Voronoi-Del complexes (C^1)

3.3 Algebraic Shell Splines (C^1)

3.3.1 Algorithm Sketch

There are four main steps in our algebraic spline molecular surface (ASMS) construction algorithm: (1) construct an initial triangular mesh of the SES; (2) build a prism scaffold surrounding the triangulation; (3) define a piecewise polynomial with certain continuity; (4) extract the 0-contour of the piecewise polynomial. We are going the explain each step in detail in the following and discuss how to make use the parametrization of the ASMS in the numerical integration.

3.3.2 Initial triangulation of the MS

So far a lot of work has been done on the triangulation of the SES or its approximation [35][2][67][122][17]. The ASMS generation could be applied to any of these triangulations. In our current research we use the triangulation generated by a

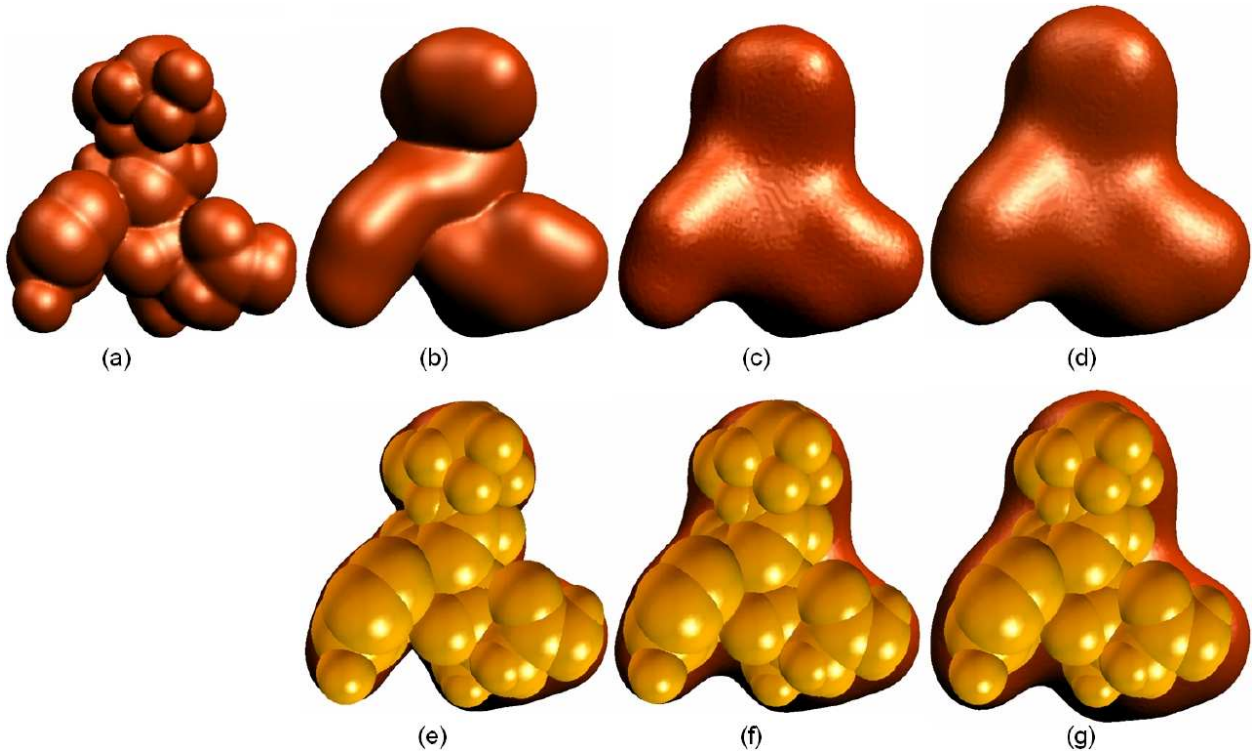


Figure 3.7: Different effects of changing p_2 and tight/non-tight approximations for an ASN-THR-TYR peptide which consists of 49 atoms. The surface (b), (c) and (d) are the same as outer surfaces of (e), (f) and (g) respectively. The inner surface of (e), (f) and (g) is the hard sphere model of three residues. (a) shows the atomic level approximation of the hard sphere model, where $p_1 = 5.0$, $p_2 = 1.0$; (b), (e), (c) and (f) show the tight approximation of the residue level with $p_1 = 0.4$. But different p_2 are used. We choose $p_2 = 2.0$ for (b) $p_2 = 0.5$ for (c). It could be observed that larger p_2 leads to closer approximation. (d) and (g) show non-tight approximations using the same p_1 and p_2 as (c) and (f). Comparing with (f), even larger error is observed in (g).

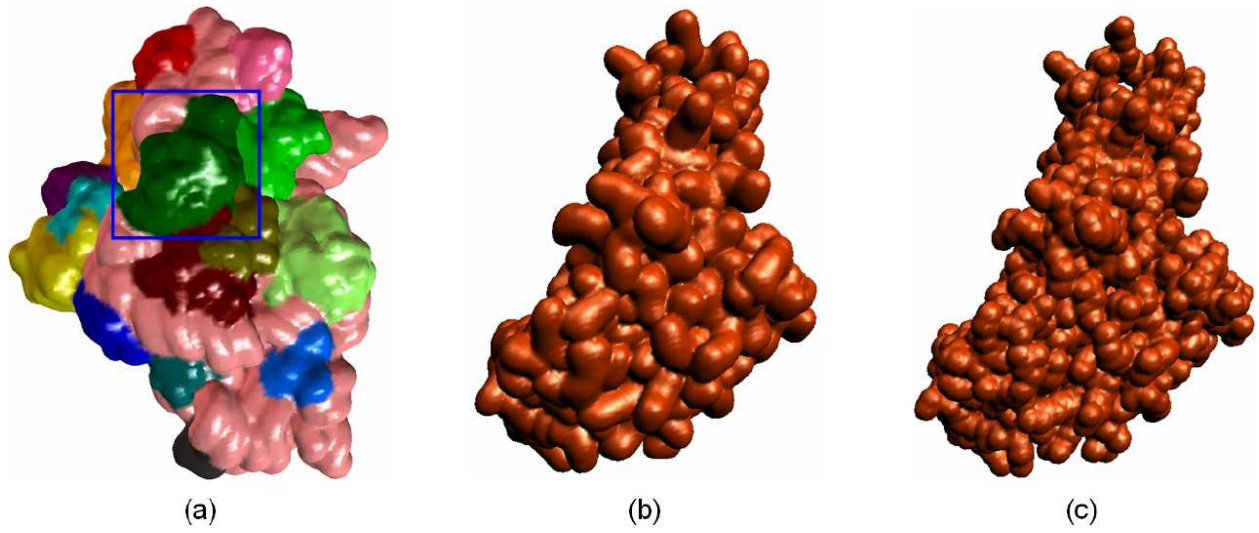


Figure 3.8: Multi-resolution models of Ribosome 30S. (a) - Ribosome 30S at the low level with $p_1 = 0.0625$, $p_2 = 1.0$ in multi-level Gaussian map. Ribosome 30S contains 22 chains and each of them is painted in a different color. The pink color shows 16S rRNA and the remaining colors are proteins. The blue box shows one protein (Chain B). (b) - Chain B at the residue level with $p_1 = 0.4$, $p_2 = 5.0$. It consists of 234 residues. (c) - Chain B at the atomic level with $p_1 = 5.0$, $p_2 = 1.0$. It consists of 1900 atoms.

program in the software TexMol [17][7] as the initial. Features of the molecular surface are well preserved in this triangulation. We then decimate the mesh [21] to obtain a coarser one.

3.3.3 Implicit/parametric patches generation

Given the triangulation mesh T, let $[\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k]$ be one of the triangles where $\mathbf{v}_i,\ \mathbf{v}_j,\ \mathbf{v}_k$ are the vertices of the triangle. Suppose the unit normals of the surface at the vertices are also known, denoted as $\mathbf{n}_l,\ (l = 1, j, k)$. Let $\mathbf{v}_l(\lambda) = \mathbf{v}_l + \lambda\mathbf{n}_l$. First we define a prism (Figure 3.9) $D_{ijk} := \{\mathbf{p} :\ \mathbf{p} = b_1\mathbf{v}_i(\lambda) + b_2\mathbf{v}_j(\lambda) + b_3\mathbf{v}_k(\lambda),\ \lambda \in I_{ijk}\}$, where (b_1, b_2, b_3) are the barycentric coordinates of points in $[\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k]$, and I_{ijk} is a maximal open interval containing 0 and for any $\lambda \in I_{ijk}$, $\mathbf{v}_i(\lambda)$, $\mathbf{v}_j(\lambda)$, $\mathbf{v}_k(\lambda)$ are not collinear and $\mathbf{n}_i$, $\mathbf{n}_j$, $\mathbf{n}_k$ point to the same side of the plane $P_{ijk}(\lambda) := \{\mathbf{p} :\ \mathbf{p} = b_1\mathbf{v}_i(\lambda) + b_2\mathbf{v}_j(\lambda) + b_3\mathbf{v}_k(\lambda)\}$.

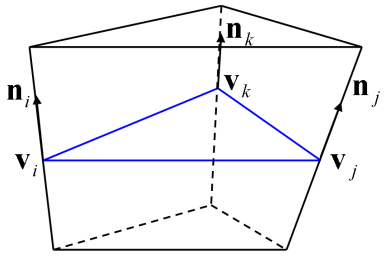


Figure 3.9: A prism D_{ijk} constructed based on the triangle $[\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k]$.

Next we define a function in the Benstein-Bezier (BB) basis over the prism D_{ijk}:

$$F(b_1, b_2, b_3, \lambda) = \sum_{i+j+k=n} b_{ijk}(\lambda) B^n_{ijk}(b_1, b_2, b_3), \tag{3.12}$$

where $B^n_{ijk}(b_1, b_2, b_3)$ is the Bezier basis

$$B^n_{ijk}(b_1, b_2, b_3) = \frac{n!}{i!\,j!k!} b_1^i b_2^j b_3^k.$$

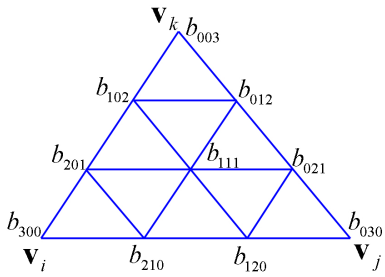


Figure 3.10: The control coefficients of the cubic Bezier basis of function F.

We approximate the molecular surface by the zero contour of F, denoted as S. In order to make S smooth, the degree of the Bezier basis n should be no less than 3. For simplicity, here we consider the case of $n = 3$. The control coefficients $b_{ijk}(\lambda)$ should be properly defined such that S is continuous. In Figure 3.10 we show the relationship of the control coefficients and the points of the triangle when $n = 3$. Next we are going to discuss these coefficients are defined.

Since S passes through the vertices $\mathbf{v}_i,\ \mathbf{v}_j,\ \mathbf{v}_k$, we define

$$b_{300} = b_{030} = b_{003} = \lambda. \tag{3.13}$$

To obtain C^1 continuity at the vertices, we require $b_{210} - b_{300} = \frac{1}{3}\nabla F(v_i) \cdot (\mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda))$, where $\nabla F(v_i) = \mathbf{n}_i$. Therefore

$$b_{210} = \lambda + \frac{1}{3}\mathbf{n}_i \cdot (\mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda)). \tag{3.14}$$

b_{120}, b_{201}, b_{102}, b_{021}, b_{012} are defined similarly.
To obtain the C^1 continuity at the midpoints of the edges of T, we define b_{111} by using the side-vertex scheme [85]:

$$b_{111} = w_1 b_{111}^{(1)} + w_2 b_{111}^{(2)} + w_3 b_{111}^{(3)}, \tag{3.15}$$

where

$$w_i = \frac{b_j^2 b_k^2}{b_2^2 b_3^2 + b_1^2 b_3^2 + b_1^2 b_2^2}, \qquad i = 1,2,3,\ i \neq j \neq k.$$

.

Next we are going to define $b_{111}^{(1)}$, $b_{111}^{(2)}$ and $b_{111}^{(3)}$ such that the C^1 continuity is obtained at the midpoint of the edge $\mathbf{v}_j\mathbf{v}_k$, $\mathbf{v}_i\mathbf{v}_k$ and $\mathbf{v}_i\mathbf{v}_j$. Consider the edge $\mathbf{v}_i\mathbf{v}_j$ for instant. Recall that any point $\mathbf{p} = (x,y,z)$ in D_{ijk} can be represented by

$$(x,y,z)^T = b_1\mathbf{v}_i(\lambda) + b_2\mathbf{v}_j(\lambda) + b_3\mathbf{v}_k(\lambda). \tag{3.16}$$

Therefore differentiating both sides of (3.16) with respect to x, y and z, respectively, yields

$$I_3 = \begin{pmatrix} \frac{\partial b_1}{\partial x} & \frac{\partial b_2}{\partial x} & \frac{\partial \lambda}{\partial x} \\ \frac{\partial b_1}{\partial y} & \frac{\partial b_2}{\partial y} & \frac{\partial \lambda}{\partial y} \\ \frac{\partial b_1}{\partial z} & \frac{\partial b_2}{\partial z} & \frac{\partial \lambda}{\partial z} \end{pmatrix} \begin{pmatrix} (\mathbf{v}_i(\lambda) - \mathbf{v}_k(\lambda))^T \\ (\mathbf{v}_j(\lambda) - \mathbf{v}_k(\lambda))^T \\ (b_1\mathbf{n}_i + b_2\mathbf{n}_j + b_3\mathbf{n}_k)^T \end{pmatrix}, \tag{3.17}$$

where I_3 is a 3×3 unit matrix. Denote

$$T := \begin{pmatrix} (\mathbf{v}_i(\lambda) - \mathbf{v}_k(\lambda))^T \\ (\mathbf{v}_j(\lambda) - \mathbf{v}_k(\lambda))^T \\ (b_1\mathbf{n}_i + b_2\mathbf{n}_j + b_3\mathbf{n}_k)^T \end{pmatrix}, \tag{3.18}$$

and let $A = \mathbf{v}_i(\lambda) - \mathbf{v}_k(\lambda)$, $B = \mathbf{v}_j(\lambda) - \mathbf{v}_k(\lambda)$ and $C = b_1\mathbf{n}_i + b_2\mathbf{n}_j + b_3\mathbf{n}_k$, then $T = (A\,B\,C)^T$.
From (3.17) we have

$$\begin{pmatrix} \frac{\partial b_1}{\partial x} & \frac{\partial b_2}{\partial x} & \frac{\partial \lambda}{\partial x} \\ \frac{\partial b_1}{\partial y} & \frac{\partial b_2}{\partial y} & \frac{\partial \lambda}{\partial y} \\ \frac{\partial b_1}{\partial z} & \frac{\partial b_2}{\partial z} & \frac{\partial \lambda}{\partial z} \end{pmatrix} = T^{-1} = \frac{1}{\det(T)}\,(B \times C,\ C \times A,\ A \times B). \tag{3.19}$$

According to (3.12), at the midpoint of $\mathbf{v}_i\mathbf{v}_j$, $(b_1,b_2,b_3) = (\frac{1}{2},\frac{1}{2},0)$, we have

$$\begin{pmatrix} \frac{\partial F}{\partial b_1} \\ \frac{\partial F}{\partial b_2} \\ \frac{\partial F}{\partial \lambda} \end{pmatrix} = \begin{pmatrix} (\mathbf{v}_i(\lambda) - \mathbf{v}_k(\lambda))^T \\ (\mathbf{v}_j(\lambda) - \mathbf{v}_k(\lambda))^T \\ (\mathbf{n}_i + \mathbf{n}_j)^T/2 \end{pmatrix} \left(\frac{\mathbf{n}_i + \mathbf{n}_j}{4}\right) + \begin{pmatrix} \frac{3}{2}(b_{210} - b_{111}) \\ \frac{3}{2}(b_{120} - b_{111}) \\ \frac{1}{2} \end{pmatrix}.$$

By (3.15), at $(b_1,b_2,b_3) = (\frac{1}{2},\frac{1}{2},0)$ we have $b_{111} = b_{111}^{(3)}$. Therefore the gradient at $(\frac{1}{2},\frac{1}{2},0)$ is

$$\begin{aligned} \nabla F &= T^{-1}\left(\frac{\partial F}{\partial b_1}, \frac{\partial F}{\partial b_2}, \frac{\partial F}{\partial \lambda}\right)^T \\ &= \frac{\mathbf{n}_i + \mathbf{n}_j}{4} + \frac{1}{2\det(T)}[3(b_{210} - b_{111}^{(3)})B \times C + 3(b_{120} - b_{111}^{(3)})C \times A + A \times B] \end{aligned} \tag{3.20}$$

Define vectors

$$\begin{aligned} &\mathbf{d}_1(\lambda) = \mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda) = B - A, \\ &\mathbf{d}_2(b_1,b_2,b_3) = b_1\mathbf{n}_i + b_2\mathbf{n}_j + b_3\mathbf{n}_k = C, \\ &\mathbf{d}_3(b_1,b_2,b_3,\lambda) = \mathbf{d}_1 \times \mathbf{d}_2 = B \times C + C \times A. \end{aligned} \tag{3.21}$$

Let

$$\mathbf{c} = C(\frac{1}{2}, \frac{1}{2}, 0), \tag{3.22}$$

$$\mathbf{d}_3(\lambda) = \mathbf{d}_3(\frac{1}{2}, \frac{1}{2}, 0, \lambda) = B \times \mathbf{c} + \mathbf{c} \times A. \tag{3.23}$$

Let $\nabla F = \nabla F(\frac{1}{2}, \frac{1}{2}, 0)$. In order to have C^1 continuity at $(\frac{1}{2}, \frac{1}{2}, 0)$, we should have $\nabla F \cdot \mathbf{d}_3(\lambda) = 0$. Therefore, by (3.20) and (3.23), we have

$$b_{111}^{(3)} = \frac{\mathbf{d}_3(\lambda)^T (3b_{210} B \times \mathbf{c} + 3b_{120} \mathbf{c} \times A + A \times B)}{3||\mathbf{d}_3(\lambda)||^2}. \tag{3.24}$$

Similarly, we may define $b_{111}^{(1)}$ and $b_{111}^{(2)}$.

Now the function $F(b_1, b_2, b_3, \lambda)$ is well defined. The next step is to extract the zero level set S. Given the barycentric coordinates (b_1, b_2, b_3) of a point in the triangle $[\mathbf{v}_i \mathbf{v}_j \mathbf{v}_k]$, we find the corresponding λ by solving the equation $F(b_1, b_2, b_3, \lambda) = 0$ for λ and this could be done by the Newton's method. Then we may get the corresponding point on S as

$$(x,\ y,\ z)^T = b_1 \mathbf{v}_i(\lambda) + b_2 \mathbf{v}_j(\lambda) + b_3 \mathbf{v}_k(\lambda). \tag{3.25}$$

3.3.4 Smoothness

Theorem 3.3.1. *The ASMS S is C^1 at the vertices of T and the midpoints of the edges of T.*

Theorem 3.3.2. *S is C^1 everywhere if every edge $\mathbf{v}_i \mathbf{v}_j$ of T satisfies $\mathbf{n}_i \cdot (\mathbf{v}_i - \mathbf{v}_j) = \mathbf{n}_j \cdot (\mathbf{v}_j - \mathbf{v}_i)$.*

Theorem 3.3.3. *S is C^1 everywhere if the unit normals at the vertices of T are the same.*

Proofs of the theorems are shown in [124].

3.3.5 Parametrization and quadrature

In this section, we would like to show how the ASMS is applied to the computation of (6.13). Since we use the ASMS to represent the molecular surface, now $\Gamma = S$. Let $f = \frac{(\mathbf{r} - \mathbf{x}_i) \cdot \mathbf{n}(\mathbf{r})}{|\mathbf{r} - \mathbf{x}_i|^4}$. We decompose the entire surface S into patches $\{S_j\}$ with S_j being the AMSM generated over triangle j, then we have

$$\int_S f(\mathbf{x})\ dS = \sum_j \int_{S_j} f(\mathbf{x})\ dS. \tag{3.26}$$

For any point $\mathbf{x} = (x,y,z)$ on S_j, by the inverse map of (3.25), one can uniquely map $\mathbf{x}$ to a point in triangle j and get its baricentric coordinates (b_1, b_2, b_3) with $b_3 = 1 - b_1 - b_2$. Therefore, x, y, z can be represented in terms of (b_1, b_2):

$$x = x(b_1, b_2,), \qquad y = y(b_1, b_2), \qquad z = z(b_1, b_2)$$

Replacing (x,y,z) with (b_1, b_1, b_3) in (3.26) and letting

$$g(b_1, b_2) = f(x(b_1, b_2), y(b_1, b_2), z(b_1, b_2)),$$

we get

$$\int_{S_j} f(\mathbf{x})\ dS = \int_{\sigma_j} g(b_1, b_2) \sqrt{EG - F^2}\ db_1 db_2, \tag{3.27}$$

where

$$E = (\frac{\partial x}{\partial b_1})^2 + (\frac{\partial y}{\partial b_1})^2 + (\frac{\partial z}{\partial b_1})^2,$$
$$F = \frac{\partial x}{\partial b_1}\frac{\partial x}{\partial b_2} + \frac{\partial y}{\partial b_1}\frac{\partial y}{\partial b_2} + \frac{\partial z}{\partial b_1}\frac{\partial z}{\partial b_2},$$
$$G = (\frac{\partial x}{\partial b_2})^2 + (\frac{\partial y}{\partial b_2})^2 + (\frac{\partial z}{\partial b_2})^2.$$

We then apply the Gaussian quadrature to (3.27):

$$\int_{\sigma_i} g(b_1,b_2)\sqrt{EG-F^2}\,db_1db_2 \approx \sum_{k=1}^{n} W_k g(b_1^k,b_2^k)\sqrt{EG-F^2}|_{b_1^k,b_2^k}, \tag{3.28}$$

where (b_1^k,b_2^k,b_3^k) and W_k are the Gaussian integration nodes and weights on the triangles.

3.3.6 Error of the ASMS model

In order to show the error of S to the true surface S_0, we do a test on some typical surfaces (Table 3.1) $S_0 := \{(x,y,z) : z = f(x,y), (x,y) \in [0,1]^2\}$ which are considered as the true surfaces. We generate a triangulation mesh over the true surface with the maximum edge length h being 0.1. Based on the mesh, we construct the ASMS model S. The error of S to S_0 is defined as $\max \frac{||\mathbf{p}-\mathbf{q}||}{||\mathbf{q}||}$, where $\mathbf{p} \in S$, $\mathbf{q} \in S_0$, and $\mathbf{p}$ and $\mathbf{q}$ have the same (b_1,b_2,b_3) volume coordinates but different λ coordinates. We sample $(\mathbf{p},\ \mathbf{q})$ on the surfaces and compute the maximum relative error.

Table 3.1: Relative error and Convergence

Function $(x,y) \in [0,1]^2$	$\max\{\frac{\|\mathbf{p}-\mathbf{q}\|}{\|\mathbf{q}\|}\}$	C
$z = 0$	0	0
$z = x^2 + y^2$	2.450030e-05	1.010636e-2
$z = x^3 + y^3$	1.063699e-04	2.610113e-2
$z = e^{-\frac{1}{4}[(x-0.5)^2+(y-0.5)^2]}$	5.286856e-07	6.288604e-5
$z = 1.25 + \frac{\cos(5.4y)}{6+6(3x-1)^2}$	2.555683e-04	4.58608e-2
$z = \tanh(9y-9x)$	1.196519e-02	1.896754e-1
$z = \sqrt{1-x^2-y^2}$	8.614969e-05	1.744051e-1 (h^4)
$z = [(2-\sqrt{1-y^2})^2 - x^2]^{1/2}$	1.418242e-05	1.748754e-02

For the point pair $\mathbf{p}(b_1,b_2,b_3,\lambda_p)$ and $\mathbf{q}(b_1,b_2,b_3,\lambda_q)$ defined above, we prove that their Euclidean distance is bounded by the difference of their λ coordinates.

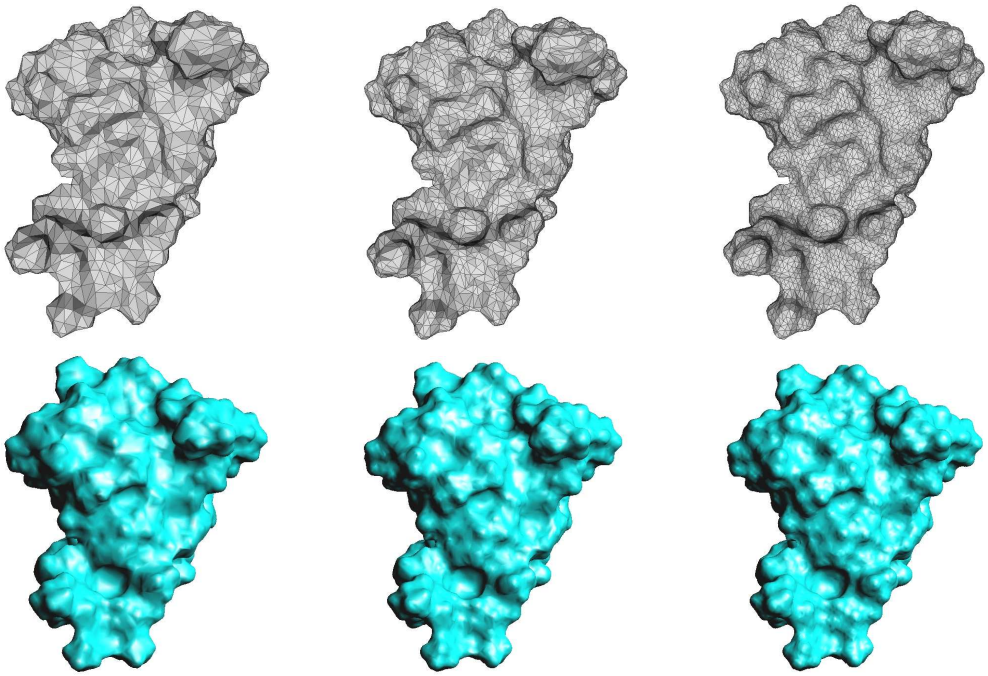
Lemma 3.3.1. *The error of the approximation point* $\mathbf{p}$ *to the true point* $\mathbf{q}$ *is bounded by* $|\lambda_p - \lambda_q|$.
PROOF.

$$\begin{aligned}||\mathbf{p}-\mathbf{q}|| &\le b_1||\mathbf{v}_i(\lambda_p) - \mathbf{v}_i(\lambda_q)|| + b_2||\mathbf{v}_j(\lambda_p) - \mathbf{v}_j(\lambda_q)|| + b_3||\mathbf{v}_k(\lambda_p) - \mathbf{v}_k(\lambda_q)|| \\ &\le |\lambda_p - \lambda_q|(b_1||\mathbf{n}_i|| + b_2||\mathbf{n}_j|| + b_3||\mathbf{n}_k||) \\ &= |\lambda_p - \lambda_q|\end{aligned}$$

□

To study the rate of converges of S to S_0, we gradually refine the initial mesh. Since the error is bounded by $|\lambda_p - \lambda_q|$, we compute the ratio of the maximum difference of λ_p and λ_q to h, h^2, h^3, and so forth. As h decreases, we check if the ratio converges or not, which allows us to know the highest rate of convergence of S to S_0. For most of the test functions in Table 3.1, we observe that S converges to S_0 as fast as $O(h^3)$. We also observe that for the case $z = \sqrt{1-x^2-y^2}$, the rate of convergence reaches $O(h^4)$. We show the limit of the ratio $\frac{|\lambda-\lambda'|}{h^3}$ as $h \downarrow 0$, denoted as C, in Table 3.1. Hence we draw the following claim:

Claim: Let h be the maximum side length of triangulation mesh T, $\mathbf{p}$ be the point on the ASMS, $\mathbf{q}$ be the corresponding point on the true surface, then $\mathbf{p}$ converges to $\mathbf{q}$ at the rate of $O(h^3)$. i.e. There exists a constant C such that $||\mathbf{p}-\mathbf{q}|| \leq Ch^3$.



4600 Triangles 9216 Triangles 18434 Triangles

Figure 3.11: The top row is the triangulation of the SES of protein 1ML0 with different number of triangles. The bottom row is the ASMS generated from the above corresponding triangulation.

We generated the ASMS for the real proteins based on different size of meshes (Figure 3.11) and show the error of the ASMS to the SES of three proteins: 1GCQ (843 atoms), 1ML0 (1051 atoms), and 1KKL (1276 atoms) in Table 3.2. Here the SES is modeled as a level set of the summation of fast decaying Gaussian functions. The ASMS is generated from the triangulation of the SES at different resolution. The number of triangles of the initial meshes are listed in Table 3.2. The error ε_{max} is defined as the one-way Hausdorff distance from the ASMS to the SES: $\varepsilon_{max} = \max_{\mathbf{p}\in \text{ASMS}} \min_{\mathbf{q}\in \text{SES}} ||\mathbf{p}-\mathbf{q}||$. As we see in the table, the errors are small and decrease rapidly as the initial triangulation becomes dense.

3.3.7 Application to the biomolecular energetic computation

We apply the ASMS model to the GB electrostatic solvation energy computations of the example proteins 1HIA (693 atoms), 1CGI (852 atoms), and 1PPE (436 atoms). The ASMS models S for the proteins are generated based on the initial mesh with different number of triangles (Table 3.3). We show the ASMS of the example molecules generated from the decimated triangulations in Figure 6.4 and Figure 3.13. As a comparison, we compute the polarization energy G_{pol} for both the ASMS and the piecewise linear (PL) surfaces and show the energy results and the timing in Table 3.3. For all the computations, a 4-point Gaussian quadrature rule over a triangle [43] is used for the numerical integration in (3.28) when computing the Born radii. The running time contains the time cost of computing the integration nodes over the surfaces, computing the Born radii,

Table 3.2: Error of ASMS to the SES

1GCQ		1ML0		1KKL	
No. of Δs	ε_{max}	No. of Δs	ε_{max}	No. of Δs	ε_{max}
16,312	0.266069	18,400	0.233949	19,968	0.260418
32,624	0.142149	36,864	0.142380	39,544	0.134689
65,456	0.082550	73,736	0.083895	79,096	0.085855

and evaluating G_{pol}. If we consider the energy computed from the dense mesh as accurate, as we see from the table, the G_{pol} computed from the coarse PL model has a large error, however for the coarse ASMS model, it is very close to the dense mesh result but with less time. On the other hand, to get a energy result of the same accuracy, fewer number of triangles are needed for the ASMS model than the PL model. For example, for the protein 1CGI, the G_{pol} computed from the ASMS with 3674 triangles is -1394.227 kcal/mol. However to get a similar result, 8712 triangles are needed for the piecewise linear model. Therefore the ASMS model is much more efficient in the energetic computation than trivial piecewise linear models.

Table 3.3: electrostatic solvation energy and timing

Protein ID	No. of Triangles	G_{pol} (kcal/mol)		Timing (s)	
		PL	AS	PL	AS
1CGI	29108	-1371.741894	-1343.1496	39.64	40.31
	8712	-1399.194841	-1346.2230	12.94	12.64
	3674	-1678.444735	-1394.2270	7.40	6.11
1HIA	27480	-1361.226603	-1340.6384	30.23	31.18
	7770	-1389.017538	-1347.8067	9.43	9.93
	3510	-1571.890827	-1388.4665	5.21	5.21
1PPE	24244	-835.563905	-825.3252	17.27	18.26
	6004	-852.713039	-828.2158	5.09	5.39
	2748	-933.956234	-845.5085	2.74	3.27

3.4 Variational B-spline Surfaces (C^2)

3.4.1 Geometric Flow for Molecular Surface Construction

Given a non-negative function $f(\mathbf{x})$ over a domain $\Omega \in \mathbb{R}^3$, we generate a molecular surface Γ in Ω, such that the energy functional

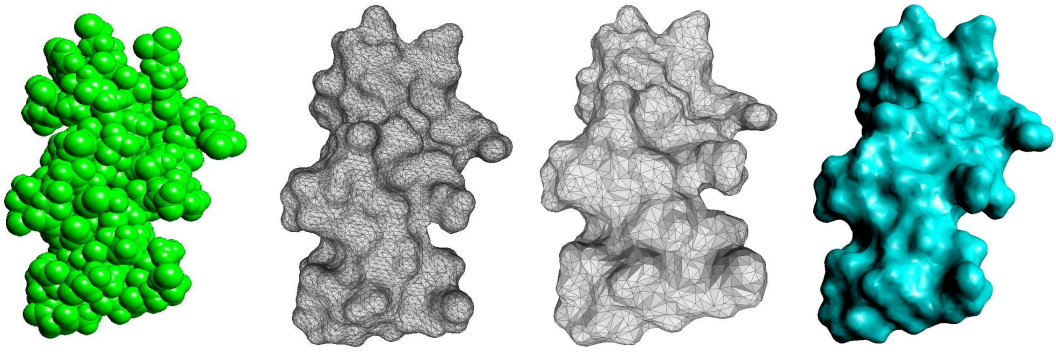
$$E(\Gamma) = \int_\Gamma f(\mathbf{x})d\mathbf{x} + \lambda \int_\Gamma h(\mathbf{x},\mathbf{n})d\mathbf{x} \tag{4.29}$$

is minimal, where $\mathbf{x}$ and $\mathbf{n}$ are the surface point and surface normal, respectively. $h(\mathbf{x},\mathbf{n})$ is another given non-negative function over $\mathbb{R}^3 \times \mathbb{R}^3$ which is used for regularizing the molecular surface. Here $\lambda \geq 0$ is a constant. From minimizing energy functional (4.29), the partial differential equation (PDE) in the level-set form is obtained as the Euler Lagrange equation [22]. Furthermore an evolutionary PDE equation is obtained as an iterative (time dependent) geometric flow approximation to the PDE.

Given a molecular representation (i.e., PDB) which consists of a sequence of atoms with centers $\{\mathbf{x}_i\}_{i=1}^m$ and radii $\{r_i\}_{i=1}^m$ (see Fig. 3.14(a)), we construct molecular surfaces which minimize the general energy functional (4.29). In [25], we select $f(x) = g(x)^2$ and $h(\mathbf{x},\mathbf{n}) = 1$ with

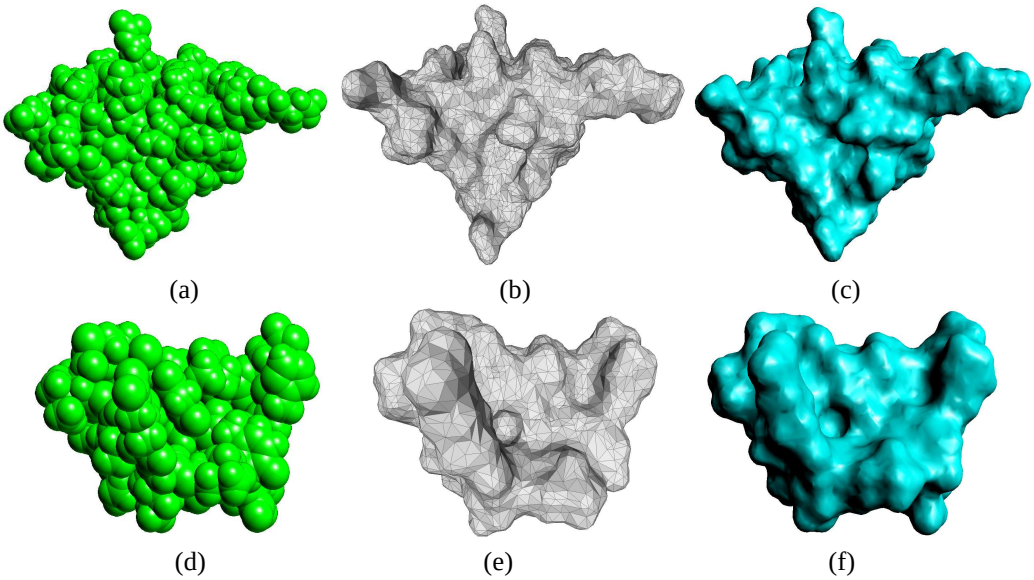
$$g(\mathbf{x}) = 1 - \sum_{i=1}^{m} e^{-C_i(\|\mathbf{x}-\mathbf{x}_i\|^2 - \tilde{r}_i^2)}, \quad \tilde{r}_i = r_i + r_b,$$

where r_b is the probe radius, which usually is 1.4 $\mathring{A}$ (the radius of water). The constant $C_i > 0$, which is also called the Gaussian decay rate, is determined so that $g(\mathbf{x}) = 0$ is an approximation of the solvent accessible surface within a given tolerance ε. We



(a) (b) (c) (d)

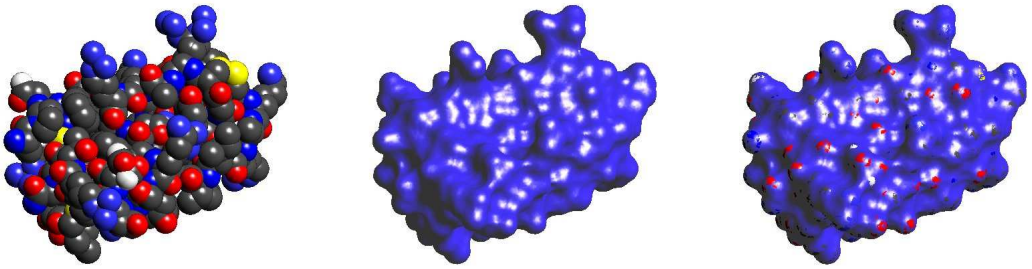
Figure 3.12: Molecular models of a protein(1HIA). (a) is The atomic model. (b) is the initial dense mesh of the SES (27480 triangles). (c) is the decimated mesh of the SES model (7770 triangles). (d) is the ASMS (7770 patches) generated from (c).



(a) (b) (c)

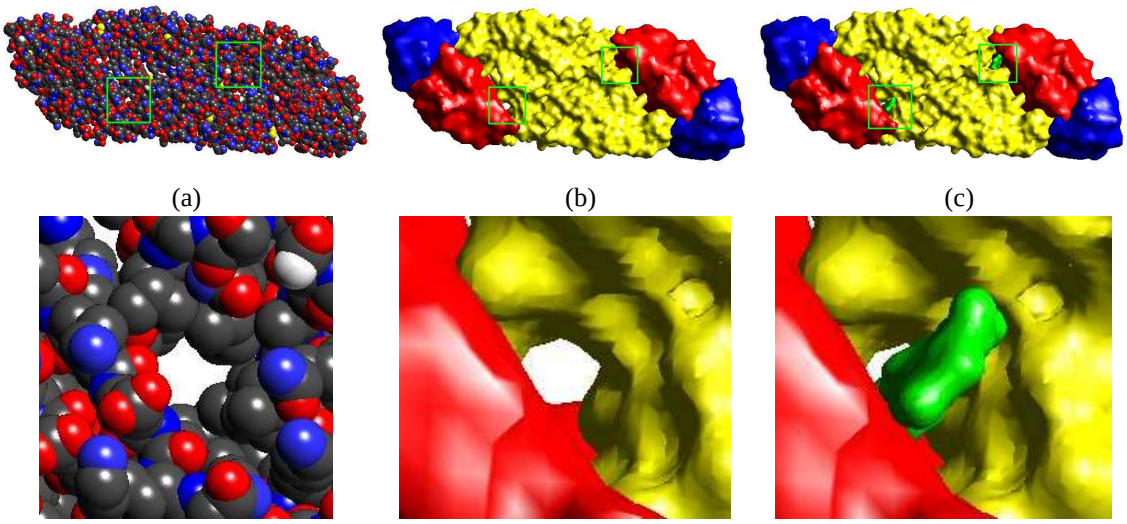
(d) (e) (f)

Figure 3.13: The top row are the models of 1CGI and the bottom row are the models of 1PPE. (a) and (d) are the atomic structures of the proteins. (b) and (e) are the decimated triangular meshes of the proteins with 8712 triangles and 6004 triangles, respectively. (c) and (f) are the ASMS models generated from (b) and (e), respectively.



(a) (b) (c)

Figure 3.14: Molecular Surfaces of Protein (PDB Id: 6PTI). (a) the van der Waals surface. (b) shows our C^2 smooth implicit B-spline molecular surface. (c) shows the tight enclosure of the implicit B-spline molecular surface superimposed with the van der Waals surface.



(a) (b) (c)

(d) (e) (f)

Figure 3.15: Resolution adaptive molecular surfaces of the E - Glycoprotein (PDB Id: 1OKE) of the envelope of the Dengue virus. (a) the van der Waals surface. (b) our C^2 implicit B-spline molecular surface of 1OKE at 10 $\mathring{A}$ resolution (Residues 1-52, 133-193 and 281-296 are colored red. Residues 53-132 and 194-280 are colored yellow. Residues 297-394 are colored blue. The coloring method is based on [84]). (c) our C^2 implicit B-spline molecular surface complexed with a ligand (green) at 10 $\mathring{A}$ resolution. (d) (e) and (f) are a zoomed view of the boxed portions in (a), (b) and (c) respectively (only one of the two boxes are shown in each case as they are identical).

choose C_i as $\frac{\ln 2}{\varepsilon \tilde{r}_i}$. The second term of (4.29) is used to regularize the constructed surface, where λ is a small number. In the examples provided in the following, we choose λ as 0.01. To further eliminate depression and smooth out high curavtures, we select function $h(\mathbf{x},\mathbf{n}) = \|\nabla g(\mathbf{x})\|^2$ in this paper and demonstrate its efficiency by comparing it to a number of prior analytic surfaces [18, 123]. Minimizing energy functional (4.29) for this choice of $h(\mathbf{x},\mathbf{n})$ yields the following Euler-Lagrange equation

$$(g^2 + \lambda\|\nabla g\|^2)\mathrm{div}\left(\frac{\nabla\phi}{\|\nabla\phi\|}\right) + \frac{2g(\nabla g)^T\nabla\phi + \lambda[\nabla(\|\nabla g\|^2)]^T\nabla\phi}{\|\nabla\phi\|} = 0.$$

Thus the corresponding level-set formulation of the evolution equation (see [118] for derivation details or [47]) is

$$\begin{aligned}\frac{\partial\phi}{\partial t} &= (g^2 + \lambda\|\nabla g\|^2)\mathrm{div}\left(\frac{\nabla\phi}{\|\nabla\phi\|}\right)\|\nabla\phi\| + 2g(\nabla g)^T\nabla\phi + \lambda[\nabla(\|\nabla g\|^2)]^T\nabla\phi \\ &= H(\phi) + L(\nabla\phi),\end{aligned} \tag{4.30}$$

where

$$(\phi) = (g^2 + \lambda\|\nabla g\|^2)\mathrm{div}\left(\frac{\nabla\phi}{\|\nabla\phi\|}\right)\|\nabla\phi\|, \quad L(\nabla\phi) = 2g(\nabla g)^T\nabla\phi + \lambda[\nabla(\|\nabla g\|^2)]^T\nabla\phi.$$

This evolution equation is solved by our higher-order level set methods [25]. The first order term $L(\nabla\phi)$ is computed using an upwind scheme over a finer grid, and the higher order term $H(\phi)$ is computed using a spline presentation defined on a coarser grid. If ϕ is a signed distance function and a steady solution of equation (4.30), then the iso-surface $\phi = -r_b$ is an approximation of molecular surface (see Fig. 3.14(b)).

We suppose that equation (4.30) is solved in a bounding box domain Ω which is uniformly partitioned with vertices $G_0 = \{\mathbf{x}_{ijk}\}_{ijk=0}^{n}$. Let G_l be the set of vertices of the grid which is generated by binary subdivision G_0 uniformly l times. Let ϕ be a piecewise trilinear level set function defined on a grid G_l, with l = 0 or 1 or 2, and Φ be the cubic spline approximation of ϕ over grid G_0. Without loss of generality, we take l = 1 for simplicity. Thus the main algorithm in [22] can be repeated as follows.

Algorithm 1. *Smooth molecular surface construction*

1. Compute $g(\mathbf{x})$ over the grid G_l.
2. Compute an initial ϕ (= ϕ_0) by taking $\phi(\mathbf{x}) = g(\mathbf{x})$ and then update it using a reinitialization step, such that $\|\nabla\phi\| = 1$. Let Γ^0 be the initial closed level set surface of ϕ_0 with interior $\mathscr{D} \subset \mathbb{R}^3$.
3. Update ϕ by solving equation (4.30) for one time step using a finite difference method.
4. Reinitialize ϕ, convert ϕ to Φ, and then return to the previous step if the stopping criterion is not satisfied.
5. Generate a level set surface $\{\mathbf{x} : \Phi(\mathbf{x}) = -r_b\}$, which is the required approximation of the smooth molecular surface.

In the following subsections, we summarize the main issues in the implementation of the above algorithm.

Level set evolution

Equation (4.30) is solved in a thin shell of the moving surface to accelerate computation and reduce errors. We first initialize ϕ^0 to be the signed distance function of Γ^0, if necessary, reinitialization step can be done first (see subsubsection 3.4.1). Then we define a thin shell around Γ^0 by

$$N^0 = \{\mathbf{x} \in G_l : |\phi^0(\mathbf{x})| < \mathscr{H}\},$$

where $\mathscr{H}$ is the thickness of the shell should be evaluated first. To prevent numerical oscillations at the boundary of the thin shell, we should introduce a cut-off function $c(x)$ in (4.30) as

$$\begin{cases} \frac{\partial\phi}{\partial t} = c(\phi)[H(\phi) + L(\nabla\phi)], \\ \phi(\mathbf{x},0) = \phi^0(\mathbf{x}), \end{cases} \tag{4.31}$$

on N^0 for one time step and get $\phi^1(\mathbf{x})$. The time step is chosen such that the interface moves less than one grid size Δx. At each grid point $\mathbf{x}_{ijk}$ in the thin shell N^0, compute $v^0(\mathbf{x}_{ijk}) = c(\phi^0(\mathbf{x}_{ijk}))[H(\Phi^0(\mathbf{x}_{ijk})) + L(\nabla\phi^0(\mathbf{x}_{ijk})]$. Let

$$\tau = \min\left\{\Delta x, \Delta x / \max_{\mathbf{x}_{ijk}\in N^0} |v^0(\mathbf{x}_{ijk})|\right\}.$$

Then update ϕ^0 by the explicit Euler scheme

$$\phi^1(\mathbf{x}_{ijk}) = \phi^0(\mathbf{x}_{ijk}) + \tau v^0(\mathbf{x}_{ijk}), \ \ \mathbf{x}_{ijk} \in N^0.$$

Since ϕ^1 is no longer a signed distance function, a reinitialization step is required to get a new ϕ^1 and a new thin shell N^1. The process from ϕ^0 to ϕ^1 described above is repeated to get a sequence $\{\phi^m\}_{m\geq 0}$ of ϕ, and a sequence of thin shells $\{N^m\}_{m\geq 0}$, until the following termination conditions

$$\max_{\mathbf{x}_{ijk}\in N^m} |v^m(\mathbf{x}_{ijk})| < \varepsilon \ \text{ and } \ m < M$$

are satisfied. We choose $\varepsilon = 0.001$. M is a given upper bound of the iteration number, we choose $M = n$, where n^3 is the number of grid points in G_l.

The construction of an initial surface Γ^0

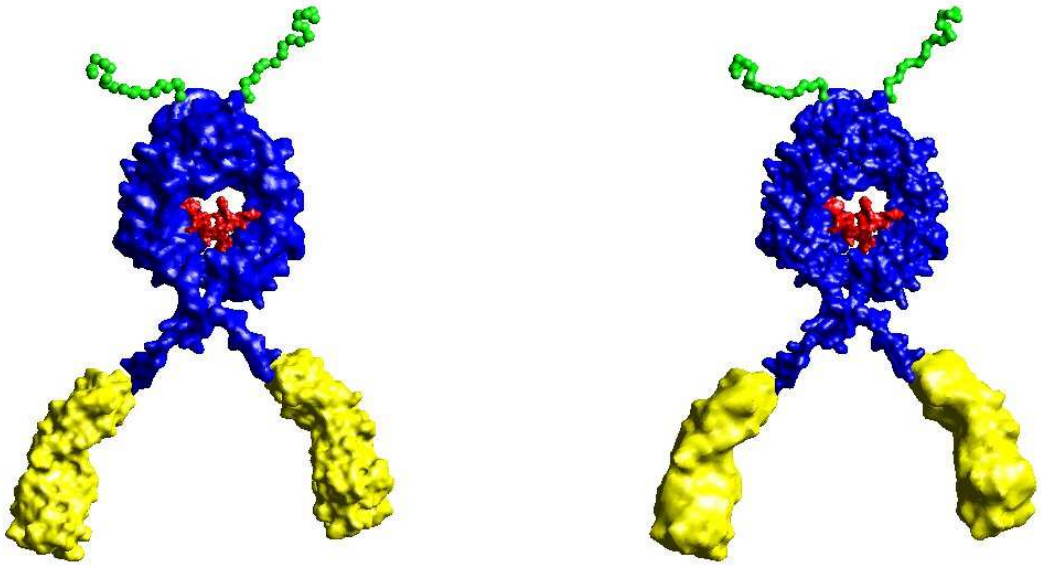
The construction of an initial surface is pivotal to the level-set methods. If the initial surface is near to the final minimal surface, a few evolution steps are enough. In our variational molecular surface construction, we use $g(\mathbf{x}) = 0$ as an initial surface Γ^0. To further speed the computation, we use a local fast computation of the Gaussian function $e^{-C[\|\mathbf{x}-\mathbf{x}_i\|^2-\tilde{r}_i^2]}$ in $g(\mathbf{x})$ around $\mathbf{x}_i$. For details on fast Gaussian summation, refer to [18, 23].

Adaptive reinitialization

In general, it is impossible to prevent the evolving level set function $\phi(\mathbf{x})$ from deviating away from a signed distance function. Flat or steep regions could develop around the interface, making further computation and level set determination highly inaccurate. Hence a reinitialization step to reset the level set function ϕ to be a signed distance function is usually necessary. This problem has been carefully studied in [90]. The main idea is to solve a Hamilton-Jacobi equation. We omit the details here and refer the reader to [22].

3.4.2 Illustrative Examples

In this subsection, we provide implementation details of several applications of the methods. Our variational molecular surface algorithm has been implemented in our molecular visualization software package TexMol [8]. We now present illustrative examples of variational molecular surfaces, such as multiresolution molecular models and hierarchical macromolecular structure surfaces. Quantitative comparative surface analysis with Gaussian and adaptive grid molecular surfaces methods are also described.



(a) (b)

Figure 3.16: Adaptive Resolution C^2 Smooth Implicit B-spline Molecular Surfaces. (a) yellow legs and green antenna at about 5 $\mathring{A}$ resolution, and the blue body at about 10 $\mathring{A}$ resolution of 1HZH (PDB Id). (b) adaptive resolution with legs and antenna at about 10 $\mathring{A}$ resolution, and the body at about 5 $\mathring{A}$ resolution.

Adaptive resolutions of molecular surface models

To capture molecular features, such as pockets and tunnels, one can model molecular surfaces with varying and adaptive resolution. Our variational molecular surface method provides an approach to achieve this. Since the initial surface is an approximation surface, one can select a spatially adaptive decay rate C to capture the initial surface at adaptive resolution. In Fig. 3.15, we show such an example of a ligand-binding pocket in the dengue virus (DV) envelope (E) Glycoprotein. Fig. 3.16 shows another example, where different portions of the molecular surface of a monomer of the intact human Immunoglobulin B12 (PDB Id 1HZH) are shown at different resolutions.

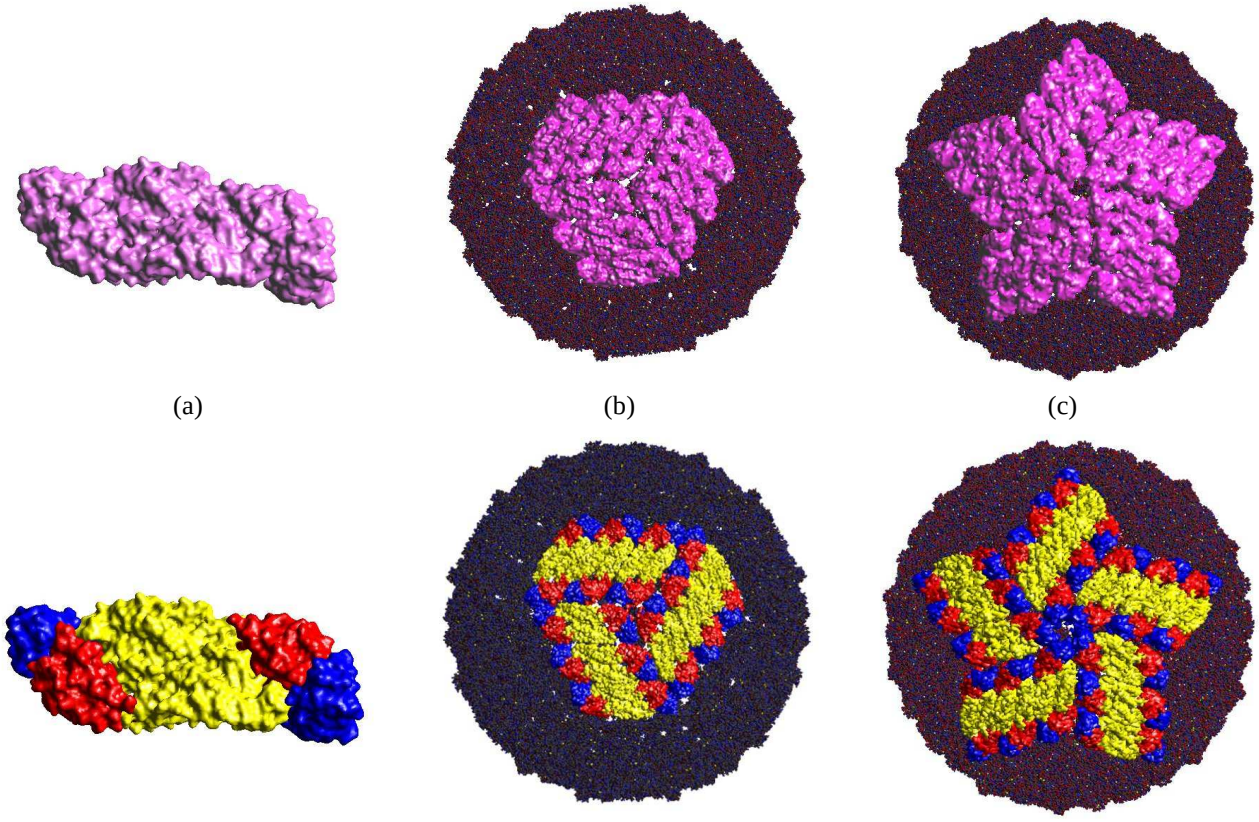


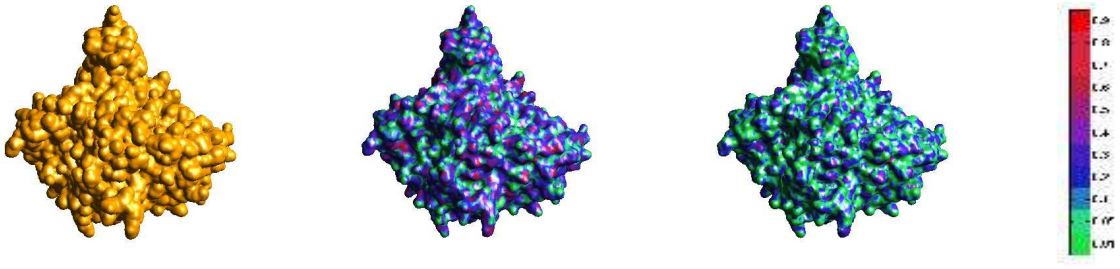
Figure 3.17: Hierarchical C^2-Smooth Implicit B-Spline Molecular Surface Models of the Envelope of the Dengue Virus Figure. (a) Two chains of the monomeric envelope glycoprotein. (b) 1 three-fold envelope and the van der Waal of the other part of the envelope. (PDB Id: 1K4R) (c) 1 five-fold envelope and the van der Waal of the remain part of the envelope. (PDB Id: 1K4R). (d) the smooth implicit B-Spline molecular surface of two chains of the monomeric envelope glycoprotein colored using [84]. (e) similar to (b) using the coloring of (d). (f) similar to (b) using the coloring of (d).

Hierarchical molecular surface segmentation of large macro-molecular complexes

Large biomolecular complexes, such as ribosomes and viruses are assemblies of multiple proteins and nucleic acids and dozens to thousands of structural/functional biomolecular subunits. Hierarchical molecular surface segmentation with distinguishable coloring is extremely useful in achieving better understanding of the structural organization of such assemblies. Here we present one example of a hierarchical structure organization of the molecular surface of the icosahedral envelope of the Dengue Virus in Fig. 3.17. Where figure (a) is the molecular surface of two chains. Figure (b) is a molecular surface of a three-fold part of the envelope with the other parts as van der Waals surfaces. Figure (c) is a molecular surface of a five-fold part of the envelope with the other parts as van der Waals surfaces. In figure (d), molecular surfaces of different residues groups are colored differently. The other two figures are similar to (b) and (c) separately.

Comparative examples

In this subsubsection, we compare our C^2 smooth B-spline molecular surface with molecular surfaces generated by level sets of Gaussian functions [123] and molecular surfaces generated by adaptive grid methods [18]. We also compare the difference between different regularization terms in the generation of variational molecular surfaces, in particular with respect to [22, 25].



(a) (b) (c) (d)

Figure 3.18: Comparison of Three Different Molecular Surface Models (PDB Id: 1FSS). (a) the Gaussian molecular surface. (b) molecular surface by the adaptive grid method with the difference between this surface and the surface of (a), displayed as a color mapped function on surface. (c) our C^2 molecular surface and the difference between this surface and the surface of (a), again displayed as color mapped function on surface. (d) is the color bar of the difference.

Fig. 3.18 shows this comparison for Acetylcholinesterase complexed with Fasciculin-II, having 1FSS as its PDB Id. The molecular surface using Gaussian functions is defined by $\{\mathbf{x} \in \mathbb{R}^3 : g(\mathbf{x}) = 1\}$, where $g(\mathbf{x}) = \sum_{i=1}^{n} e^{-\frac{d}{r_i^2}\left[\|\mathbf{x}-\mathbf{x}_i\|^2-(r_i)^2\right]}$. In figure (a), we show Gaussian molecular surface. Figures (b) and (c) show the results of the adaptive grid method and our variational method. The differences between the Gaussian molecular surface with the surfaces generated by the adaptive grid method and our variational method are also respectively plotted as color mapped functions on the Gaussian surfaces. For two surfaces S_1, S_2, we calculated the difference by the following simple method

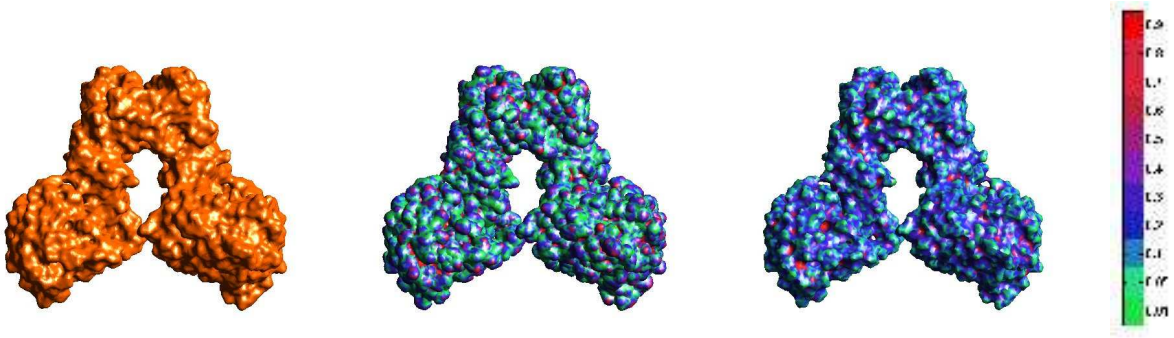
$$\text{Diff}(\mathbf{x}, S_2) = \min\{\text{dist}(\mathbf{x}, \mathbf{y}), \mathbf{y} \in S_2\}, \mathbf{x} \in S_1,$$

where $\text{dist}(\mathbf{x}, \mathbf{y})$ is the Euclidean distance of two points. Then we displayed the difference by a color mapped function defined on the initial surface S_1. In this example, we select Gaussian surface as S_2 and adaptive grid surface and our variational surface as S_1. In Fig. 3.19, an illustrative example is given for the Aspartate Carbamoyltransferase (PDB Id 4AT1). We show our molecular surface in figure (a). Similar to the above example, we depict in figure (b) the difference between our variational molecular surface and the Gaussian surface, S_2 is the molecular surface produced by our variational method. Similarly in figure (c), we show the difference between the adaptive grid molecular surface with our molecular surface. From these figures, we can see that the variational molecular surface is different from the Gaussian and adaptive grid surface.

To better quantitate this difference, in Table 3.4 and Table 3.5, we compare the results of area and volume computation using our variational method, and that of Gaussian, of adaptive grid molecular surfaces. The results of Gaussian molecular surfaces and adaptive grid molecular surfaces are implemented in TexMol. Since the surfaces produced by Gaussian functions often yield artifacts such as narrow depressions or tunnels and furthermore are quite inflated [9] (see Fig. 3.18 and 3.19), the surface area is enlarged and the enclosed volume is smaller. The result shows that our method gives larger volumes to Gaussian molecular surfaces but much smaller surfaces areas. They are also free from the topological surface artifacts. Comparing results with the adaptive grid method, we find that both the surface areas and volumes of our method are larger.

On comparison between different regularization terms

In this paper, we use another regularization term to decrease unwanted surface depression or narrow tunnel artifacts on the molecular surface. Compared with the regularization term we used in [25], while no obvious visual differences can be observed, we show example of the molecular surfaces with our different regularization term in Fig. 3.20. Figures (a) and (d) are the variational molecular surface enclosing the van der Waals surface. Figures (b) and (e) are the mean curvature and Gaussian curvature plots of the functional with area as regularization term. Figures (c) and (f) are the mean curvature and Gaussian curvature plots of the functional with the new regularization term used in this paper.



(a) (b) (c) (d)

Figure 3.19: Comparison of Three Different Molecular Surface Models of the Aspartate Carbamoyltransferase (PDB Id: 4AT1). (a) our B-spline molecular surface. (b) Gaussian molecular surface with difference with (a) shown a color mapped function on surface. (c) molecular surface using an adaptive grid with grid resolution 128^3 and with difference with (a) shown as a color mapped function on surface. (d) is the color bar.

Molecule	PDB Id	A1	A2	A3
Acetylcholinesterase Fasciculin	1FSS	25638.6	20233.6	20335.6
Fas.2 Mouse Acetylcholinesterase Complex	1MAH	16920.2	17061.0	17121.8
Glutamine Synthetase	2GLS	262196.6	172746.3	158346.8
Aspartate Carbamoyltransferase	4AT1	45059.5	32608.0	32300.5
HIV Capsid C	1A8O	11294.4	7015.9	6334.3
GroEL-GroES Complex	1AON	404391.4	283704.5	204322.9
Quinoprotein Methylamine Dehydrogenase	2BBK	63868.9	25349.6	21396.8
Scapharca Inaequivalvis	2Z8A	22827.4	11228.7	9997.8

Table 3.4: Surface area of different proteins computed using three different methods. A1 is computed for Gaussian molecular surface. A2 is computed for adaptive grid molecular surface. A3 is computed for the molecular surface using our methods.

Molecule	PDB Id	V1	V2	V3
Acetylcholinesterase Fasciculin	1FSS	94653.5	78017.2	84453.0
Fas.2 Mouse Acetylcholinesterase Complex	1MAH	116586.9	113528.9	119776.8
Glutamine Synthetase	2GLS	48724.5	46250.6	42617.0
Aspartate Carbamoyltransferase	4AT1	96081.7	90061.6	97158.3
HIV Capsid C	1A8O	22997.3	21801.2	24718.3
GroEL-GroES Complex	1AON	43316.1	38817.6	30928.9
Quinoprotein Methylamine Dehydrogenase	2BBK	130205.1	133399.8	144046.2
Scapharca Inaequivalvis	2Z8A	44518.0	45002.0	50330.7

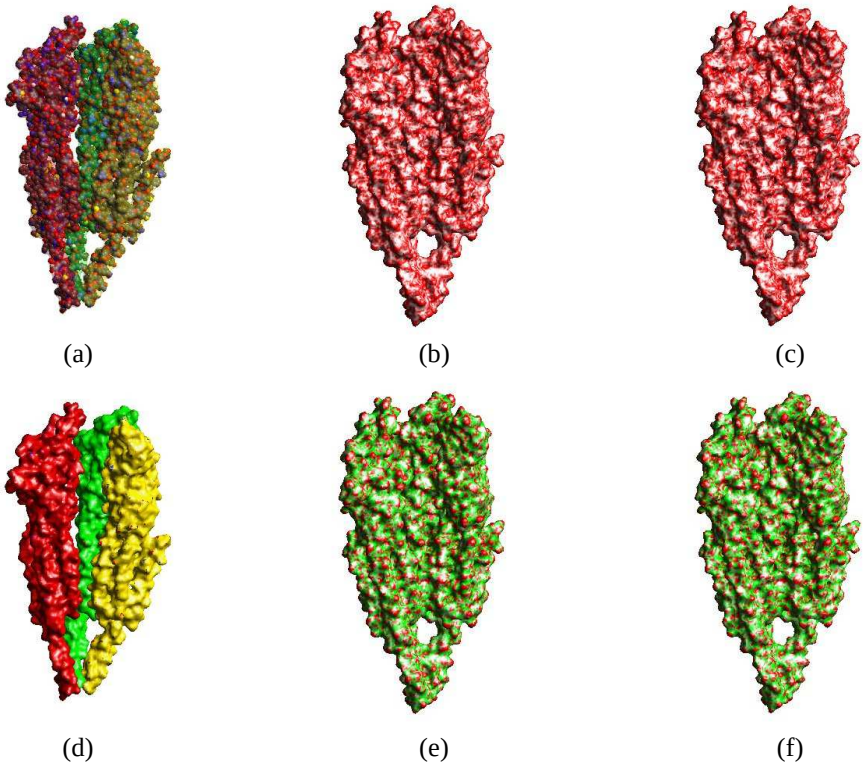
Table 3.5: Molecular volume of different proteins computed using three different methods. V1 is computed via Gaussian molecular surface. V2 is computed for adaptive grid method surface. V3 is computed for the molecular surface using our methods.

3.5 Meshing of Molecular Interfaces

In this subsection, we describe an approach to generate quality triangular/tetrahedral meshes for complicated biomolecular structures directly from the PDB format data, conforming to a good implicit solvation surface approximation. There are three main steps in our mesh generation process:

1. Implicit Solvation Surface Construction – A smooth implicit solvation model is constructed to approximate the Lee-Richards molecular surface by using weighted Gaussian isotropic atomic kernel functions and a two-level clustering techniques. See subsection 3.1 for details

2. Mesh Generation – A modified dual contouring method is used to extract triangular and interior/exterior tetrahedral



(a) (b) (c)

(d) (e) (f)

Figure 3.20: Feature Preserving Adaptive Resolution Molecular Surfaces of the Nicotonic AcetylCholine Receptor Protein 2BG9. (a) (d) our molecular surface enclosing the the van der Waals surface. (b) and (c) are the mean curvature plots of the variational molecular surface generated by the minimal area regularization and the gradient magnitude squares regularization functional. (e) and (f) are the respective Gaussian curvature plots.

meshes, conforming to the implicit solvation surface. The dual contouring method [120, 121] is selected for mesh generation as it tends to yield meshes with better aspect ratio. In order to generate exterior meshes described by biophysical applications [108, 109, 119], we add a sphere or box outside the implicit solvation surface, and create an outer boundary. Our extracted tetrahedral mesh is spatially adaptive and attempts to preserve molecular surface features while minimizing the number of elements.

3. Quality Improvement – Geometric flows are used to improve the quality of extracted triangular and tetrahedral meshes.

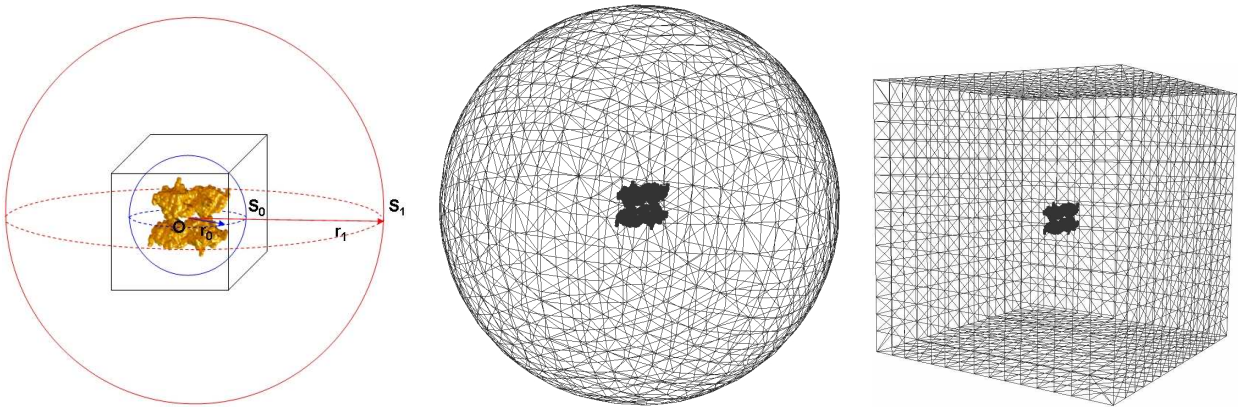
The generated tetrahedral meshes of the monomeric and tetrameric mouse acetylcholinesterase (mAChE) [31, 32] have been successfully used in solving the steady-state Smoluchowski equation using the finite element method [108, 109, 119].

3.5.1 Mesh Generation

There are two main methods for contouring scalar fields, primal contouring [77] and dual contouring [61]. Both of them can be extended to tetrahedral mesh generation. The dual contouring method [120, 121] is often the method of choice as it tends to yield meshes with better aspect ratio.

Triangular Meshing

Dual contouring [61] uses an octree data structure, and analyzes those edges that have endpoints lying on different sides of the isosurface, called *sign change edges*. The mesh adaptivity is determined during a top-down octree construction. Each sign change edge is shared by either four (uniform case) or three (adaptive case) cells, and one minimizer point is calculated for



(a) (b) (c)

Figure 3.21: The analysis domain of exterior meshes. (a) - 'O' is the geometric center of the molecule, suppose the circum-sphere of the biomolecule has the radius of r. The box represents the volumetric data, and 'S_0' is the maximum sphere inside the box, the radius is $r_0(r_0 > r)$. 'S_1' is an outer sphere with the radius of $r_1(r_1 = (20 \sim 40)r)$. (b) - the diffusion domain is the interval volume between the molecular surface and the outer sphere 'S_1', here we choose $r_1 = 5r$ for visualization. (c) - the outer boundary is a cubic box.

each of them by minimizing a predefined Quadratic Error Function (QEF) [54]:

$$QEF[x] = \sum_i [n_i \cdot (x - p_i)]^2, \tag{5.32}$$

where p_i, n_i represent the position and unit normal vectors of the intersubsection point respectively. For each sign change edge, a quad or triangle is constructed by connecting the minimizers. These quads and triangles provide a 'dual' approximation of the isosurface.

A recursive cell subdivision process was used to preserve the trilinear topology [121] of the isosurface. During cell subdivision, the function value at each newly inserted grid point can be exactly calculated since we know the volumetric function (Eqn. (1.4)). Additionally, we can generate a more accurate triangular mesh by projecting each generated minimizer point onto the isosurface (Eqn. (1.3)).

Tetrahedral Meshing

The dual contouring method has already been extended to extract tetrahedral meshes from volumetric scalar fields [120, 121]. The cells containing the isosurface are called boundary cells, and the interior cells are those cells whose eight vertices are inside the isosurface. In the tetrahedral mesh extraction process, all the boundary cells and the interior cells need to be analyzed in the octree data structure. There are two kinds of edges in boundary cells, one is a sign change edge, the other is an interior edge. Interior cells only have interior edges. In [120, 121], interior edges and interior faces in boundary cells are dealt with in a special way, and the volume inside boundary cells is tetrahedralized. For interior cells, we only need to split them into tetrahedra.

Adding an Outer Boundary In biological diffusion systems, we need to analyze the electrostatic potential field which is faraway from the molecular surface [60, 71]. Assume that the radius of the circum-sphere of a biomolecule is r. The computational model can be approximated by a field from an outer sphere S_1 with the radius of $(20 \sim 40)r$ to the molecular surface. Therefore the exterior mesh is defined as the tetrahedralization of the interval volume between the molecular surface and the outer sphere S_1 (Fig. 3.21(b)). Sometimes the outer boundary is chosen to be a cubic box as shown in Fig. 3.21(c).

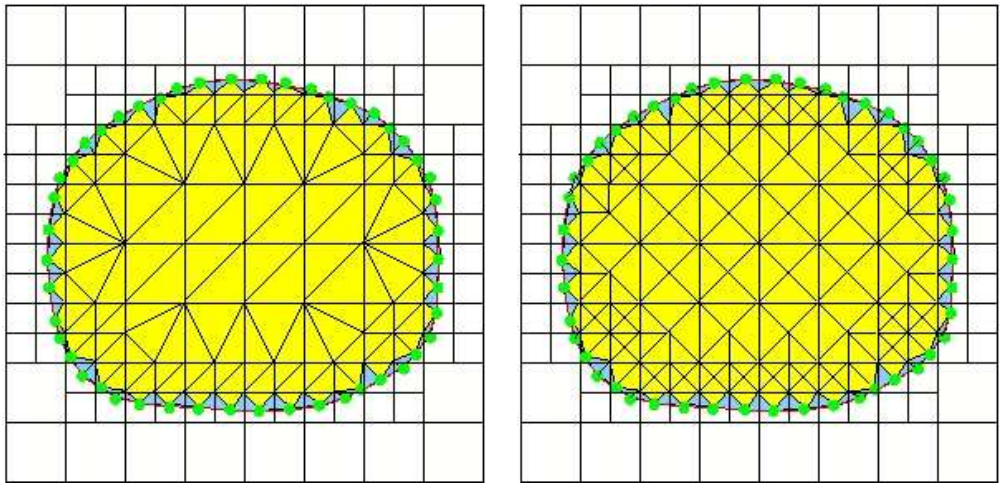
First we add a sphere S_0 with the radius of r_0 (where $r_0 > r$ and $r_0 = 2^n/2 = 2^{n-1}$) outside the molecular surface, and generate meshes between the molecular surface and the outer sphere S_0. Then we extend the tetrahedral meshes from the sphere S_0 to the outer bounding sphere S_1. For each data point inside the molecular surface, we keep the original function value. While for each data point outside the molecular surface, we reset the function value as the smaller one of $f(x) - \alpha$ and the shortest distance from the grid point to the sphere S_0. Eqn. (5.33) shows the newly constructed function $g(x)$ which provides

a grid-based volumetric data containing the biomolecular surface and an outer sphere S_0.

$$g(x) = \begin{cases} \min(\|x-x_0\| - r_0, f(x) - \alpha), & \text{if } f(x) < \alpha, \|x-x_0\| < r_0, \\ \|x-x_0\| - r_0, & \text{if } f(x) < \alpha, \|x-x_0\| \geq r_0, \\ f(x) - \alpha, & \text{if } f(x) \geq \alpha, \end{cases} \tag{5.33}$$

where x_0 are coordinates of the molecular geometric center. The isovalue α = 0.5 for volumetric data generated from the characteristic function, and α = 1.0 for volumetric data generated from the summation of Gaussian kernels.

The molecular surface and the outer sphere S_0 can be extracted as an isosurface at the isovalue 0, $S_g(0) = \{x|g(x) = 0\}$. All the grid points inside the interval volume $I_g(0) = \{x|g(x) \leq 0\}$ have negative function values, and all the grid points outside it have positive values.



(a) (b)

Figure 3.22: 2D triangulation. (a) Old scheme, (b) New scheme. Blue and yellow triangles are generated for sign change edges and interior edges respectively. The red curve represents the molecular surface, and the green points represent minimizer points.

Mesh Extraction

Here we introduce a different scheme from the algorithm presented in [120, 121], in which we do not distinguish boundary cells and interior cells when we analyze edges. We only consider two kinds of edges - sign change edges and interior edges. For each boundary cell, we can obtain a minimizer point by minimizing its Quadratic Error Function. For each interior cell, we set the middle point of the cell as its minimizer point. Fig. 3.22(b) shows a simple 2D example. In 2D, there are two cells sharing each edge, and two minimizer points are obtained. For each sign change edge, the two minimizers and the interior vertex of this edge construct a triangle (blue triangles). For each interior edge, each minimizer point and this edge construct a triangle (yellow triangles). In 3D as shown in Fig. 3.23, there are three or four cells sharing each edge. Therefore, the three (or four) minimizers and the interior vertex of the sign change edge construct one (or two) tetrahedron, while the three (or four) minimizers and the interior edge construct two (or four) tetrahedra.

Compared with the algorithm presented in [120, 121] as shown in Fig. 3.22(a), Fig. 3.22(b) generates the same surface meshes, and tends to generate more regular interior meshes with better aspect ratio, but a few more elements for interior cells. Fig. 3.22(b) can be easily extended to large volume decomposition. For arbitrary large volume data, it is difficult to import all the data into memory at the same time. So we first divide the large volume data into some small subvolumes, then mesh each subvolume separately. For those sign change edges and interior edges lying on the interfaces between subvolumes, we analyze them separately. Finally, the generated meshes are merged together to obtain the desired mesh. The mesh adaptivity is controlled by the structural properties of biomolecules. The extracted tetrahedral mesh is finer around the molecular surface, and gradually gets coarser from the molecular surface out towards the outer sphere, S_0. Furthermore, we generate the finest mesh around the active site, such as the cavity in the monomeric and tetrameric mAChE shown in Fig.3.29 (a$\sim$b), and a coarse mesh everywhere else.

Mesh Extension

We have generated meshes between the biomolecular surface and the outer sphere S_0, the next step is to construct tetrahedral meshes gradually from the sphere S_0 to the bounding sphere S_1 (Fig. 3.21). The sphere S_0 consists of triangles, so we extend

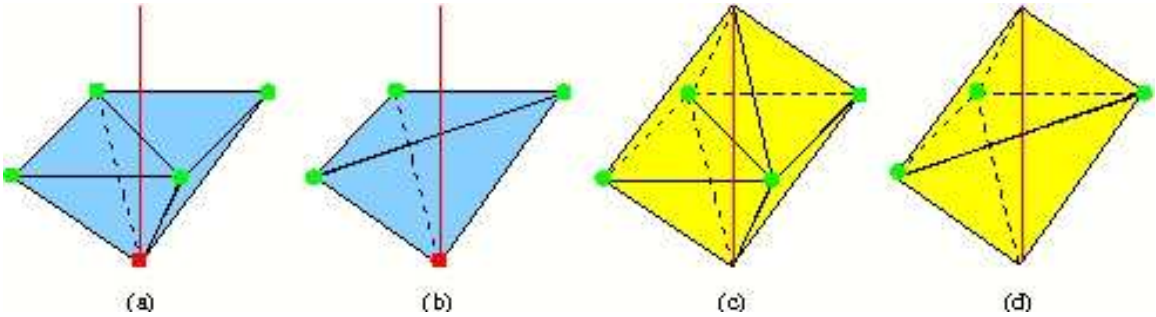


Figure 3.23: Sign change edges and interior edges are analyzed in 3D tetrahedralization. (a)(b) - sign change edge (the red edge); (c)(d) - interior edge (the red edge). The green solid points represent minimizer points, and the red solid points represent the interior vertex of the sign change edge.

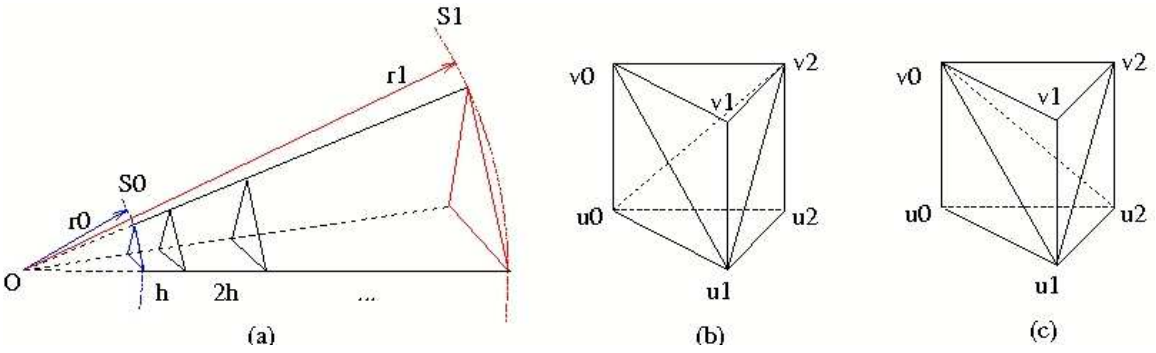


Figure 3.24: (a) - one triangle in the sphere S_0 (blue) is extended n steps until arriving at the sphere S_1 (red); (b) and (c) - a prism is decomposed into three tetrahedra in two different ways.

each triangle radially as shown in Fig. 3.24 and a prism is obtained for each extending step. The prism can be divided into three tetrahedra. The extension step length h can be calculated by Eqn. (5.34). It is better for the sphere S_0 to be triangulated uniformly since the step length is fixed for each extension step.

$$r_0 + h + 2h + \cdots + nh = r_1 \quad \Longrightarrow \quad h = \frac{2(r_1 - r_0)}{n(n+1)} \tag{5.34}$$

where n is the step number. In Figure 3.24, suppose $u_0u_1u_2$ is a triangle on sphere S_0, and u_0, u_1, u_2 are the unique index numbers of the three vertices, where $u_1 < u_0$ and $u_1 < u_2$. For one extension step, $u_0u_1u_2$ is extended to $v_0v_1v_2$, and the two triangles construct a prism, which can be decomposed into three tetrahedra. In order to avoid the diagonal conflict problem, a different decomposition method (Fig. 3.24(b∼c)) is chosen based on the index number of the three vertices. If $u_0 < u_2$, then we choose Fig. 3.24(b) to split the prism into three tetrahedra. If $u_2 < u_0$, then Fig. 3.24(c) is selected

Assume there are m triangles on the sphere S_0, which is extended n steps to arrive at the sphere S_1. m prisms or $3m$ tetrahedra are generated in each extending step, and a total of $3mn$ tetrahedra are constructed in the extension process. Therefore, it is better to keep a coarse and uniform triangular mesh on the sphere S_0.

3.5.2 Quality Improvement

There are two sub-steps in mesh quality improvement:

1. Denoising and improving the aspect ratio of the surface mesh (surface vertex adjustment in the normal and the tangent directions).

2. Improving the aspect ratio of the volumetric mesh (vertex adjustment inside the volume).

We use geometric partial differential equations (PDEs) to handle the first step. Geometric PDEs, such as the mean curvature flow, the surface diffusion flow and Willmore flow, have been intensively used in surface and imaging processing [117, 116]. Here we choose surface diffusion flow to smooth the molecular surface because of its volume preserving, and furthermore it approximates spheres accurately (quadratic precision).

$$\frac{\partial x}{\partial t} = \Delta H(x)\mathbf{n}(x) + v(x)\mathbf{T}(x), \tag{5.35}$$

where H is the mean curvature, $\mathbf{n}$ is the unit surface normal vector, $v(x)$ is the velocity in the tangent direction $\mathbf{T}(x)$, and Δ is the Laplace-Beltrami operator, $\Delta = (\frac{\partial^2}{\partial x^2}, \frac{\partial^2}{\partial y^2}, \frac{\partial^2}{\partial z^2})$.

Eqn. (5.35) is solved over a triangular mesh with vertices $\{x_i\}$ by discretizing each of its terms. In temporal space, $\frac{\partial x}{\partial t}$ is approximated by the Euler scheme $\frac{x_i^{n+1} - x_i^n}{\tau}$, where τ is time step-length. x_i^n is the approximating solution at $t = n\tau$, x_i^{n+1} is the approximating solution at $t = (n+1)\tau$, and $x_i^0 = x_i$ serves as the initial value. Discretizing schemes for Δ and H in the spatial space are given in [116], we do not go to detail here. Further $v(x)\mathbf{T}(x)$ is approximated by

$$[m(x_i^n) - x_i^n] - \mathbf{n}(x_i^n)^T [m(x_i^n) - x_i^n]\mathbf{n}(x_i^n), \tag{5.36}$$

where $m(x_i^n)$ is defined as the mass center of all the surface triangles incident to x_i^n. A mass center P of a region V is defined by finding $p \in V$, such that $\int_V \| y - p \|^2 \, d\sigma = min$, where V could be a piece of surface or a volume in $\mathbb{R}^3$. For our surface mesh case, V consists of triangles around vertex x_i^n. Then we could derive that

$$m(x_i^n) = \frac{1}{3}x_i^n + \frac{1}{3}\sum_{j \in N(i)} x_j^n(\triangle_j + \triangle_{j+1})/A(x_i^n), \tag{5.37}$$

where $N(i)$ is the index set of the one ring vertex neighbors of x_i^n, $\triangle_j$ is the area of the triangle $[x_i^n x_{j-1}^n x_j^n]$. $A(x_i^n)$ is the total of triangle areas.

Usually, people use the geometric center [116], instead of the mass center, however we observed that the mass center works better for biomolecules. The discretization leads to a positive-definite linear system, and the approximate solution is obtained by solving this linear system.

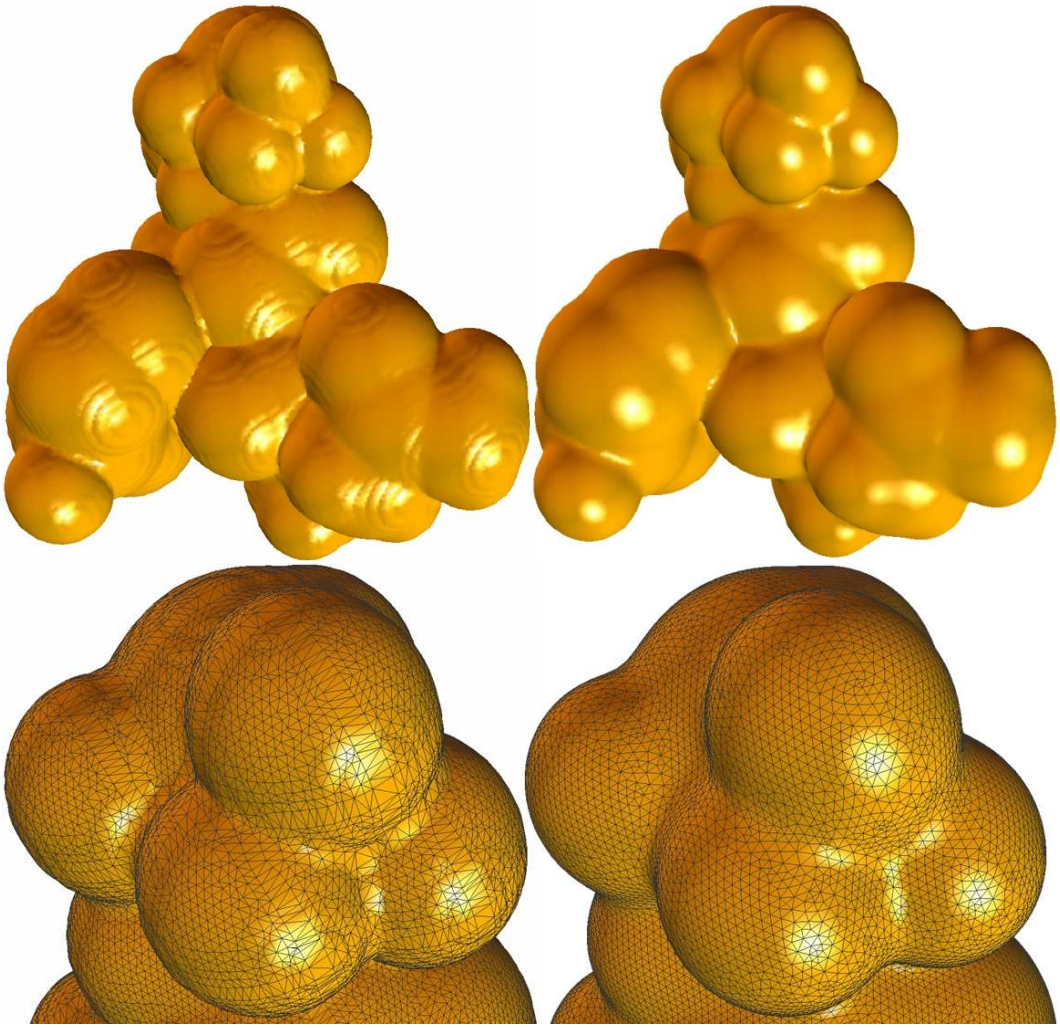


Figure 3.25: The surface comparison before/after quality improvement. The left column shows the original surface of an ASN-THR-TYR peptide, and the right column shows the surface after mesh regularization. The top row shows the smooth shading surfaces, and the bottom row shows snapshots of the meshes.

After the molecular surface is regularized, the next step is to improve the volumetric mesh by relocating each interior vertex to the mass center of its surrounding tetrahedra. Let p_i be an interior vertex, p_j be one of its neighboring vertices, then the mass center of all tetrahedra around p_i is computed by $m(p_i) = \frac{1}{4}p_i + \frac{1}{4V_i}\sum_j V_{ij}p_j$, where V_{ij} is the volume summation of all the tetrahedra around the edge $[p_i p_j]$, V_i is the volume summation of the tetrahedra around the vertex p_i. This is similar in spirit to the multi-linear centroid smoothing scheme [14].

Fig. 3.25 shows the difference of the mesh before and after the quality improvement steps. The left column shows the original iso-surface of an ASN-THR-TYR peptide, and the right column shows the results after mesh regularization. It is obvious that after quality improvement, the surface mesh is more regular and has better aspect ratio (twice the ratio of incircle radius to circumcircle radius).

The left picture in Fig.3.28 shows the improvement of the aspect ratio, and Fig.3.26$\sim$3.27 show the improvement in mesh regularization. We can see that noises are removed and features are preserved since the surface diffusion flow preserves volume and spherical geometry. The surface error is restricted within half of the grid size for the binary data from the characteristic function, and almost zero for the data from Gaussian density map since we have projected each boundary vertex onto the isosurface.

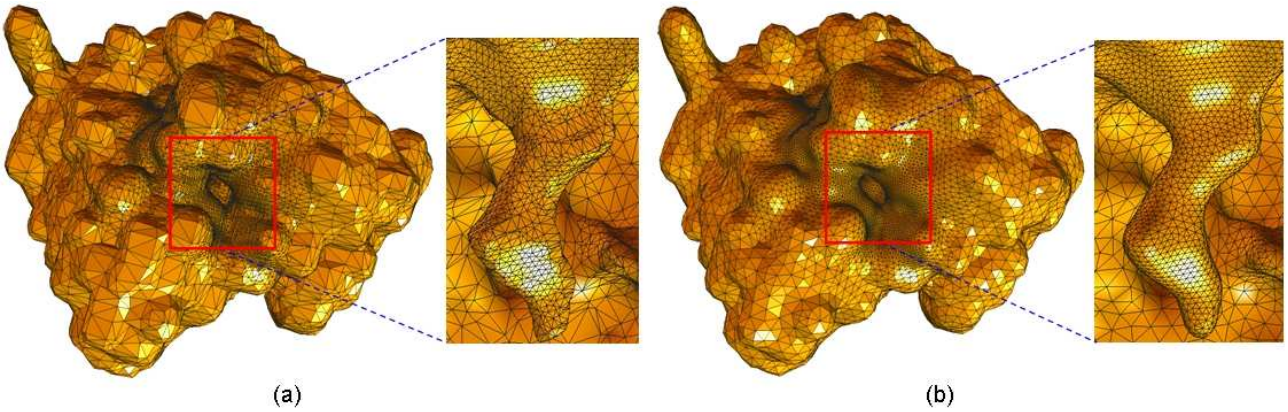


Figure 3.26: Comparison of mAChE (9308 vertices, 18612 triangles) before and after surface mesh regularization. (a) - original; (b) - after mesh regularization.

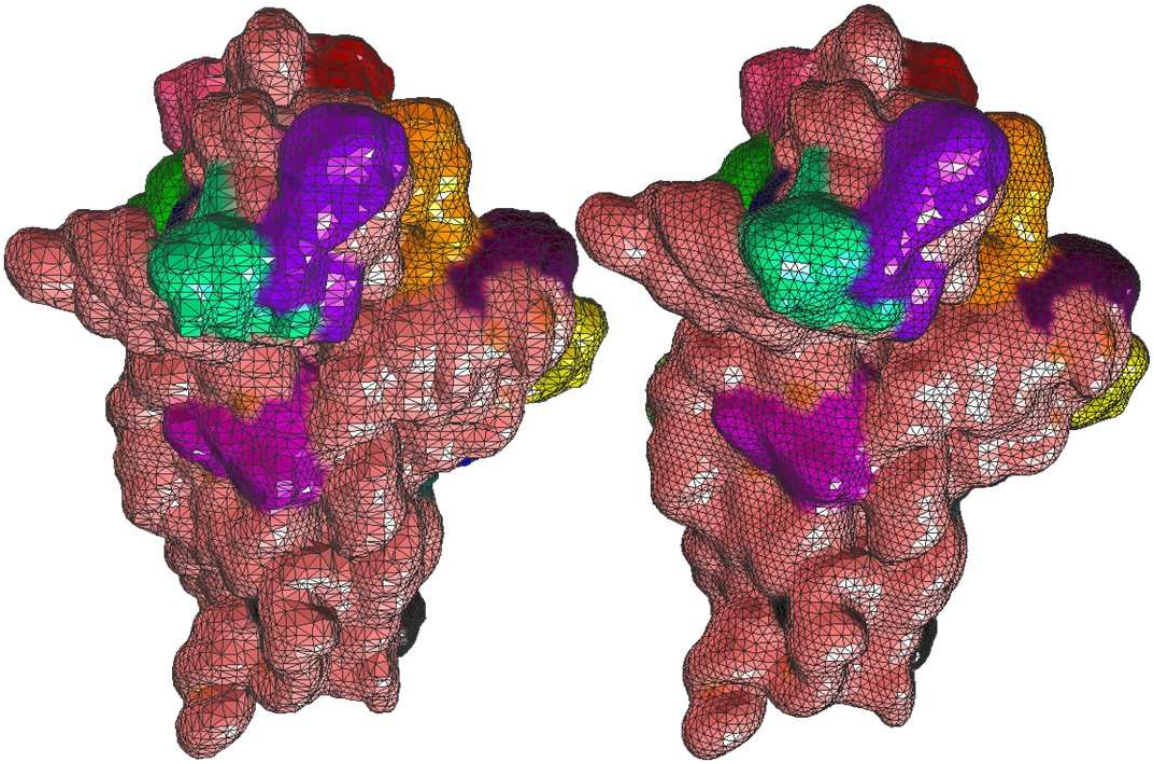


Figure 3.27: Comparison of Ribosome 30S (13428 vertices, 26852 triangles) before and after surface mesh regularization. Left - original; Right - after mesh regularization.

In [121], the edge contraction and linear averaging method was used to improve the quality of tetra meshes with the edge-ratio (the longest edge length over the shortest edge length) and Joe-Liu parameter ($2^{\frac{4}{3}} \times 3 \times (|V|)^{\frac{2}{3}} / \sum_{0 \le i < j \le 3} |e_{ij}|^2$, where $|V|$ denotes the volume, e_{ij} represents the edge connecting vertex v_i and v_j) as metrics. The goal is to improve the worst parameters in each iteration. Here we still use the same edge contraction scheme, but relocate each interior vertex to its mass center (Eqn. (5.37)) since it can minimize the energy defined earlier ($\int_V \| y - p \|^2 \, d\sigma$). From the right picture in Fig. 3.28, we can see that the worst Joe-Liu parameter is improved significantly after quality improvement. Fig. 3.29 and 3.31 show interior tetra meshes of mAChE and Ribosome 30S.

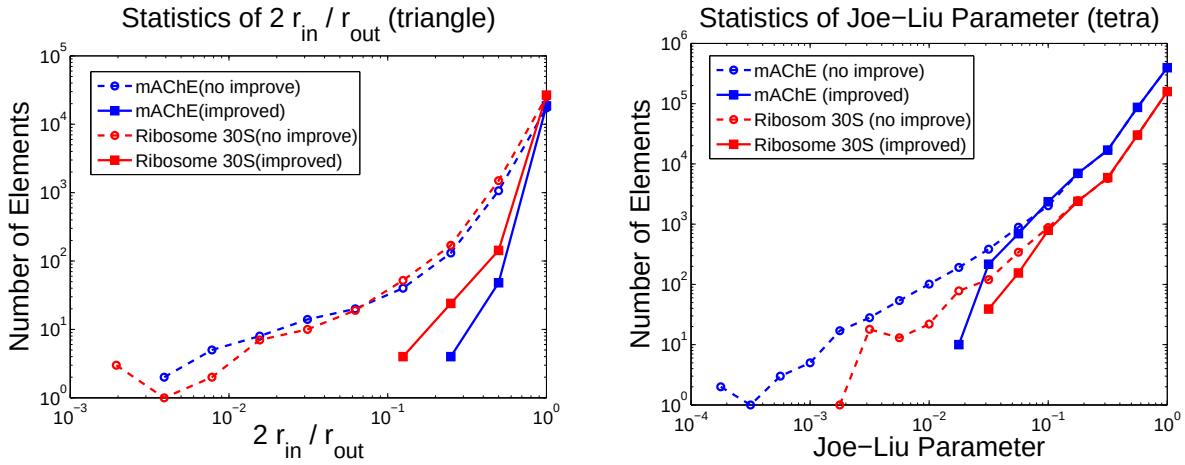


Figure 3.28: The histogram of the aspect-ratio and Joe-Liu parameter.

3.5.3 Results and Conclusion

Monomeric mAChE: The extracted tetrahedral meshes of the monomer as shown in Fig. 3.29 have been used in the finite element analysis of the steady-state Smoluchowski equation (SSSE) for diffusion rate constant calculations [108] [109]. Note the adaptive meshes around the narrow gorge region (the active site in mAChE is at the bottom of this gorge).

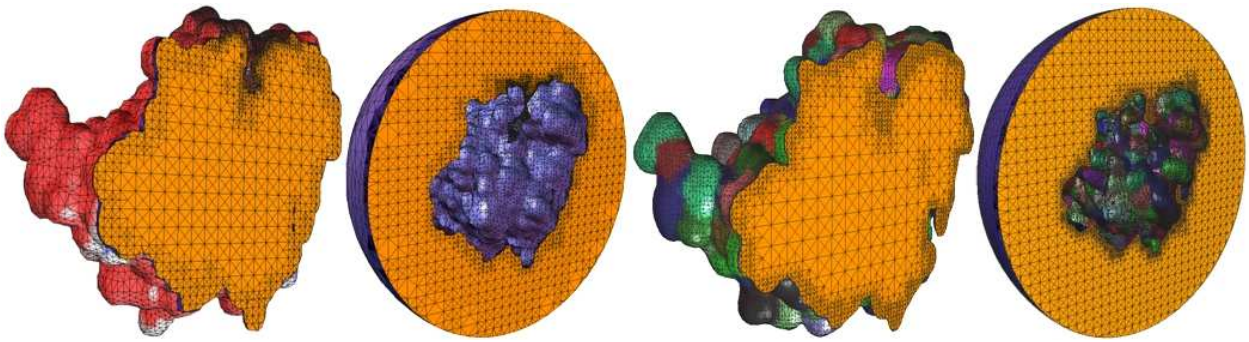


Figure 3.29: Interior and exterior tetrahedral meshes of monomeric mAChE. The left two pictures conform to the SAS with inflation $\sigma = 2$, and the right two pictures conform to the surface constructed from Gaussian summation with $p_1 = 0.25$, $p_2 = 1.0$. From left to right: (65147 vertices, 323442 tets), (121670 vertices, 656823 tets), (103680 vertices, 509597 tets) and (138967 vertices, 707284 tets). The color shows electrostatics potential (leftmost) and residues (the right two).

Tetrameric mAChE: We also generated adaptive tetrahedral meshes for the acetylcholinesterase in tetrameric form, with two different arrangement of the monomers. Each monomer has an active site accessible though a long narrow gorge (20 Angstrom), so there are a total of four gorges. Fig. 3.30 shows the two crystal structures. In the first crystal structure, two gorges are partially blocked, while the other two are completely accessible to solvent. In the second one, all the four gorges are open. Each of the adaptive meshes have finer triangles around the region of the four gorges. These meshes are also used in calculating the diffusion rate constant [119].

Ribosome: Ribosomes are macromolecular complexes responsible for the translation of mRNA into proteins. These complexes consist of two subunits: the larger 50S and the smaller 30S, both of the subunits are composed of rRNA and protein constituents. Atomic level, residue-level and low resolution structure models were constructed from density maps as shown in Fig. 3.3 and 3.5. The constructed exterior meshes are being used for the finite element solution of the Possion-Boltzmann equation [27]. Fig. 3.31 show interior and exterior meshes of the Ribosome 30S/50S.

We have developed a quality molecular meshing approach directly from PDB molecular structural data, with adaptivity at prescribed active sites on the molecular surface. Our generated meshes continue to being used in several boundary/finite

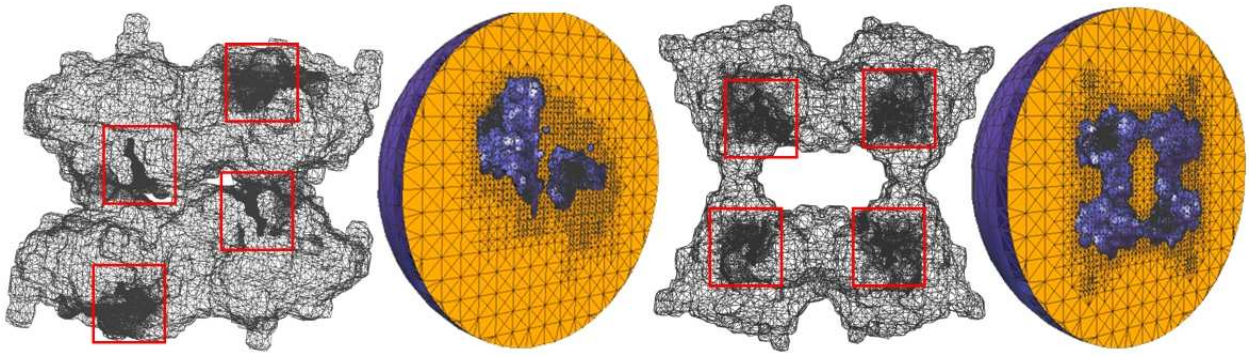


Figure 3.30: Interior and exterior tetrahedral meshes of tetrameric mAChE, $p_1 = 0.5$, $p_2 = 1.0$. The left two pictures show the 1st crystal structure 1C2O (133078 vertices, 670950 tets), and the right two pictures show the 2nd one 1C2B, (106463 vertices, 551074 tets). Gorges are shown in red boxes.

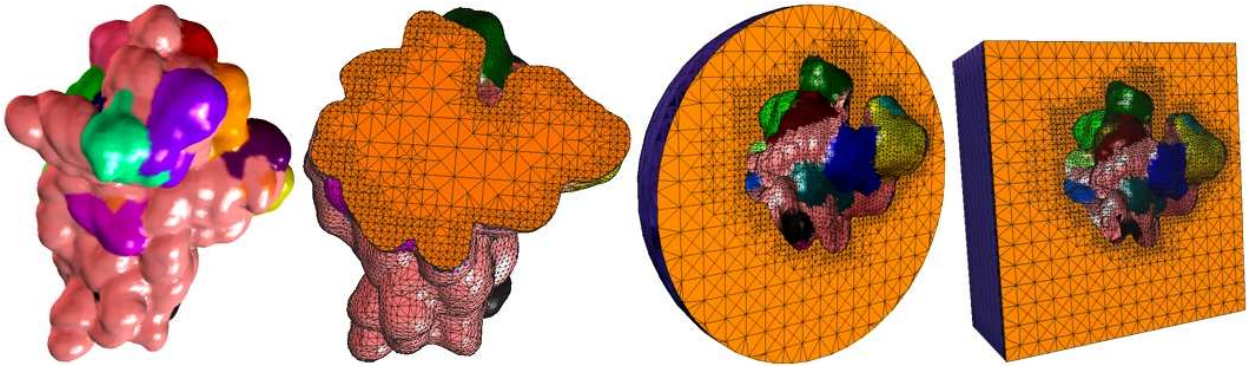


Figure 3.31: Interior and exterior tetrahedral meshes of Ribosome 30S, low resolution, $p_1 = 0.03125$, $p_2 = 1.0$. From left to right: (33612 vertices, 163327 tets), (37613 vertices, 186496 tets) and (40255 vertices, 201724 tets). The pink color shows 16S rRNA and other colors show proteins.

element biophysics simulations [108, 109] [119].

3.6 Multi-resolution Surfaces

Exploratory visualization of large molecules and even larger biomolecular complexes (LBCs) is useful in understanding the structure of the entity in question and visually inspect its bio-chemical properties. Although, advancement in computer graphics has made this exploration possible, it is still extremely memory intensive and far from being real-time when it comes to visualization of large macromolecules and molecular complexes at high resolution. As the resolution of the representation is varied, different features of the molecules are discernable. At high resolutions, near the 1 and 2 Å range, one can distinguish atoms from each other, showing a very high detailed map of the molecule. At this atomistic resolution, biologists can study the active sites in detail. Active sites, or activity sites where different proteins bind and interact, or small molecules (ligands) dock with proteins are often studied by molecular biologists at this fine resolution. For example, during the oxy-deoxy process in the hemoglobin molecule and the binding/dissociation of the oxygen ion with the Iron ion, the activity site is the region surrounding the heme group of atoms containing the Iron ion, and hence the area of interest and activity is necessarily best visualized there at atomistic detail(resolution), while the rest of the atoms of the protein scaffolding the heme could be captured at coarser resolutions. In figure 3.32, we show an example of this multiresolution visualization of the hemoglobin molecule with the region surrounding one of the active sites depicted at a finer resolution. At lower resolutions, between 5 to 10 Å, secondary structures become more apparent in volumetric maps. Imaging data of large structures like icosahedral viruses, captured at very low resolutions show the arrangement of protein capsomers on their genomic enclosing shell. When scientists look at each capsomer in a higher resolution, the arrangement of proteins, then the structures in the proteins and finally the atomic arrangements become apparent. In figure 3.33, we show the Large Ribosomal Subunit at two different resolutions, bringing out different sets of features independently. Hence we see that studying large biomolecular complex (LBC) structures involves visualization at multiple levels of resolution.

3.6.1 Interactive Exploratory Visualization

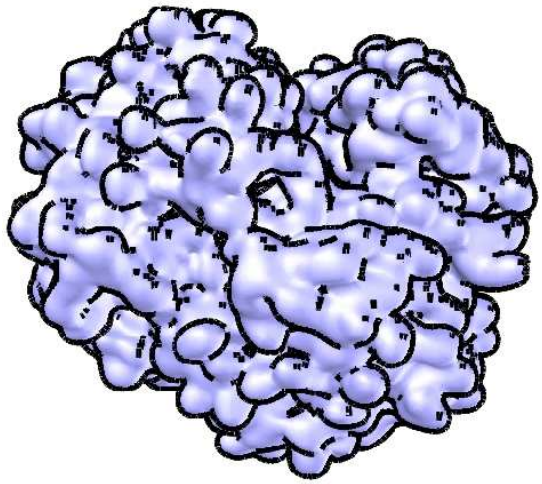
This program takes an ASCII formatted file containing the center coordinates commonly referred to as the PDB (Protein Data Bank). Each atom's type gives it a unique radius. The user can also select to color the resulting adaptive meshes by properties like the chain number etc. The interface allows the user to change the resolution inside and outside the *rover* and the isovalues. A set of three axis, aligned with the global x, y and z axes (colored red, green and blue for the positive axes) are used to both move and resize the *rover* . A wireframe of a cube is rendered to show the outline of the *rover* . These geometry are rendered with no depth in OpenGL to always keep them on top (In figure 3.34, we have added depth to the *rover* to show the reader the actual position). Knobs at the end of the axes are used to resize the inner region. The traditional user interface transformations like translation, zoom and rotation are provided.

On loading a PDB file, an adaptive surface is extracted with the *rover* in a default position, the outer region at low resolution 5Å and the inner region at atomic resolution and more densely sampled. UI widgets are provided for the user to change all of the above parameters.

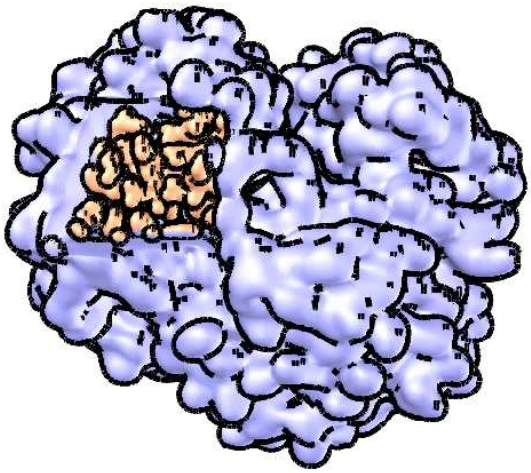
Autonomous Movement of *Rover*

We also generate a trajectory for the *rover* to move autonomously by tracing a path within the pockets (depressions) and tunnels (through holes) on the molecular surface that we identify. For this purpose, we realy heavily on the critical points of the distance function imposed by the molecular surface. The distance function h_S assigns every point x its distance to the surface S. When S is known only via a discrete representation, for example a set of points P, one approximates h_S by h_P where at every point $x \in \mathbb{R}^3$, the function value of h_P gives the distance to the nearest sample point $p \in P$. The critical points of h_P are of four types - maxima, index-1 saddle, index-2 saddle, minima and all four types of critical points can be detected along with their indices by using Voronoi and Delaunay diagram of P [107].

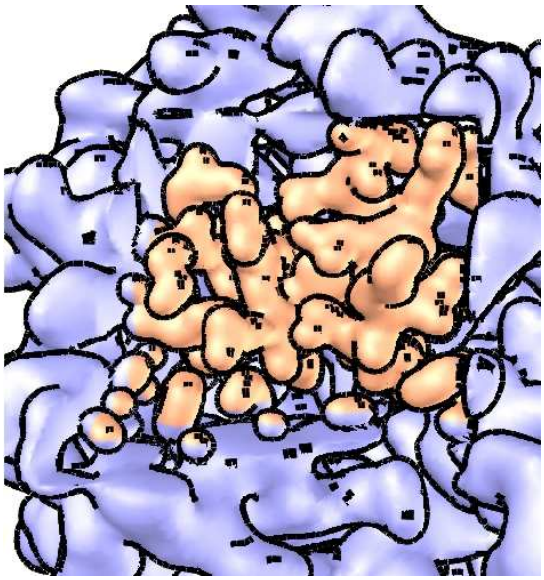
To automate the movement of the *rover* , we first detect the pockets and tunnels of the molecular surface. A pocket is a depression on the surface with narrower mouth and a tunnel is a through hole on the surface that contributes to the genus of the surface. In order to detect these topological features, we cluster the maxima and index 2 saddle points lying exterior to the surface and compute their stable manifolds. For details of the algorithm, see [9].



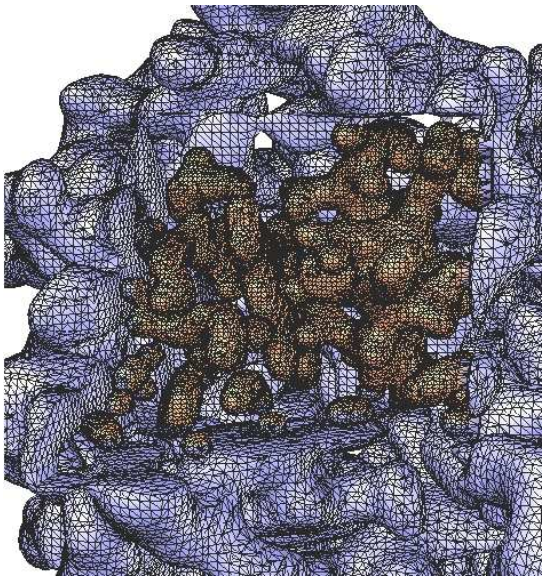
(a) The entire hemoglobin molecule is represented at a coarse resolution, showing the main globular chains.



(b) The *Rover* is moved to a region of interest, which is resolved at a higher resolution.

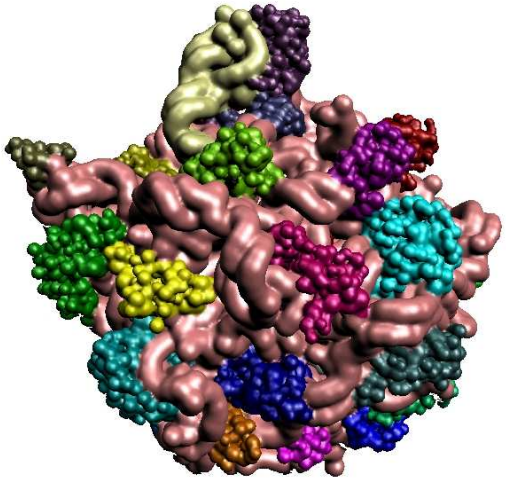


(c) A zoom into the *Rover* showing smooth adaptive surfaces.

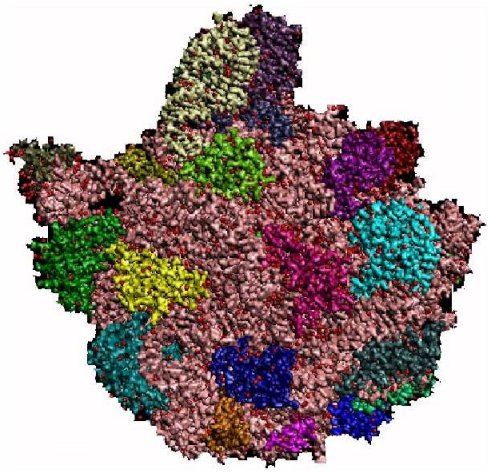


(d) The wireframe rendering shows crack free isocontouring.

Figure 3.32: A single resolution image of the hemoglobin molecule hides important active site details. We provide the user with a *rover* to dynamically compute regions of interest with higher resolution using a combination of novel fast summation algorithms and smooth dual contouring techniques.



(a) A lower resolution model showing the different RNA chains and proteins in different colors. The tertiary and global structure of the molecule is apparent at this resolution.



(b) The Ribosome is rendered at higher resolution, showing near atomic details. While this is useful to study a small portion like an active site, this destroys all global information in the image.

Figure 3.33: Comparison between details revealed by high and low resolution molecular surfaces.

Once these features are detected, the task is to build a trajectory through these features so that the *rover* can follow the path. To build one such path, we restrict our computation only within each pocket or tunnel and collect the index 1 and index 2 saddle points lying inside these features. We then compute the unstable manifold of the saddle points. The unstable manifold of an index 2 saddle point (U_2) is one dimensional, where as the unstable manifold of the index 1 saddle point (U_1) is two dimensional. Computation of these structures has been described in detail in [57]. The U_1s are further starred to remove the tiny, thin patches along the path and a clean one dimensional polylinear trajectory is created. We can this trajectory T. The distance function is then sampled on T at close enough intervals in order to estimate the dimensions of the *rover* at every sample point on T. The dimension of the *rover* is then set to the maximum distanxe function value within the pocket/tunnel. Figure 3.35 shows the performance of the feature (pocket and tunnel) identfication along with the trajectories of the *rover* computed inside them.

3.6.2 Multiresolution Molecular Surfaces

Let us consider an LBC with M atoms centered at $\mathbf{c}_i$, and radii $r_i, 0 \leq i \leq M$. Since we are dealing with proteins and RNA, we can simplify our problem by assuming that the radii come from a small discrete set, so each set can be computed separately and summed up (the fast summation algorithms do not consider different radii for the kernels). Let the roving cube break this set into two disjoint sets V_{out}, V_{in} with $M_{out}, M_{in} : M_{out} + M_{in} = M$ atoms each. We allow three different kinds of dynamic updates. First, the resolution of the inner and outer functions (f_{out}, f_{in}), controlled by changing the parameters β_{out} and β_{in}. The isovalues in each region (iso_{out}, iso_{in}) can be changed, showing the skin (molecular surface at isovalue 1) and the backbone (regions of higher density). The user is also allowed to roam the dataset, visualizing regions in higher resolution using the *Rover* . In figure 3.36, we show the main steps in our algorithm. These interactions require maintaining a set of active atoms in the *Rover* , dynamically updating the new function and a smooth adaptive isocontouring algorithm.

Atom set query

To speed up the query of particles lying in and out of the sub-volume we construct of a range tree on the input centers [65]. A range tree is an $O(\log(M))$ method (for an M atom system) for determining a subset of the input which lies inside any given range. We consider the construction of the range to be pre-processing and is not included in the actual computation. On every repositioning of the sub-volume, we have to query the range tree to obtain the subset of centers inside the sub-volume and its complement set (centers that are outside the sub-volume.) Note that this is two range queries and not one. The complexity

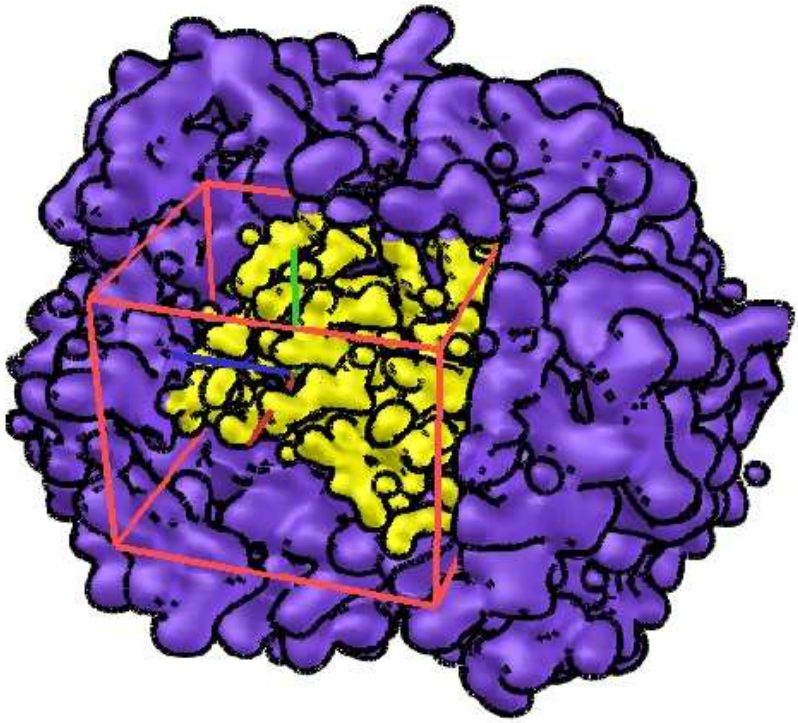


Figure 3.34: The *rover* in action.

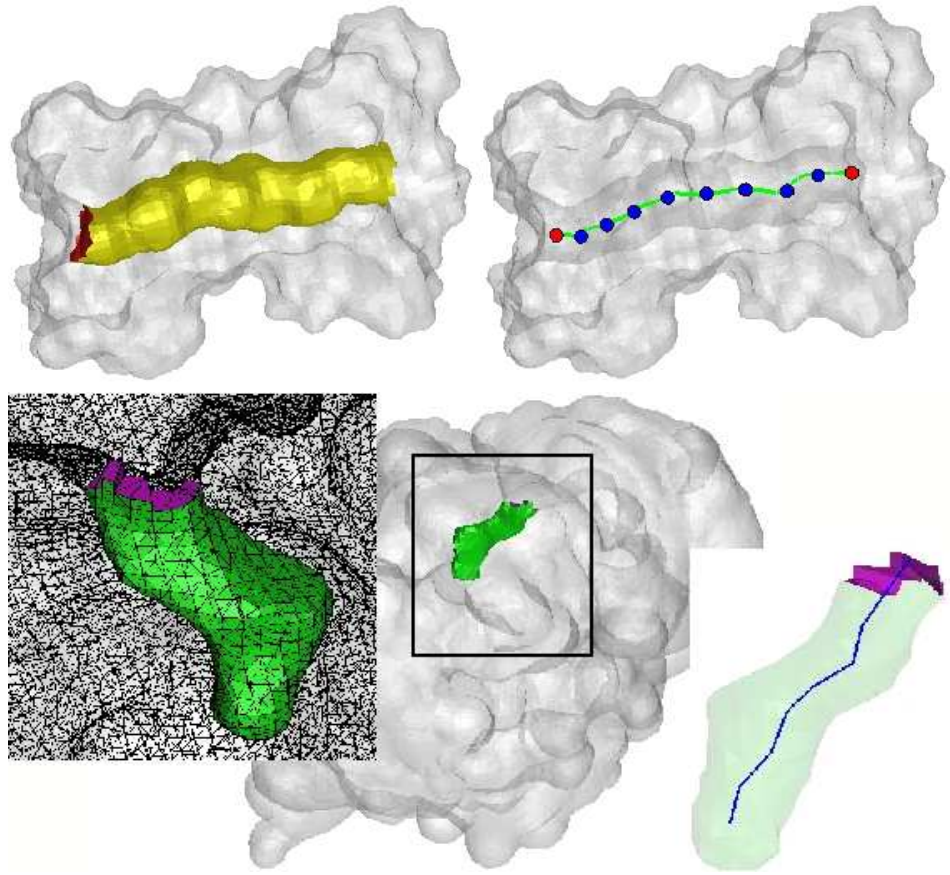


Figure 3.35: Top row: The surface of gramicidin D ion channel is shown (transparent) with the detected tunnel inside (yellow body and red mouths). The subfigure in the right pannel shows the path detected via the unstable manifolds of index 1 and 2 saddle points falling inside the tunnel. Bottom row: The molecular surface of mouse Acetylcholinesterase (mAchE) is shown (transparent) with the pocket (green) identified. The active site of this molecule resides inside the gorge enclosed by the pocket. A closeup of the triangulation is shown in the left corner and the path detected inside the pocket is shown in the right corner.

for the range query given a bound is $O(\log(M+k)))$ for some constant k. Range tree also carries the storage overhead of $O(M \log M)$.

Fast density function update

For interactive visualization of dynamically updated *Rover* and resolution, isovalue parameters, we need algorithms to compute the functions f_{out} and f_{in} efficiently. When the *Rover* cube is moved, a new volume f_{in} needs to be computed for isosurfacing. This is probably the most common update operation performed during interaction. We provide precomputation based algorithms to speed up this update.

Direct summation: If we have N output points and M Gaussians, then we can compute the sum at all the points in O(NM) time and O(M+N) memory. If the rate of decay of the Gaussian is high, then we can use a truncated Gaussian and update only around it. Consider a width of w for a truncated Gaussian. Using local summations, we get a computation cost of Mw^3. For grids (where N is large, typically 128^3 to 256^3), and slow decay Gaussians, this operation can be expensive.

Fast summation algorithms The Gaussian function summation can be expressed as a convolution. This allows the functions f_{out}, f_{in} to be quickly computed with a change of β_{out}, β_{in} using the Fourier transform. There are fast Gaussian summation algorithms including general multi-pole methods. We follow the method outlined in [91],[16] to compute both the functions. The cost of updating the function in the *Rover* is seen to vary as $\bar{M} \log \bar{M}$ for $\bar{M}$ atoms influencing the inner function. In the

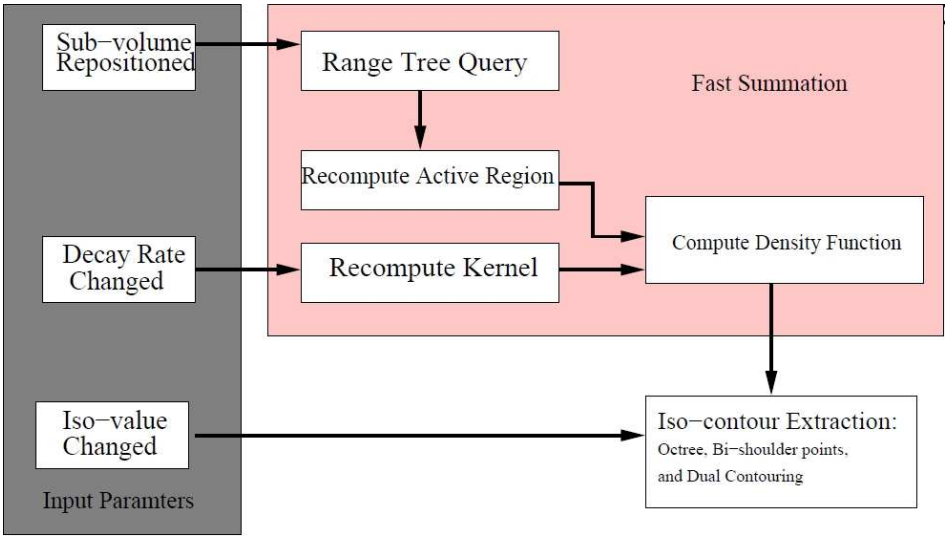


Figure 3.36: A block diagram showing the system implementation.

case of LBCs where a large number of atoms are represented in a smaller subgrid, this dependence on the number of atoms will be a bottleneck to the update process. In [16] the summation is approximated as a sum of locally defined functions ϕ such as bsplines or truncated Gaussians (see also [92]): $f(\mathbf{x}) = G \otimes \sum_{i=0}^{M-1} \delta(\mathbf{x}-\mathbf{c}_i) \approx \phi \otimes \sum_{\mathbf{i}\in I_\mathbf{n}} k_{\lfloor \mathbf{x}-\mathbf{i} \rfloor} \delta(\lfloor \mathbf{x}-\mathbf{i} \rfloor)$ where ϕ is a locally supported function, k is a the set of coefficients for ϕ. $I_\mathbf{n}$ is a cubic set of indices: $\{\{i,j,k\} : i,j,k \in -n/2 \,..\, n/2-1\}$. Their algorithm uses $O(n^3 \log n + Mm^3 + N^3 \log N)$ time and $O(n^3+N^3+M)$ memory, where n is taken to be the same order of M and controls the error with m for this precomputation. The truncated function has support of size $2m+1$. Using this higher order grid, a simple convolution with ϕ is used to compute f_{in}. Thus, for grids with low atom density, a range tree is used to obtain atoms within the new *rover* and the function computed. For grids with high atom density, we switch to using a precomputed higher order grid. A convolution with ϕ with the correct subset of k gives us f_{in}. This step is independent of M, but requires the precomputation of the coefficients k.

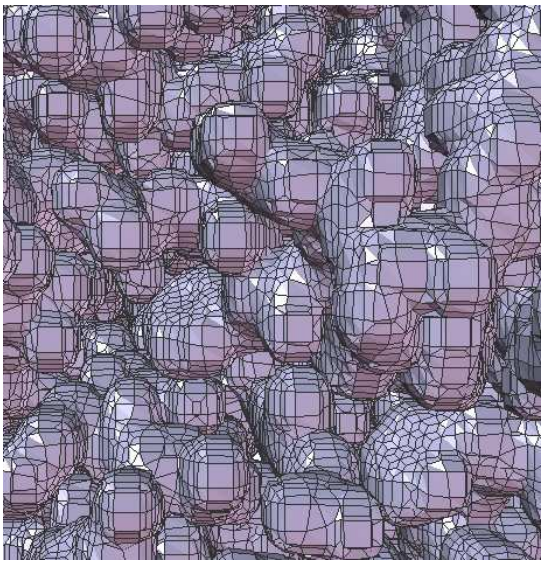
Update β_{out}: The Fourier transform of the new Gaussian can be done analytically. The cost is in multiplying the two set of frequencies ($O(N_{out}^3)$) and performing an inverse Fourier transform ($O(N_{out}^3 \log N_{out})$). We also need to isocontour the outer and boundary regions.

Update β_{in}: For the range tree partitioning method, we obtain the list of atoms within the *Rover* in $O(M \log M)$ time and perform a fast summation in approximately $O(M_{in} + N_{in}^3 \log N_{in})$ time [16]. The higher order grid update is more expensive. We would need to precompute the coefficients k again everywhere, and then perform the convolution of ϕ with a subset to update f_{in}.

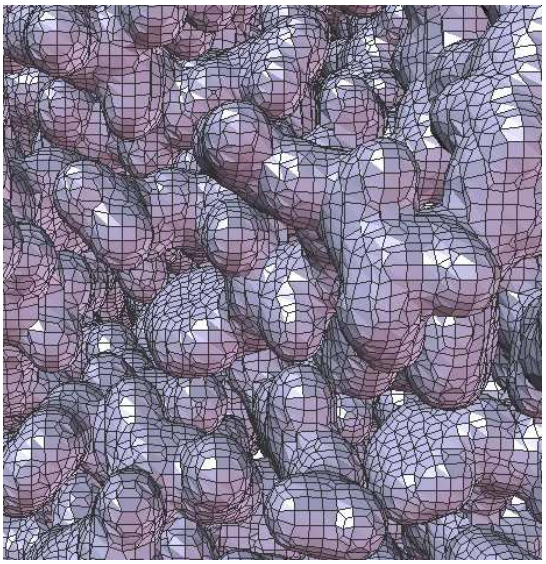
Rover *update:* The current β_{in} gives us the width of the truncated kernels. Using the range tree data structure, we query for the set of atoms influencing the *rover* and perform a fast summation to update f_{in}. The mesh needs to be recomputed. If the movement of the rover was small we can reuse the previously compute f_{in} and its corresponding isosurface. The movement of the *rover* also allows us to perform any well known caching algorithm to store frequently visited active sites in high resolution in our cache.

Smooth adaptive isocontouring

We use dual contouring [62] as our surface extraction algorithm. Dual contouring uses the dual map of the primal contouring (Marching Cube) and normal tagging (Hermite Data) to extract the iso-contour. We choose this method primarily because of it avoids the degenerate cases of primary contouring where cracks are introduced. Our algorithm differs from that of Ju et el [62] in the minimizer computation. Instead of Hermite data and QEF minimizer, we compute a vertex termed *bishoulder point* [76] that closely approximates the true contour. In figure 3.37, we show how the bishoulder point is a better minimizer for smooth functions (If we were to perform isocontouring of objects with sharp edges, we would not use this technique). Lopes



(a) The traditional QEF minimizer forces sharp edges in the function, leading to unrealistic isosurfaces for the smooth electron density function of molecules.



(b) A bishoulder point based dual contouring algorithm results in smooth contours.

Figure 3.37: The hemoglobin molecule (PDB:1A00) is rendered using the traditional QEF minimizer and our bishoulder point based dual contouring algorithm. We see that our method leads to smooth realistic isosurfaces.

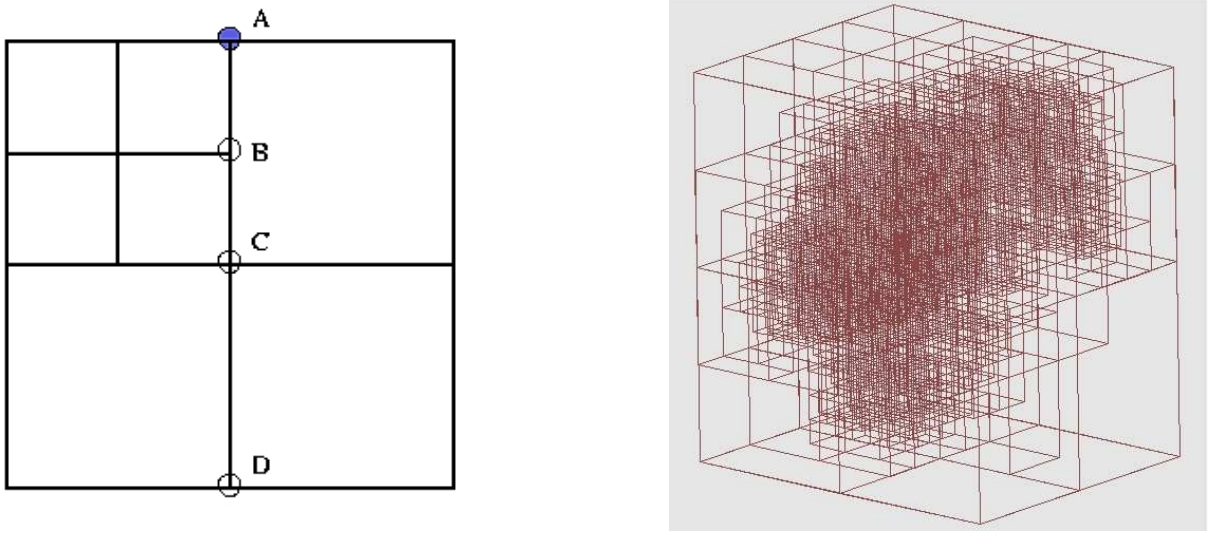
and Brodlie [76] detailed in their reports the mathematical reasoning behind bishoulder point's accuracy. We retain the general principle of dual contouring in generating quads. The algorithm traverses each cell recursively. Upon locating a leaf node that contains iso-contour, we check the *sign-change edges* of the cell. Consider the interval defined by the function values of an edge's endpoints. If the iso-value is within that interval, then the edge is a sign-change edge. For example, in Figure 3.38(a), AB, AC, and AD are sign-change edges. We refer to a sign-change edge as being *minimal* when it is an edge of the smallest cell neighboring itself. In Figure 3.38(a), AB is the minimal sign-change edge. If the edge is not minimal, then it is skipped. If the sign-change edge is minimal, then a quad is created, connecting the four bishoulder points of the four cells neighboring the sign-change edge.

Our dual contouring algorithm uses an octree as its surface extraction data structure. The octree is a recursive subdivision of space into variable-sized cells. Its structure is inherently similar to that of a uniform cell division in MC. We can exploit its subdivision structure to produce adaptive cells. This is a simple matter of subdividing cells that borders the sub-volume. Figure 3.38(b) illustrates the use of adaptive cell construction with octree.

3.6.3 Examples and Timings

We have implemented the algorithm for the *Rover* using standard C++ and QT (www.trolltech.com). All the experiments were done on an AMD Opteron 246. We looked at different size molecules, ranging from the hemoglobin (PDB:1A00) which has approximately 4000 atoms, to the Ribosomal subunits (PDB:1J5E,PDB:1JJ2) which have around 100,000 atoms and the Human Rhinovirus Virus, which has over a million atoms.

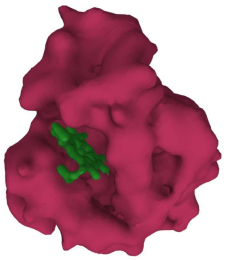
In figures 3.39 and 3.40, we show both the methods of multiresolution our algorithm provides. First, the site of interest in the molecule is smoothed using a sharper kernel than the rest of the molecule to differentiate it and provide higher detail in that region. Next, we use our smooth, adaptive dual contouring algorithm to extract an adaptive mesh which provides high detail at the region of interest and lower resolution elsewhere. These two parameters, the kernel decay rate and the isocontouring mesh refinement can be controlled by the user to obtain feature based functions and visualizations. In figure 3.39, the heme is the active site of the molecule and the region of interest. Hence, the atoms of the heme are smoothed using a gaussian at a 1A resolution while the rest of the molecule was blurred a coarser resolution of 3A. Also, by allowing the user to provide



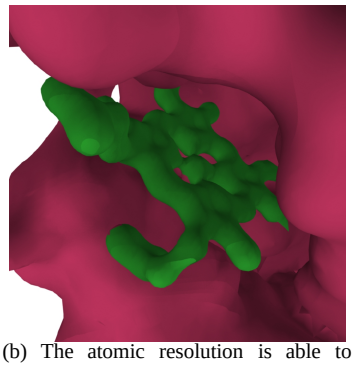
(a) (b)

Figure 3.38: (a) is a 2D analog for the sign-changed edge. (b) is an example of adaptive cell construction with octree.

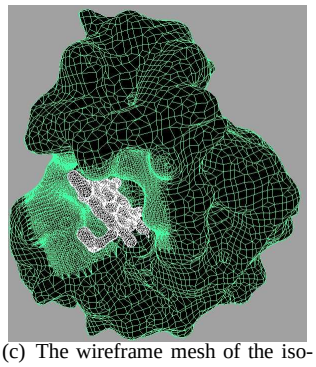
a cube around the heme, the adaptive isocontouring algorithm was used to extract a higher resolution mesh around the active site. In figure 3.40, we use the same two multiresolution techniques to show the trimer, a unit of symmetry of the icosahedral Human Rhinovirus. The Large and small Ribosomal subunits are responsible in part for the creation of proteins and widely studied. The main active sites are called as the A,P and E sites. There is also the formation of the cavities and exit tunnels in the bound complex. It is a rich complex with various features including small proteins helping its activity. In figure 3.41, we see that increasing the resolution in the *Rover* and changing properties like the grid spacing and color helps the user focus on the regions of interest. The smoothness of the dual contouring is maintained even though there is a sharp increase in both resolution and sampling density, as shown in figure 3.42. Timing results and the number of quads generated in the adaptive isocontouring are presented in table 3.6.3.



(a) The heme of the myoglobin is colored differently using an additional color function summation. The function around the heme is extracted at higher resolution.

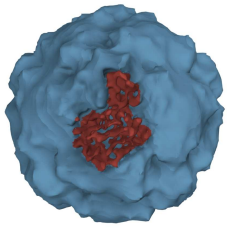


(b) The atomic resolution is able to clearly distinguish the ring of atoms composing the heme structure.

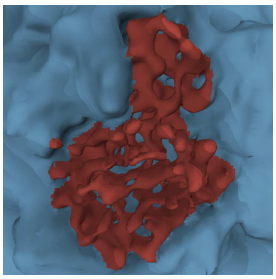


(c) The wireframe mesh of the isocontour.

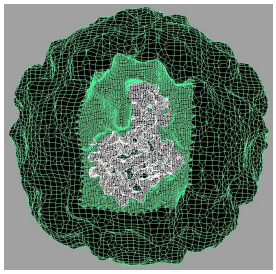
Figure 3.39: The myoglobin molecule showing the heme structure. We used a kernel with sharp decay rate, modeling the atomic structure for the heme where the oxy-deoxygenation takes place. A coarser rate of decay was used for the rest of the molecule as the active site is of primary interest for the end user. The region around the heme was extracted at a higher resolution using an adaptive isocontouring algorithm. To maintain the required features, fewer frequencies were required for most of the molecule as compared with the heme.



(a) The virus with one capsid shown in higher resolution.



(b) A zoomed view to present the smooth isocontour.



(c) The wireframe mesh of the adaptive isocontour.

Figure 3.40: The Human Rhinovirus, an icosahedral virus (1FPN.PDB) showing a trimer (a symmetry unit) in higher resolution. The trimer was smoothed using a sharper gaussian than the rest of the virus. Also, an adaptive isocontouring technique was employed to extract a higher resolution mesh in a cube containing the trimer.

Related Work

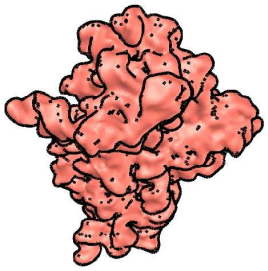
Implicit Solvation Surface from volumetric Density Maps (Radial Basis Splines, $C^i n f$)

There are three different yet often used molecular interfaces [98], the van der Waals surface (VWS), the solvent-accessible surface (SAS) and the solvent-excluded surface (SES) [41] or sometimes called the Lee-Richards surface [69]. The SES or the Lee-Richards surface is by far the molecular surface of choice for solvation energy calculations [60, 71], and is the surface for our meshing approximations.

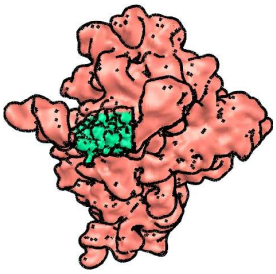
According to the properties of molecular structures, Laug and Borouchaki used a combined advancing front and generalized Delaunay approach to mesh molecular surfaces [68]. Algorithms were developed for sampling and triangulating a smooth surface with correct topology [3]. Skin surfaces, introduced by Edelsbrunner in [44], have a rich combinational structure and provide a smooth alternative to the Lee-Richard's surface. Cheng et. al [36] maintained an approximating triangulation of a deforming skin surface. Bajaj et. al [10] give NURB approximateion of Lee-Richards molecular surfaces as well as present methods to maintain molecular surfaces for varying solvent radii [13]. Compressed volumetric representation of molecular surfaces is given in [6]. Simplex subdivision schemes are used to generate tetrahedral meshes for molecular structures in

PDB ID	Number of particles	Coarse Volume Summation (secs)	Fine Volume Summation (secs/atoms)	Polygonization Time	Number of Quads
Coarse Volume Resolution: 64^3 | Subvolume Resolution: 52^3					
1A00	4770	0.94	0.32/470	0.72	73181
1J5E	51743	0.94	0.66/10997	0.68	43092
1JJ2	90418	0.93	0.63/10426	0.66	49176
Coarse Volume Resolution: 128 | Subvolume Resolution: 104^3					
1A00	4770	2.95	3.09/474	4.32	313421
1J5E	51743	3.68	2.95/11180	3.82	216307
1JJ2	90418	3.02	2.95/10598	3.54	251904
Coarse Volume Resolution: 256 | Subvolume Resolution: 206^3					
1A00	4770	10.81	25.79/474	28.76	1257644
1J5E	51743	10.89	27.97/11180	29.04	926541
1JJ2	90418	10.35	26.32/10598	23.55	1074166

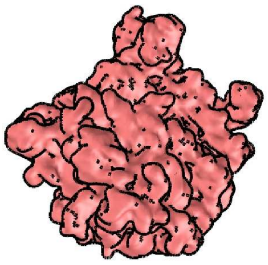
Table 3.6: This table shows the timing results of our method. All tests are performed on AMD Opteron 246 with 16GB of memory. The subvolume is sampling $(40\%)^3 = 6.4\%$ of the entire input domain. In the third set of results, we perform the surface extraction at very high resolutions, where the small domain in the *Rover* is sampled at 206^3.



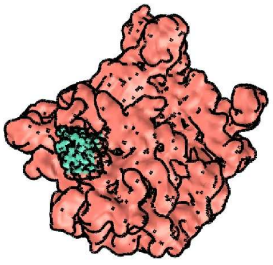
(a) The Small Ribosomal Subunit at a single resolution.



(b) The *rover* is used to visualize a region of interest at higher resolution.



(c) The Large Ribosomal Subunit at a single resolution.



(d) An active site is extracted at higher resolution showing atomic details. The rest of the molecule still presents the global features.

Figure 3.41: We present global and local views in the Ribosomal subunits and contrast it with using a single overall resolution.

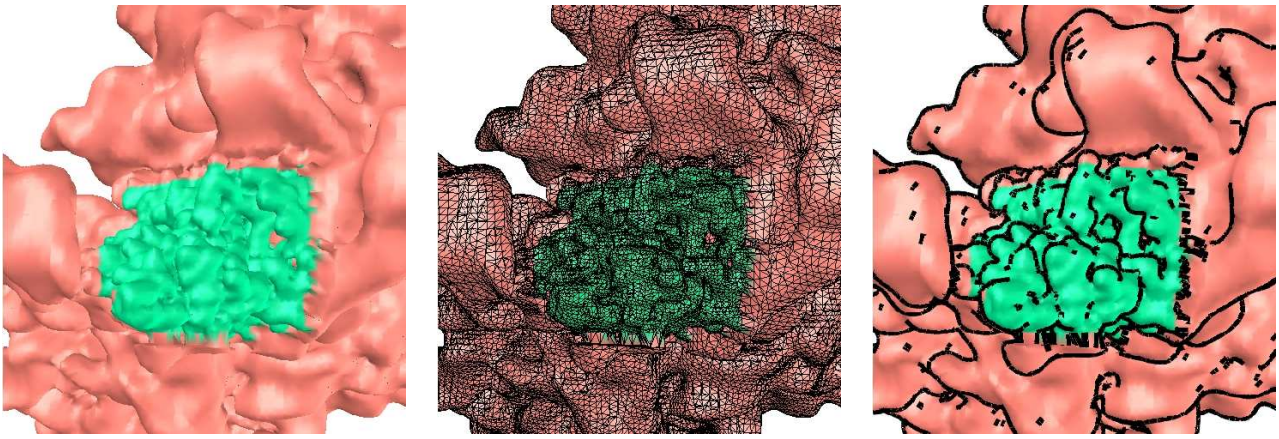


Figure 3.42: Close up images of an active site in the Small Ribosomal subunit, showing the smooth adaptive dual contour.

solving the Poisson-Boltzmann equation [60]. Gaussian functions have been used to construct density maps [29, 58, 83, 1, 81], from which implicit solvation models are approximated as an isocontour [58, 71, 52]. However, it still remains a challenge to generate quality and adaptive triangular and tetrahedral meshes for arbitrary molecular structures.

Algebraic Shell Splines (C^1)

One way to generate an analytical representation of the molecule is to define an analytical volumetric density function, for example, the summation of Gaussian functions [59], Fermi-Dirac switching function [73], or piecewise polynomials [72], and approximate the SES by an iso-contour of the density function. Techniques of fast extracting an iso-contour of smooth kernel functions are developed in [5][19]. However the error of the generated isosurface could be large and result in inaccurate energy computation. A NURBS representation for the SES is presented in [11]. Although it provides a parametric approximation to the SES, it does not solve the singularity problem. Edelsbrunner [45] defines another paradigm of a smooth surface referred to as *skin* which is based on the Voronoi, Delaunay, and Alpha complexes of a finite set of weighed points. The *skin* model has good geometric properties such as it is free of singularity and it can be decomposed into a collection of quadratic patches. Triangulation schemes based on the *skin* model are provided in [34][35]. However when applied to the energetic computation, the *skin* triangulation which in fact is a linear approximation to the SES has to be very dense to gain accuracy, which causes oversampling on the surface and hence makes the computation very slow.

Meshing of Molecular Interfaces

Mesh Generation: As reviewed in [89, 112], octree-based, advancing front based and Delaunay like techniques were used for triangular and tetrahedral mesh generation. The octree technique recursively subdivides the cube containing the input geometry until the desired resolution is reached [104]. Advancing front methods start from a boundary and move a meshed front from the boundary towards empty space within the domain [50, 75]. Delaunay refinement is used to refine triangles or tetrahedra locally by inserting new nodes to maintain the Delaunay criterion ('empty circumsphere') [38]. Sliver Exudation [37] was used to eliminate slivers (bad aspect ratio). Shewchuk [105] solves the problem of enforcing boundary conformity by use of constrained Delaunay triangulations (CDT).

The predominant algorithm for isosurface extraction from volume data is Marching Cubes (MC) [77], which computes a local triangulation within each cube to approximate the isosurface by using a case table of edge intersections. MC was extended to extract tetrahedral meshes between two isosurfaces [51]. A different and systematic algorithm was proposed for interval volume tetrahedralization [86]. By combining SurfaceNets [56] and the extended Marching Cubes algorithm [64], octree based dual contouring [61] generates adaptive multiresolution isosurfaces with preservation of sharp features. The dual

contouring method has also been extended to extract adaptive and quality tetrahedral meshes from volumetric imaging data [120, 121].

Quality Improvement: Algorithms for mesh quality improvement can be classified into three categories [112, 89]: local coarsening/refinement by inserting/deleting points, local remeshing by face/edge swapping and mesh smoothing by relocating vertices.

Laplacian smoothing relocates vertex position at the average of the nodes (vertices) incident to it [48]. Instead of relocating vertices based on a heuristic algorithm, the optimization technique measures the quality of the surrounding elements to a node and attempts to optimize it. The optimization-based smoothing yields better results, nevertheless it is more expensive than Laplacian smoothing. Therefore, a combined Laplacian/Optimization-based approach was recommended [33, 49]. The Laplacian operator was discretized over triangular meshes [82], and geometric flows have been used in surface and imaging processing [101, 117]. Physically-based simulations are used to reposition nodes [74]. Anisotropic meshes are obtained from bubble placement and equilibrium [106]. Mesh regularization was discussed in [88, 115].

Multi-resolution Surfaces

The electron density and shape are used in a similar sense in the literature with respect to molecular surface modeling. There have been many definitions for the molecular surface of biomolecules.

Molecular surface visualization Many smooth models of the molecular surface have been proposed, including explicit rolling ball blend models and implicit Gaussian models. The Solvent Accessible Surface (SAS) [70] is defined as the locus of the center of the probe when it is in touch with the molecule and not intersecting it. The Solvent Excluded Surface (SES) [94] provides a smooth contour of the molecular surface. It is the surface which envelopes the region not accessible to the solvent. Later analytical expressions for the patches which make up the surface were given in [42] and [39]. A general surface description for a set of points is given in [46]. Non Uniform Rational BSplines (NURBS) descriptions for the patches of the molecular surfaces are given in [12] and [24]. A triangulation based on a similar idea of using weighted Voronoi and Delaunay triangulations is given in [4]. Fast computations of the SES is described in [99] and [97]. In [100], the authors describe an algorithm to update the triangulation if only a part of the molecule changes positions. Using Gaussians around the atom centers to represent the van der Waals region of influence, many authors represented molecular surfaces as isocontours of this field [30], [58]. A similar function was used at the interface to model varying probe radii by [113]. Fast computation of the NURBS representation when the probe radius changes was given in [24]. Due to its computational efficiency, the gaussian function is being used more frequently even in docking algorithms [95].

Resolution of density maps The electron density of an atom at a point $\mathbf{x}$ is represented as a Gaussian function: $f(\mathbf{x}) = e^{\beta(\frac{|\mathbf{x}-\mathbf{c}|^2}{r^2}-1)}$, where $\mathbf{c}, r$ are the center and radius of the atom. If we consider the function value of 1, we see that it is satisfied at the surface of the sphere $(\mathbf{x} : |\mathbf{x}-\mathbf{c}| = r)$. Using this model, the electron density of a LBC with M atoms at $\mathbf{x}$ is just a summation of Gaussians: $f(\mathbf{x}) = \sum_{i=0}^{M-1} e^{\beta(\frac{|\mathbf{x}-\mathbf{c}_i|^2}{r_i^2}-1)}$. Here β is a parameter used to control the rate of decay of the Gaussian and known as the *blobbiness* of the Gaussian [30], [58]. Isosurfaces of this function with isovalue 1 are extracted using traditional isosurfacing methods like primal/dual marching cubes. In [58] $\beta = -2.3, f(\mathbf{x}) = 1$ is provided as a good approximation to the Molecular Surface. This is used as the *atomic resolution* (res) model's parameter. Through correspondence with Dr Wah Chiu's group from from Baylor College of Medicine, Houston USA, and from EMAN [79], we also have the following parameters for the Gaussians: The Gaussian is weighted by the number of electrons τ_i for the i^{th} atom. The resolution is taken as the distance in Å where the Gaussian function decays to either 0.5 or 1/e of the peak. Hence, $f(\mathbf{x}) = \sum_{i=0}^{M-1} \tau_i e^{-a|\mathbf{x}-\mathbf{c}_i|}$, $a = \frac{\log 2}{res^2}$, *or* $a = \frac{1}{res^2}$. Another commonly used definition is method is in the Fourier domain: $\mathscr{F}(e^{-ax^2})(k) = e^{-\pi^2 k^2/a}$, $a = \frac{\log 2\pi^2}{res^2}$, *or* $a = \frac{\pi^2}{res^2}$.

Adaptive isocontouring Marching Cubes (MC) has been a widely used uniform grid isocontouring algorithm [78]. By storing only the cells containing an iso-contour, uniform data grids are converted adaptively into octrees [103]. Several post-

processing are performed in which cells meeting certain criteria are merged, producing an adaptive contour. Westermann et al. described an octree-based extraction method [114] that patches the cracks produced by traditional MC by limiting difference of levels between neighboring cells to two and by applying a pyramid averaging scheme to cover the cracks with additional polygons. Dual contouring reported in [62] describes a method that contours on the dual graph of MC, producing crack free isosurfaces.

Multiresolution modeling There have been various algorithms used in multiresolution modeling in research areas both in and outside of molecular modeling. A good survey of existing techniques are presented in [53].

Relevant Mathematics

Partial Differential Equations

Fast Summations and Fourier Transforms

Bibliography

[1] R. C. Agarwal. A new least-squares refinement technique based on the fast fourier transform algorithm. *Acta Cryst.*, A34:791–809, 1978.

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

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

[4] N. Akkiraju and H. Edelsbrunner. Triangulating the surface of a molecule. *Discrete Applied Mathematics*, 71(1-3):5–22, 1996.

[5] C. Bajaj, J. Castrillon-Candas, V. Siddavanahalli, and Z. Xu. Compressed representations of macromolecular structures and properties. *Structure*, 13:463–471, 2005.

[6] C. Bajaj, J. Castrillon-Candas, V. Siddavanahalli, and Z. Xu. Compressed representations of macromolecular structures and properties structure. *Discrete Applied Mathematics*, 13(3):463–471, 2005.

[7] C. Bajaj, P. Djeu, V. Siddavanahalli, and A. Thane. Texmol: Interactive visual exploration of large flexible multi-component molecular complexes. *Proc. of the Annual IEEE Visualization Conference*, pages 243–250, 2004.

[8] C. Bajaj, P. Djeu, V. Siddavanahalli, and A. Thane. Texmol: interactive visual exploration of large flexible multi-component molecular complexes. In *VIS'04: Proceedings of the conference on Visualization '04*, pages 243–250, Washington, DC, USA, 2004.

[9] C. Bajaj, A. Gillette, and S. Goswami. Topology based selection and curation of level sets. In *TopoInVis 2007*, Accepted.

[10] C. Bajaj, H. Lee, R. Merkert, and V. Pascucci. NURBS based B-rep models from macromolecules and their properties. In *Proceedings 4th Symposium on Solid Modeling and Applications*, pages 217–228, 1997.

[11] C. Bajaj, H. Lee, R. Merkert, and V. Pascucci. Nurbs based b-rep models from macromolecules and their properties. *In Proceedings of Fourth Symposium on Solid Modeling and Applications*, pages 217–228, 1997.

[12] C. Bajaj, H. Y. Lee, R. Merkert, and V. Pascucci. Nurbs based b-rep models for macromolecules and their properties. In *Proceedings of the fourth ACM symposium on Solid modeling and applications*, pages 217–228. ACM Press, 1997.

[13] C. Bajaj, V. Pascucci, A. Shamir, R. Holt, and A. Netravali. Dynamic maintenance and visualization of molecular surfaces. *Discrete Applied Mathematics*, 127:23–51, 2003.

[14] C. Bajaj, S. Schaefer, J. Warren, and G. Xu. A subdivision scheme for hexahedral meshes. *The Visual Computer*, 18(5-6):343–356, 2002.

[15] C. Bajaj and V. Siddavanahalli. Fast and adaptive surfaces and derivatives computations for volumetric particle data. *CS and ICES Technical Report, the University of Texas at Austin*, 2005.

[16] C. Bajaj and V. Siddavanahalli. Fast error-bounded surfaces and derivatives computation for volumetric particle data. Ices report, The University of Texas Austin, 2005.

[17] C. Bajaj and V. Siddavanahalli. An adaptive grid based method for computing molecular surfaces and properties. ICES Technical Report TR-06-57, 2006.

[18] C. Bajaj and V. Siddavanahalli. Adaptive grid based methods for computing molecular surfaces and properties. Technical Report TR-06-56, Department of Computer Science, The University of Texas at Austin, 2006.

[19] C. Bajaj and V. Siddavanahalli. Fast error-bounded surfaces and derivatives computation for volumetric particle data. ICES Technical Report TR-06-06, 2006.

[20] C. Bajaj, V. Siddavanahalli, and W. Zhao. Fast algorithms for molecular interface triangulation and solvation energy computations. *ICES Technical Report TR-07-06*, 2007.

[21] C. Bajaj, G. Xu, R. Holt, and A. Netravali. Hierarchical multiresolution reconstruction of shell surfaces. *Computer Aided Geometric Design*, 19:89–112, 2002.

[22] C. Bajaj, G. Xu, and Q. Zhang. Smooth surface construction via a higher order level set method. ICES Report 06-18, Institute for Computational and Applied Mathematics, The University of Texas at Austin, 2006.

[23] C. Bajaj and W. Zhao. Fast molecular energetics and forces computation. ICES Report 08-20, Institute for Computational Engineering and Sciences, The University of Texas at Austin, 2008.

[24] C. L. Bajaj, V. Pascucci, A. Shamir, R. J. Holt, and A. N. Netravali. Dynamic maintenance and visualization of molecular surfaces. *Discrete Applied Mathematics*, 127(1):23–51, 2003.

[25] C. L. Bajaj, G.-L. Xu, and Q. Zhang. Higher-order level-set method and its application in biomolecular surfaces construction. *Journal of Computer Science and Technology*, 23(6):1026–1036, 2008.

[26] N. Baker, M. Holst, and F. Wang. Adaptive multilevel finite element solution of the poisson-boltzmann equation ii. refinement at solvent-accessible surfaces in biomolecular systems. *Journal of Comput Chem.*, 21:1343–1352, 2000.

[27] N. Baker, D. Sept, S. Joseph, M. Holst, and J. McCammon. Electrostatics of nanosystems: application to microtubules and the ribosome. *Proceedings of National Academy of Science*, 98:10037–10041, 2001.

[28] D. Bashford and D. A. Case. Generalized born models of macromolecular solvation effects. *Annual Review of Physical Chemistry*, 51:129–152, 2000.

[29] J. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.

[30] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Trans. Graph.*, 1(3):235–256, 1982.

[31] Y. Bourne, J. Grassi, P. E. Bougis, and P. Marchot. Conformational flexibility of the acetylcholinesterase tetramer suggested by x-ray crystallography. *Journal of Biological Chemistry*, 274(43):30370–30376, 1999.

[32] Y. Bourne, P. Taylor, and P. Marchot. Acetylcholinesterase inhibition by fasciculin: crystal structure of the complex. *Cell*, 83:503, 1995.

[33] S. Canann, J. Tristano, and M. Staten. An approach to combined laplacian and optimization-based smoothing for triangular, quadrilateral and quad-dominant meshes. In *7th International Meshing Roundtable*, pages 479–494, 1998.

[34] H. Cheng and X. Shi. Guaranteed quality triangulation of molecular skin surfaces. *IEEE Visualization*, pages 481–488, 2004.

[35] H. Cheng and X. Shi. Quality mesh generation for molecular skin surfaces using restricted union of balls. *IEEE Visualization*, pages 51–58, 2005.

[36] H.-L. Cheng, T. Dey, H. Edelsbrunner, and J. Sullivan. Dynamic skin triangulation. *Discrete Computational Geometry*, 25:525–568, 2001.

[37] S.-W. Cheng, T. Dey, H. Edelsbrunner, M. Facello, and S. Teng. Sliver exudation. *Proceeding Journal of ACM*, 47:883–904, 2000.

[38] P. Chew. Guaranteed-quality delaunay meshing in 3d. In *13th ACM Annual Symposium on Computational Geometry*, pages 391–393, 1997.

[39] M. Connolly. Solvent-accessible surfaces of proteins and nucleic acids. *Science*, 221(4612):709–713, 19 August 1983.

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

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

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

[43] D. Dunavant. High degree efficient symmetrical gaussian quadrature rules for the triangle. *International Journal for Numerical Methods in Engineering*, 21:1129–1148, 1985.

[44] H. Edelsbrunner. Deformable smooth surface design. *Discrete Computational Geometry*, 21:87–115, 1999.

[45] H. Edelsbrunner. Deformable smooth surface design. *Discrete Computational Geometry*, 21:87–115, 1999.

[46] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[47] O. D. Faugeras and R. Keriven. Variational principles, surface evolution, PDE's, level-set methods, and the stereo problem. *IEEE Transactions on Image Processing*, 7(3):336–344, 1998.

[48] D. Field. Laplacian smoothing and delaunay triangulations. *Communications in Applied Numerical Methods*, 4:709–712, 1988.

[49] L. Freitag. On combining laplacian and optimization-based mesh smoothing techniqes. *AMD-Vol. 220 Trends in Unstructured Mesh Generation*, pages 37–43, 1997.

[50] P. Frey, H. Borouchaki, and P.-L. George. Delaunay tetrahedralization using an advancing-front approach. In *5th International Meshing Roundtable*, pages 31–48, 1996.

[51] I. Fujishiro, Y. Maeda, H. Sato, and Y. Takeshima. Volumetric data exploration using interval volume. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):144–155, 1996.

[52] R. Gabdoulline and R. Wade. Analytically defined surfaces to analyze molecular interaction properties. *Journal of Molecular Graphics*, 14(6):341–353, 1996.

[53] M. Garland. Multiresolution modeling: Survey and future opportunities. In *Eurographics: State of The Art Report*. 1999.

[54] M. Garland and P. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *IEEE Visualization*, pages 263–270, 1998.

[55] A. Ghosh, C. S. Rapp, and R. A. Friesner. Generalized born model based on a surface integral formulation. *Journal of Physical Chemistry B*, 102:10983–10990, 1998.

[56] S. Gibson. Using distance maps for accurate surface representation in sampled volumes. In *IEEE symposium on Volume visualization*, pages 23–30, 1998.

[57] S. Goswami, T. K. Dey, and C. L. Bajaj. Identifying flat and tubular regions of a shape by unstable manifolds. In

Proceedings of the 11th ACM Symposium Solid and Physical Modeling, pages 27–37, 2006.

[58] J. Grant and B. Pickup. A gaussian description of molecular shape. *Journal of Physical Chemistry*, 99:3503–3510, 1995.

[59] J. A. Grant and B. T. Pickup. A gaussian description of molecular shape. *Journal of Physical Chemistry*, 99:3503–3510, 1995.

[60] M. Holst, N. Baker, and F. Wang. Adaptive multilevel finite element solution of the poisson-boltzmann equation i: Algorithms and examples. *Journal of Compututational Chemistry*, 21:1319–1342, 2000.

[61] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. In *SIGGRAPH*, pages 339–346, 2002.

[62] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. *Proceedings of SIGGRAPH*, pages 339–346, 2002.

[63] M. Karplus and J. A. McCammon. Molecular dynamics simulations of biomolecules. *Nature Structural Biology*, 9:646–652, 2002.

[64] L. Kobbelt, M. Botsch, U. Schwanecke, and H. Seidel. Feature sensitive surface extraction from volume data. In *SIGGRAPH*, pages 57–66, 2001.

[65] M. V. Kreveld, M. Overmars, O. Schwarzkopf, and M. D. Berg. *Computational Geometry:Algorithms and Applications*. Springer-Verlag Telos, 1997.

[66] B. Kuhn and P. A. Kollman. A ligand that is predicted to bind better to avidin than biotin: insights from computational fluorine scanning. *Journal of American Chemical Society*, 122:3909–3916, 2000.

[67] P. Laug and H. Borouchaki. Molecular surface modeling and meshing. *Engineering with Computers*, 18:199–210, 2002.

[68] P. Laug and H. Borouchaki. Molecular surface modeling and meshing. *Engineering with Computers*, 18:199–210, 2002.

[69] B. Lee and F. M. Richards. The interpretation of protein structure: estimation of static accessiblilty. *Journal of Molecular Biology*, 55:379–400, 1971.

[70] B. Lee and F. M. Richards. The interpretation of protein structures: estimation of static accessibility. *Journal of Molecular Biology*, 55(3):379–400, February 1971.

[71] M. Lee, M. Feig, F. Salsbury, and C. Brooks. New analytic approximation to standard molecular volume definition and its application to generalized born calculation. *Journal of Computational Chemistry*, 24:1348–1356, 2003.

[72] M. S. Lee, M. Feig, F. R. Salsbury, and C. L. Brooks. New analytic approximation to the standard molecular volume definition and its application to generalized born calculations. *Journal of Comput Chem.*, 24:1348–1356, 2003.

[73] M. S. Lee, F. R. Salsbury, and C. L. Brooks. Novel generalized born methods. *Journal of Chemical Physics*, 116:10606–10614, 2002.

[74] R. Lohner, K. Morgan, and O. Zienkiewicz. Adaptive grid refinement for compressible euler equations. *Accuracy Estimates and Adaptive Refinements in Finite Element Computations, I. Babuska et. al. eds. Wiley*, pages 281–297, 1986.

[75] R. Lohner and P. Parikh. Three dimensional grid generation by the advancing-front method. *International Journal for Numerical Methods in Fluids*, 8:1135–1149, 1988.

[76] A. Lopes and K. Brodlie. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):16–29, 2003.

[77] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*, pages 163–169, 1987.

[78] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH '87*, pages 163–169, 1987.

[79] S. Ludtke, W. Jiang, L. Peng, G. Tang, P. Baldwin, S. Fang, H. Khant, and L. Nason. Eman. Software for Single Particle Analysis and Electron Micrograph Analysis by National Center for Macromolecular Imaging, 2005.

[80] J. D. Madura, J. M. Briggs, R. C. Wade, M. E. Davis, B. A. Luty, A. Ilin, J. Antosiewicz, M. K. Gilson, B. Bagheri, L. R. Scott, and J. A. McCammon. Electrostatics and diffusion of molecules in solution: simulations with the university of houston brownian dynamics program. *Computer Physics Communications*, 91:57–95, 1995.

[81] N. L. Max. Approximating molecular surfaces by spherical harmonics. *Journal of Molecular Graphics*, 6(4):210, 1988.

[82] M. Meyer, M. Desbrun, P. Schroder, and A. Burr. Discrete differential-geometry operators for triangulated 2-manifolds. *VisMath'02, Berlin*, 2002.

[83] P. G. Mezey. *Shape in Chemistry: An Introduction to Molecular Shape and Topology*. VCH Publishers, New York, 1993.

[84] Y. Modis, S. Ogata, D. Clements, and S. C. Harrison. A ligand-binding pocket in the dengue virus envelope glycoprotein. *Proceedings of the National Academy of Sciences of the United States of America*, 100(12):6986–6991, 2003.

[85] G. Nielson. The side-vertex method for interpolation in triangles. *J. Apprrox. Theory*, 25:318–336, 1979.

[86] G. Nielson and J. Sung. Interval volume tetrahedrization. In *IEEE Visualization 97*, pages 221–228, 1997.

[87] M. Nina, D. Beglov, and B. Roux. Atomic radii for continuum electrostatics calculations based on molecular dynamics

free energy simulations. *Journal of Physical Chemistry B*, 101:5239–5248, 1997.

[88] Y. Ohtake, A. Belyaev, and I. Bogaevski. Mesh regularization and adaptive smoothing. *Computer Aided Design*, 33:789–800, 2001.

[89] S. Owen. A survey of unstructured mesh generation technology. In *7th International Meshing Roundtable*, pages 26–28, 1998.

[90] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. A PDE-based fast local level set method. *Journal of Computational Physics*, 155(2):410–438, 1999.

[91] D. Potts, G. Steidl, and A. Nieslony. Fast convolution with radial kernels at nonequispaced knots. *Numerical Mathematics*, 98(2):329–351, 2004.

[92] D. Potts, G. Steidl, and M. Tasche. *Fast Fourier transforms for nonequispaced data: A tutorial*, chapter 12, pages 247–269. Birkhäuser, 2001.

[93] F. M. Richards. Areas, volumes, packing, and protein structure. *Annu. Rev. Biophys. Bioeng.*, 6:151–176, 1977.

[94] F. M. Richards. Areas, volumes, packing, and protein structure. *Annual Review of Biophysics and Bioengineering*, 6:151–176, June 1977.

[95] D. W. Ritchie. Evaluation of protein docking predictions using hex 3.1 in capri rounds 1 and 2. *Proteins: Structure, Function, and Genetics*, 52(1):98–106, July 2003.

[96] B. Roux and T. Simonson. Implicit solvent models. *Biophysical Chemistry*, 78:1–20, 1999.

[97] M. Sanner, A. Olson, and J. Spehner. Fast and robust computation of molecular surfaces. In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 406–407. ACM Press, 1995.

[98] M. Sanner, A. Olson, and J. Spehner. Reduced surface: An efficient way to compute molecular surfaces. *Biopolymers*, 38:305–320, 1996.

[99] M. Sanner, A. Olson, and J. Spehner. Reduced surface: an efficient way to compute molecular surfaces. *Biopolymers*, 38(3):305–320, March 1996.

[100] M. F. Sanner and A. J. Olson. Real time surface reconstruction for moving molecular fragments. In *Proceedings of the Pacific Symposium on Biocomputing '97*, Maui, Hawaii, January 1997.

[101] G. Sapiro. Geometric partial differential equations and imaging analysis. *Cambridge University Press*, 2001.

[102] M. Schaefer and M. Karplus. A comprehensive analytical treatment of continuum electrostatics. *Journal of Physical Chemistry*, 100:1578–1599, 1996.

[103] R. Shekhar, E. Fayyad, R. Yagel, and J. F. Cornhill. Octree-based decimation of marching cubes surfaces. In *VIS '96: Proceedings of the 7th conference on Visualization '96*, pages 335–ff., Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.

[104] M. Shephard and M. Georges. Three-dimensional mesh generation by finite octree technique. *International Journal for Numerical Methods in Engineering*, 32:709–749, 1991.

[105] J. Shewchuk. Constrained delaunay tetrahedrizations and provably good boundary recovery. In *11th International Meshing Roundtable*, pages 193–204, 2002.

[106] K. Shimada, A. Yamada, and T. Itoh. Anisotropic triangular meshing of parametric surfaces via close packing of ellipsoidal bubbles. In *6th International Meshing Roundtable*, pages 375–390, 1997.

[107] D. Siersma. Voronoi diagrams and morse theory of the distance function. In O. E. Barndorff and E. B. V. Jensen, editors, *Geometry in Present Day Science*, pages 187–208. 1999.

[108] Y. Song, Y. Zhang, C. Bajaj, and N. Baker. Continuum diffusion reaction rate calculations of wild type and mutant mouse acetylcholinesterase: Adaptive finite element analysis. *Biophysical Journal*, 87(3):1558–1566, 2004.

[109] Y. Song, Y. Zhang, T. Shen, C. Bajaj, J. McCammon, and N. Baker. Finite element solution of the steady-state smoluchowski equation for rate constant calculations. *Biophysical Journal*, 86(4):2017–2029, 2004.

[110] J. Srinivasan, T. E. Cheatham, P. Cieplak, P. A. Kollman, and D. A. Case. Continuum solvent studies of the stability of dna, rna, and phosphoramidate-dna helices. *Journal of American Chemical Society*, 120:9401–9409, 1998.

[111] W. C. Still, A. Tempczyk, R. C. Hawley, and T. Hendrickson. Semianalytical treatment of solvation for molecular mechanics and dynamics. *Journal of American Chemical Society*, 112:6127–6129, 1990.

[112] S.-H. Teng and C. Wong. Unstructured mesh generation: theory, practice and perspectives. *International Journal of Computational Geometry Applications*, 10(3):227–266, 2000.

[113] R. Voorintholt, M. T. Kosters, G. Vegter, G. Vriend, and W. G. Hol. A very fast program for visualizing protein surfaces, channels and cavities. *Journal of Molecular Graphics*, 7(4):243–245, December 1989.

[114] R. Westermann, L. Kobbelt, and T. Ertl. Real-time exploration of regular volume data by adaptive reconstruction of iso-surfaces. *The Visual Computer*, pages 100–111, 1999.

[115] Z. Wood, D. Breen, and M. Desbrun. Semi-regular mesh extraction from volumes. In *IEEE Visualization*, pages 275–282, 2000.

[116] G. Xu. Discrete laplace-beltrami operator on sphere and optimal spherical triangulations. *Research Report, Institute of Comput. Math. and Sci./Engin. Comput., Chinese Academy of Sciences, No. ICM-04-11*, 2004.

[117] G. Xu. Discrete laplace-beltrami operators and their convergence. *Computer Aided Geometric Design*, 21:767–784, 2004.

[118] G. Xu and Q. Zhang. Construction of geometric partial differential equations in computational geometry. *Mathematica Numerica Sinica*, 28(4):337–356, 2006.

[119] D. Zhang, J. Suen, Y. Zhang, Y. Song, Z. Radic, P. Taylor, M. Holst, C. Bajaj, N. Baker, and J. McCammon. Tetrameric mouse acetylcholinesterase: Continuum diffusion rate calculations by solving the steady-state smoluchowski equation using finite element methods. *Biophysical Journal*, 88(3):1659–1665, 2004.

[120] Y. Zhang, C. Bajaj, and B.-S. Sohn. Adaptive and quality 3d meshing from imaging data. In *ACM Solid Modeling and Applications*, pages 286–291, 2003.

[121] Y. Zhang, C. Bajaj, and B.-S. Sohn. 3d finite element meshing from imaging data. *Special issue of Computer Methods in Applied Mechanics and Engineering on Unstructured Mesh Generation, in press*, 2004.

[122] Y. Zhang, G. Xu, and C. Bajaj. Quality meshing of implicit solvation models of biomolecular structures. *Computer Aided Geometric Design*, 23:510–530, 2006.

[123] Y. Zhang, G. Xu, and C. Bajaj. Quality meshing of implicit solvation models of biomolecular structures. *Computer Aided Geometric Design*, 23(6):510–530, 2006.

[124] W. Zhao, G. Xu, and C. Bajaj. Modeling with cubic A-patches. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2009.

Chapter 4

Complementary Space

Molecular surfaces are solvent contact interfaces between the strongly covalent bonded atoms of the molecule and the ionic solvent environment which is mostly water. Molecular surfaces often contain a number of pockets, holes and interconnected tunnels with many openings (mouths), aka molecular features in contact with the solvent. Several of these molecular features are biochemically significant as pockets are often active sites for ligand binding or enzymatic reactions[7], and tunnels are often solvent ion conductance zones [63]. Since pockets or holes or tunnels share similar surface feature visavis their openings (mouths), we shall sometimes refer to these molecular features collectively as generalized pockets or pockets.

The surface of a protein can be represented as a closed compact surface S in $\mathbb{R}^3$ and the closed interior V as the region bounded by S. It is important to correctly identify the main biophysical features of S in the protein surfaces, i.e. "pockets", so that they can be used for quantitatively analyzing the binding affinities of ligands and other biochemical reactions. Studying the shapes and other biochemical properties of protein pockets has many potential applications in biomedical research, for example screening potential drug molecules in computational drug design.

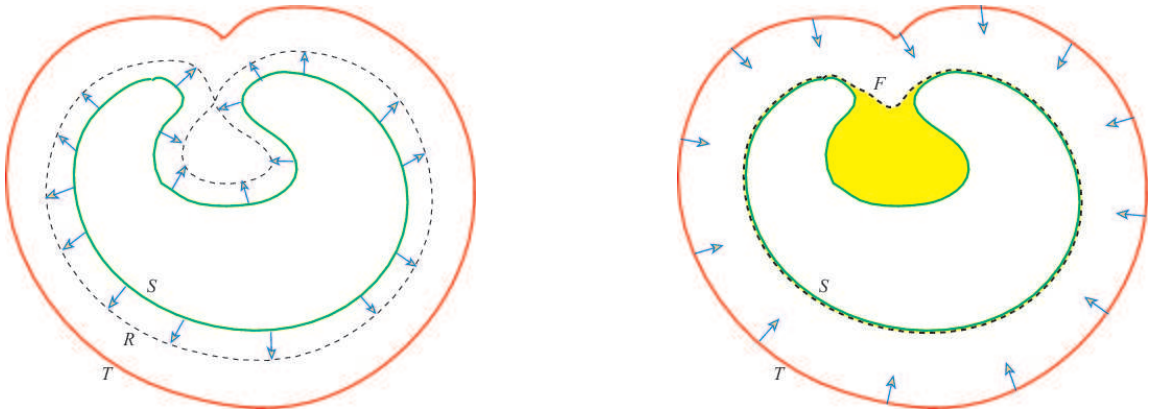
4.1 Mouths and Pockets

The surface of a protein can be represented as a closed compact surface S in $\mathbb{R}^3$ and the closed interior V as the region bounded by S. It is important to correctly identify the main biophysical features of S in the protein surfaces, i.e. "pockets", so that they can be used for quantitatively analyzing the binding affinities of ligands and other biochemical reactions. Studying the shapes and other biochemical properties of protein pockets has many potential applications in biomedical research, for example screening potential drug molecules in computational drug design.

In this subsection, we present a simple and fast geometric algorithm for extracting pockets of any closed compact smooth surface, particularly complicated molecular surfaces of proteins. This algorithm employs a two-step level-set marching method, first outward from the original protein surface S, and then backward from a topological simple enclosing shell T resulted from the first marching. The backward marching step would move the front back to the original surface S, except the "pocket" regions on S. Thus the pockets are defined as the regions outside S and not reached by the backward propagation, as illustrated in Figure 4.1.

The result of this marching algorithm is computed as a $3D$ volumetric "*pocket function*" $P(\mathbf{x})$, where $P(\mathbf{x}) > 0$ if x is inside the pocket regions on S and $P(\mathbf{x}) < 0$ if x is outside. This volumetric representation of pockets is very convenient, since it allows us to compute the pocket bounding surfaces as a level set $P(\mathbf{x}) = 0$, quantify shape attributes of the pockets , and visualize them with volume and surface visualization techniques.

This pocket extraction algorithm is a geometrical method independent of any particular model of the protein surface. It works for any representation of protein surfaces, as long as the final protein surface is described as a closed compact surface. It constructs a smooth representation of the pockets as a volumetric pocket function $P(\mathbf{x})$, from which the pocket envelops can be extracted as the level set $P(\mathbf{x}) = 0$. The extracted envelops match very well with the geometric shapes of the protein pockets,



(a) (b)

Figure 4.1: (a) Outward propagation from S to the shell T. (b) Backward propagation from T to the final front F. Pockets are extracted as the yellow shaded regions between F and S.

as demonstrated by the examples in the later subsections. Our Pocket extraction and feature analysis algorithm is also clearly generalizable to many other classes of free-form surfaces.

4.1.1 Pocket Extraction Algorithm

We present in subsection 4.1.1 our pocket extraction algorithm, which applies a two-step marching approach to compute the pocket function $P(\mathbf{x})$. subsection 4.1.1 describes our method of computing signed distance function (SDF) that is central to the pocket extraction algorithm in subsection 4.1.1.

Pocket Extraction

In Figure 4.1, we first use a simple 2D example to illustrate the basic ideas of our pocket extraction algorithm, i.e. the two-step surface marching method. Consider the closed compact surface S, the green inner curve in Figure 4.1 in 2D space. The two-step marching method consists of a outward propagation (fill) step and a backward propagation (removal) step, in order to extract pockets on S.

The first marching step starts from the original surface S and moves outwards at a constant speed to fill all pockets, voids, and depressions on the surface. As shown in Figure 4.1(a), the front propagates outward from the surface S to a final shell surface T. The propagation front would change its shape and topology during the marching. For example, the topology of the intermediate front R (dashed line) in Figure 4.1(a) is different from both S and T. Eventually the topology of the front would become the same as that of a simple sphere. The outward marching stops at the final shell surface T, which is far enough away from S with a distance t such that T has the same topology as a simple sphere and its topology would not change any more by further outward propagation. The exact value of the marching distance t from S to T is not significant in the following computations of the pocket function, as long as it is sufficiently large to ensure that T is a simple shell. For a protein, we typical choose t to be as least twice the largest dimension of the protein.

The second marching step starts from the shell T backward towards the original surface S and moves at a constant speed, in order to reveal the pockets on S. The marching distance of the second step is selected to be the same as t in the first step, so that the backward propagation front would not penetrate S and stops when it reaches S. However, notice the outward marching in the first step is irreversible and the final front of the second marching cannot extend into the depressed regions, i.e. pockets, on the surface S. Therefore in our algorithm, *pockets* are defined as the regions between the final front F of the backward marching, dashed line in Figure 4.1(b), and the original surface S. In the simple 2D example in Figure 4.1, the pocket region is illustrates as a shaded (yellow) area. This definition intuitively captures the main geometric characteristics of protein pockets.

Next we need to develop a mathematical representation of the pocket regions.

Starting from the initial surface S, the propagation front of the first outward marching step moves along its outer normal directions at a speed $v(\mathbf{x})$. The marching front $R(t)$ at time t can be determined according to the level set method [56], i.e. $R(t)$ is the zero level set of a function $\phi(\mathbf{x},t)$ satisfying the evolution equation:

$$\phi_t + v|\nabla\phi| = 0$$

with initial condition $\phi(\mathbf{x},t=0) = d(\mathbf{x})$, where $d(\mathbf{x})$ is the signed distance function (SDF) from S, defined as

$$|d(\mathbf{x})| = \min_{\mathbf{y}\in S} ||\mathbf{x}-\mathbf{y}||. \tag{1.1}$$

The SDF $d(\mathbf{x})$ is positive/negative when $\mathbf{x}$ is outside/inside the surface S, and the marching front $R(t)$ is the level set $\phi(\mathbf{x},t) = 0$. If the speed $v = 1$ is constant, as we would assume in our two-step marching algorithm, the marching front $R(t)$ at time(distance) t is simply the level set

$$d(\mathbf{x}) = t. \tag{1.2}$$

We will discuss an efficient algorithm to compute the signed distance functions (SDF) of a closed compact surface in subsection 4.1.1, based on the fast distance transform. Applying the SDF algorithm, we present here the main algorithm of computing a volumetric *pocket function* $P(\mathbf{x})$ to represent the pockets on the protein surface S as follows:

1. *Compute the SDF for S:* Compute the signed distance function $d_S(\mathbf{x})$ from the original surface S, where $d_S(\mathbf{x}) > 0$ if $\mathbf{x}$ is outside S and $d_S(\mathbf{x}) < 0$ if $\mathbf{x}$ is inside.

2. *Extract the Shell Surface T:* Extract the shell surface T as the level set $d_S(x) = t$, where the distance $t > 0$ is large enough so that T has the same topology as a simple sphere. As mentioned earlier, the exact value of t is not significant in the algorithm.

3. *Compute the SDF for T:* Compute the signed distance function $d_T(x)$ from the surface T, where the sign of $d_T(x)$ is inverted, i.e. $d_T(x) > 0$ if x is inside T and $d_T(x) < 0$ if x is outside T.

4. *Construct the Pocket Function P(x):* The volumetric pocket function $P(\mathbf{x})$ is constructed as:

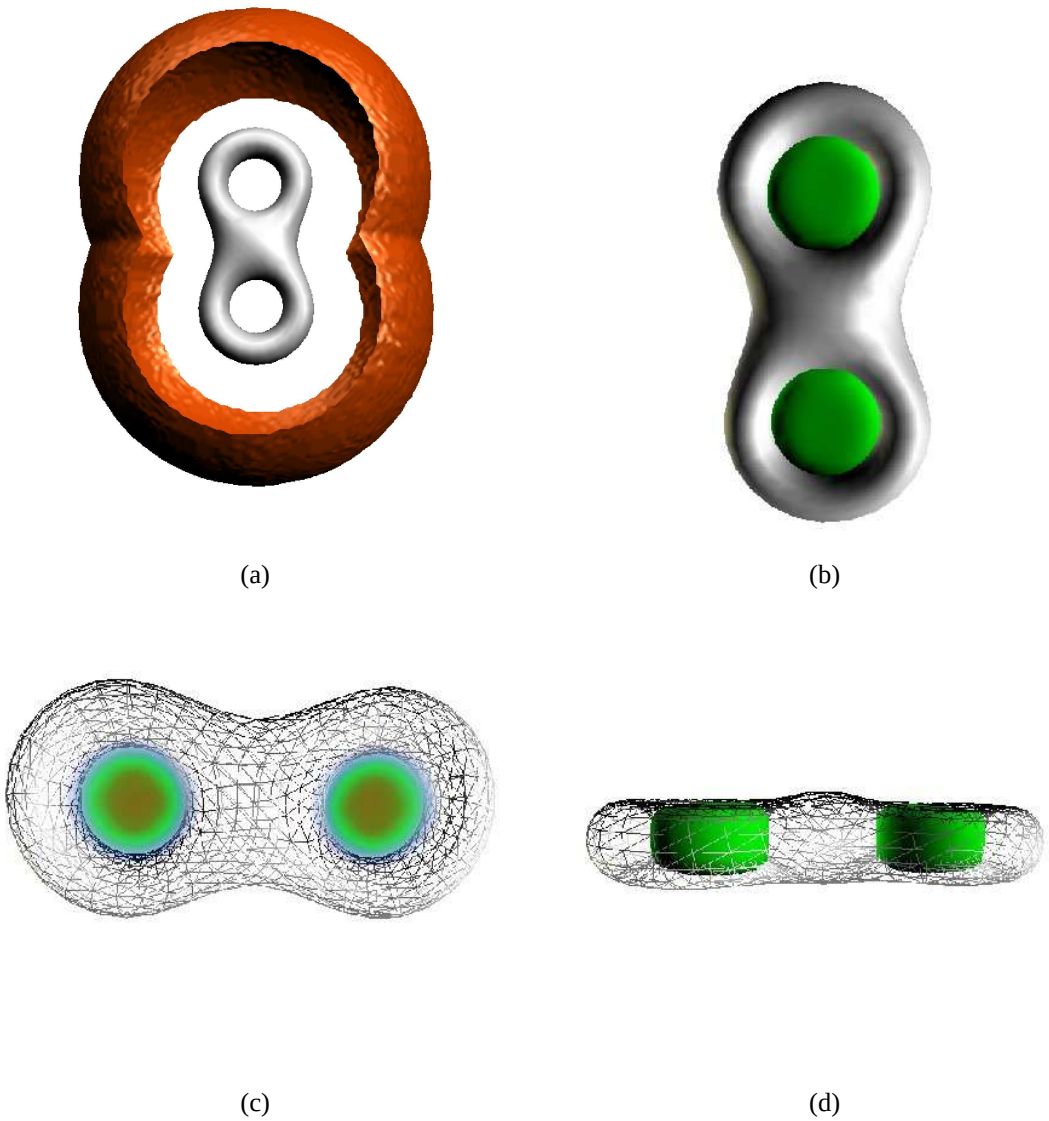
$$P(\mathbf{x}) = \min(d_S(x), d_T(x) - t) - \varepsilon, \tag{1.3}$$

   where $d_S(x)$ and $d_T(x)$ are the SDFs computed in step 1 and 3. Notice that $\min(d_S(x), d_T(x) - t) > 0$ only for points outside the surface S and not reached by backward marching from the shell T, i.e. points in the pockets. A small positive value ε is introduced to account for the size of solvent atoms, which is typically set to a value between 1 and 1.5 Å. The bounding surfaces of pockets (pocket envelops) then can be extracted as the level set $P(\mathbf{x}) = 0$.

5. *Smooth the Pocket Function P(x):* Although $P(\mathbf{x})$ constructed in the previous step matches well with geometric features of the pockets on the surface S, it can be noisy because of the small bumpiness on S. In order to remove small noises and focus on the main features of the pockets, we apply a smoothing step to the pocket function $P(\mathbf{x})$. In our implementation, one step of bilateral filtering [62] is applied to smooth $P(\mathbf{x})$.

This pocket extraction algorithm is simple, flexible, and robust. Figure 4.2 shows a successful extraction of two tunnels in an "8" shape. It works for any closed compact surfaces in $\mathbb{R}^3$ space. Particularly it can be used for any molecular surface models: union of balls, solvent accessible surface, or contours of electron density functions.

In order to show the effectiveness of our algorithm, we randomly select the "Bacteriochlorophyll Containing Protein" (PDB ID: 4BCL) from the protein data bank (PDB) [6]. This protein has a very complex molecular surface and contains one large binding site in the middle and small dents on its surface, as shown in Figure 4.3(a). The pocket function $P(\mathbf{x})$ of the protein (4BCL) is computed using the algorithm described in this subsection. To better reveal the geometric relations between the extracted pockets and the protein surface, we look at a 2D slice of the data. Figure 4.3(b) shows a slice of the protein 4BCL, as a color-map of its SDF $d_S(\mathbf{x})$. The cross-subsection of the protein surface is displayed in Figure 4.3(b) as white curves, on which the SDF $d_S(\mathbf{x}) = 0$. The large tunnel in the middle of the protein is clearly visible, with several additional small surface dents and internal voids.

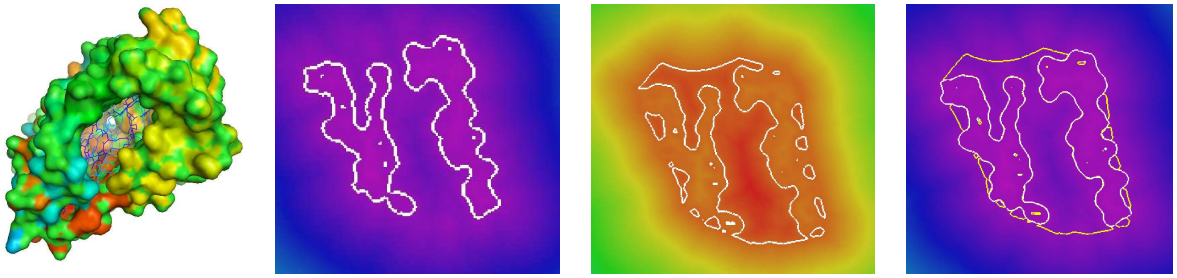


(a) (b)

(c) (d)

Figure 4.2: (a) The original gray surface of the "8" shape and the final shell surface (clipped) in dark red. (b) The "8" shape and its tunnel envelops shown in green. (c) Volume rendering of the pocket function $P(\mathbf{x})$ of the "8" shape. (d) Another view of the tunnel envelops extracted as a level set of $P(\mathbf{x})$.

Figure 4.3(c) shows the color-mapped slice of the pocket function $P(\mathbf{x})$ and the cross-subsection of the pocket envelop as white curves. The cross-subsections of the protein surface and the pocket envelop are finally superimposed in Figure 4.3(d). We can see that our pocket extraction algorithm has almost perfectly located all pockets and holes in the molecular surface and how well they match with the geometric features on the protein surface. The pocket openings are extracted and displayed as yellow line segments, where the two separate openings of middle tunnel are clearly visible.



(a) (b) (c) (d)

Figure 4.3: Pocket extraction example for the "Bacteriochlorophyll Containing Protein" (PDB ID: 4BCL). (a) The protein surface and the big tunnel in the middle. (b) A cross-subsection of the protein surface shown as white curves on a slice of the color-mapped protein SDF. (c) A cross-subsection of the pocket envelop is shown as curves on the slice of the color-mapped pocket function. (d) The pocket envelop in (c) is superimposed onto the protein surface in (b), and they match perfectly.

Signed Distance Functions

Efficient and stable computation of signed distance functions (SDF) $d(\mathbf{x})$ from a surface plays a central role in the pocket extraction algorithm described in subsection 4.1.1. A number of SDF algorithms were developed in recent years. In this subsection, we present a method of computing the SDF $d(\mathbf{x})$ based on fast distance transforms [30]. Other stable SDF algorithms may also be applied, like SDF algorithms running on GPU [59] for better performance.

Given a 2D/3D binary image as input, the distance transform calculates the shortest distance from each pixel (voxel) to the nearest non-zero pixels (voxels). The distance transform computation is very efficient and can be done in time linear to the number of pixels (voxels). We extend the distance transforms to compute SDF for any closed compact surface.

Considering a closed compact surface S embedded in a regular grid, we define a grid point p as a *near point*, highlighted in Figure 4.4(a) for a 2D example, if at least a cell containing p intersects S. Otherwise p is defined as a *far point*. The signed distance function $d_S(x)$ to the surface S is computed as follows:

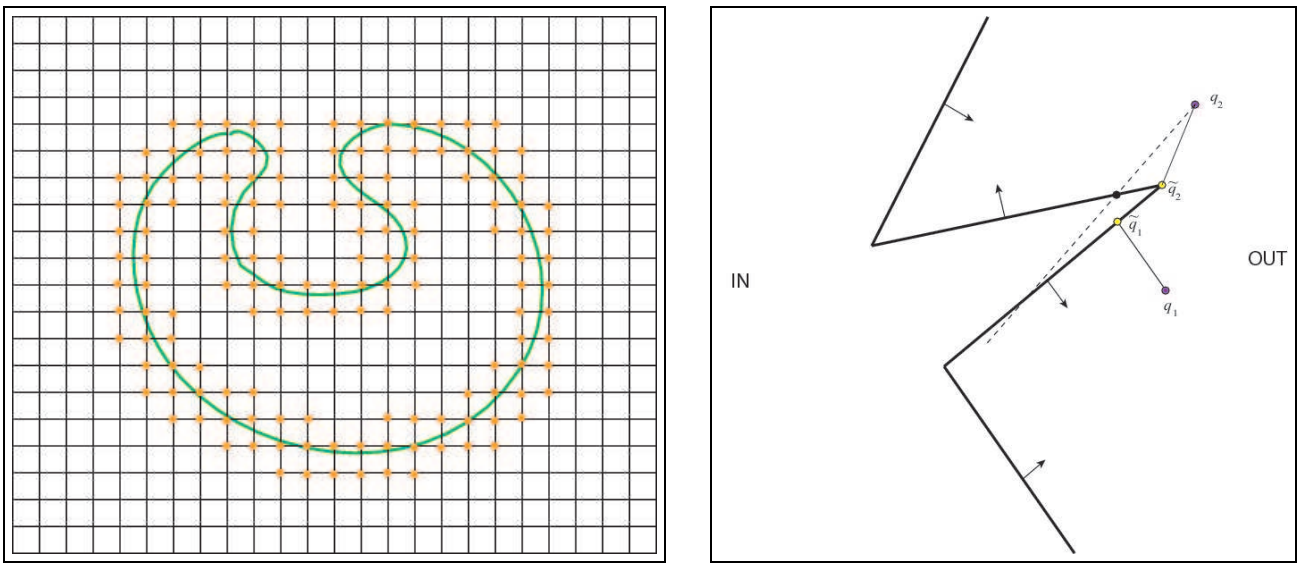
1. Construct a binary image I_0 by setting the values of near points to 1 and far points to 0.

2. Compute the distance transform for the binary image I_0. As the result of the distance transform, the value of a near point is now 0 and the value of a far point is the distance to the closest near point. Furthermore, the closest near point c_p to each far point p is recorded. c_p is called the *near cousin* of p. The time for this step is linear in the number of grid points.

3. For each near point q, compute the shortest distance $d_S(q)$ from q to the surface S and set the sign of $d_S(q)$ positive/negative if q is outside/inside S, and record the point $\tilde{q}$ on S that is nearest to q. So now we have the SDF for the near points, and will compute the SDF for the far points in the next step.

4. The SDF $d_S(p)$ of a far point p has the same sign as that of its near cousin c_p. The magnitude of $d_S(p)$ is approximated as $|p - \tilde{c_p}|$, where $\tilde{c_p} \in S$ is the point on S nearest to c_p recorded in step 3.

We now explain more details of the step 3 in the above SDF algorithm. Without loss of generality, we assume the surface S is decomposed into simplices, e.g. triangles in 3D, and the normal vectors of all its vertices always point towards the outside of S. We need to compute the shortest distance from a near point q to S, and determine whether a near point q is outside or inside



(a) (b)

Figure 4.4: (a) A 2D example of SDF computation, all near points in the grid are highlighted in orange. (b) A 2D example of determining whether near points are inside/outside the surface S. For the point q_1, its nearest point $\tilde{q}_1 \in S$ is within a single simplex and it is simply determined by the dot-product rule. But the nearest $\tilde{q}_2 \in S$ of q_2 is on the corner shared by two simplices. A ray (the dashed line) is cast from q_2 to find the nearest intersecting simplex.

S. First, we calculate the point $\tilde{q} \in S$ nearest to q, which is done by examining the triangles close to the point q. Next we need to decide whether q is outside/inside S. In $\mathbb{R}^3$, the nearest point $\tilde{q} \in S$ of q may be inside a triangle, on a triangle edge, or on a triangle vertex.

- If $\tilde{q}$ belongs to only a triangle $t \in S$, i.e. $\tilde{q}$ is within the interior of t, then q is outside S if $(q-\tilde{q}) \cdot \mathbf{n}_t > 0$, where $\mathbf{n}_t$ is the normal vector of the triangle t. But this simple dot-product rule fails if $\tilde{q}$ is a shared point of two or more simplices, i.e. $\tilde{q}$ is on a corner or edge of S.
- If $\tilde{q}$ is on a edge of corner of S, we use a ray-shooting method to find the closest simplex to q. We cast a ray R_q from q through an interior point of a simplex containing $\tilde{q}$ and compute the intersubsection points between R_q and all other simplices sharing the same $\tilde{q}$. A 2D example is illustrated in Figure 4.4(b). The first simplex t_0 intersected by R_q is chosen and the sign of $d_S(q)$ is set as the sign of $(q-\tilde{q}) \cdot \mathbf{n}_{t_0}$, where $\mathbf{n}_{t_0}$ is the normal of the simplex t_0.

We prove two propositions about the signed distance functions $d_S(p)$ computed by the above algorithm.

Proposition 4.1.1. *The sign of SDF $d_S(p)$ is correctly set for every far point p .*

Proof: We prove this by contradiction. The sign of $d_S(p)$ of the far point p is the same as that of its closest near point c_p. If p is outside S, then its near cousin c_p is inside S. Let us follow the path from p to c_p that consists of three segments along the x, y, and z axes. The last outside point on the path must be a near point and is closer to p than c_p. This contradicts the definition that c_p is the closest near point to p. The same arguments hold if p is inside S.

Proposition 4.1.2. *The error of $d_S(p)$ is not accumulative and is bounded by the same order as the grid cell side δ.*

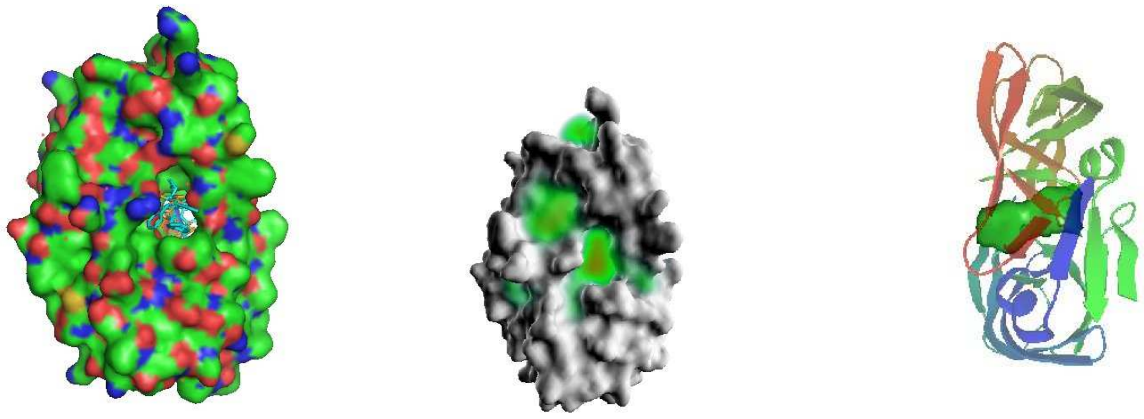
Proof: Clearly the magnitude of $d_S(p)$ is larger than the distance $|d(p,c_p)|$ from p to its near cousin c_p and less than $|d(p,c_p)|+|d(c_p,\tilde{c_p})|$. The distance $|d(c_p,\tilde{c_p})|$ from the near point c_p to the closest point $\tilde{c_p}$ on S is in the order of $O(\delta)$. Thus we have the following inequality,

$$\begin{aligned} |d(p,c_p)| &\leq |d_S(p)| \leq |d(p,c_p)|+|d(c_p,\tilde{c_p})| \\ &= |d(p,c_p)|+O(\delta). \end{aligned}$$

Therefore error between $d_S(p)$ and its approximate $d(p,c_p)+d(c_p,\tilde{c_p})$ is bounded by $O(\delta)$.

Since the computed SDF always achieves the correct signs and has bounded errors for the SDF, the above algorithm is very robust. It was also shown to be very efficient and works even for highly complicated protein surfaces. The running time of each step of algorithm is $O(N)$, linear to the number of grid points N, except for step (3). In the worst case, step (3) has computational complexity $O(s \cdot N_n)$, where s is the number of simplices in the surface S and N_n is the number of near points. However, the regular grid provides a natural decomposition of the space, and typically only a small subset of simplices need to be examined to compute the SDF of a near point q in step 3. On the average case, the complexity of step (3) is $O(N_n)$, proportional to the number of simplices in S, which makes the computation of the SDF efficient even for highly complicated protein surfaces in our tests.

4.1.2 Visualization and Quantitative Analysis



(a) (b) (c)

Figure 4.5: The molecular surface and pockets of HIV-I protease (1HOS): (a) Protein surface with the ligand in the middle tunnel; (b) Pocket function visualized using volume rendering on the top of the protein surface; (c) The largest pocket (tunnel) shown as a green surface in the cartoon of the protein structure.

Representing the protein pockets implicitly as a volumetric pocket function $P(\mathbf{x})$ allows for various ways to visualize and analyze the pocket structures quantitatively. subsection 4.1.2 describes the visualizations of the protein pocket functions and pocket envelops. subsection 4.1.2 discusses some analyses that can be performed on the protein pocket function.

Visualization

Because the pocket function $P(\mathbf{x})$ is a 3D volumetric scalar function, we can visualize it using various volume visualization techniques, e.g. ray-cast or texture based volume rendering and isosurface rendering. As an example of visualization, Figure 4.5 displays the HIV-I protease (PDB ID: 1HOS), an important protein for the maturation of HIV-I virus. An ligand can bind in the tunnel of the HIV-I protease, which also has a few more large pockets on its surface. Figure 4.5(a) shows the protein surface colored by the residues and the ligand bound in its middle tunnel. The pocket function $P(\mathbf{x})$ of the HIV-1 protease is computed by using algorithm in subsection 4.1.1. Figure 4.5(b) renders the pocket function using 3D volume rendering together with the protein surface to illustrate the overall distribution of the pocket regions. The pocket envelops are extract as the level set $P(\mathbf{x}) = 0$, of which each connected component is the bounding surface of a pocket. We are often interested in the largest pockets, which may serve as active binding sites of ligands. as the large pocket region of the function. The pocket (tunnel) with the largest volume is extracted from the HIV-1 protease pocket function, which is actually the middle tunnel that binds the ligand. Figure 4.5(c) displays the bounding surface of the largest pocket of the HIV-1 protease, i.e. the ligand-binding tunnel, together with the cartoon of the protein structure. The visualization proves that our pocket extraction algorithm correctly

captured the main pocket of the protein.

Quantitative Analysis

In this subsection we discuss some analyses based on the pocket function $P(\mathbf{x})$. They demonstrate the benefits of using the pocket function to represent the protein pockets. **Signatures of Protein Pockets** First we look at calculating quantitative measures such as volumes and surfaces areas, i.e. signatures, of the pockets of a protein. They can be easily derived from the pocket function $P(\mathbf{x})$. As mentioned earlier, the pocket envelops are extract as the level set $P(\mathbf{x}) = 0$. Each connect contour of the level set is the bounding surface of a pocket. This becomes just a problem of computing signatures of contours of a scalar function, which were described as "contour spectrum" in [3]. Basically, quantitative measures like the volume and surface area of each pocket can be computed by summing up the contributions from individual cells that belong completely or partially to the pocket. If the 3D domain is decomposed into simplices, the contribution from each simplex to the volume or surface area of the level set $P(\mathbf{x}) = 0$ can be quickly evaluated with a B-spline function[3].

Additional geometric and shape properties can also be computed for protein pockets based on the pocket function $P(\mathbf{x})$, for example curvatures distributions [17, 60], shape histograms [53, 40], coefficients of volumetric function expansions [39], and shape context [4]. Those shape properties of protein pockets may be used for building a database of the proteins pocket structures, and applied to the problem of ligand binding prediction [44].

Pocket Mouth In some applications we wish to find a pocket's openings (mouths), the interface connecting the pocket to the space outside the protein. The number of mouths m of a pocket (or tunnel) classifies the type of the pocket (or tunnel):

- *void* if $m = 0$
- *normal pocket* if $m = 1$
- *hole or simple tunnel (simple conector)* if $m = 2$
- *arbitrary tunnel (multiple connector)* if $m >= 3$

The pocket function $P(\mathbf{x})$ can be used to help obtaining any pocket's mouths. The bounding surfaces of a pocket, extracted as a connect contour of the level set $P(\mathbf{x}) = 0$, can be divided into the mouth patches and the patches that interface to the protein surface S. According to our pocket extraction algorithm described in subsection 4.1.1 and illustrated in Figure 4.1(b), pockets mouths are the patches on the backward marching front F, which do not reach the original surface S. Based on equation 1.3, we have $d_S(\mathbf{x}) > P(x) + \varepsilon$ on the pocket mouths, where $d_S(\mathbf{x})$ is the SDF from the surface S. Therefore, in our algorithm the pocket mouths are defined as the points satisfying the conditions:

$$P(\mathbf{x}) = 0 \quad \text{and} \quad d_S(\mathbf{x}) > \varepsilon.$$

Residues Near a Pocket Given a protein pocket, we often need to know the set of amino acid residues surrounding it. The information would help to identify the protein sequence motifs that are involved in protein binding and related to certain protein functions. This can be achieved by applying the SDF algorithm mentioned above to the envelop of the protein pocket:

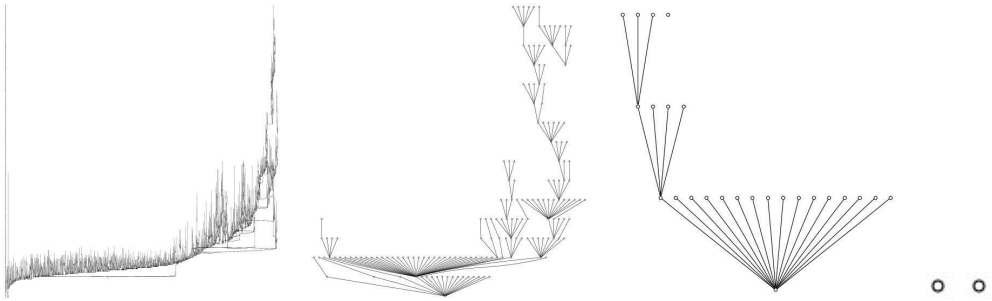
1. Extract the envelop E of the protein pocket being studied. Notice the level set $P(\mathbf{x}) = 0$ may contain multiple connected contours, which are the envelops of separate pockets on the protein. Here we only extract the envelop surrounding the pocket under consideration, as discussed in subsection 4.1.2.
2. Compute the SDF $d_E(\mathbf{x})$ for the envelop E.
3. For each residue R, evaluate $d_E(\mathbf{x}_R)$ at its coordinates $\mathbf{x}_R$. If $d_E(\mathbf{x}_R)$ is less than some given threshold, then the residue R is considered to be near the pocket.

Data Structures for the Pocket Function

Another advantage of representing protein pockets as a volumetric function is to allow to construct data structures, such as Contour Trees (CT) and Multi-Resolution Attributed Dual Contour Trees (MACT), for the pocket function $P(\mathbf{x})$. These data structures have a number of applications such as pocket selection and protein structure matching.

Contour Tree and Pocket Selection Contour tree (CT) is an affine-invariant data structure that captures the topological structures of the level sets of a volumetric function $F(x)$ [11]. Each node of the CT corresponds to a critical point of the function and each arc corresponds to a contour class connecting two critical points. A contour class is a maximal set of continuous contours which do not contain any critical points. If the CT is cut at the isovalue w, the number of connected contours of the level set $F(x) = w$ is equal to the number of intersubsections (cuts) to CT arcs. In the case of pocket function $P(\mathbf{x})$, the number of cuts to its CT at $P(\mathbf{x}) = 0$ gives the number of connected pocket envelops, i.e. the number of separate pockets.

The pocket extraction algorithm described in this paper works for general 3D surface models, e.g. the tunnels in the "8" shape shown in Figure 4.2, and complex protein surfaces. Because protein surfaces are highly complicated, they usually contain many small pockets and voids in addition to the major binding pockets. Our algorithm is very sensitive. For a complicated surface like the molecular surfaces computed from electron density functions, the level set of the pocket function $P(\mathbf{x}) = 0$ will capture the large binding sites as well as small depressions and voids on the surface. While it is good to have the capability of obtaining the details of the protein shape, the large number of small pockets may make the pocket function $P(\mathbf{x})$ and its corresponding CT quite complex. For example, the CT of the pocket function $P(\mathbf{x})$ for the "Bacteriochlorophyll A Protein (4BCL)" is shown in Figure 4.6 (a). It is quite complex and contains 2,063 nodes (critical points). The level set at $P(\mathbf{x}) = 0$ would contain a large number of individual contours, many of which are very small and of little importance. In real applications, we often want to focus on one or a few biologically important active/binding pockets which have enough size to hold the solvated ligand.



(a) (b) (c) (d)

Figure 4.6: (a) The contour tree of the pocket function for the "Bacteriochlorophyll A Protein" (4BCL) (b), (c), and (d) are the DCTs of the pocket function at three different resolutions of 16, 4, and 1 intervals.

The CT of the pocket function $P(\mathbf{x})$ can assist the selection of the most significant pockets from the level set $P(\mathbf{x}) = 0$. During the construction of the CT, each CT arc can be tagged with additional geometric attributes such as the volume and surface area of the contour and a seed cell, from which the entire connect contour can be constructed by cell propagation [12]. Therefore the CT can be simplified by suppressing contours of small geometrical measures. Furthermore, it can be applied to pick the most significant pockets. For instance, if we are only interested in the top three largest pockets, from the cut CT arcs we select the three ones tagged with largest volumes and propagate from their seed cells to get the envelops of the three largest pockets. Figure 4.5(c) shows the largest pocket of HIV-I protease, constructed by using the pocket function and its CT.

Protein Structure Matching Protein structure matching is important for classifying proteins into different families and predicting the functions of new proteins. The volumetric pocket function $P(\mathbf{x})$ can be readily used to compare protein structures.

An efficient method of comparing protein structures is described in [69] by using the affine-invariant multi-resolution attributed dual contour trees (MACT) of some volumetric shape functions, combined with additional geometric, topological, and electrostatic potential properties. The shape function used in [69] was solvent accessibility of the protein. As we know, proteins perform their functions through docking/undocking with other proteins and binding/unbinding of ligands in their active

pockets, satisfying the complementary shape and electrical properties. It might be more accurate to compare the protein pocket structures than the overall protein shapes. This is especially useful for proteins whose their overall shapes are very similar but functionally important active binding sites are different.

The MACT method can be applied to compare the pocket functions of different proteins. We summarize here the steps of matching protein structures using the volumetric pocket functions [67]:

1. Compute the volumetric pocket function $P(\mathbf{x})$ for the protein.
2. Compute the contour tree (CT)[11] for the pocket function $P(\mathbf{x})$.
3. Construct the finest-level dual contour tree (DCT) [68] from the CT in the previous step.
4. Build the multi-resolution attributed dual contour tree (MACT) from the DCT by merging adjacent functional intervals.
5. Compare two protein structures by computing the similarity score of their MACTs.

Step 1 and 2 have been discussed in the upper subsections. Next we briefly describe the data structures and algorithms involved in step 3 to 5.

Dual Contour Tree Although the contour tree (CT) captures the topological structures of the level sets of a function, it is not practical to compare CT's of the pocket functions because of their complexities, as shown in Figure 4.6(a). The dual contour tree (DCT) is a simplified data structure constructed from the CT, which can then be compared to determine the similarities among the protein structures [68, 69].

To construct the DCT, we partition a functional range $[f_1, f_2]$ ($f_1 < f_2$) of the volumetric function into a number of intervals, and cut the corresponding CT arcs into disjoint sets of connected segments [69]. Each disjoint set of connect segments has functional values within an interval and represents a connected region in the 3D volume, which is called an *interval volume*. Each node of the DCT represents such an interval volume. The DCT is considered to be "dual" to the CT because a DCT node corresponds to a disjoint set of CT arc segments. The complexity of the DCT can be controlled by choosing the functional range $[f_1, f_2]$ and the number of intervals partitioned from it. Figure 4.6(b) shows the DCT of the pocket function $P(\mathbf{x})$ of the protein 4BCL, constructed from the CT in Figure 4.6(a) by dividing the functional range $[0, \max(P(\mathbf{x}))]$ into 16 intervals. For each node of the DCT, geometric, topological and functional properties of the corresponding interval volume can be computed for the purpose of matching. We refer to [68, 69] for details of constructing DCT and computing the volume and other attributes of the DCT nodes.

Multi-resolution Attributed Dual Contour Tree (MACT) As mentioned above, the complexities of DCT's are controlled by the number of partitioned intervals, which is here referred as the resolution of the DCT. It is helpful for the matching purpose to keep a set of DCT's for a volumetric functions at different resolutions and use the match of low-resolution DCT's to guide the match of high-resolution DCT's.

This hierarchy of multi-resolution DCT's for a given volumetric function is called a Multi-resolution Attributed Dual Contour Tree (MACT) [69]. The MACT can be constructed from the highest-resolution DCT by merging its adjacent functional intervals recursively. Without loss of generality, we assume the highest-resolution DCT $\mathscr{D}_0$ has $N = 2^k$ functional intervals. Every two adjacent intervals are merged into one interval for the next lower-resolution DCT $\mathscr{D}_1$, which would then have $N/2$ intervals. A node n in $\mathscr{D}_1$ is merged from a set of nodes (S) in $\mathscr{D}_0$. For each node $m \in S \subset \mathscr{D}_0$, the node n is called the parent of m and m is a child of n. The merging process can be recursively applied to the lower-resolution DCTs until there is only a single interval spanning the entire functional range $[f_1, f_2]$. Figure 4.6 (c) and (d) show two levels of the MACT with four and one intervals respectively, which are merged from the intervals of the DCT in Figure 4.6 (b). The complexity of the DCTs at lower-resolutions are significantly reduced and the hierarchy makes it much easier to find the match between higher-resolution DCT's for the pocket functions.

Because protein surfaces are highly complicated, the compute pocket functions usually contain many small pockets and voids, which reflect as small subtrees in DCT's. On the other hand, biologically important active/binding pockets must have enough size to hold the solvated ligands. Therefore the DCT's are further simplified by pruning the nodes corresponding to

the very small pockets whose volumes fall under a given threshold. The pruning process is started from the lowest-resolution DCT in the MACT hierarchy. If a lower-resolution DCT node is pruned, all its child nodes should be removed as well in the next higher-resolution DCT. The DCT's in Figure 4.6 have been simplified by pruning. In the lowest-resolution DCT in Figure 4.6(d), only two nodes are left after pruning, which means only two pockets are considered significant. One of them contains more than 94% of total pocket volume and actually represents the large binding site in the middle of the "Bacteriochlorophyll A Protein (4BCL)" (Figure 4.3).

Computing Similarity Scores of Protein Pockets Functions We now compare protein structures by matching the MACT's of their pocket functions. The matching process is performed in the same way as that in [69]. It starts from the lowest-resolution to the highest-resolution of the MACT hierarchies. The geometrical, topological and functional attributes of DCT nodes are used to compute the similarity between the matched nodes. The similarity score between two DCTs of the same resolution is computed as the average of the scores of their matched node pairs, weighted by the volume of the nodes. Finally the similarity score between the volumetric pocket functions is evaluated as the average of the similarity scores of DCTs from all level of the MACT hierarchies.

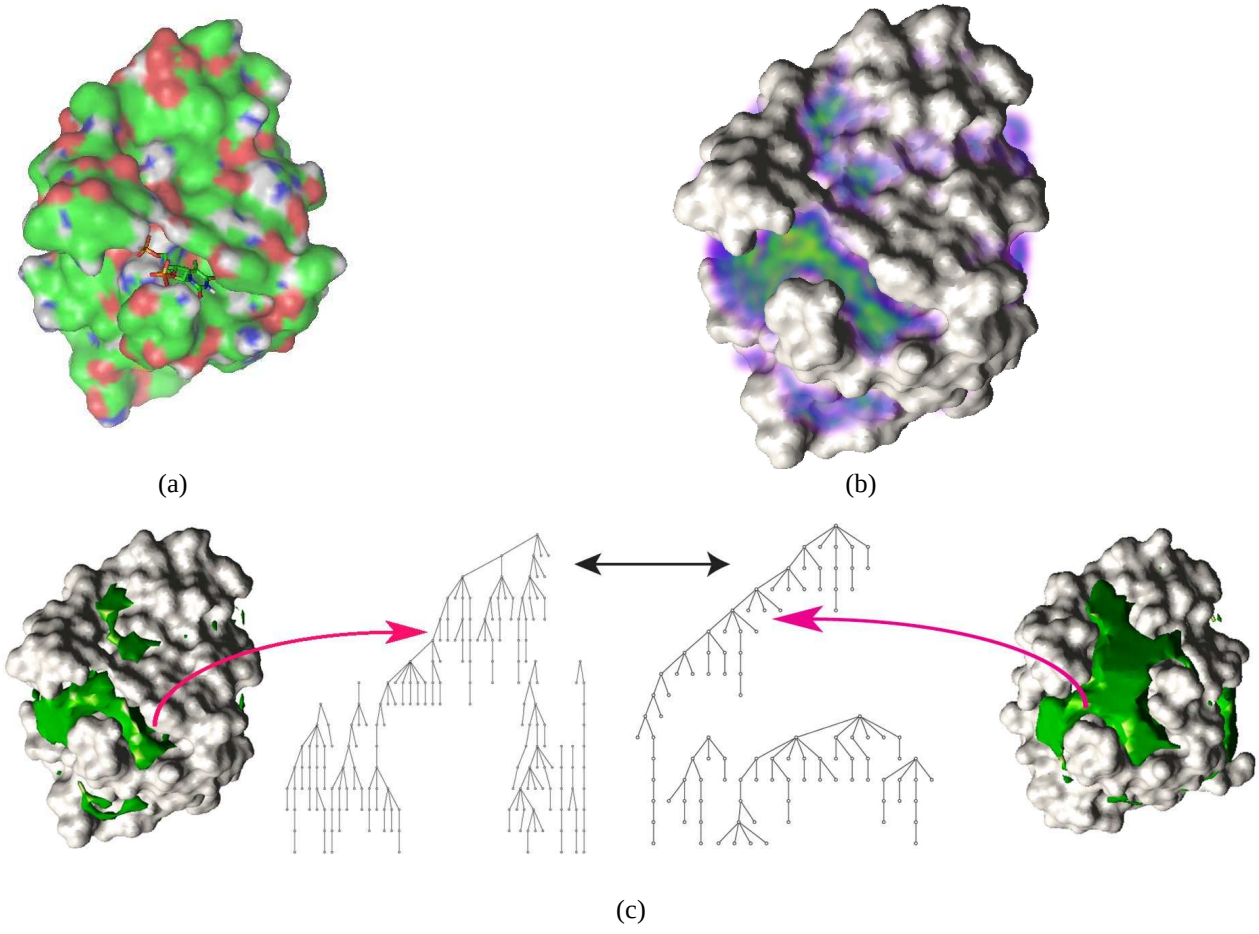


Figure 4.7: Comparison of pocket functions for the protein "Staphylococcal Nuclease" in the binding (PDB code: 1ATT, (c) left) and unbinding (PDB code: 1KAA, (c) right) states.

The most significant difference here is that pocket functions are used instead of overall protein functions such as the solvent accessibility used in [69]. This is based on the assumption that pockets are the most important features of protein structures, and is particularly useful for studying the mutants of the same protein where their overall shapes are similar but have different

pocket structures and bound ligands.

Figure 4.7 shows an example of of comparing the structures of "Staphylococcal Nuclease" in bound state (PDB code 1A2T) and its unbound state (PDB code 1KAA). "Staphylococcal Nuclease" is a protein for nucleic acid binding and binds two ligands "S-(Thioethylhydroxy)Cystine"(CME) and "Thymidine-3',5'-Diphosphate"(THP). The protein 1A2T and one of its bound ligand (THP) are shown in Figure 4.7 (a) while the other ligand is on the back side. Figure 4.7 (b) displays the pocket function of 1ATT over its molecular surface with volume rendering. Due to the bound ligands, the pocket shapes of 1A2T would be quite different from those of its unbound sibling (1KAA), although the overall their shapes remain very similar.

We computed the pocket functions and used the MACT algorithm described above to compare the structures of 1A2T and 1KAA. The correspondences between the pockets of 1ATT and 1KAA are easily established by using the MACT hierarchy, as illustrated in Figure 4.7 (c). It clearly shows the structural differences between pockets of the two proteins, which are not obvious by looking at the molecular surfaces directly. The pocket functions are very useful for studying the family of protein mutants, for example the family of HIV protease and inhibitors bound to them.

4.1.3 Implementation and Examples

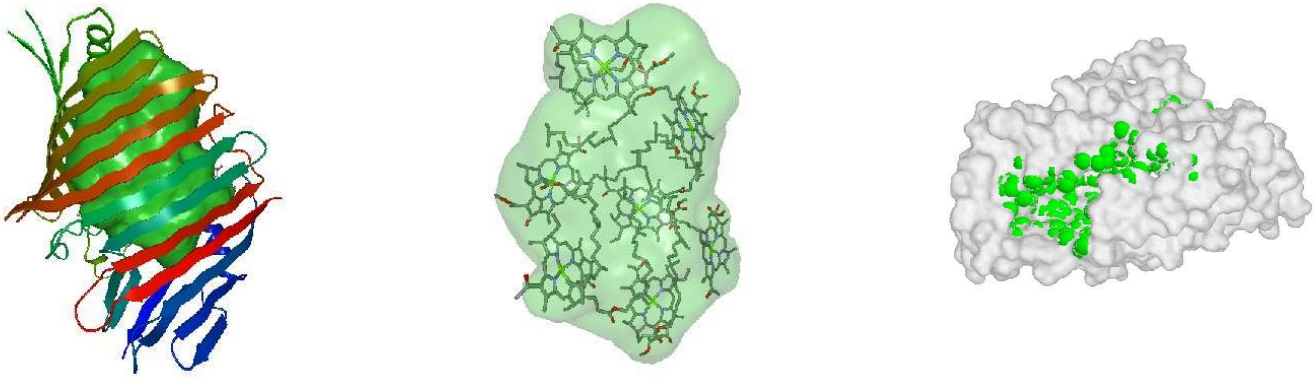
The pocket extraction algorithm and MACT protein matching algorithm have been implemented in C++ and made publicly available. The pocket functions $P(\mathbf{x})$ are computed as described in subsection 4.1.1. The pocket algorithm described in this paper is applicable to any model of protein surfaces, for example the simple union-of-ball model or the more sophisticated solvent accessible surfaces. It may actually provide a way of comparing different models of protein surfaces. For the results discussed below, we used the model of protein surfaces as the smooth level sets of the electron density functions $E(\mathbf{x})$), which are computed as the summation of the Gaussian density kernels for all atoms contained in the corresponding proteins. As mentioned earlier, the protein surface is extracted as the level set $E(\mathbf{x}) = 1$.

Table 4.1: Computational time for some pocket extraction examples on a low-end laptop. T1, T2, and T3 are the time for computing $d_S(x)$, $d_T(x)$, and pocket functions respectively.

data	# of triangles	T_1(s)	T_2(s)	T_3(s)	total (s)
"8" shape	1,536	2.1	5.45	0.33	7.88
4BCL	275,456	10.25	6.38	0.33	16.96
1C2B	268,876	9.92	5.63	0.45	16

The implementation is portable across multiple compute platforms and tested to be robust for various protein structures. It is also efficient, being able to compute pocket functions for complicated proteins with thousands of atoms in seconds. Table 4.1 shows the computation time without optimization on a DELL Laptop with 1.6 GHz processor and 1GB memory for three examples: the "8" shape, "Bacteriochlorophyll A Protein" (PDB ID: 4BCL) and "Hydrolase" (PDB ID: 1C2B). The "Hydrolase" (1C2B) is a protein complex containing four similar subunits. The dimensions of the volumetric functions in the computation were set to $128 \times 128 \times 128$, which are more than sufficient for extracting pocket functions and subsequent visualizations and quantitative analyses. The higher the dimensions, the smaller are the grid sizes and the more accurate are the SDF computations. However higher dimensions also require more memory and longer computation time since the time of distance transform is proportional to the number of grid points. Our experiments showed that increasing the dimensions would not change the number of pockets or significantly alter their quantitative measurements. The dimensions of $128 \times 128 \times 128$ provide good balance between accuracy and memory requirements for commodity PC's. If the speed is very important, lower dimensions such as 96^3 may also be used to achieve results sufficient for all applications discussed above.

The last column in Table 4.1 shows the total time of pocket extraction for the three examples : "8" shape is the simple surface shown in Figure 4.2; 4BCL and 1C2B are two example protein surfaces. The second column shows the number of triangles in the original surfaces. T_1 is the time for computing the SDF $d_S(x)$ for the original surface S, T_2 is the time for extracting the shell surface T and computing the SDF $d_T(x)$, and T_3 is the time for constructing the final pocket function $P(\mathbf{x})$ and extracting the pocket envelops. T_1 is longer for the 4BCL and 1C2B protein surfaces than the "8" shape because the 4BCL and 1C2B data have more simplices (triangles) in their original surfaces. T_2 and T_3, which are proportional to the dimensions of the sampling grid, are similar for all three data sets.



(a) (b) (c)

Figure 4.8: Visualizations of the largest pocket extracted using our algorithm for the "Bacteriochlorophyll A Protein" (4BCL). (a) The pocket envelop (green surface) within the cartoon diagram of the protein; (b) The seven BCL ligands shown inside the transparent pocket envelop; (c) The protein surface and residues near the pocket marked green [9].

Table 4.2: Pocket statistics of some sample proteins.

PDB ID	# of pockets	Largest pocket (^{3})	Ligand	Rank
1SNC	1	948.7	THP	1
1HVI	1	1137.5	A77	1
2ACK	6	1448.5	EDR	5
2POR	1	13126.4	C8E	1
1ROB	1	1883.9	C2P	1
2NPX	1	7743.8	CYO, FAD & NAD	1
4BCL	2	9184.8	BCL×7	1
1GPD	5	16182.8	NAD	1
3EST	2	2747.8	-	-
1ELA	4	1249.5	ISO	2
1FKF	1	975.5	FK5	1
11BG	1	5320.8	U2G×4	1

Compared to many other pocket extraction algorithms, e.g. CAST [47] and CASTp [9], the pocket functions computed in our algorithm provide more flexible visualizations and powerful ways for quantitative analyses. Most importantly, the extracted pockets are shown to be correct and overlap with the binding ligands very well. Figure 4.8 shows the result of pocket extraction for the protein 4BCL. The envelop of the largest pocket is displayed as a green surface inside the cartoon diagram of the protein in Figure 4.8 (a). The largest pocket almost perfectly encompass all seven BCL ligands of the protein, as illustrated in Figure 4.8 (b). Table 4.2 gives more examples to demonstrate the effectiveness of the algorithm. Those examples are selected without any special consideration towards our algorithm, most of which are examples used in [9].

In Table 4.2, the third column shows the number of pockets extracted for each protein. To remove small pockets, we used the MACT data structure discussed above in the implementation to prune geometrically insignificant nodes. We set a very conservative threshold of 1%, which means a pocket would be discarded only if its volume is less than 1% of the total pocket volume, because a pocket of such a small volume is too small to be a binding site. For most proteins, the pockets with the largest volumes often act as the ligand binding sites. The fourth column is the computed size of the largest pocket. The last two columns contain the name of the bound ligands and the rank of the binding pocket among all pockets in terms of their sizes.

Figure 4.9 visualize the extracted pockets for the proteins listed in 4.2. The proteins are visualized as cartoons to illustrate their secondary structures, and ligands as sticks. The envelops of the binding pockets are rendered as transparent yellow surfaces to show the ligands inside them, while other pocket envelops are rendered as green opaque surfaces. For all the examples, we

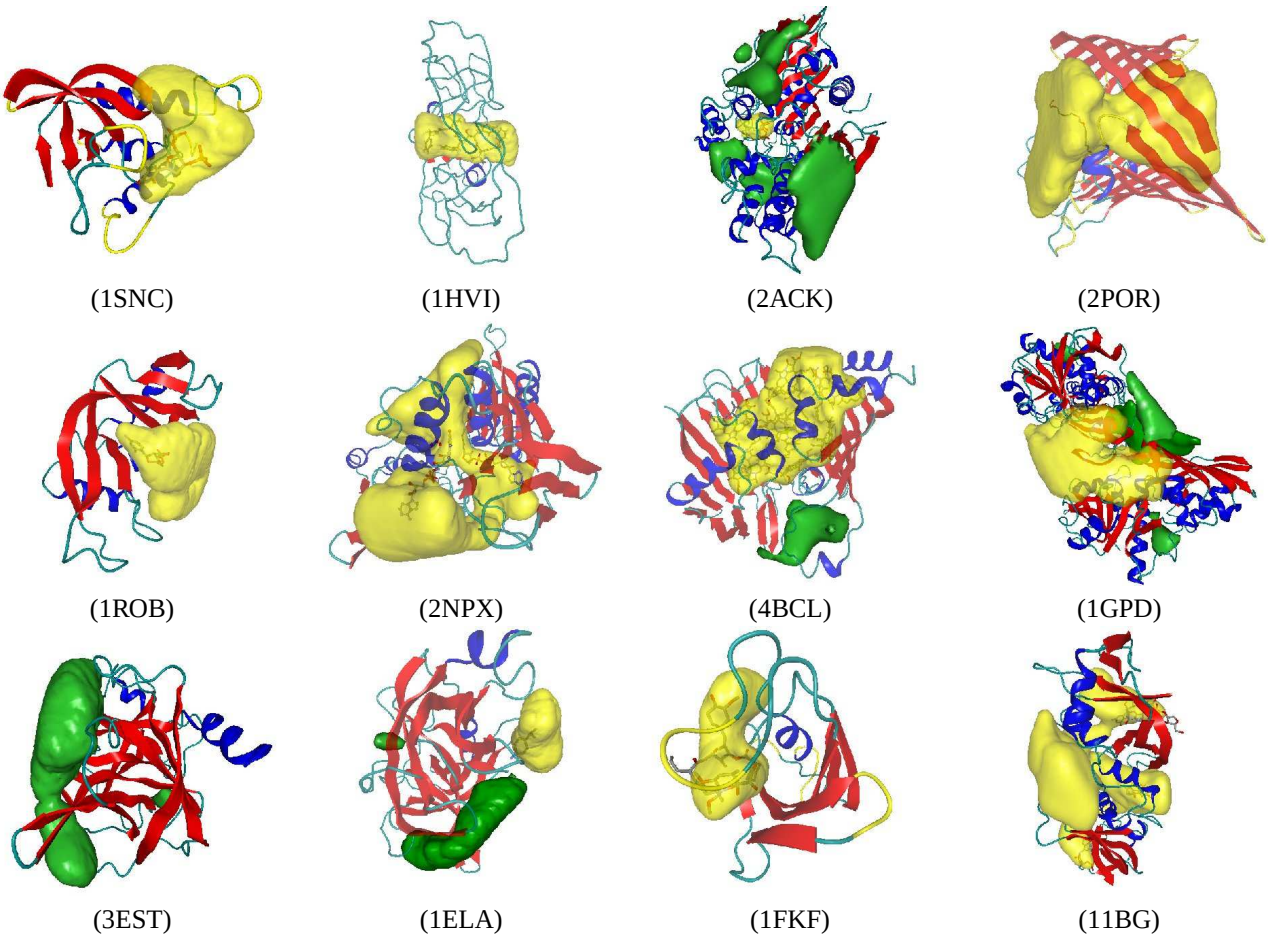
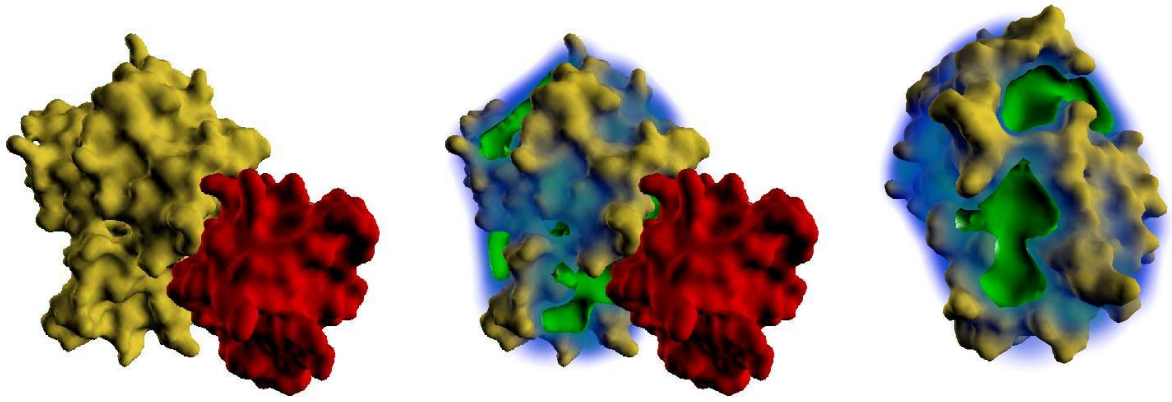


Figure 4.9: Visualizations of pockets for the proteins listed in Table 4.2

can see that the extracted pockets fit very well with the shapes of the proteins. While the quantitative measures of the pockets correlate with the results in the CASTp[9] reasonably well, they certainly don't match exactly. This may be due to two major reasons. First we used a more sophisticated model of protein surfaces in the implementation than union-of-ball used in the CASTp. Second the algorithms may segment the pockets differently. For example "Porin" (2POR) has 45 pockets computed in the CASTp compared to only one in our algorithm.

We believe our algorithm is quantitatively more accurate according to the visualizations. Our definition of pockets is mathematically sound and the extracted pockets match the protein shapes and overlap well with the binding ligands as in Figure 4.9.

Furthermore, the CASTp algorithm has its limitations in finding shallow pockets (depressions) with large openings on protein surfaces, but our pocket algorithm still works correctly in those cases. Figure 4.10 (a) shows a docking example of a protein chain (PDB ID: 1YCC) on another protein (PDB ID: 1CCP) in a protein complex (PDB ID: 2PCC). The docking interface has large but shallow pockets, which the CASTp[9] fails to extract. However, our algorithm successfully computes the pockets on the docking interface, as shown in Figure 4.10 (b) and (c). In Figure 4.10 (b), the extracted pocket function is rendered with volume rendering and the top four pocket envelops are extracted from the pocket function and displayed as green surfaces. We can see that the pockets on the docking interface are successfully calculated and the docked protein (1YCC) interacts with two major pocket envelops on the substrate protein (1CCP), which are the largest and fourth largest computed pockets respectively. Figure 4.10 (c) looks at the volume rendered pocket function and the two pocket envelops on the docking interface from a direct angle. The pocket functions successfully computed by our algorithm may be applied to improve the solutions to the docking problem, which tries to find the best configuration of how a ligand docks on the substrate protein.



(a) (b) (c)

Figure 4.10: (a) Docking example with the protein chain 1YCC (red) on the protein 1CCP (yellow). (b) The pocket function of the protein 1YCC is rendered with volume rendering and four largest pocket envelops are displayed as green surfaces. (c) The volume rendered pocket function and the two pocket envelops on the docking interface.

4.2 Tunnels

Many applications in shape modeling require to identify the salient features of a given shape. Some of them such as assembly planning, feature tracking, animations, structure elucidation of bio-molecules, human-body modeling benefit from a semantic annotation of the features. One such natural annotation is achieved by classifying the features as 'tubular' and 'flat'. Obviously, this annotation is ambiguous since the feature-space is a continuum resulting into features that cannot be simply classified as tubular or flat. Nevertheless, many designed and organic shapes have pronounced features that are perceived to be tubular and flat. We seek to identify these features using a topological method. The unstable manifolds induced by a shape distance function identify some one- and two-dimensional subsets of the medial axis. The preimage of a function that maps the points on the surface to the medial axis provides an association of the shape to these one- and two-dimensional subsets. The preimage of the one-dimensional subset is called tubular whereas that of the two-dimensional subset is called flat. Our experimental result

shows that this classification can be effectively approximated for many datasets in practice.

Given a compact surface Σ smoothly embedded in $\mathbb{R}^3$, a distance function h_Σ can be assigned over $\mathbb{R}^3$ that assigns to each point its distance to Σ.

$$h_\Sigma : \mathbb{R}^3 \to \mathbb{R}, \; x \mapsto \inf_{p \in \Sigma} \|x - p\|$$

In applications, Σ is often known via a finite set of sample points P of Σ. Therefore it is quite natural to approximate the function h_Σ by the function

$$h_P : \mathbb{R}^3 \to \mathbb{R}, \; x \mapsto \min_{p \in P} \|x - p\|$$

which assigns to each point in $\mathbb{R}^3$ the distance to the nearest sample point in P.

In this section, we start with a finite sample P of Σ and identify the index 1 and index 2 saddle points of h_P from the Voronoi diagram $\mathrm{Vor}P$ and its dual Delaunay triangulation $\mathrm{Del}P$ of P. We then select only the saddle points of both indices which lie on the interior medial axis of Σ and compute their unstable manifolds. The unstable manifold of index 1 saddle points (U_1) are two dimensional whereas those of index 2 (U_2) are one dimensional. Exact computations of U_1 is prone to numerical error. See Section 4.3 for details about Voronoi diagram, Delaunay triangulation, and definition, properties and computation of saddle points. Then, the algorithm given below maps the points belonging to U_1 and U_2 back to Σ. The image of U_1 under the mapping gives the flat regions of Σ and that of U_2 gives its tubular regions.

4.2.1 Feature Annotation Algorithm

Mapping of Unstable Manifolds to Σ

There is a natural association between the medial axis M_Σ and Σ via the map $\phi : \Sigma \to M_\Sigma$ where $\phi(x)$ is the center of the medial ball touching Σ at x. Following this map, any subset $A \subseteq M_\Sigma$ can be associated with $\phi^{-1}(A) \subseteq \Sigma$. Let A_1 and A_2 be the closure of the unstable manifolds of index 2 and index 1 saddles in M_Σ defined by the distance function h_Σ. Recall that, generically, A_1 is one-dimensional and A_2 is two-dimensional. Ideally, we would like to identify $\phi^{-1}(A_1) \subseteq \Sigma$ as tubular and $\phi^{-1}(A_2) \subseteq \Sigma$ as flat. As we have an approximation of h_Σ by h_P, we compute these tubular and flat regions for the unstable manifolds in the approximate medial axis which we denote also as M_Σ for convenience.

We face a difficulty to compute an approximation of the preimage of ϕ from the approximate medial axis M_Σ. We are interested in computing an approximation of the preimage of $M'_\Sigma = A_1 \cup A_2 \subseteq M_\Sigma$ under the map ϕ.

Unfortunately, this requires an expensive computation to cover the entire M'_Σ which often spans a substantial portion of M_Σ. A naive approach is to take only a sample of M'_Σ, namely the Voronoi vertices, and then associate them to P, a sample of Σ, via the Voronoi-Delaunay duality. This also proves useless because M'_Σ does not contain all the Voronoi vertices and therefore many points in P cannot be covered by this Voronoi-Delaunay duality.

It turns out that the distance function h_P again proves to be useful to establish a correspondence between Σ and M'_Σ. Recall that, the stable manifold of a critical point is a collection of points whose orbits terminate at that critical point. Let X and Y be the set of maxima in $A_1 \subseteq M'_\Sigma$ and $A_2 \subseteq M'_\Sigma$ respectively. Consider the stable manifolds of the maxima in X and Y. The points in P that are in the stable manifolds of X are associated with the tubular regions and those in the stable manifolds of Y are associated with the flat regions. If a point belongs to the stable manifolds of maxima in X as well as in Y, we tag it arbitrarily. These points belong to the regions where a tubular part meets a flat part. Subsequently, every triangle of the surface reconstructed by TIGHT COCONE is tagged as flat or tubular if at least two of its vertices are already marked as flat or tubular respectively.

Computation of stable manifold of maxima has been described in [31] and its approximation was given in [19]. We follow the approximate algorithm to compute the stable manifolds of the local maxima lying on M'_Σ.

Figure 4.11 shows the set M'_Σ of the molecule data 1IRK, and the set of maxima belonging to that set and identified as linear or planar. The corresponding flat and tubular portions of the surface captured by the mapping via stable manifold of these maxima - colored golden and cyan respectively - are shown in Figure 4.13. We collected the protein from Protein Data Bank [5] and blurred the molecule at a resolution 8 angstrom. Further we took the vertex set of a suitable level set as the input to our

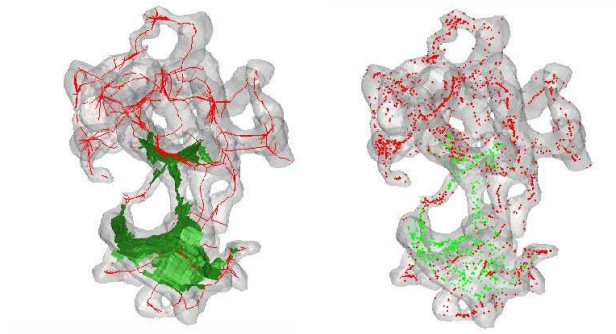


Figure 4.11: One dimensional subset of the interior medial axis is drawn in red and the two dimensional subset of the medial axis is drawn in green for the molecule data 1IRK. The right subfigure shows the selection of local maxima of the distance function in those two parts, colored accordingly.

program. We verified the result with the existing literature in structural biology and we have seen that the flat regions identified by our algorithm correspond to the β-sheets of the protein molecule.

Annotation Algorithm

The modules described in the previous sections and subsections can thus be combined to devise an algorithm for automatic feature annotation of Σ. We give the pseudo-code of this annotation algorithm here.

IDENTIFY_FLAT_AND_TUBULAR_REGIONS(P)
1 Compute Vor$P$ and Del$P$.
2 Compute the interior Medial Axis M_Σ
  by TIGHTCOCONE_AND_MA (P)
3 C_1 = set of index 1 saddle points lying on M_Σ and
  C_2 = set of index 2 saddle points lying on M_Σ,
4 $A_1 = A_2 = \emptyset$
5 for all $c \in C_2$
6 $A_1 = A_1 \cup$ UM_INDEX_2(c)
7 for all $c \in C_1$
8 $A_2 = A_2 \cup$ APPROX_UM_INDEX_1(c)
9 X = maxima in A_1
10 Y = maxima in A_2
11 Σ_{Tubular} = MAPPING_VIA_STABLE_MANIFOLD(A_1)
12 Σ_{Flat} = MAPPING_VIA_STABLE_MANIFOLD(A_2)
13 return Σ_{Tubular} and Σ_{Flat}.

Figure 4.12: Pseudo-code of the feature annotation algorithm.

4.2.2 Results

Implementation Issues

The algorithm works on the Voronoi-Delaunay diagram of the set of sample points lying on the surface. To robustly compute the Delaunay triangulation and its dual Voronoi diagram for the input set of points we use the library CGAL [13] which is freely available.

Even in CGAL-framework, we sometimes face the degenerate case of five or more points being cospherical. This case has to be handled with special care because only one Voronoi vertex is repeated and therefore the flow along the Voronoi edges is not well-defined anymore. To deal with such situations, we modify the algorithm slightly. At the start of the algorithm we collect the sets of tetrahedra which are cospherical. While computing the unstable manifold of index 2 saddle points, if the polyline hits a Voronoi vertex whose dual is a member of one such cospherical cluster, the algorithm automatically advances through the non-degenerate Voronoi edges which are dual to the triangles bounding the cospherical lump. This degeneracy poses a more serious threat to the computation of unstable manifold of index 1 saddle points and at this stage, we do not extend the manifold through any Voronoi edge whose dual Delaunay triangle is shared by two cospherical tetrahedra.

There are some parameters involved in the full feature annotation process. For surface reconstruction and medial axis approximation we used the software [16]. The parameters for these routines are described in [23], [25]. For noisy inputs we replace TIGHT COCONE by ROBUST COCOCNE and the parameters for that step are again described in [24]. The rest of the algorithm requires only one parameter k which is the number of flat regions to be output.

Performance

Figure 4.13 shows the performance of the annotation algorithm on six datasets. The datasets have been chosen to represent different domains this algorithm can possibly be applied in. PIN is a CAD dataset which has two tubular parts joined in the middle through a flat portion. The algorithm can identify them correctly. Similarly the method can correctly identify the handle as the tubular and the body as the flat region for the MUG dataset. In the second row we show the performance of our method on two protein molecules obtained from Protein Data Bank [5]. We took the crystal structure of these two molecules (PDB ID 1CID and 1IRK) and blurred them with Gaussian kernel. We further took a level set which represents a molecular surface and used the vertex set of that isosurface as the input to our algorithm. The flat features identified by our method correspond to the β-sheets of the secondary structure of those two proteins. In the last row we show the result on two free form objects containing both flat and tubular features. As we can see, the palm of the HAND has been detected as flat whereas the fingers have been detected as tubular. Our method can also capture the major flat and tubular features of ALIEN.

We purposefully show the performance of the algorithm on ALIEN as it brings forth the limitations of our algorithm. We see that a portion of the arm has been identified as flat. This is because the initial reconstruction phase could not separate the beginning of the arm from the torso due to lack of sampling. Secondly, one of the feet could not be fully identified as flat by our algorithm. This is because the approximate medial axis, that we started with, is not a close approximation of the true medial axis in that region, again due to lack of sampling. Because of that, our method fails to collect sufficiently many index 1 saddle points leading to incomplete identification of flat features in that region.

Figure 4.14 shows the performance of our method on noisy dataset HORSE. Instead of applying TIGHT COCONE, we first mark the interior and exterior of the closed surface from its noisy point sample by ROBUST COCONE ([24]) and then obtain the interior medial axis and proceed further with the unstable manifold computation and feature identification. Originally there were some thin flat regions due to the unstable manifold of some index 1 saddle points near the hind legs which we filtered out by thresholding in order to get a clean skeleton of the HORSE. In the rightmost picture we see some white triangles near the ears. These triangles appear as the mapping via stable manifold misses some points on the surface in that portion.

Timings

The time and space complexity of the algorithm is dominated by the complexity of Delaunay triangulation. We report the timings of the entire execution into four major steps

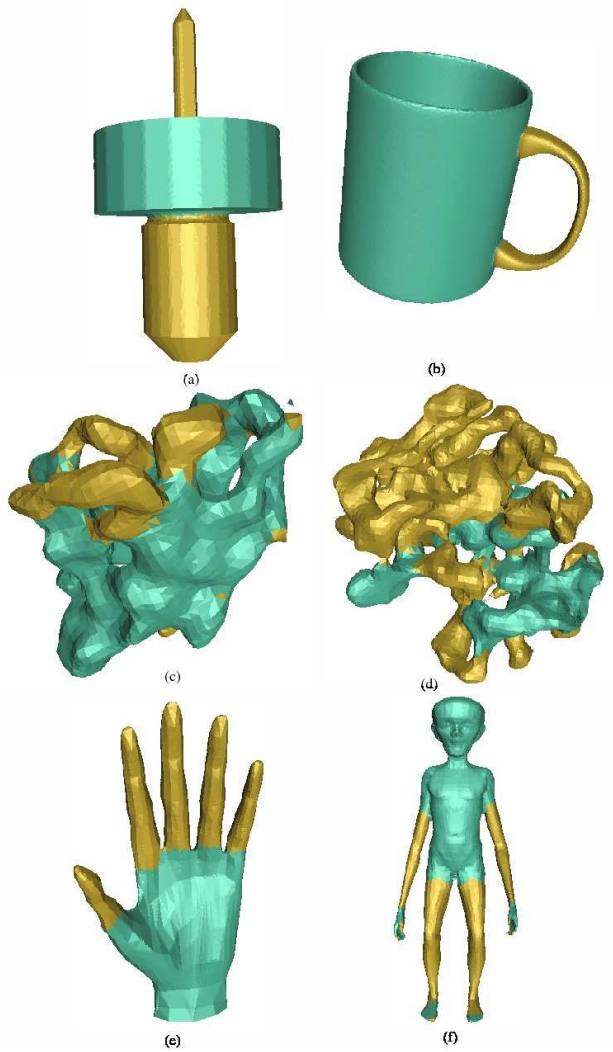


Figure 4.13: Performance of the feature annotation algorithm. The models are (a) PIN, (b) MUG, (c) molecule 1CID (d) molecule 1IRK, (e) HAND, (f) ALIEN

1. Step 1: Building the Voronoi-Delaunay diagram of the point set (Line 1 of Figure 4.12).
2. Step 2: Computation of interior medial axis. (Line 2 of Figure 4.12).
3. Step 3: Computation of unstable manifold of index 1 and index 2 saddle points lying on the interior medial axis. (Line 3-8 of Figure 4.12).
4. Step 4: Mapping the maxima in the planar and linear portion of the medial axis back to the surface. (Line 9-13 of Figure 4.12).

We built the code using CGAL [13] and gnu C++ libraries. The code is compiled at an optimization level $-O2$. We run the experiments in a machine with INTEL XEON processor with 1GB RAM running at 1GHz cpuspeed. Table 4.3 reports the time taken in the four steps of the algorithm for a number of datasets. It is clear from the breakup of timing that the first two steps of building the Delaunay triangulation and then computing the interior medial axis are the two most expensive steps. For noisy datasets, additionally ROBUST COCONE is used to obtain an initial in-out marking. This step is comparatively inexpensive. For

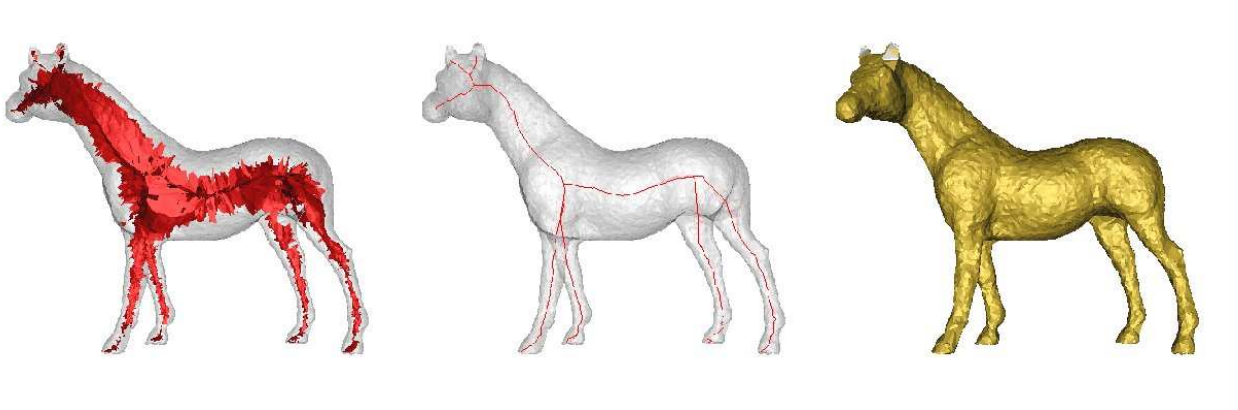


Figure 4.14: Results on Noisy Data.

example, for NOISY HORSE (48,000 points) this step only adds 10 sec to the whole computation time which is approximately 100 sec.

object	# points	Step 1 (sec.)	Step 2 (sec.)	Step 3 (sec.)	Step 4 (sec.)
1CID	5170	7.59	15.63	6.69	0.39
1IRK	13940	29.88	43.93	15.63	1
HEADLESS MAN	16287	18.63	51.30	16.01	1.26
PIN	15530	15.73	41.4	21.53	0.92
CLUB	16864	20.54	47.3	19.83	1.24
MUG	27109	37.68	83.28	47.14	2.19
HAND	40573	53.48	120.16	40.67	2.69
P8	48046	33.46	136.59	39.97	3.22
1BVP	53392	148.18	159.52	62.19	3.53
ALIEN	78053	102.62	242.33	64.11	5.4

Table 4.3: Timings

4.3 Curation of Surface

Related Work

Pockets, Tunnels

Pocket Extraction

Several pocket extraction methods have been developed and published. Delaney [18] used cellular logic operations on grid points in a spirit similar to our two-step marching algorithm, but its results were very rough approximations and difficult for further visualizations and analyses. LIGSITE is a simple algorithm that scans along the x , y, and z axes and the cubic diagonals for areas that are enclosed on both sides by a protein [35], in order to identify pockets. It would not be accurate if the pockets are not along those predefined directions. Edelsbrunner et al. [27] computed pockets for molecular surfaces based on the union-of-balls model by using Delaunay triangulations and alpha shapes. In their method, the Delaunay triangulation D_B (and its dual Voronoi diagram) are first constructed for the set B of all atomic centers[27]. A flow relation can then be defined for two Delaunay tetrahedra, $\tau \in D_B$ and $\sigma \in D_B$, if they share a common plane and the dual Voronoi vertex of τ lies on different sides of the plane from σ. If $\tau \prec \sigma$, τ is called a predecessor of σ and σ a successor of τ. A tetrahedron flows to infinity, if its dual Voronoi vertex is outside D_B or its successor flows to infinity. The alpha-shape $A_B \subset D_B$ at $\alpha = 0$ is the sub-simplex of D_B contained in the union of balls. Pockets P are defined in[27] as the set of Delaunay tetrahedra that do not flow to infinity and do not belong to the alpha-shape A_B, i.e.

$$P \subset D_B - A_B.$$

The alpha-shape based algorithm was implemented and tested for a number of sample proteins [47]. This method has the shortcoming of being dependent on the union-of-balls model and the pockets represented by alpha-shapes are usually not smooth. The SURFNET method [34, 42] tries to detect the clefts on the protein surfaces by placing spheres between all pairs of atoms and shrinking their sizes until they do not intersect any atoms. The sphere fitting process results in a number of separate groups of interpenetrating spheres, corresponding to the cavities and clefts of the protein. However, a set of spheres do not match well with the actual pocket shapes. Masuya and Doi described a method of computing protein cavities using digital morphological operations [49]. They represented the surfaces and interior of the molecular surfaces as a set of discrete grid points X. Given a probe sphere, represented also as a set of grid points P, the dilation and erosion operations of X by P are defined as

$$\begin{aligned} X \oplus P &= \bigcup_{x_i \in X} \bigcup_{p_j \in P} \{x_i + p_j\} \\ X \ominus P &= \bigcap_{x_i \in X} \bigcup_{p_j \in P} \{x_i + p_j\} \end{aligned}$$

The closing operation of X by P defined as a dilation followed by an erosion, $X \bullet P = (X \oplus P) \ominus P$, is used to obtain a filled molecule X^f. The set of grid points in $X^f - X$ are taken as the cavities. While the spirit of the dilation and erosion operations is the most similar to that of the two-step marching algorithm in this paper, the algorithms and implementations are vastly different. Our algorithm can much more accurately model the molecular surfaces. The dilation and erosion operations in [49] depend critically on the radius of the probe sphere P. If the radius of P is too large, the algorithm would miss important pockets and tunnels on the molecular surface; if the radius of P is too small, many small cavities would be generated from a large real pocket. So it requires an "optimal" radius to be selected for each protein individually. Our algorithm in this paper does not have those limitations. The PASS (Putative Active Sites with Spheres) [10] is a geometric pocket extraction algorithm that uses small probe spheres to fill the "buried" regions on the protein surfaces. A number of parameters such as the probe radius and "burial count" are chosen a prior for this algorithm. The results are set of probe spheres represent the protein pockets. PocketFinder [1] is not a geometrical method, but uses a volumetric function of the smoothed van der Waals potential of the protein. The van der Waals potential is approximated as the sum of Lennard-Jones formula of all atoms and then smoothed using a simple iterative averaging process. The bounding surfaces (pocket envelops) of the protein pockets are then computed as a level set of the potential function. However, the parameters for the smoothing process and the isovalue of the level set are determined ad hoc and the pocket surfaces do not match the geometrical shapes of the proteins very well. Q-SiteFinder [43] is another energy-based method for predicting protein-ligand binding sites. Although it is different from the geometric algorithm in this paper, the interaction energy computation described in [43] may be combined with our geometric algorithm to improve the accuracy of predictions.

Pockets Representation

It is advantageous to represent the pocket shapes with a 3D volumetric function, which allows further analyses of the shapes. Complicated shapes are often captured via volumetric functions coupled to morphological operations on the functions. In 2D range images, Krishnapuram and Gupta [41] uses dilation and erosion operations to detect and classify edges; Gil and Kimmel [33] discussed algorithms for computing one-dimensional dilation and erosion operators. In addition to the extraction of polygonal surfaces from volumetric functions, 3D polygonal models are also converted into volumetric representations and then modified, repaired and simplified using morphological operations [52, 29].

Shape Segmentation

Shape segmentation is a problem related to finding pockets in the molecular surfaces, which has been studied using different geometric and topological methods such as shock graphs [55], medial axes [45], skeletons [61], Reeb graphs [36], and others [48, 37, 46]. It is applicable to consider the pocket extraction as the problem of segmenting the complementary space outside the molecular surface S. A notable approach of shape segmentation is based on the Morse theory, which segments the domain manifold M into stable (unstable) manifolds [20] or Morse-Smale cells [28] of critical points of a Morse function. The Morse function commonly used for shape segmentation is the distance function $h(\mathbf{x})$ to a set of discrete points P [20, 32, 27]:

$$h(\mathbf{x}) = \min_{\mathbf{p}\in P} \|\mathbf{x}-\mathbf{p}\|.$$

Here again the Delaunay triangulation (and the dual Voronoi decomposition) can be computed for the points in P. The critical points of h are the intersections of Delaunay elements with their Voronoi complements. The stable-manifolds of the critical points of the distance function to a set of discrete points are called the flow complex in [32], and which is homotopy equivalent to its alpha-shape [21]. The stable manifolds of maxima has the same dimension as the the manifold M and give a segmentation of M. However, a large number of points are necessary to sample complex surfaces and a large number of maxima and stable manifolds would segment the space into many small pieces that have no direct correspondence to the pockets.

Identifying Tunnels

Because of the significance of the problem, quite a few work spanning various approaches have been reported in the literature. To mention a few, we refer to the curvature based methods of [64] and [50, 51], the fuzzy clustering method of [38], the method based on PCA of surface normals by [54], the hybrid variational surface approximation by [66] and the Reeb graph approach of [57] and [65]. Remarkably the distance function over $\mathbb{R}^3$ which is defined by the distance to the boundary of the shape has not been fully used for feature annotation. In the context of surface reconstruction, topological structures induced by distance functions have been analyzed by Edelsbrunner [26], Chaine [14] and Giesen and John [31]. Chazal and Lieutier [15] and Siddiqi et al. [58] have used it for medial axis approximations. Dey, Giesen and Goswami used the topological structures induced by the distance function to segment a shape [19]. However, this work stops short of using the topological structures for feature annotations. In this paper we complete this step.

Relevant Mathematics

Voronoi-Delaunay Diagram of P

[*Repeated from Chapter 2]

For a finite set of points P in $\mathbb{R}^3$, the Voronoi cell of $p \in P$ is

$$V_p = \{x \in \mathbb{R}^3 : \forall q \in P - \{p\}, \|x-p\| \leq \|x-q\|)\}.$$

If the points are in general position, two Voronoi cells with non-empty intersection meet along a planar, convex Voronoi facet, three Voronoi cells with non-empty intersection meet along a common Voronoi edge and four Voronoi cells with non-empty intersection meet at a Voronoi vertex. A cell decomposition consisting of the *Voronoi objects*, that is, Voronoi cells, facets, edges and vertices is the Voronoi diagram $\mathrm{Vor}P$ of the point set P.

The dual of $\mathrm{Vor}P$ is the Delaunay diagram $\mathrm{Del}P$ of P which is a simplicial complex when the points are in general position. The tetrahedra are dual to the Voronoi vertices, the triangles are dual to the Voronoi edges, the edges are dual to the Voronoi facets and the vertices (sample points from P) are dual to the Voronoi cells. We also refer to the Delaunay simplices as *Delaunay objects*.

Induced Flow

The distance function h_P induces a flow at every point $x \in \mathbb{R}^3$. This flow has been characterized earlier [31]. See also [26]. For completeness we briefly mention it here.

Critical Points. The critical points of h_P are those points where h_P has no non-zero gradient along any direction. These are the points in $\mathbb{R}^3$ which lie within the convex hull of its closest points from P. It turns out that the critical points of h_P are the intersection points of the Voronoi objects with their dual Delaunay objects.

- *Maxima* are the Voronoi vertices contained in their dual tetrahedra,
- *Index 2 saddles* lie at the intersection of Voronoi edges with their dual Delaunay triangles,
- *Index 1 saddles* lie at the intersection of Voronoi facets with their dual Delaunay edges, and
- *Minima* are the sample points themselves as they are always contained in their Voronoi cells.

In this discrete setting, the index of a critical point is the dimension of the lowest dimensional Delaunay simplex that contains the critical point.

Flow. For every point $x \in \mathbb{R}^3$, let $V(x)$ be the lowest dimensional Voronoi object that contains x and $D(x)$ be its dual. Now *driver* of x, denoted as $d(x)$, is defined as

$$d(x) = \mathrm{argmin}_{y \in D(x)} \|x-y\|$$

The direction of steepest ascent can be uniquely determined by a unit vector in the direction of $x - d(x)$. The critical points coincide with their drivers. Now one can assign a vector v at every x with a zero vector assigned at the critical points. The resulting vector field is not necessarily continuous. Nevertheless, it induces a *flow* in $\mathbb{R}^3$. This flow tells how a point x moves in $\mathbb{R}^3$ along the steepest ascent of h_P and the corresponding path is known as the *orbit* of x. We can also define an *inverted orbit* of x where x moves in the direction of steepest descent.

Stable and Unstable Manifolds. For a critical point c its stable manifold is the set of points whose orbits end at c. The unstable manifold of a critical point c is the set of points whose inverted orbits end at c. The structure and computation of stable manifolds of the critical points of h_P were described in [31]. They can be computed from the Delaunay triangulations of the given point sets though they may not be subcomplexes of the Delaunay triangulations. For computational advantages they are also approximated by Delaunay subcomplexes as in [19].

As the Delaunay and Voronoi diagrams, the structures of stable and unstable manifolds have a duality. Interestingly, one can compute the unstable manifolds and their approximations from the Voronoi diagrams. Here we state some of the facts about the unstable manifolds of the critical points.

1. MAXIMA. The unstable manifold is the local maximum itself.
2. INDEX 2 SADDLES. The unstable manifold of an index 2 saddle point is a polyline starting at the saddle point and ending at a maximum.
3. INDEX 1 SADDLES. The unstable manifold of an index 1 saddle point is a two dimensional surface patch which is bounded by the unstable manifold of index 2 saddle points.
4. MINIMA. The unstable manifold of a local minimum is a three dimensional polytope bounded by the unstable manifold of critical points with higher indices.

Flow on Voronoi Objects

Before we state the connection between the flow induced by h_P and the Vor-Del diagram of P, we would like to state some facts about the relative position of Voronoi and Delaunay objects. These relative positions can describe the nature of flows in the Voronoi objects. These facts were clearly explained in [26] for a more general setting of power distance.

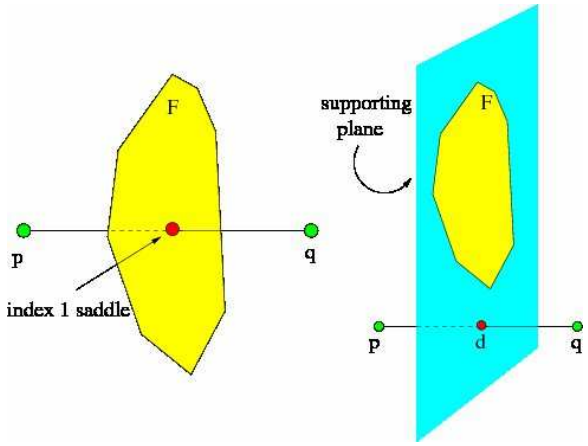


Figure 4.15: Relative position of a Voronoi facet F with respect to its dual Delaunay edge pq. The left picture shows the creation of an index 1 saddle point. The right picture shows the position of the driver d of F.

Fact 1. *The unoriented normal to the supporting plane of a Voronoi facet is along its dual Delaunay edge and the plane passes through the midpoint of the edge. The Delaunay edge, though, may or may not intersect the dual Voronoi face.*

Figure 4.15 illustrates the two possibilities that may arise. The left figure corresponds to the situation that results in an index 1 saddle point.

Fact 2. *The supporting line of a Voronoi edge always intersects the plane of the dual Delaunay triangle at its circumcenter and is along its unoriented normal. The Voronoi edge may or may not intersect the interior of the Delaunay triangle.*

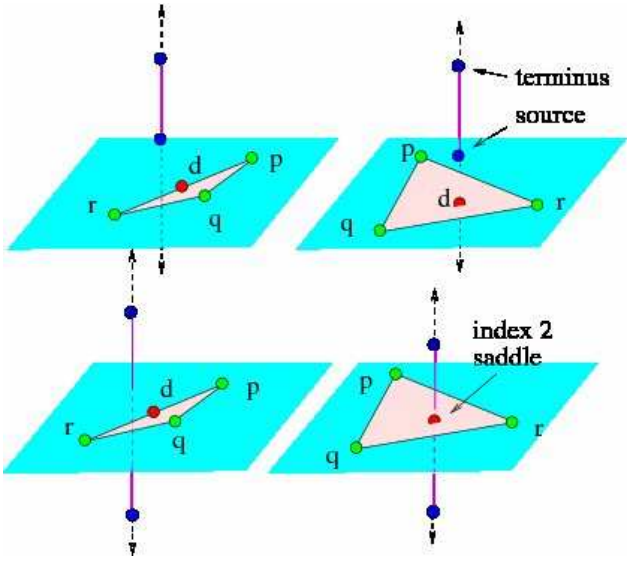


Figure 4.16: Relative position of a Voronoi edge e with respect to its dual Delaunay triangle pqr. The blue circles denote the two Voronoi vertices defining e. The driver of e is marked d and the supporting plane of triangle pqr is drawn in cyan.

Figure 4.16 lists the four possible scenarios. The bottom right corresponds to the generation of an index 2 saddle point.

We have already seen that the critical points of h_P can be computed from $\mathrm{Vor}P$ and $\mathrm{Del}P$. Also, the driver of a point x comes from the Delaunay object dual to the Voronoi object x lies in. In this context we would like to state the following lemma which is key to the further computations.

Lemma 4.3.1. *All interior points of a Voronoi object have the same driver.*

This result can be easily proved by considering all the different cases regarding the dimension of the Voronoi object and its position with respect to its dual Delaunay object.

By Lemma 4.3.1 and Facts 1 and 2 we can list the possible position of the drivers of the points lying in the interior of a certain dimensional Voronoi object.

Position of Drivers

Voronoi Cell

For a Voronoi cell V_p, the dual Delaunay object is a singleton set containing the sample point p and therefore all points x in the interior of V_p has p as their driver.

Voronoi Facet

Consider a Voronoi facet in the intersection of V_p and V_q. The dual Delaunay edge is pq and the midpoint of pq is the driver of all x lying in the interior of the Voronoi facet (Figure 4.15(right)).

Voronoi Edge

Next, consider a Voronoi edge in the intersection of V_p, V_q, V_r. As Fact 2 and Figure 4.16 indicate, the infinite line segment containing the Voronoi edge may or may not intersect the convex hull of p, q, r leading to two different possibilities

Case 1.1 In case of intersection, the circumcenter of *pqr* is the driver. Such Voronoi edges will be termed *non-transversal* edges as the flow is along the edge itself. The Voronoi edge has two Voronoi vertices as its endpoints. If both of them are in the same half-space defined by *pqr*, the closer Voronoi vertex is called *source* and the further one is called *terminus* of the Voronoi edge because the flow is directed from the closer to the further vertex. Figure 4.16 (top right) illustrates this case.

Case 1.2 If the Voronoi edge does not intersect the affine hull of $p, q,$ and r, the midpoint of the edge opposite to the largest angle of *pqr* is the driver. These Voronoi edges will be termed as *transversal*. If any point x moving along its orbit hits one such edge, the position of the driver implies that it will enter the Voronoi facet dual to the Delaunay edge opposite to the largest angle in *pqr*. Such Voronoi facet will be termed *acceptor* facets of that *transversal* Voronoi edge. Figure 4.17 illustrates the situation.

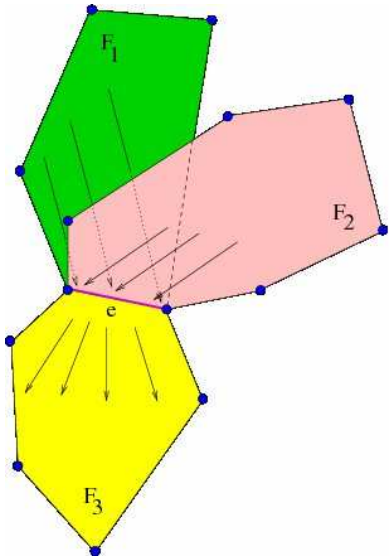


Figure 4.17: Transversal Voronoi edge e is shown in red with three incident Voronoi facets. Flow direction is shown with arrows. Flow from either of F_1 or F_2 hits e and enters F_3, the acceptor of e.

Voronoi Vertex

The case of Voronoi vertex again requires the analysis of two different cases. We assume, that it is outside its dual tetrahedron because otherwise it is a local maximum and hence is its own driver. Let v be a Voronoi vertex with the dual tetrahedron σ whose four neighbors are $\sigma_i, i = 1 \ldots 4$. Further, let the corresponding shared triangles between σ and σ_i be $t_i, i = 1 \ldots 4$ where $w_i, i = 1, \ldots 4$ is its opposite vertex in σ.

Case 2.1 There is only one triangle t_i of σ for which the Voronoi vertex v and the opposite vertex w_i lie in two different half-spaces defined by t_i. Let e_i be the Voronoi edge between the duals of σ and σ_i. Then, the driver for v (dual to σ) is same as the driver of e_i. In such cases, e_i is termed as the *outgoing* Voronoi edge of v. See top row of Figure 4.18 for an illustration.

Case 2.2 There are two triangles t_i, t_j of σ for which the Voronoi vertex v and the opposite vertex (w_i and w_j) lie in two different half-spaces defined by the corresponding triangles. Let e_i, e_j be the Voronoi edges defined as in Case 2.1. Note, in this case, both e_i, e_j are the *outgoing* Voronoi edges of v. There are two possibilities that we need to consider further.

Case 2.2.1 Both e_i, e_j are *transversal*: In this case the *acceptor*s of both of them is dual to the Delaunay edge $t_i \cap t_j$ and the corresponding driver is the midpoint of $t_i \cap t_j$. See bottom-left subfigure of Figure 4.18.

Case 2.2.2 One of e_i, e_j is *transversal*: The driver is same as that of the non-transversal Voronoi edge. See bottom-right subfigure of Figure 4.18.

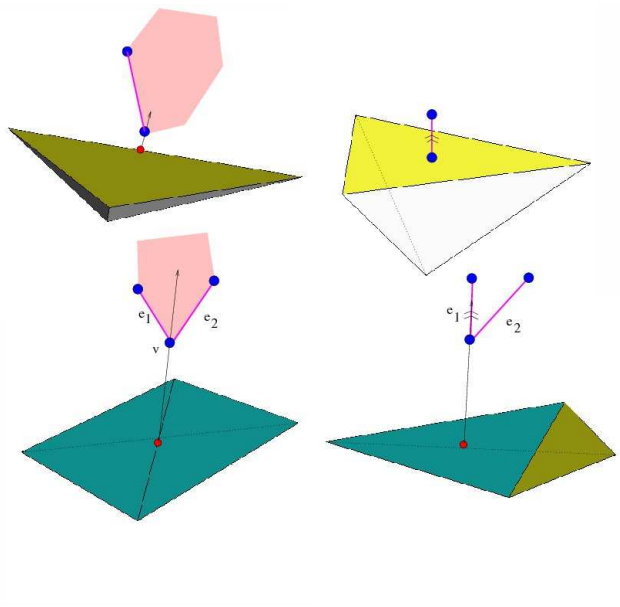


Figure 4.18: Possible driver positions of a Voronoi vertex v according to the cases 2.1 and $2.2.(1-2)$. The acceptor Voronoi facet is shown in pink. The flow along a non-transversal Voronoi edge is shown with a double arrow. The driver is shown in red circle.

In this context we state another lemma that is important for subsequent developments.

Lemma 4.3.2. *Let F be an acceptor Voronoi Facet for the transversal Voronoi edges $e_1 = (v_1, v_2) \dots e_k = (v_k, v_{k+1})$ around it.*

1. *The Voronoi edges $e_1 \dots e_k$ form a continuous chain around F.*
2. *The Voronoi vertices $v_2 \dots v_k$ fall in the category 2.2.1. The Voronoi vertices v_1 and v_{k+1} fall in the category 2.2.2.*
3. *F, $e_1 \dots e_k$, $v_2 \dots v_k$ have same driver which is the midpoint of the Delaunay edge dual to F.*

We omit the proofs of all of the above claims.

Computing Unstable Manifolds

Unstable Manifold of Index-2 Saddle Points

In this section we describe the structure and computation of the unstable manifolds of index 2 saddle points.

The unstable manifold of an index 2 saddle point is one dimensional. In our discrete setting it is a polyline with one endpoint at the saddle point and the other endpoint at a local maximum. The polyline consists of segments that are either subsets of non-transversal Voronoi edges or lie in the Voronoi facets. Due to the later case, the polyline may not be a subcomplex of $\mathrm{Vor}\,P$.

Let us consider an index 2 saddle point, c, at the intersection of a Delaunay triangle t with a Voronoi edge e. Let the two tetrahedra sharing f be σ_1, σ_2. The edge e has the endpoints at the dual Voronoi vertices of σ_1 and σ_2, denoted as v_1, v_2 respectively. The unstable manifold $U(c)$ of c, has two intervals - one from c to v_1 and the other from c to v_2. We look at the structure of one of them, say the one from c to v_1, and the other one is similar.

At any point on the subsegment cv_1, the flow is toward v_1 from c. Once the flow reaches v_1, the subsequent flow depends on the driver of v_1. Instead of just looking at v_1, we consider a generic step, where the flow reaches at a Voronoi vertex v and we enumerate the possible situations that might occur depending on the position of driver of v. If v is a local maximum, the flow stops there, as the driver of v is v itself. Otherwise there are two cases to consider.

- v **falls into Case 2.1:** Let the dual tetrahedron be σ and the driver of v is same as that of the Voronoi edge e which is between the dual of σ and one of its neighbors, say σ'. If e is non-transversal, the flow will be along the Voronoi edge e till it hits the Voronoi vertex at the other endpoint (dual to σ'). Otherwise, the flow enters the acceptor Voronoi facet F of e. Due to Lemma 4.3.2, the driver of F is same as the driver of e. Therefore the next piece of the unstable manifold can be uniquely determined by the driver of e, say d and the Voronoi vertex v. It is the segment between v and the point where the ray $\overrightarrow{dv}$ intersects a Voronoi edge of F.
- v **falls under Case 2.2.x:** This situation is similar to the one described above. In case of both of the Voronoi edges being *transversal* (Case 2.2.1), the flow enters the acceptor Voronoi facet. In the other case (Case 2.2.2), the flow follows the non-transversal Voronoi edge.

Some segments of $U(c)$ are not along the Voronoi edges. Wherever the flow encounters a transversal Voronoi edge, it seizes to follow the Voronoi edge and enters a Voronoi facet which is acceptor for that Voronoi edge. This calls for the analysis of the flow when it crosses an acceptor Voronoi facet and hits a Voronoi edge. We have already characterized the position of the driver for a Voronoi edge and thereby classified those edges as either transversal or non-transversal. If the current edge intersected by the ray from the driver to v is a non-transversal edge, the flow will follow that Voronoi edge and hit a Voronoi vertex. Otherwise, it will enter the acceptor Voronoi facet of the Voronoi edge again. There is a technical difficulty we need to point out. Unless the acceptor for this Voronoi edge is different from the Voronoi facet the flow came from, we may encounter a cycle. The following lemma saves us from this awkward situation.

Lemma 4.3.3. *Let F be a Voronoi facet and let d be its driver. Let e be a Voronoi edge for which F is acceptor and x be any point on e. Also assume the ray from d to x intersects a Voronoi edge e'. If e' is transversal, the acceptor of e' is different from F.*

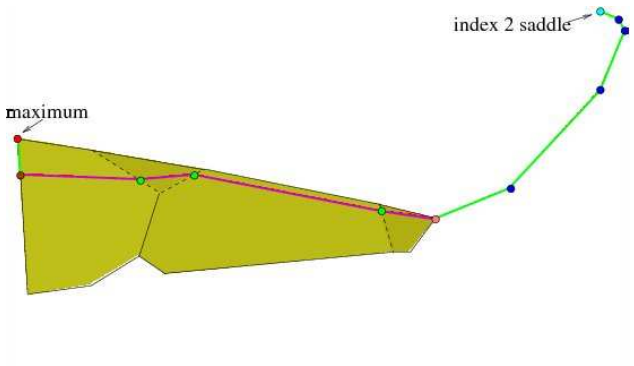


Figure 4.19: Unstable manifold $U(c)$ of an index 2 saddle point c. c is drawn with a cyan circle. The portion of $U(c)$ which is a collection of Voronoi edges is drawn in green with intermediate Voronoi vertices drawn in blue. The pink circle is a Voronoi vertex on $U(c)$ where the flow enters a Voronoi facet. The portion of $U(c)$ which lies inside the Voronoi facets is drawn in magenta. The transversal Voronoi edges intersected by this portion of $U(c)$ are dashed. $U(c)$ ends at a local maximum which is drawn in red.

Figure 4.19 shows an example of the unstable manifold of an index 2 saddle point.

Following the above discussion on the structure of $U(c)$ we devise the algorithm to compute the unstable manifold of an index 2 saddle point c. We assume, the saddle point c carries the information about the two neighboring tetrahedra σ_1, σ_2 and additionally we have access to Del$P$ which is used to evaluate the utility routines like acceptor() , terminus() etc. The pseudo-code of the algorithm is given in Figure 4.20.

UM_INDEX_2(c)
1 $U_1 = cv_1$ and $U_2 = cv_2$
2 $v = v_1$
3 end$(U_1) = v_1$
4 while (v is not a maximum) do
5 if(v is not a Voronoi vertex)
6 $e =$ Voronoi edge containing v
7 if(e is non-transversal)
8 end$(U_1) =$ terminus(e)
9 $U_1 = U_1 \cup$ segment$(v,$ end(U_1)
10 $v =$ terminus(e)
11 else
12 $F =$ acceptor(e)
13 $d =$ driver$(F) =$ driver(e)
14 $x = \overrightarrow{dv} \cap e' \neq \emptyset$, e' is a Voronoi edge of F
15 end$(U_1) = x$
16 $U_1 = U_1 \cup$ segment$(v,$ end(U_1)
17 $v = x$
18 else
19 if(v falls under Case 2.1)
20 $e =$ *outgoing* Voredge (v)
21 repeat steps 7-17.
22 else if(v falls under Case 2.2)
23 $F =$ acceptor(v)
24 repeat steps 13-17.
25 endwhile
26 Similarly compute U_2.
27 return $U_1 \cup U_2$.

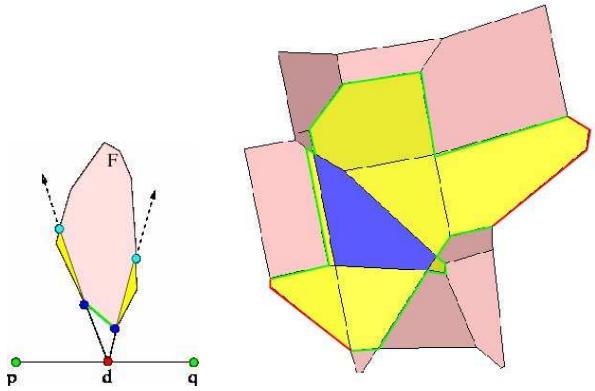
Figure 4.20: Pseudo-code for computation of unstable manifold of an index 2 saddle point.

Unstable Manifold of Index-1 Saddle Points

Unstable Manifold of index 1 saddle points are two dimensional. Due to hierarchical structure, they are bounded by the unstable manifold of index 2 saddle points. In this section we first describe the structure of the unstable manifolds and then describe an algorithm that computes an approximation of the unstable manifold of an index 1 saddle point.

Let us consider an index 1 saddle point, c. This point lies at the intersection of a Voronoi facet F and a Delaunay edge. For any point $x \in F \setminus c$, the driver is c. For all such x, if they are allowed to move in the direction of flow, they will move radially outward and hit the Voronoi edges bounding F. Thus F is in $U(c)$. Now we analyze the flow when a point hits a Voronoi edge.

We have characterized the position of the drivers for a Voronoi edge and we have also seen that depending on the driver, one can classify the Voronoi edges into two categories - transversal and non-transversal. For a non-transversal Voronoi edge, the flow is along the Voronoi edge. Such Voronoi edges lie on the boundary of $U(c)$. On the other hand, $U(c)$ grows via the acceptor facets of transversal Voronoi edges. Depending on the position of the driver, which by Lemma 4.3.2 is same for both the edge and the acceptor facet, a *truncated cone* defines the extension of $U(c)$ into the acceptor Voronoi facet. Consider the cone defined by the two rays emanating from the driver and passing through the endpoints of the transversal Voronoi edge. The intersection of the acceptor facet with the cone defines the truncated cone. The truncated cone hits a continuous chain of Voronoi edges in the acceptor facet. Some of them are completely contained in the truncated cone and some of them are intersected by the two rays and hence are partially contained in it. This chain of edges defines the new boundary of $U(c)$ through some of which $U(c)$ can be extended further recursively. Figure 4.21 shows an example truncated cone in a Voronoi facet F by the driver d and the end Voronoi vertices of the transversal Voronoi edge (green).



(a) (b)

Figure 4.21: (a) Truncated Cone. Accurate computation selects only the pink region from the yellow Voronoi facet as part of unstable manifold of an index 1 saddle point c (not shown). (b) Snapshot of approximate computation of $U(c)$ at a generic stage.

To compute $U(c)$ accurately, one therefore needs to compute the intersection of a ray and a line segment in three dimension. Such computations are prone to numerical errors. Therefore, we rely on an approximation algorithm that computes a superset of $U(c)$. The algorithm works as follows.

Starting from the Voronoi facet F containing c, we maintain a list of Voronoi facets which are already in $U(c)$ and a list of active Voronoi edges which are transversal edges and lie on the boundary of the current approximation of $U(c)$. Through these transversal edges we collect their acceptor facets and grow $U(c)$. Instead of computing the new set of active edges by an expensive numerical calculation of ray-segment intersection, we collect all the transversal edges of this new acceptor Voronoi facets. This way we grow $U(c)$ recursively till we have a set of Voronoi facets which are bounded by only a set of transversal Voronoi edges.

Figure 4.21(b) illustrates an intermediate stage of this computation. The index 1 saddle point c is contained in the blue Voronoi facet. The yellow Voronoi facets are already in $U(c)$. The red edges designate the static boundary as they are non-transversal and the green edges designate the active boundary through which the pink facets are included in $U(c)$ in the later stage of the algorithm. Following is the pseudo-code for this algorithm. Given an index 1 saddle point c it computes an approximation of $U(c)$. We assume c also has information about the Voronoi facet F it is contained in.

```
APPROX_UM_INDEX_1(c)
1  U = F
2  B = Voronoi edges of F
3  while (B ≠ ∅) do
4     e = pop(B)
5     if (e is transversal)
6        U = U ∪ acceptor(e)
7        B = B ∪ unvisited edges of acceptor(e)
8  endwhile
9  return U.
```

Figure 4.22: Pseudo-code for approximate computation of unstable manifold of an index 1 saddle point.

Classification of Medial Axis

In the previous two subsections we have described the structures of the unstable manifolds of an index 1 and index 2 saddle

points. We have also given an accurate and an approximate algorithm to compute them. Our goal is to identify the unstable manifolds near the medial axis of Σ. Ultimately these manifolds are mapped back to Σ for the feature annotation. For this we first compute a Voronoi subcomplex that approximates the medial axis M_Σ and then identify different regions of this approximate medial axis as the unstable manifolds computed by the two subroutines UM_INDEX_2 and APPROX_UM_INDEX_1.

Before we describe our approach, we briefly mention a recent result by Dey, Giesen, Ramos and Sadri [22] where they proved that under sufficient sampling of Σ by P, the critical points of h_P lie either close to Σ or close to M_Σ. This motivates our approach. Applying the same result, we filter out only the index 1 and index 2 saddle points near M_Σ instead of Σ. Further, we consider only the components of M_Σ which lie in the interior of the solid bounded by Σ. For this purpose we use the TIGHTCOCONE algorithm by Dey and Goswami [23]. The implementation of this algorithm is freely available in the public domain [16] along with the software for medial axis approximations which is computed as a Voronoi subcomplex according to the algorithm by Dey and Zhao [25]. For the purpose of reconstruction, any other reconstruction algorithm also could be used [8, 2]. Applying TIGHTCOCONE followed by medial axis approximation we get the approximate interior medial axis of Σ. We perform the critical point detection only within the Voronoi subcomplex that approximates this medial axis. Let us call this set of index 1 saddle points C_1 and that of index 2 saddle points C_2. We then apply UM_INDEX_2(c) for all $c \in C_2$ and APPROX_UM_INDEX_1(c) for all $c \in C_1$. $U(c \in C_1)$ is two dimensional and $U(c \in C_2)$ is one dimensional. Therefore, by restricting the unstable manifold computation only within M_Σ we obtain two subsets of M_Σ. In the next section, we describe how this classification can be mapped back to Σ for automatic identification of its flat and tubular regions.

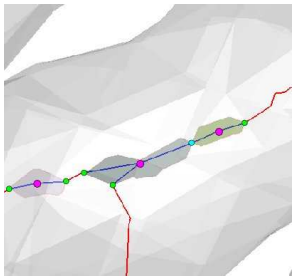


Figure 4.23: Removal of small patches in the tubular region via starring. Magenta circles indicate the centroids of these patches, green circles are the boundary vertices which connect a patch with a linear portion (red line) and cyan circle indicates where two different patches join at a common vertex. Blue lines are the replacements of these small patches obtained by the starring process.

Because of sampling artifacts, sometimes the interior medial axis in the tubular regions have a few index 1 saddle points. The unstable manifold of these saddle points need to be detected and approximated by lines. We partition the set C_1 based on the connectivity of their unstable manifolds via a common edge and every partition creates a patch which is the union of the unstable manifolds of all the index 1 saddle points falling into that partition. We further assign an *importance* value based on the area of the patch and sort the patches according to their *importance*. One could also employ other attributes like diameter, width etc. to evaluate the importance. The small clusters are then detected either by a user-specified threshold value or by simply selecting the k-smallest clusters where k is also a user-supplied parameter. These insignificant planar regions are then approximated by a set of straight lines emanating from the centroid of the patch to the boundary points which are connected to either a polyline (green circles in Figure 4.23) or another patch (cyan circle in Figure 4.23). We call this process *starring*.

The resulting one dimensional and two dimensional subsets of the interior medial axis is shown in Figure 4.24. Left column shows the approximate medial axis computed by [25]. The right column shows the subset of medial axis captured by $U(C_2)$ and $U(C_1)$.

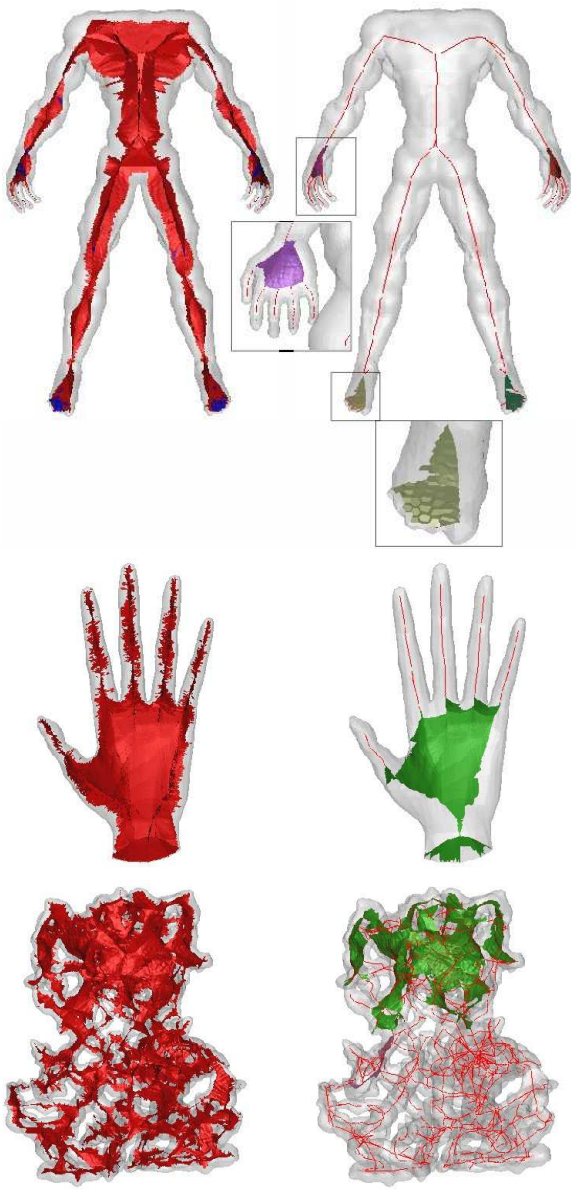


Figure 4.24: Results of Medial Axis classification. Top row shows the result for HEADLESS MAN. Two closeups have been shown to highlight the planar clusters in the palm of the hand and the feet. The closeup of hand has been rotated for visual clarity. The middle row shows the result on HAND dataset and the bottom row shows the result on a molecule data 1BVP.

Bibliography

[1] J. An, M. Totrov, and R. Abagyan. Pocketome via comprehensive identification and classification of ligand binding envelopes. *Mol. Cell Proteomics*, 4(6):752–761, 2005.

[2] C. Bajaj, F. Bernardini, and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3D scans. In *ACM SIGGRAPH*, pages 109–118, 1995.

[3] C. L. Bajaj, V. Pascucci, and D. R. Schikore. The contour spectrum. In *IEEE Visualization Conference*, pages 167–173, 1997.

[4] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions*

on Pattern Analysis and Machine Intelligence, 24(4):509–522, 2002.

[5] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Bourne. The Protein Data Bank. *Nucleic Acids Research*, pages 235–242, 2000.

[6] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic Acids Res*, 28(1):235–42, 2000. 0305-1048 Journal Article.

[7] J. Berman. Structural properties of acetylcholinesterase from eel electric tissue and bovine erythrocyte membranes. *Biochem*, 12:1710, 1973.

[8] F. Bernardini, C. Bajaj, J. Chen, and D. Schikore. Automatic reconstruction of 3d cad models from digital scans. *Int. J. on Comp. Geom. and Appl.*, 9(4-5):327–369, 1999.

[9] T. A. Binkowski, S. Naghibzadeh, and J. Liang. CASTp: Computed Atlas of Surface Topography of proteins. *Nucl. Acids Res.*, 31(13):3352–3355, 2003.

[10] G. P. Brady Jr. and P. F. Stouten. Fast prediction and visualization of protein binding pockets with pass. *Journal of Computer-Aided Molecular Design*, 14:383–401, 2000.

[11] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry: Theory and Applications*, 24(2):75–94, 2003.

[12] H. Carr, J. Snoeyink, and M. van de Panne. Simplifying flexible isosurfaces using local geometric measures. In *VIS '04: Proceedings of the conference on Visualization '04*, pages 497–504, Washington, DC, USA, 2004. IEEE Computer Society.

[13] CGAL Consortium. CGAL: Computational Geometry Algorithms Library.

[14] R. Chaine. A geometric convection approach of 3-d reconstruction. In *Proc. Eurographics Sympos. on Geometry Processing*, pages 218–229, 2003.

[15] F. Chazal and A. Lieutier. Stability and homotopy of a subset of the medial axis. In *Proc. 9th ACM Sympos. Solid Modeling and Applications*, pages 243–248, 2004.

[16] Cocone. Tight Cocone Software for surface reconstruction and medial axis approximation.

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

[18] J. S. Delaney. Finding and filling protein cavities using cellular logic operations. *J. Mol. Graph.*, 10(3):174–177, 1992. 159108.

[19] T. K. Dey, J. Giesen, and S. Goswami. Shape segmentation and matching with flow discretization. In F. Dehne, J.-R. Sack, and M. Smid, editors, *Proc. Workshop Algorithms Data Strucutres (WADS 03)*, LNCS 2748, pages 25–36, Berlin, Germany, 2003.

[20] T. K. Dey, J. Giesen, and S. Goswami. Shape segmentation and matching with flow discretization. In *Workshop on Algorithms and Data Structures*, 2003.

[21] T. K. Dey, J. Giesen, and M. John. Alpha-shapes and flow shapes are homotopy equivalent. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 493–502, 2003.

[22] T. K. Dey, J. Giesen, E. Ramos, and B. Sadri. Critical points of the distance to an epsilon-sampling of a surface and flow-complex-based surface reconstruction. In *Proc. 21st ACM-SIAM Sympos. Comput. Geom.*, pages 218–227, 2005.

[23] T. K. Dey and S. Goswami. Tight cocone: A water-tight surface reconstructor. In *Proc. 8th ACM Sympos. Solid Modeling and Applications*, pages 127–134, 2003.

[24] T. K. Dey and S. Goswami. Provable surface reconstruction from noisy samples. In *Proc. 20th ACM-SIAM Sympos. Comput. Geom.*, pages 330–339, 2004.

[25] T. K. Dey and W. Zhao. Approximating the Medial axis from the Voronoi diagram with convergence guarantee. *Algorithmica*, 38:179–200, 2004.

[26] H. Edelsbrunner. Surface reconstruction by wrapping finite point sets in space. In B. Aronov, S. Basu, J. Pach, and M. Sharir, editors, *Ricky Pollack and Eli Goodman Festschrift*, pages 379–404. Springer-Verlag, 2002.

[27] H. Edelsbrunner, M. Facello, and J. Liang. On the definition and the construction of pockets in macromolecules. *Discrete Applied Mathematics*, pages 83–102, 1998.

[28] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-smale complexes for piecewise linear 3-manifold. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 361 – 370, 2003.

[29] J. El-Sana and A. Varshney. Topology simplification for polygonal virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):133–144, 1998.

[30] P. F. Felzenszwalb and D. P. Huttenlocher. Distance transforms for sampled functions. Technical Report Technical Report TR2004-1963,, Cornell University, 2004.

[31] J. Giesen and M. John. The flow complex: a data structure for geometric modeling. In *Proc. 14th ACM-SIAM Sympos.*

Discrete Algorithms, pages 285–294, 2003.

[32] J. Giesen and M. John. The flow complex: a data structure for geometric modeling. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 285–294, 2003.

[33] J. Y. Gil and R. Kimmel. Efficient dilation, erosion, opening, and closing algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(12):1606–1617, 2002.

[34] F. Glaser, R. J. Morris, R. J. Najmanovich, R. A. Laskowski, and J. M. Thornton. A method for localizing ligand binding pockets in protein structures. *Proteins: Structure, Function, and Bioinformatics*, 62:479–488, 2006.

[35] M. Hendlich, F. Rippmann, and G. Barnickel. Ligsite: Automatic and efficient detection of potential small molecule-binding sites in proteins. *Journal of Molecular Graphics and Modelling*, 15:359–363, 1998.

[36] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *Siggraph 2001*, pages 203–212, Los Angeles, USA, 2001.

[37] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics*, 22(3):954–961, 2003.

[38] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. In *Trans. on Graphics*, volume 3, pages 954–961, 2003.

[39] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3d shape descriptors. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 156–164. Eurographics Association, 2003.

[40] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Shape matching and anisotropy. *ACM Trans. Graph.*, 23(3):623–629, 2004.

[41] R. Krishnapuram and S. Gupta. Morphological methods for detection and classification of edges in range images. *Journal of Mathematical Imaging and Vision*, 2(4):351–375, 1992.

[42] R. A. Laskowski. Surfnet: A program for visualizing molecular surfaces, cavities, and intermolecular interactions. *Journal of Molecular Graphics*, 13:323–330(8), October 1995.

[43] A. T. R. Laurie and R. M. Jackson. Q-sitefinder: an energy-based method for the prediction of protein ligand binding sites. *Bioinformatics*, 21:1908–1916, 2005.

[44] C. Lee and A. Varshney. *Computing and Displaying Intermolecular Negative Volume for Docking*. Springer Berlin Heidelberg, 2006.

[45] F. F. Leymarie and B. B. Kimia. The shock scaffold for representing 3d shape. In *Proceedings of the 4th International Workshop on Visual Form*, Lecture Notes In Computer Science, pages 216 – 228. Springer-Verlag, 2001.

[46] X. Li, T. W. Woon, T. S. Tan, and Z. Huang. Decomposing polygon meshes for interactive applications. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 35–42, 2001.

[47] J. Liang, H. Edelsbrunner, and C. Woodward. Anatomy of protein pockets and cavities: measurement of binding site geometry and implications for ligand design. *Protein Sci*, 7(9):1884–97, 1998.

[48] A. P. Mangan and R. T. Whitaker. Partitioning 3d surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, 1999.

[49] M. Masuya and J. Doi. Detection and geometric modelling of molecular surfaces and cavities using digital mathematical morphological operations. *Journal of Molecular Graphics*, 13:331–336, 1995.

[50] M. Mortara, G. Patanè, M. Spagnuolo, B. Falcidieno, and J. Rossignac. Blowing bubbles for the multi-scale analysis and decomposition of triangle meshes. *Algorithmica*, 38:227–248, 2004.

[51] M. Mortara, G. Patanè, M. Spagnuolo, B. Falcidieno, and J. Rossignac. Plumber: a multi-scale decomposition of 3d shapes into tubular primitives and bodies. In *Proc. 9th ACM Sympos. Solid Modeling and Applications*, pages 139–158, 2004.

[52] F. S. Nooruddin and G. Turk. Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):191–205, 2003.

[53] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Matching 3d models with shape distributions. In *Proceedings of the International Conference on Shape Modeling & Applications*, page 154. IEEE Computer Society, 2001.

[54] H. Pottmann, M. Hofer, B. Odehnal, and J. Wallner. Line geometry for 3d shape understanding and reconstruction. *Computer Vision - ECCV*, 3021(1):297–309, 2004.

[55] T. B. Sebastian, P. N. Klein, and B. B. Kimia. Recognition of shapes by editing shock graphs. In *International Conference on Computer Vision*, pages 755–762, 2001.

[56] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *PNAS*, 93(4):1591–1595, 1996.

[57] Y. Shinagawa, T. Kunni, A. Belayev, and T. Tsukioka. Shape modeling and shape analysis based on singularities. *Internat.*

J. Shape Modeling, 2:85–102, 1996.

[58] K. Siddiqi, A. Shokoufandeh, J. Dickinson, and S. Zucker. Shock graphs and shape matching. *Computer Vision*, pages 222–229, 1998.

[59] C. Sigg, R. Peikert, and M. Gross. Signed distance transform using graphics hardware. In *IEEE Vis2003*, 2003.

[60] R. Sonthi, G. Kunjur, and R. Gadh. Shape feature determination usiang the curvature region representation. In *Proceedings of the fourth ACM symposium on Solid modeling and applications*, pages 285–296. ACM Press, 1997.

[61] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. In *Shape Modelling and Applications Conference*, May 2003.

[62] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proceedings of the 1998 IEEE International Conference on Computer Vision*, pages 839–846, New Delhi, India, 1998.

[63] N. Unwin. Refined structure of the nicotinic acetylcholine receptor at 4 a resolution. *J. Mol. Biol.*, 346:967–989, 2005.

[64] T. Várady, R. Martin, and J. Cox. Reverse engineering of geometric models - an introduction. *Computer Aided Design*, 29:255–268, 1997.

[65] A. Verroust and F. Lazarus. Extracting skeletal curves from 3d scattered data. *The Visual Computer*, 16:15–25, 2000.

[66] J. Wu and L. Kobbelt. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum*, 24(3):277–284, 2005.

[67] X. Zhang. Complementary shape comparison with additional properties. In *Proceedings of Eurographics/IEEE volume graphics 2006*, pages 79–86, 2006.

[68] X. Zhang, C. Bajaj, and N. Baker. Affine invariant comparison of molecular shapes with properties. Technical report, University of Texas at Austin, 2005.

[69] X. Zhang, C. L. Bajaj, B. Kwon, T. J. Dolinsky, J. E. Nielsen, and N. A. Baker. Application of new multi-resolution methods for the comparison of biomolecular electrostatic properties in the absence of global structural similarity. *SIAM Multiscale Modeling and Simulation*, 5:1196–1213, 2006.

Chapter 5

Integral and Differential Properties

5.1 Areas, Volumes

5.2 Gradients, Curvatures

Related Work

Relevant Mathematics

Multivariate Calculus

Differential Geometry

Weingarten Map

Chapter 6

Energetics

In [6], the Gibbs free energy of a system of an assemble of solutes with arbitrary shape and composition surrounded by a dielectric solvent is expressed as

$$G = pV + \int \gamma(\mathbf{r}) \|\nabla \mathscr{H}(\mathbf{r})\| \, \mathrm{d}\mathbf{r} + \int \rho(\mathbf{r}) U(\mathbf{r}) \, \mathrm{d}\mathbf{r} + \frac{\varepsilon_0}{2} \int \|\nabla \phi(\mathbf{r})\|^2 \varepsilon(\varepsilon) \, \mathrm{d}\mathbf{r}. \tag{0.1}$$

The first term is energy of creating a cavity in the solvent against the difference in bulk pressure between the liquid and vapor. The second term is the energetic cost due to solvent arrangement close to the cavity surface. This interfacial energy penalty is thought to be the main driving force for hydrophobic phenomena [3]. They assume γ is a function of the local mean curvature of the cavity interface. The third term is the total non-electrostatic solute-solvent interaction. It is represented as an isotropic Lennard-Jones potential. The fourth term is the total electrostatic energy. The electrostatic potential ϕ is evaluated by Poisson's equation.

In [20], they rewrite the Poinsson-Boltzmann (PB) equation into the form of the Euler-Lagrange equation the solution of which minimizes the total electrostatic free energy of the system:

$$G^{\mathrm{ele}} = \int \left(\rho^f \phi - kTc^b [2\cosh(\phi) - 2] - \frac{1}{2} \mathbf{E} \cdot \mathbf{D} \right) \mathrm{d}\mathbf{r}, \tag{0.2}$$

where ρ^f is the charge density of the fixed charges (of the proteins or other macromolecules) and c^b is the bulk salt concentration. When $\phi \ll 1$, the PB equation can be linearized, so the electrostatic free energy can be written as

$$G^{\mathrm{ele}} = \int \left(\rho^f \phi - \frac{\varepsilon \kappa^2}{8\pi} \phi^2 - \frac{1}{2} \mathbf{E} \cdot \mathbf{D} \right) \mathrm{d}\mathbf{r}. \tag{0.3}$$

In [18], they write the total potential of mean force $G(\mathbf{X})$ for the configuration $\mathbf{X}$ as

$$G(\mathbf{X}) = U_u(\mathbf{X}) + \Delta G^{(np)}(\mathbf{X}) + \Delta G^{(elec)}(\mathbf{X}), \tag{0.4}$$

where $U_u(\mathbf{X})$ is the intramolecular solute potential. The representation of the non-polar solvation contribution $\Delta G^{(np)}(\mathbf{X})$ can be obtained from the scaled particle theory (SPT). To produce a cavity in a liquid,

$$\Delta G^{(np)}(\mathbf{X}) = pV + \gamma A, \tag{0.5}$$

where p is the isotropic pressure, V is the volume of the cavity, γ is a function of the surface tension of the solvent and the curvature of the interface, and A is the area of the interface. The continuum electrostatic contribution can be expressed as a surface integral

$$\Delta G^{(elec)}(\mathbf{X}) = \frac{1}{2} \int_\Gamma \sum_i q_i \frac{\sigma(\mathbf{r})}{|\mathbf{r} - \mathbf{x}_i|} \, \mathrm{d}\mathbf{r}, \tag{0.6}$$

where $\sigma(\mathbf{r})$ is the surface charge density. This is because the solvent charge density is a sharply peaked function localized at the solute-solvent interface. One can further rewrite (0.6) as

$$\begin{aligned}\Delta G^{(elec)}(\mathbf{X}) &= \frac{1}{2}\int_{\Gamma}\sum_{i,j} q_i q_j \frac{\sigma(\mathbf{r}:\mathbf{x}_j)}{|\mathbf{r}-\mathbf{x}_i|}\,\mathrm{d}\mathbf{r} \\ &= \frac{1}{2}\sum_{i,j} q_i q_j F(\mathbf{x}_i,\mathbf{x}_j),\end{aligned} \tag{0.7}$$

where $\sigma(\mathbf{r}:\mathbf{x}_j)$ is the surface charge induced at $\mathbf{r}$ by a unit solute charge located at $\mathbf{x}_j$. The geometry-dependent coupling function $F(\mathbf{x}_i,\mathbf{x}_j)$ has different approximations, for example FIESTA [22], IMS [5], ACE [19], and GB [23].

In the Weeks-Chandler-Anderson (WCA) model [26], the intermolecular potential is separated into two parts:

$$w(r) = u_0(r) + u(r), \tag{0.8}$$

where $u_0(r)$ is the reference system pair potential which includes all the repulsive forces in the Lennard-Jones potential and $u(r)$ is the perturbation potential which includes all the attractive forces. So one can also write $w(r) = U_{rep}(r) + U_{att}(r)$. Because the non-polar free energy can be decomposed into the repulsive cavity hydration free energy and the solute-solvent van der Waals dispersion interaction [16],

$$\Delta G^{(np)}(\mathbf{X}) = \Delta G_{cav} + \Delta G_{vdW}, \tag{0.9}$$

in [28] they use the WCA decomposition scheme to represent ΔG_{cav} and ΔG_{vdW} by the repulsive potential $U_{rep}(r)$ and attractive potential $U_{att}(r)$, respectively. Explicit solvent simulation has shown that ΔG_{cav} is approximately proportional to the solvent accessible area. $U_{att}(r)$ is obtained by summing the average van der Waals solute-solvent energy U^i_{att} of each atom

$$U_{att} = \sum_{i=1}^{M} U^i_{att}, \tag{0.10}$$

where

$$U^i_{att} = \int_{\text{solvent}} \bar{\rho}\, u^i_{att}(|\mathbf{r}-\mathbf{x}_i|)\,\mathrm{d}\mathbf{r}. \tag{0.11}$$

$\bar{\rho}$ is the bulk density of the solvent, $u^i_{att} = -\varepsilon_i\left(\frac{\sigma_i+\sigma_s}{|\mathbf{r}-\mathbf{x}_i|}\right)^6$, ε_i measures the depth of the attractive well at $|\mathbf{r}-\mathbf{x}_i| = \sigma_i+\sigma_s$, σ_i and σ_s are the radii of the solute atom and the solvent probe. The integration domain in (0.11) can be converted to a bounded domain

$$U^i_{att} = U^i_{att}(\text{isolated}) - \bar{\rho}\int_{\text{solute}\setminus\text{atom}_i} u^i_{att}(|\mathbf{r}-\mathbf{x}_i|)\,\mathrm{d}\mathbf{r}, \tag{0.12}$$

where first term is the solute-solvent attractive van der Waals energy when the solute is composed solely of atom i. This can be analytically obtained by integrating (0.11) for the single atom solute model.

6.1 Hamiltonian, Lagrangian

6.2 Partial Charge Assignment

6.3 Lennard Jones Potential (vander Waals)

The Lennard-Jones (LJ) potential between molecules A and B is given by the following expression.

$$LJ(A,B) = \sum_{i\in A, j\in B} lj(i,j), \quad lj(i,j) = a_{ij}/r_{ij}^{12} - b_{ij}/r_{ij}^{6},$$

where r_{ij} is the distance between atoms $i \in A$ and $j \in B$, constants a_{ij} and b_{ij} depend on the type (e.g., C, H, O, etc.) of the two atoms involved. One can evaluate the LJ potential among the atoms of a single molecule A by setting $B = A$ and considering only non-bonded atom pairs.

Observe that direct computation of $LJ(A,B)$ requires $\mathscr{O}(M_A M_B)$ time, where M_A (resp. M_B) is the number of atoms in molecule A (resp. B). However, since the terms in the summation diminish quickly with the increase of r_{ij}, one can evaluate $LJ_{\delta^-}(A,B) = \sum_{i\in A,\, j\in B, r_{ij}\le\delta} lj(i,j)$ as an approximation of $LJ(A,B)$, where δ is a given distance cutoff.

Suppose $M_A > M_B$. Then one can evaluate $LJ_{\delta^-}(A,B)$ in $\mathscr{O}((M_A+M_B)\log M_B + m)$ time and $\mathscr{O}(M_A+M_B)$ space, where m is the total number of $\langle i\in A, j\in B\rangle$ pairs within distance δ. The trick is to use an octree [12] to store the atoms of B, and then use it to locate the atoms of B within distance δ from each atom of A (see [4] for details). Use of a 3D grid instead of an octree may result in $\Theta\left({M_B}^3\right)$ space usage in the worst case [4].

In the rest of this section we first outline our approach to evaluating $LJ_{\delta^-}(A,B)$ faster than $\mathscr{O}((M_A+M_B)\log M_B + m)$ time while still using $\mathscr{O}(M_A+M_B)$ space, followed by our approach to fast approximation of $LJ(A,B)$ to within a factor of $1+\varepsilon$ of the exact value for any given $\varepsilon > 0$.

6.3.1 Faster Evaluation of $LJ(A,B)$ with a Distance Cutoff

We store the atoms of B in our *Dynamic Packing Grid* (DPG) [1] data structure instead of an octree. The DPG can maintain the atoms of a molecule in space linear in the number of atoms, while allowing a range of spherical range queries and updates (i.e., insertion/deletion) very efficiently. An update takes $\mathscr{O}(1)$ time (w.h.p.[1]), while a range query returns all atoms within a given distance δ from any given atom center in $\mathscr{O}(k)$ time[2] (w.h.p.), where k is the number atoms returned. Therefore, all atoms of B can be inserted into the DPG in $\mathscr{O}(M_B)$ time (w.h.p.), and the total time required to find all atoms of B within distance δ of the atoms of A is $\mathscr{O}(M_A+m)$. Hence, $LJ_{\delta^-}(A,B)$ can be evaluated exactly in $\mathscr{O}(M_A+M_B+m)$ time (w.h.p.) and $\mathscr{O}(M_A+M_B)$ space.

6.3.2 Fast $(1+\varepsilon)$-Approximation of $LJ(A,B)$

Observe that $LJ(A,B) = LJ_{\delta^-}(A,B) + LJ_{\delta^+}(A,B)$, where $LJ_{\delta^+}(A,B) = \sum_{i\in A,\, j\in B, r_{ij}>\delta} lj(i,j)$. We outline below how to obtain an error-bounded approximation of $LJ(A,B)$ through a fast approximation of $LJ_{\delta^+}(A,B)$ in addition to the exact evaluation of $LJ_{\delta^-}(A,B)$. More precisely, given any user-defined constant $\varepsilon > 0$, we will approximate $LJ(A,B)$ to within a $(1+\varepsilon)$ factor of its exact value.

In the expression of $LJ(A,B)$, a_{ij} and b_{ij} are fixed for any fixed pair of atom types, and can be calculated from the Amber force field using well depths μ_{XY} and equivalence contact distances of homogeneous pairs $r_{eqm,XY}$, where $X = atomType(i\in A)$ and $Y = atomType(j\in B)$) [27, 14]. By definition, $a_{ij}/b_{ij} = r^6_{eqm,XY}/2$ (see [14]). We assume $X,Y \in \{\text{C, H, N, O, P, S}\}$.

Let $\mathscr{M}_X$ denote the subset of atoms of type X in molecule $\mathscr{M}\in\{A,B\}$. Then $LJ(A,B) = \sum_{X,Y\in\{\text{C, H, N, O, P, S}\}} LJ(A_X,B_Y)$, where, $LJ(A_X,B_Y) = LJ_{\delta^-_{XY}}(A_X,B_Y) + LJ_{\delta^+_{XY}}(A_X,B_Y)$, for some constant $\delta_{XY}\ge 0$ (to be defined later). We outline below how to approximate $LJ(A_X,B_Y)$ for a given pair $\langle X,Y\rangle$. We evaluate $LJ_{\delta^-_{XY}}(A_X,B_Y)$ exactly, and approximate $LJ_{\delta^+_{XY}}(A_X,B_Y)$ to within a factor of $(1+\varepsilon)$ of its exact value.

Let $\delta_{XY} \ge (1/2+1/\varepsilon)^{1/6} r_{eqm,XY}$. If we approximate each b_{ij}/r^6_{ij} with $r_{ij} > \delta_{XY}$ to within a factor of $1+\varepsilon/(2+\varepsilon)$, simple algebraic manipulations show $|lj(i,j)| < \left[b_{ij}/r^6_{ij}\right]_{approx} < (1+\varepsilon)|lj(i,j)|$.

In order to approximate $LJ(A_X,B_Y)$ as mentioned above, we construct two octrees $\mathscr{T}_{A_X}$ and $\mathscr{T}_{B_Y}$ from the atoms in A_X and B_Y, respectively, and compute a $(1+\varepsilon)$-approximation of $LJ(A_X,B_Y)$ by simultaneous recursive traversals of $\mathscr{T}_{A_X}$ and $\mathscr{T}_{B_Y}$ starting from their root nodes. Suppose at some point we are at node x of $\mathscr{T}_{A_X}$ and node y of $\mathscr{T}_{B_Y}$. If both x and y are leaf nodes, potential between the atoms contained in x (say, $\mathscr{M}_x$) and y (say, $\mathscr{M}_y$) is computed exactly. Otherwise if x and y are far enough (i.e., at least δ_{XY} apart), and small enough[3] the potential between $\mathscr{M}_x$ and $\mathscr{M}_y$ is approximated by assuming that x and y are single pseudo atoms centered at the center of gravity of $\mathscr{M}_x$ and $\mathscr{M}_y$, respectively, and taking $|\mathscr{M}_x||\mathscr{M}_y|\left(b_{ij}/r^6_{xy}\right)$ as

[1] For an input of size n, an event E occurs w.h.p. (with high probability) if, for any $\alpha \ge 1$ and c independent of n, $Pr(E) \le 1 - \frac{c}{n^\alpha}$.

[2] The actual complexities also depend on $\log w$ and $\log\log w$, respectively, where w is the RAM word size (e.g., 32 or 64) of the machine, which is a constant for a given machine.

[3] i.e., $r_{x,y} + (r_x + r_y) < (1+\varepsilon/(2+\varepsilon))^{\frac{1}{6}}(r_{x,y} - (r_x+r_y))$, where, r_x (resp. r_y) is the radius of the smallest ball centered at the atom centers of x (resp. y) that encloses all atom centers of x (resp. y).

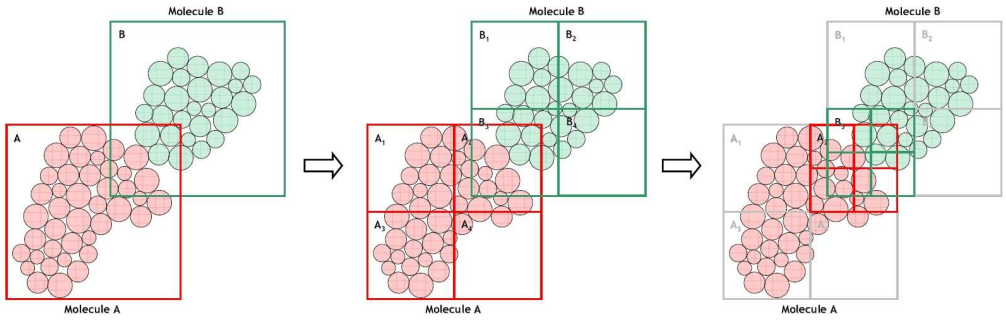


Figure 6.1: Approximation of LJ potential in 2D using quadtrees [9] (i.e., 2D variant of octrees): In the leftmost figure the bounding box of molecule A (resp. B) represents the root node of the quadtree storing A (resp. B). The smallest boxes in the middle and the rightmost figures represent quadtree nodes at levels 2 (i.e., children of the root) and 3, respectively. Let us assume for simplicity that if two nodes of the two quadtrees do not intersect they are far enough so that the LJ potential between their atoms can be approximated by treating them as pseudo-atoms. In the leftmost figure the two root nodes (nodes A and B) intersect, and so we move to their children nodes in the middle figure. In the middle figure only nodes A_2 and B_3 intersect, and so while the potential between the atoms of all other $\langle A_i, B_j \rangle$ pairs can be approximated, we need to move to the children of A_2 and B_3 in order to compute the potential between them (see the rightmost figure).

the approximated potential, where r_{xy} is the distance between the centers of the two pseudo atoms. Otherwise we subdivide x and/or y (i.e., move to their children), and approximate the potential recursively. Figure 6.1 explains the approach in 2D. The pseudocode is given in Figure 6.2.

APPROXLJ(x, y)

(Inputs are two octree nodes $x \in \mathscr{T}_{A_X}$ and $y \in \mathscr{T}_{B_Y}$, and the the output is a floating point number V such that $U \leq V \leq (1+\varepsilon)\cdot U$, where $U = \sum_{i \in \mathscr{M}_x \wedge j \in \mathscr{M}_y} \left(a_{ij}/r_{ij}^{12} - b_{ij}/r_{ij}^{6}\right)$. By CHILD$(x)$ (resp. CHILD(y)) we denote the set of non-empty octree nodes obtained by subdividing node x (resp. y). We denote by b_{XY} the value of the constant b_{ij} for atom types X and Y, and by $r_{x,y}$ the distance between the centers of x and y.)

1. ***if*** LEAF$(x) \wedge$ LEAF(y) ***then return*** $\sum_{i \in \mathscr{M}_x \wedge j \in \mathscr{M}_y} \left(\frac{a_{ij}}{r_{ij}^{12}} - \frac{b_{ij}}{r_{ij}^{6}}\right)$ {*exact value*}
2. ***else if*** $r_{x,y} - (r_x + r_y) > \delta_{XY} \;\wedge\; \frac{r_{x,y}+(r_x+r_y)}{r_{x,y}-(r_x+r_y)} < \left(1+\frac{\varepsilon}{2+\varepsilon}\right)^{\frac{1}{6}}$ ***then return*** $-\frac{M_X \cdot M_Y \cdot b_{XY}}{\left(r_{x,y}-(r_x+r_y)\right)^6}$ {*approximation*}
3. ***else if*** LEAF(x) ***return*** $\sum_{cy \in \text{CHILD}(y)}$ APPROXLJ(x, cy) {*recursive approximation*}
4. ***else if*** LEAF(y) ***return*** $\sum_{cx \in \text{CHILD}(x)}$ APPROXLJ(cx, y) {*recursive approximation*}
5. ***else return*** $\sum_{cx \in \text{CHILD}(x) \wedge cy \in \text{CHILD}(y)}$ APPROXLJ(cx, cy) {*recursive approximation*}

APPROXLJ ENDS

Figure 6.2: Recursive approximation of $\sum_{i \in \mathscr{M}_x \wedge j \in \mathscr{M}_y} \left(a_{ij}/r_{ij}^{12} - b_{ij}/r_{ij}^{6}\right)$ to within a factor of $1+\varepsilon$. The initial call is APPROXLJ(ROOT$(\mathscr{T}_{A_X})$, ROOT$(\mathscr{T}_{B_Y})$) for the approximation of $\sum_{i \in A_X \wedge j \in B_Y} \left(a_{ij}/r_{ij}^{12} - b_{ij}/r_{ij}^{6}\right)$.

In order to obtain an upper bound on the time required for approximating $LJ(A_X, B_Y)$ we assume that the initial bounding box of both A_X and B_Y have exactly the same size, and each non-root node of the two octrees has at least one sibling[4]. Then it can be shown that each node $x \in \mathscr{T}_{A_X}$ will be paired with $\mathscr{O}\left(\frac{1}{\varepsilon^3}\right)$ nodes of $\mathscr{T}_{B_Y}$ of the same or larger size during recursive calls, and vice versa. In order to see that this is indeed the case, suppose the diameter of the smallest ball containing node x is d (i.e., equal to the length of the longest diagonal of x.). Then a node $y \in \mathscr{T}_{B_Y}$ of the same size will be paired with x for approximating the potential by treating both nodes as pseudo atoms provided the (center-to-center) distance $r_{x,y}$ between them is in the range $[\varepsilon' d,\ 2\varepsilon' d)$, where $\varepsilon' = (\varepsilon''+1)/(\varepsilon''-1)$, and $\varepsilon'' = (1+\varepsilon/(2+\varepsilon))^{1/6}$. Since the volume of node y is $\Theta\left(d^3\right)$, there can be

[4] if not, we directly connect the node to its nearest ancestor that has at least two children.

$\mathscr{O}\left(\frac{1}{\varepsilon^3}\right)$ such nodes within that range. Also since each internal node of $\mathscr{T}_{B_Y}$ is assumed to have at least two children, the number of nodes of $\mathscr{T}_{B_Y}$ larger than y with which x can be paired as pseudo atom is also bounded by $\mathscr{O}\left(\frac{1}{\varepsilon^3}\right)$. Observing that there are $\mathscr{O}(|A_X|)$ (resp. $\mathscr{O}(|B_Y|)$) nodes in $\mathscr{T}_{A_X}$ (resp. $\mathscr{T}_{B_Y}$), and taking the construction times of the octrees into account, the total running time of the algorithm for atom-type pair $\langle X,Y\rangle$ is $\mathscr{O}\left(|A_X|\log|A_X|+|B_Y|\log|B_Y|+\frac{1}{\varepsilon^3}(|A_X|+|B_Y|)\right)$. Summing over all possible pairs of atom types, the total running time for approximating $LJ(A,B)$ is $\mathscr{O}\left(\left(\frac{1}{\varepsilon^3}+\log(M_A+M_B)\right)(M_A+M_B)\right)$. However, assuming that the octrees are already constructed in a preprocessing step[5], the running time of the algorithm is only $\mathscr{O}\left(\frac{1}{\varepsilon^3}(M_A+M_B)\right)$.

6.4 Coulombic

Long range Coulomb potential plays a role in forming stable complexes due to partially charged bio-molecules and solvent atoms, and is given by $\mathscr{Q}=\sum_{i,j}\frac{q_iq_j}{\varepsilon(r_{ij})r_{ij}}$. Assuming $\varepsilon(r_{ij})=r_{ij}$, $\mathscr{Q}$ is also approximated as $\sum_{i,j}q_iq_j/r_{ij}^2$, where pairwise interactions fall off more sharply with distance.

We can obtain an approximation $\mathscr{Q}'$ of $\mathscr{Q}$ using an algorithm similar to the $1+\varepsilon$ approximation algorithm in Section 6.3. Since contributions due to positive and negative pairwise potentials tend to cancel out the algorithm does not guarantee a multiplicative (i.e., $1+\varepsilon$) error bound. Instead the error bound can be obtained as follows. Suppose $\mathscr{Q}=\mathscr{Q}_{\mathscr{P}}-\mathscr{Q}_{\mathscr{N}}$, where $\mathscr{Q}_{\mathscr{P}}$ (resp. $\mathscr{Q}_{\mathscr{N}}$) is the sum of all positive (resp. negative) pairwise potentials in $\mathscr{Q}$. Now if $\mathscr{Q}>0$, then simple algebraic manipulations show that $\mathscr{Q}-\varepsilon\mathscr{Q}_{\mathscr{P}}\leq\mathscr{Q}'\leq\mathscr{Q}+\varepsilon\mathscr{Q}_{\mathscr{P}}$. Similarly, if $\mathscr{Q}<0$ the bound is $\mathscr{Q}-\varepsilon\mathscr{Q}_{\mathscr{N}}\leq\mathscr{Q}'\leq\mathscr{Q}+\varepsilon\mathscr{Q}_{\mathscr{N}}$. We cannot guarantee an error bound if $\mathscr{Q}=0$.

6.5 Dispersion

The solute-solvent van der Waals interaction energy (also known as *dispersion energy*) is modeled as [7, 25]: $E_{\text{vdw(s-s)}}=\rho_0\sum_{i=1}^{M}\int_{\text{ex}}u_i^{(\text{att})}(\mathbf{x}_i,\mathbf{r})d^3\mathbf{r}$, where ρ_0 is the bulk density, and $u_i^{(\text{att})}(\mathbf{x}_i,\mathbf{r})=\frac{1}{|\mathbf{r}-\mathbf{x}_i|^6}$ is the van der Waals dispersive component of the interaction between atom $i\in[1,M]$ and the solvent. Thus $E_{\text{vdw(s-s)}}=\rho_0\sum_{i=1}^{M}\int_{\text{ex}}\frac{1}{|\mathbf{r}-\mathbf{x}_i|^6}d^3\mathbf{r}$.

The following discrete surface formulation of the equation above is obtained by applying the divergence theorem and Gaussian quadrature: $E_{\text{vdw(s-s)}}\approx\frac{\rho_0}{3}\sum_{i=1}^{M}\sum_{k=1}^{m}w_k\frac{(\mathbf{r}_k-\mathbf{x}_i)\cdot\mathbf{n}_k}{|\mathbf{r}_k-\mathbf{x}_i|^6}$. If R_i is the Born radius of atom i calculated using the r^6-approximation, then $E_{\text{vdw(s-s)}}\approx\rho_0\frac{4\pi}{3}\sum_{i=1}^{M}\frac{1}{R_i^3}$.

Therefore, $E_{\text{vdw(s-s)}}$ can be approximated in $\mathscr{O}\left(M\log M+M/\varepsilon^3\right)$ time and $\mathscr{O}(M)$ space using $m=\mathscr{O}(M)$ quadrature points and the technique described in Section 6.6.2 for the simultaneous approximation of Born radius of all atoms in a molecule, where $\varepsilon>0$ is the approximation parameter used for Born radius approximation. In fact, $E_{\text{vdw(s-s)}}$ can be approximated slightly (a constant factor) faster than approximating all Born radii since we do not need to approximate the Born radius integral for each atom and instead we can simply compute the sum of those integrals. The simplified pseudocode is given in Figure 6.3.

The algorithm runs in $\mathscr{O}\left(M/\varepsilon^3\right)$ time if the octrees are already available.

NFFT based Fast Summation. The inner summation in the discrete surface formation of $E_{\text{vdw(s-s)}}$ can be written as: $\sum_{k=1}^{m}w_k\frac{(\mathbf{r}_k-\mathbf{x}_i)\cdot\mathbf{n}_k}{|\mathbf{r}_k-\mathbf{x}_i|^6}=\sum_{k=1}^{m}\frac{w_k\mathbf{r}_k\cdot\mathbf{n}_k}{|\mathbf{r}_k-\mathbf{x}_i|^6}-x_i\sum_{k=1}^{m}\frac{w_kn_k^x}{|\mathbf{r}_k-\mathbf{x}_i|^6}-y_i\sum_{k=1}^{m}\frac{w_kn_k^y}{|\mathbf{r}_k-\mathbf{x}_i|^6}-z_i\sum_{k=1}^{m}\frac{w_kn_k^z}{|\mathbf{r}_k-\mathbf{x}_i|^6}$, where, $\mathbf{x}_i=\langle x_i,y_i,z_i\rangle$ and $\mathbf{n}_k=\langle n_k^x,n_k^y,n_k^z\rangle$. The summations on the right hand side are of the common form $G(\mathbf{x}_i)=\sum_{k=1}^{m}c_kg(\mathbf{x}_i-\mathbf{r}_k)$, $i\in[1,M]$, with $g(\mathbf{x}_i-\mathbf{r}_k)=1/|\mathbf{x}_i-\mathbf{r}_k|^6$ and coefficients $c_k=w_k\mathbf{r}_k\cdot\mathbf{n}_k$, $w_kn_k^x$, $w_kn_k^y$ and $w_kn_k^z$ for the 1st, 2nd, 3rd and 4th summation, respectively. All such $G(\mathbf{x}_i)$ for $i\in[1,M]$ can be

[5] e.g., in rigid-body pairwise docking, where LJ potential is computed for numerious relative translations and orientations of the same two input molecules, the octrees can be constructed only once, and the locations of the octree nodes can be transformed on-the-fly during potential approximation based on the relative transformation of the two molecules.

APPROX-DISPE(A, Q)

(For the given node A in the atoms octree and node Q in the integration/quadrature points octree approximate $\frac{\rho_0}{3}\sum_{a\in A}\sum_{q\in Q} w_q \frac{(\mathbf{p}_q-\mathbf{p}_a)\cdot\mathbf{n}_q}{|\mathbf{p}_q-\mathbf{p}_a|^6}$. By $\mathbf{p}_a = \langle x_a, y_a, z_a\rangle$ we denote the center of an atom a, while by $\mathbf{p}_q = \langle x_q, y_q, z_q\rangle$, w_q and $\mathbf{n}_q = \langle nx_q, ny_q, nz_q\rangle$ we denote the location of a quadrature point q, weight assigned to q, and the unit outward normal on the molecular surface at q, respectively. By $\langle x_A, y_A, z_A\rangle$ (resp. $\langle x_Q, y_Q, z_Q\rangle$) we denote the geometric center of the atoms (resp. integration points) under A (resp. Q). By r_A (resp. r_Q) we denote the radius of the smallest ball centered at $\langle x_A, y_A, z_A\rangle$ (resp. $\langle x_Q, y_Q, z_Q\rangle$) that encloses all atom centers (resp. integration points) under A (resp. Q). The distance between the geometric centers of A and Q is given by $r_{A,Q}$. We also assume $\widetilde{nx}_Q = \sum_{q\in Q} w_q nx_q$. Similarly for $\widetilde{ny}_Q$ and $\widetilde{nz}_Q$. By CHILD(A) (resp. CHILD(Q)) we denote the set of non-empty octree nodes obtained by subdividing node A (resp. Q), and M_A (resp. m_Q) denotes the number of atoms (resp. integration points) under node A (resp. Q).)

1. ***if*** $r_{A,Q} - (r_A + r_Q) > 0 \ \wedge\ \frac{r_{A,Q}+(r_A+r_Q)}{r_{A,Q}-(r_A+r_Q)} > (1+\varepsilon)^{\frac{1}{6}}$ ***then*** {*far enough to approximate*}

   return $\frac{\rho_0}{3}\frac{M_A m_Q\left(\widetilde{nx}_Q\cdot(x_A-x_Q)+\widetilde{ny}_Q\cdot(y_A-y_Q)+\widetilde{nz}_Q\cdot(z_A-z_Q)\right)}{(r_{A,Q})^6}$

2. ***else if*** LEAF$(A)\wedge$LEAF(Q) ***then*** {*too close to approximate; compute exact value*}

   return $\frac{\rho_0}{3}\sum_{a\in A}\sum_{q\in Q}\frac{w_q\left(nx_q\cdot(x_a-x_q)+ny_q\cdot(y_a-y_q)+nz_q\cdot(z_a-z_q)\right)}{(r_{a,q})^6}$

3. ***else if*** LEAF(A) ***then return*** $\sum_{Q'\in\text{CHILD(Q)}}$ APPROX-DISPE(A, Q') {*recurse on Q*}

4. ***else if*** LEAF(Q) ***then return*** $\sum_{A'\in\text{CHILD(A)}}$ APPROX-DISPE(A', Q) {*recurse on A*}

5. ***else return*** $\sum_{A'\in\text{CHILD(A)}}\sum_{Q'\in\text{CHILD(Q)}}$ APPROX-DISPE(A', Q') {*recurse on A and Q*}

APPROX-DISPE ENDS

Figure 6.3: Octree-based algorithm for approximating the dispersion energy. Given the atoms octree $\mathscr{T}_{\mathscr{A}}$ and quadrature/integration points octree $\mathscr{T}_{\mathscr{Q}}$, the dispersion energy can be approximated (controlled by a given approximation parameter $\varepsilon > 0$) by making calling APPROX-DISPE(ROOT$(\mathscr{T}_{\mathscr{A}})$, ROOT$(\mathscr{T}_{\mathscr{Q}})$).

simultaneously approximated using the NFFT based fast summation technique [15] in $\mathscr{O}\left(M+m+n^3\log n\right)$ time , where n^3 is the size of the NFFT grid. Hence, $E_{\text{vdw(s-s)}}$ can also be approximated within the same asymptotic time bound.

6.6 Generalized Born

In this section, we describe a method for fast computation of the GB solvation energy, along with the energy derivatives for the solvation forces, based on a discrete and continuum model of the molecules (Figure 6.4). An efficient method of sampling quadrature points on the nonlinear patch is given. We also show that the error of the Born radius calculation is controlled by the size of the triangulation mesh and the regularity of the periodic function used in the fast summation algorithm. The time complexity of the forces computation is reduced from the original $O(MN+M^2)$ to nearly linear time $O(N+M+n^3\log n+M\log M)$, where M is the number of atoms of a molecule, N is the number of integration points that we sample on the surface of the molecule when we compute the Born radius for each atom, and n is a parameter introduced in the fast summation algorithm. The fast summation method shows its advantage when it is applied to the Born radius calculations for macromolecules, where there could be tens of thousands or millions of atoms, and N could be even larger. In the fast summation method, one only need to choose a small n which is much smaller than M and N to get a good approximation, which makes the new fast summation based GB method more efficient.

6.6.1 Fast solvation energy computation

Method

Similarly to what is done for other GB models, we use (**??**) as the electrostatic solvation energy function. Before we compute (**??**), we need to first compute the effective Born radius R_i for every atom which reflects the depth a charge buried inside the molecule (Figure: 6.5). An atom buried deep in a molecule has a larger Born radius, whereas an atom near the surface has a smaller radius. Hence surfactant atoms have a stronger impact on the polarization. Given a discrete van der Waals (vdW) atom model, as long as we know R_i for each atom, we can compute (**??**) by using the fast multipole method (FMM) [11] with the time complexity $O(M\log M)$. However the Born radii computation is not easy and is very time-consuming. There are

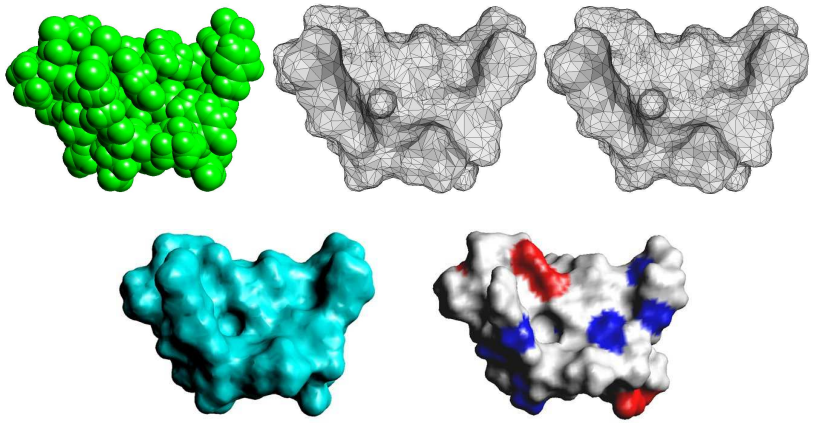


Figure 6.4: Top left: the discrete van der Waals surface model (436 atoms); top middle: the triangulation of the continuum Gaussian surface model with 6004 triangles; top right: the regularized triangular mesh where the quality of the elements is improved (making each as close as possible to an equilateral triangles); bottom left: the continuum ASMS model generated from the triangular mesh up right; bottom right: the molecular surface rendered according to the interaction with the solvent where red means strong and blue means weak interaction.

various ways of computing the Born radius as summarized in [8]. These methods can be divided into two categories: volume integration based methods and surface integration based methods. In general, the surface integration methods are more efficient than the volume integration methods due to the decreased dimension. So we adopt the surface integration method given in [10] to compute the Born radius:

$$R_i^{-1} = \frac{1}{4\pi}\int_\Gamma \frac{(\mathbf{r}-\mathbf{x}_i)\cdot\mathbf{n}(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_i|^4}\,\mathrm{d}S \qquad i=1,\ldots,M, \tag{6.13}$$

where Γ is the molecule-solvent interface, $\mathbf{x}_i$ is the center of atom i, and $\mathbf{n}(\mathbf{r})$ is the unit normal on the surface at $\mathbf{r}$ and we use ASMS as the model of Γ.

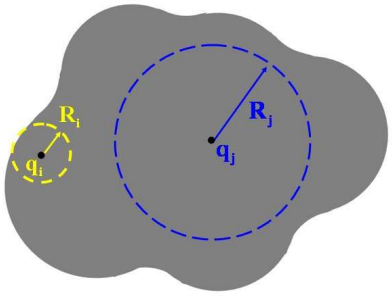
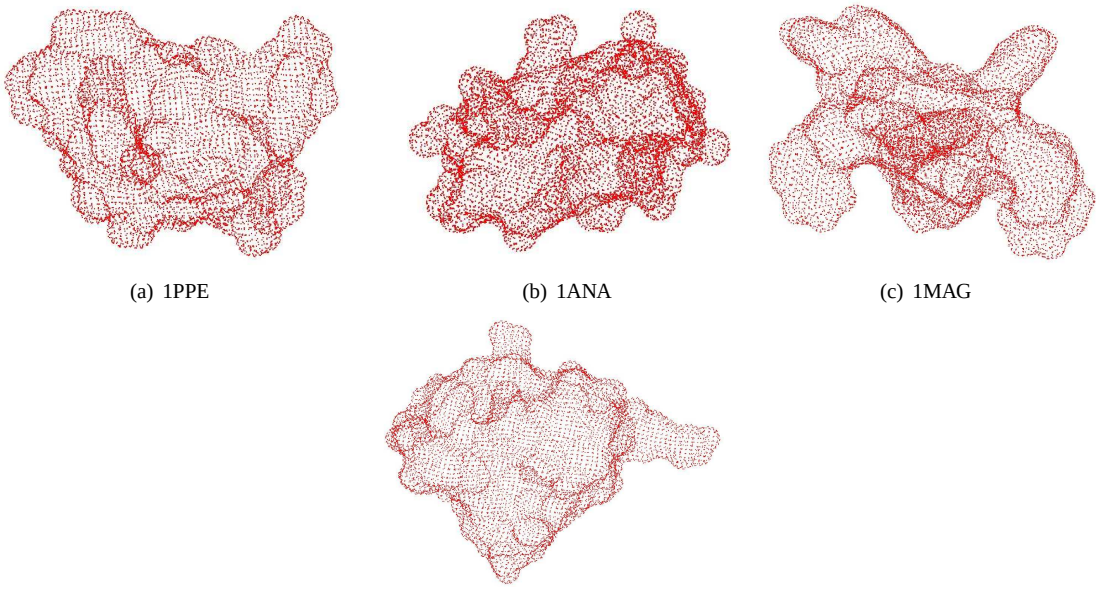


Figure 6.5: The effective Born radius reflects how deep a charge is buried inside the molecule. The Born radius of an atom is small if the atom is close to the surface of the molecule, otherwise the Born radius is large therefore has weaker interaction with the solvent.

Applying the Gaussian quadrature, We compute (6.13) numerically:

$$R_i^{-1} = \frac{1}{4\pi}\sum_{k=1}^{N} w_k \frac{(\mathbf{r}_k-\mathbf{x}_i)\cdot\mathbf{n}(\mathbf{r}_k)}{|\mathbf{r}_k-\mathbf{x}_i|^4} \qquad i=1,\ldots,M, \tag{6.14}$$

where w_k and $\mathbf{r}_k$ are the Gaussian integration weights and nodes on Γ (Figure 6.6). $\mathbf{r}_k$ are computed by mapping the Gaussian nodes of a master triangle to the algebraic patch via the transformation $\mathscr{T}$. Let $\mathbf{r}_k^0$ and w_k^0 be one of the Gaussian nodes and



(a) 1PPE (b) 1ANA (c) 1MAG

(d) 1CGI_l

Figure 6.6: Gaussian integration points on the surface of protein (a) 1PPE, (b) 1ANA, (c) 1MAG, and (d) 1CGI_l. The surfaces are partitioned into 24244 triangular patches for (a), 28620 triangular patches for (b), 30624 triangular patches for (c), and 29108 triangular patches for (d). There are three Gaussian quadrature nodes per triangle. The nodes are then mapped onto the ASMS to form the red point cloud.

weights on the master triangle. Then the corresponding node $\mathbf{r}_k$ and weight w_k are $\mathbf{r}_k = \mathscr{T}(\mathbf{r}_k^0)$ and $w_k = w_k^0 |J(\mathscr{T})|$ where $|J(\mathscr{T})|$ is the Jacobian determinant of $\mathscr{T}$.

We formalize (6.14) in two steps. First we split it into two parts:

$$R_i^{-1} = \frac{1}{4\pi} \sum_{k=1}^{N} \frac{w_k \mathbf{r}_k \cdot \mathbf{n}(\mathbf{r}_k)}{|\mathbf{r}_k - \mathbf{x}_i|^4} - \frac{1}{4\pi} \sum_{k=1}^{N} \frac{w_k \mathbf{x}_i \cdot \mathbf{n}(\mathbf{r}_k)}{|\mathbf{r}_k - \mathbf{x}_i|^4}. \tag{6.15}$$

Then we split the second summation in (6.15) into three components:

$$\sum_{k=1}^{N} \frac{w_k \mathbf{x}_i \cdot \mathbf{n}(\mathbf{r}_k)}{|\mathbf{r}_k - \mathbf{x}_i|^4} = x_i \sum_{k=1}^{N} \frac{w_k n_x^k}{|\mathbf{r}_k - \mathbf{x}_i|^4} + y_i \sum_{k=1}^{N} \frac{w_k n_y^k}{|\mathbf{r}_k - \mathbf{x}_i|^4} + z_i \sum_{k=1}^{N} \frac{w_k n_z^k}{|\mathbf{r}_k - \mathbf{x}_i|^4}. \tag{6.16}$$

The first summation in (6.15) and the three summations in (6.16) without the coefficients in front are of the common form:

$$G(\mathbf{x}_i) = \sum_{k=1}^{N} c_k g(\mathbf{x}_i - \mathbf{r}_k) \qquad i = 1, \ldots, M, \tag{6.17}$$

with the kernel function $g(\mathbf{x} - \mathbf{r}_k) = \frac{1}{|\mathbf{x} - \mathbf{r}_k|^4}$ and the coefficient $c_k = w_k \mathbf{r}_k \cdot \mathbf{n}(\mathbf{r}_k)$, $w_k n_x^k$, $w_k n_y^k$, $w_k n_z^k$, respectively. (6.17) can be efficiently computed by using the fast summation algorithm introduced in [15] with complexity $O(M + N + n^3 \log n)$, where n is a parameter used in the fast summation algorithm.

Fast summation

The fast summation algorithm is published in [15]. For convenience, we discuss this algorithm in this section briefly. The fast summation algorithm is often applied to compute the summations of the form

$$G(\mathbf{x}_i) = \sum_{k=1}^{N} c_k g(\mathbf{x}_i - \mathbf{r}_k), \qquad i = 1, \ldots, M, \tag{6.18}$$

where the kernel function g is a fast decaying function. Cutting off the tail of g, one can assume that the support of g is bounded. In our Born radii computation, since the distance between $\mathbf{x}_i$ and $\mathbf{r}_k$ is no less than the smallest radius of the atoms, there is no singularity in g. Without loss of generality, we assume $\mathbf{x} - \mathbf{r}_k \in \Pi := [-\frac{1}{2}, \frac{1}{2}]^3$. After duplicating g in the other intervals, g can be extended to be a periodic function of period one in $\mathbb{R}^3$ and this periodic function can be decomposed into the Fourier series:

$$g(\mathbf{x} - \mathbf{r}_k) = \sum_{\boldsymbol{\omega} \in I_\infty} g_{\boldsymbol{\omega}} e^{2\pi i \boldsymbol{\omega} \cdot (\mathbf{x} - \mathbf{r}_k)}, \tag{6.19}$$

where $I_\infty := \{(\omega_1, \omega_2, \omega_3) \in \mathbb{Z}^3\}$ and $g_\omega = \int_\Pi g(\mathbf{x}) e^{-2\pi \mathrm{i} \boldsymbol{\omega} \cdot \mathbf{x}} \, \mathrm{d}\mathbf{x}$. We approximate (6.19) by a truncated series:

$$g(\mathbf{x} - \mathbf{r}_k) \approx \sum_{\boldsymbol{\omega} \in I_n} g_{\boldsymbol{\omega}} e^{2\pi i (\mathbf{x} - \mathbf{r}_k) \cdot \boldsymbol{\omega}}, \tag{6.20}$$

where $I_n := \{(\omega_1, \omega_2, \omega_3) \in \mathbb{Z}^3 : -\frac{n}{2} \le \omega_i < \frac{n}{2}\}$. We compute the Fourier coefficients g_ω numerically by

$$g_{\boldsymbol{\omega}} = \frac{1}{n^3} \sum_{\mathbf{j} \in I_n} g(\frac{\mathbf{j}}{n}) e^{-2\pi \mathrm{i} \boldsymbol{\omega} \cdot \mathbf{j}/n}, \qquad \boldsymbol{\omega} \in I_n. \tag{6.21}$$

by using the fast Fourier transform (FFT) algorithm with complexity $O(n^3 \log n)$.

Plugging (6.20) into (6.18), we get

$$\begin{aligned} G(\mathbf{x}_i) &\approx \sum_{k=1}^{N} c_k \left(\sum_{\boldsymbol{\omega} \in I_n} g_{\boldsymbol{\omega}} e^{2\pi i (\mathbf{x}_i - \mathbf{r}_k) . \boldsymbol{\omega}} \right) = \sum_{\boldsymbol{\omega} \in I_n} g_{\boldsymbol{\omega}} \left(\sum_{k=1}^{N} c_k e^{-2\pi i \boldsymbol{\omega} \cdot \mathbf{r}_k} \right) e^{2\pi i \boldsymbol{\omega} \cdot \mathbf{x}_i} \\ &= \sum_{\boldsymbol{\omega} \in I_n} g_{\boldsymbol{\omega}} a_{\boldsymbol{\omega}} e^{2\pi i \boldsymbol{\omega} \cdot \mathbf{x}_i} \end{aligned} \tag{6.22}$$

where

$$a_{\boldsymbol{\omega}} = \sum_{k=1}^{N} c_k e^{-2\pi i \boldsymbol{\omega} \cdot \mathbf{r}_k}. \tag{6.23}$$

(6.22) is computed by using the NFFT algorithm with complexity $O(n^3 \log n + M)$ and (6.23) is computed by the NFFT$^{\mathrm{T}}$ algorithm with complexity $O(n^3 \log n + N)$. Hence the total complexity of computing (6.18) is $O(N + M + n^3 \log n)$, which is significantly faster than the the trivial $O(MN)$ summation method once the number of terms in the Fourier series n is much smaller than M and N. We explain the NFFT algorithm, the NFFT$^{\mathrm{T}}$ algorithm, and the error computations in the Related Math section at the end of the chapter.

6.6.2 Fast solvation energy computation using Oct-Trees

we first describe an $\mathscr{O}(M \log M)$ algorithm for fast approximation of the Born radii of all M atoms in a molecule, followed by another $\mathscr{O}(M \log M)$ time algorithm for approximating E_{pol} from the approximated Born radii.

Born Radii

Let $\mathscr{A}$ be a set of M atoms in a molecule, and let $\mathscr{Q}$ be a set of $m = \mathscr{O}(M)$ Gauss quadrature points (denoted q-points) sampled on the MS. For each $q \in \mathscr{Q}$, let $\tilde{n}_q = w_q n_q$, where n_q is the outward unit normal on the MS at point q, and w_q is the weight assigned to q.

Our approach is to use a near and far decomposition of the elements in $\mathscr{A}$ and $\mathscr{Q}$. Hence, we build two octrees $\mathscr{T}_{\mathscr{A}}$ and $\mathscr{T}_{\mathscr{Q}}$ for $\mathscr{A}$ and $\mathscr{Q}$, respectively (see Figure 6.8).

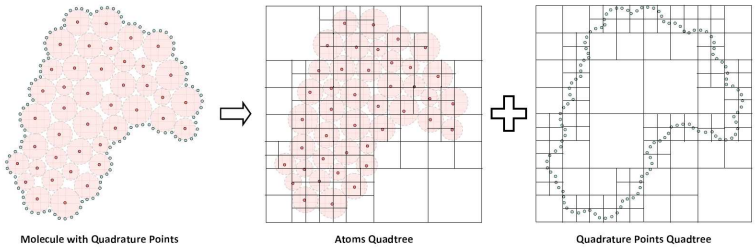


Figure 6.7: In our Born radius approximation algorithm we construct two octrees: one for the atoms in the molecule, and the other for the quadrature points. Born radii of all atoms are approximated by simultaneous recursive traversal of both octrees. Here the octrees are drawn as quadtrees [9] for simplicity.

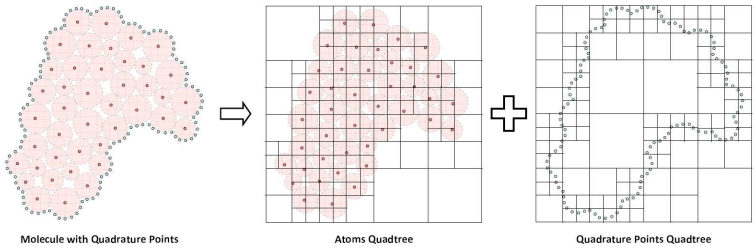


Figure 6.8: In our Born radius approximation algorithm we construct two octrees: one for the atoms in the molecule, and the other for the quadrature points. Born radii of all atoms are approximated by simultaneous recursive traversal of both octrees. Here the octrees are drawn as quadtrees [9] for simplicity.

We traverse $\mathscr{T}_{\mathscr{A}}$ and $\mathscr{T}_{\mathscr{Q}}$ simultaneously starting at their root nodes, and collect the approximated integrals at appropriate internal nodes of $\mathscr{T}_{\mathscr{A}}$ and atoms of $\mathscr{A}$. Suppose at some point during this traversal we are at node A of $\mathscr{T}_{\mathscr{A}}$ and node Q of $\mathscr{T}_{\mathscr{Q}}$. Let r_A (resp. r_Q) be the radius[6] of A (resp. Q). If A and Q are far enough, i.e., the distance between their centers is larger than $(r_A + r_Q)\,\frac{(1+\varepsilon)^{1/4}+1}{(1+\varepsilon)^{1/4}-1}$ for some user-defined approximation parameter $\varepsilon > 0$, then the contribution of all q-points in Q to the Born radius integral of each atom in A can be approximated by treating A (resp. Q) as a single pseudo atom (resp. pseudo q-point) centered at the geometric center of the atoms (resp. q-points) under it, and assuming $\tilde{n}_Q = \sum_{q \in Q} \tilde{n}_q$. This approximated contribution is collected in A. If A and Q are not far enough but at least one of them is a nonleaf, we recurse using the children of the nonleaf/nonleaves. If both are leaves then we compute the contribution exactly using the atoms under A and the q-points under Q, and collect it in the respective atoms. Finally, we traverse $\mathscr{T}_{\mathscr{A}}$ top-down and add the collected partial integrals to each atom from its ancestors and compute its Born radius from these accumulated values. The pseudocode is given in Figure 6.9.

The accuracy and running time of the algorithm depends on the approximation parameter $\varepsilon > 0$. The smaller the value of ε is the more accurate the approximated Born radii are, and the larger ε is the faster the algorithm runs. The running time is dominated by the time required for approximating the interactions between the atoms and the q-points through the simultaneous traversal of the two octrees. The analysis of the running time is similar to the one given in Section 6.3 for approximating LJ potential, and can be shown to be $\mathscr{O}\left(M \log m + m \log M + \frac{1}{\varepsilon^3}(M+m)\right)$ which reduces to $\mathscr{O}\left(M \log M + M/\varepsilon^3\right)$ for $m = \mathscr{O}(M)$. Assuming the octrees are already available, the running time is only $\mathscr{O}\left(M/\varepsilon^3\right)$. The algorithm uses $\mathscr{O}(M)$ space.

Polarization Energy

Our algorithm is based on near and far decomposition of the given set of atoms using octrees. Consider a set $\mathscr{A}$ of M atoms with $R_{\min}$ and $R_{\max}$ being the minimum and the maximum of the Born radii in $\mathscr{A}$, respectively. Now given an approximation parameter $\varepsilon > 0$, we divide the atoms into $M_\varepsilon = \log_{1+\varepsilon}(R_{\max}/R_{\min}) = \mathscr{O}(\log M)$ groups, and place each atom a with Born radius $R_a \in [R_{\min}(1+\varepsilon)^k, R_{\min}(1+\varepsilon)^{k+1})$ in group $k \in [0, M_\varepsilon)$, and approximate R_a with $R_{\min}(1+\varepsilon)^k$. We build an octree $\mathscr{T}_{\mathscr{A}}$ as in Section 6.6.2. For every node $A \in \mathscr{T}_{\mathscr{A}}$ and $0 \le k < M_\varepsilon$, we precompute $q_A[k] = \sum_{(a \in A) \,\wedge\, (a \in \text{group } k)} q_a$. We now traverse $\mathscr{T}_{\mathscr{A}}$ simultaneously using two pointers both of which initially point to the root node of $\mathscr{T}_{\mathscr{A}}$. Suppose at some point during this traversal the two pointers point to nodes U and V. We first check if both U and V are leaves, and if so, we compute the interaction between the two sets of atoms under U and V directly using actual charges, Born radii and inter-atomic distances. Otherwise if the two nodes are far enough from each other the interaction between the set of atoms under them is approximated using the approximate Born radii described above and the sum of charges q_U and q_V. If the two nodes are too close for approximation, we recurse on the nonleaf node(s). The pseudocode is given in Figure 6.10.

As with most other algorithms in this paper the performance of this algorithm depends on the approximation parameter ε with smaller values resulting in better approximations and larger values leading to better running times. The algorithm runs in

[6] i.e., r_A = radius of the smallest ball centered at the geometric center of the atom centers in A that encloses all atom centers of A.

APPROX-INTEGRALS(A, Q)

(For each atom a under the subtree rooted at the given node A in the atoms octree approximate $\sum_{q\in Q} w_q \frac{(\mathbf{p}_q-\mathbf{p}_a)\cdot\mathbf{n}_q}{|\mathbf{p}_q-\mathbf{p}_a|^4}$. By $\mathbf{p}_a = \langle x_a, y_a, z_a\rangle$ we denote the center of an atom a, while by $\mathbf{p}_q = \langle x_q, y_q, z_q\rangle$, w_q and $\mathbf{n}_q = \langle nx_q, ny_q, nz_q\rangle$ we denote the location of a quadrature point q, weight assigned to q, and the unit outward normal on the molecular surface at q, respectively. By $\langle x_A, y_A, z_A\rangle$ (resp. $\langle x_Q, y_Q, z_Q\rangle$) we denote the geometric center of the atoms (resp. integration points) under A (resp. Q). By r_A (resp. r_Q) we denote the radius of the smallest ball centered at $\langle x_A, y_A, z_A\rangle$ (resp. $\langle x_Q, y_Q, z_Q\rangle$) that encloses all atom centers (resp. integration points) under A (resp. Q). The distance between the geometric centers of A and Q is given by $r_{A,Q}$. We also assume $\tilde{nx}_Q = \sum_{q\in Q} w_q nx_q$. Similarly for $\tilde{ny}_Q$ and $\tilde{nz}_Q$. Each atom a has two fields s_a and c_a, and each node A in the atoms octree has fields s_A and c_A, all of which are initialized to zero. The approximated sum is added to s_A provided A and Q are far enough in space so that the sum can be approximated reasonably well (controlled by an approximation parameter $\varepsilon > 0$). Otherwise the sums are computed recursively and added to the s field of appropriate descendants of A. We also approximate a correction term $\sum_{q\in Q} w_q \frac{(\mathbf{p}_q-\mathbf{p}_a)\cdot\mathbf{n}_q}{|\mathbf{p}_q-\mathbf{p}_a|^7}$ and add it to c_A or the c field of the appropriate descendants of A. By CHILD(A) (resp. CHILD(Q)) we denote the set of non-empty octree nodes obtained by subdividing node A (resp. Q).)

1. ***if*** $r_{A,Q} - (r_A + r_Q) > 0 \ \wedge \ \frac{r_{A,Q}+(r_A+r_Q)}{r_{A,Q}-(r_A+r_Q)} > (1+\varepsilon)^{\frac{1}{4}}$ ***then*** {*far enough to approximate*}

   $x_\Delta = x_A - x_Q, \ y_\Delta = y_A - y_Q, \ z_\Delta = z_A - z_Q$

   $s_A = s_A + \frac{\tilde{nx}_Q\cdot x_\Delta + \tilde{ny}_Q\cdot y_\Delta + \tilde{nz}_Q\cdot z_\Delta}{(r_{A,Q})^4}, \ c_A = c_A + \frac{\tilde{nx}_Q\cdot x_\Delta + \tilde{ny}_Q\cdot y_\Delta + \tilde{nz}_Q\cdot z_\Delta}{(r_{A,Q})^7}$

2. ***else if*** LEAF$(A) \wedge$ LEAF(Q) ***then*** {*too close to approximate; compute exact value*}

   for each atom $a \in A$ ***do***

   for each quadrature point $q \in Q$ ***do***

   $x_\delta = x_a - x_q, \ y_\delta = y_a - y_q, \ z_\delta = z_a - z_q$

   $s_a = s_a + \frac{w_q\cdot(nx_q\cdot x_\delta + ny_q\cdot y_\delta + nz_q\cdot z_\delta)}{(r_{a,q})^4}, \ c_a = c_a + \frac{w_q\cdot(nx_q\cdot x_\delta + ny_q\cdot y_\delta + nz_q\cdot z_\delta)}{(r_{a,q})^7}$

3. ***else if*** LEAF(A) ***then*** $\forall Q' \in$ CHILD(Q) : APPROX-INTEGRALS(A, Q') {*recurse on Q*}

4. ***else if*** LEAF(Q) ***then*** $\forall A' \in$ CHILD(A) : APPROX-INTEGRALS(A', Q) {*recurse on A*}

5. ***else*** $\forall A' \in$ CHILD(A) $\wedge \forall Q' \in$ CHILD(Q) : APPROX-INTEGRALS(A', Q') {*recurse on A and Q*}

APPROX-INTEGRALS ENDS

PUSH-INTEGRALS-TO-ATOMS(A, s, c)

(A is a node in the atoms octree, $s = \sum_{A'\in\text{ANCESTORS(A)}} s_{A'}$ and $c = \sum_{A'\in\text{ANCESTORS(A)}} c_{A'}$. This function pushes $s + s_A$ and $c + c_A$ to each descendant of A. If A is a leaf it computes the Born radius of each atom $a \in A$ using $s + s_A + s_a$ and $c + c_A + c_a$.)

1. ***if*** LEAF(A) ***then*** $\forall a \in A : R_a = \max\left\{ r_a, \ \frac{1}{\left(1-\frac{1}{\sqrt{2}}\right)\cdot\frac{s_a+s+s_A}{4\pi} + \left(\frac{c_a+c+c_A}{16\pi}\right)^{\frac{1}{4}}} \right\}$ {*compute Born radii of A's atoms*}

2. ***else*** $\forall A' \in$ CHILD(A) : PUSH-INTEGRALS-TO-ATOMS(A', $s + s_A$, $c + c_A$) {*push integrals to A's descendants*}

PUSH-INTEGRALS-TO-ATOMS ENDS

Figure 6.9: Octree-based algorithm for approximating Born radii. Given the atoms octree $\mathscr{T}_{\mathscr{A}}$ and quadrature/integration points octree $\mathscr{T}_{\mathscr{Q}}$, the Born radii of all atoms in $\mathscr{T}_{\mathscr{A}}$ can be approximated (controlled by a given approximation parameter $\varepsilon > 0$) by making the following sequence of function calls: APPROX-INTEGRALS(ROOT($\mathscr{T}_{\mathscr{A}}$), ROOT($\mathscr{T}_{\mathscr{Q}}$)), and PUSH-INTEGRALS-TO-ATOMS(ROOT($\mathscr{T}_{\mathscr{A}}$), 0, 0).

APPROX-$E_{\text{pol}}(U, V)$

(For two given nodes U and V in the atoms octree $\mathscr{T}_{\mathscr{A}}$ approximate the part of E_{pol} resulting from the interaction between the set of atoms under U and V. By (x_U, y_U, z_U) we denote the geometric center of the atoms under U. By r_U we denote the radius of the smallest ball centered at (x_U, y_U, z_U) that encloses all atom centers under U. For any atom $u \in U$, its center, radius, charge and Born radius are given by (x_u, y_u, z_u), r_u, q_u and R_u, respectively. For $0 \leq k < M_\varepsilon = \log_{1+\varepsilon}\left(\frac{R_{max}}{R_{min}}\right)$, $q_U[k] = \sum_{(u \in U) \wedge (R_u \in [R_{min}(1+\varepsilon)^k, R_{min}(1+\varepsilon)^{k+1})} q_u$, where R_{min} and R_{max} are the minimum and the maximum Born radius among all atoms in $\mathscr{A}$. By CHILD(A) (resp. CHILD(Q)) we denote the set of non-empty octree nodes obtained by subdividing node A (resp. Q).)

1. ***if*** LEAF$(U) \wedge$ LEAF(V) ***then return*** $-\frac{\tau}{2}\sum_{(u \in U) \wedge (v \in V)} \frac{q_u q_v}{\sqrt{r_{uv}^2 + R_u R_v e^{\frac{-r_{uv}^2}{4R_u R_v}}}}$ {*exact value*}
2. ***else if*** $r_{U,V} > (r_U + r_V)\left(1 + \frac{2}{\varepsilon}\right)$ ***then return*** $-\frac{\tau}{2}\sum_{0 \leq i,j < M_\varepsilon} \frac{q_U[i] \cdot q_V[j]}{\sqrt{r_{UV}^2 + R_{min}(1+\varepsilon)^{i+j} e^{\frac{-r_{UV}^2}{4R_{min}(1+\varepsilon)^{i+j}}}}}$ {*approximate*}
3. ***else if*** LEAF(U) ***then return*** $\sum_{V' \in \text{CHILD}(V)}$ APPROX-$E_{\text{pol}}(U, V')$ {*recurse on V*}
4. ***else if*** LEAF(V) ***then return*** $\sum_{U' \in \text{CHILD}(U)}$ APPROX-$E_{\text{pol}}(U', V)$ {*recurse on U*}
5. ***else return*** $\sum_{(U' \in \text{CHILD}(U)) \wedge (V' \in \text{CHILD}(V))}$ APPROX-$E_{\text{pol}}(U', V')$ {*recurse on U and V*}

APPROX-E_{pol} ENDS

Figure 6.10: Octree-based algorithm for approximating E_{pol} from Born radii. Given the atoms octree $\mathscr{T}_{\mathscr{A}}$ with all Born radii already computed, E_{pol} can be approximated (controlled by a given approximation parameter $\varepsilon > 0$) by making the following function call: APPROX-E_{pol}(ROOT$(\mathscr{T}_{\mathscr{A}})$, ROOT$(\mathscr{T}_{\mathscr{A}})$).

$\mathscr{O}\left(\frac{1}{\varepsilon^3} \cdot M \log M\right)$ time, and uses $\mathscr{O}(M)$ space.

6.7 Possion Boltzman

Related Work

Generalized Born

Because the GB calculation is much faster than solving the PB equation, the GB model is widely used in the MD simulations. Programs which implement the GB methods include CHARMM [13], Amber [2], Tinker [17], and Impact which is now part of Schrodinger, Inc.'s FirstDiscovery program suite. Even though the GB computation is much faster than the PB model, the computation of the Born radius R_i is still slow. During the MD simulation, the Born radii need to be frequently recomputed at different time steps. Because this part of computation is too time-consuming, there are attempts to accelerate the MD simulation by computing the Born radii at a larger time step. For example, in [24] in their test of a 3 ns GB simulation of a 10-base pair DNA duplex, they change the time step of computing the Born radii and long-range electrostatic energy from 1 fs to 2 fs. This reduces the time of carrying out the simulation from 13.84 hours to 7.16 hours. From this example we can see that the calculation of the Born radii takes a large percentage of total computation time in the MD simulation. In the long dynamic runs, this decrease in the frequency of evaluating the effective Born radii are not accurate enough to conserve energy which restricts the MD simulation of the protein folding process to small time scale [21]. Hence it is demanding to calculate the Born radii and the solvation energy accurately and efficiently.

Relevant Mathematics

Gaussian surface, Triangular mesh and Algebraic spline molecular surface (ASMS)
See Chapters 2 and 3.

Oct-tree

Numerical Integration

Bibliography

[1] C. Bajaj, R. A. Chowdhury, and M. Rasheed. A dynamic data structure for flexible molecular maintenance and informatics. In *SIAM/ACM Conf. Geom. Phys. Model.*, pages 259–270, New York, NY, USA, 2009. ACM.

[2] D. Case, T. Cheatham, III, T. Darden, H. Gohlke, R. Luo, K. Merz, Jr., A. Onufriev, C. Simmerling, B. Wang, and R. Woods. The Amber biomolecular simulation programs. *J. Comput Chem.*, 26:1668–1688, 2005.

[3] D. Chandler. Interfaces and the driving force of hydrophobic assembly. *Nature*, 437:640, 2005.

[4] R. A. Chowdhury and C. Bajaj. Algorithms for faster molecular energetics, forces and interfaces. ICES report 10-32, Institute for Computational Engineering & Science, The University of Texas at Austin, Austin, TX, USA 78712., August 2010.

[5] M. Davis. The inducible multipole solvation model: A new model for solvation effects on solute electrostatics. *J. Chem. Phys.*, 100:5149–5159, 1994.

[6] J. Dzubiella, J. Swanson, and J. McCammon. Coupling hydrophobicity, dispersion, and electrostatics in continuum solvent models. *Phys. Rev. Lett.*, 96:087802, 2006.

[7] D. Eisenberg and A. Mclachlan. Solvation energy in protein folding and binding. *Nature*, 319:199–203, 1986.

[8] M. Feig, A. Onufriev, M. S. Lee, W. Im, D. A. Case, and C. Brooks III. Performance comparison of generalized Born and Poisson methods in the calculation of electrostatic solvation energies for protein structures. *J. Comput Chem.*, 25:265–284, 2004.

[9] R. A. Finkel and J. L. Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4:1–9, 1974.

[10] A. Ghosh, C. S. Rapp, and R. A. Friesner. Generalized Born model based on a surface integral formulation. *J. Phys. Chem. B*, 102:10983–10990, 1998.

[11] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J Chemical Physics*, 73:325–348, 1987.

[12] C. L. Jackins and S. L. Tanimoto. Oct-trees and their use in representing three-dimensional objects. *Computer Graphics and Image Processing*, 14(3):249–270, 1980.

[13] A. MacKerel Jr., C. Brooks III, L. Nilsson, B. Roux, Y. Won, and M. Karplus. *CHARMM: The Energy Function and Its Parameterization with an Overview of the Program*, volume 1 of *The Encyclopedia of Computational Chemistry*, pages 271–277. John Wiley & Sons: Chichester, 1998.

[14] G. M. Morris, D. S. Goodsell, R. Huey, and A. J. Olson. Distributed automated docking of flexible ligands to proteins: Parallel applications of AutoDock 2.4. *Journal of Computer-Aided Molecular Design*, 10:293–304, 1996. 10.1007/BF00124499.

[15] D. Potts and G. Steidl. Fast summation at nonequispaced knots by NFFTs. *SIAM J. Sci. Comput.*, 24:2013–2037, 2003.

[16] L. Pratt and D. Chandler. Theory of the hydrophobic effect. *J. Chem. Phys.*, 67:3683–3704, 1977.

[17] P. Ren and J. W. Ponder. Polarizable atomic multipole water model for molecular mechanics simulation. *J. Phys. Chem. B*, 107:5933–5947, 2003.

[18] B. Roux and T. Simonson. Implicit solvent models. *Biophys. Chem.*, 78:1–20, 1999.

[19] M. Schaefer and M. Karplus. A comprehensive analytical treatment of continuum electrostatics. *J. Phys. Chem.*, 100:1578–1599, 1996.

[20] K. Sharp and B. Honig. Calculating total electrostatic energies with the nonlinear Poisson-Boltzmann equation. *J. Phys. Chem.*, 94:7684–7692, 1990.

[21] A. Y. Shih, I. G. Denisov, J. C. Phillips, S. G. Sligar, and K. Schulten. Molecular dynamics simulations of discoidal bilayers assembled from truncated human lipoproteins. *Biophys. J.*, 88:548–556, 2005.

[22] H. Sklenar, F. Eisenhaber, M. Poncin, and R. Lavery. *Theoretical biochemistry and molecular biophysics*. Adenine Press, 1990.

[23] W. C. Still, A. Tempczyk, R. C. Hawley, and T. Hendrickson. Semianalytical treatment of solvation for molecular mechanics and dynamics. *J. Am. Chem. Soc*, 112:6127–6129, 1990.

[24] V. Tsui and D. A. Case. Theory and applications of the generalized Born solvation model in macromolecular simulations. *Biopolymers*, 56:275–291, 2001.

[25] J. Wagoner and N. Baker. Assessing implicit models for nonpolar mean solvation forces: The importance of dispersion and volume terms. *PNAS*, 103:8331–8336, 2006.

[26] J. Weeks, D. Chandler, and H. Andersen. Role of repulsive forces in determining the equilibrium structure of simple liquids. *Journal of Chemical Physics*, 54:5237–5247, 1971.

[27] S. J. Weiner, P. A. Kollman, D. A. Case, U. C. Singh, C. Ghio, G. Alagona, S. Profeta, and P. Weiner. A new force field for molecular mechanical simulation of nucleic acids and proteins. *Journal of the American Chemical Society*, 106:765–784, 1984.

[28] R. L. L. Zhang, E. Gallicchio, and A. Felts. On the nonpolar hydration free energy of proteins: Surface area and continuum solvent models for the solute-solvent interaction energy. *J. Am. Chem. Soc.*, 125:9523–9530, 2003.

Chapter 7

Forces

Force field, used for molecular mechanics simulation, is a set of parameters and functions describing the potential energy of a system atoms. The functions and parameters are derived from both experimental work and high-level quantum mechanical calculations.

In CHARMM, the potential energy function has the form

$$\begin{aligned} E = & \sum_{\text{bonds}} K_b(b-b_0)^2 + \sum_{\text{UB}} K_{UB}(S-S_0)^2 + \sum_{\text{angle}} K_\theta(\theta-\theta_0)^2 \\ & + \sum_{\text{dihedrals}} K_\chi(1+\cos(n\chi-\delta)) + \sum_{\text{impropers}} K_{\text{imp}}(\phi-\phi_0)^2 \\ & + \sum_{\text{nonbond}} \varepsilon_{ij}\left[\left(\frac{R_{\min_{ij}}}{r_{ij}}\right)^{12} - \left(\frac{R_{\min_{ij}}}{r_{ij}}\right)^{6}\right] + \frac{q_i q_j}{\varepsilon r_{ij}} \end{aligned} \tag{0.1}$$

where K_b, K_{UB}, K_θ, K_χ, and K_{imp} are the bond, Urey-Bradley, angle, dihedral angle, and improper dihedral angle force constants; b, S, θ, χ, and ϕ are the bond length, Urey-Bradleu 1,3-distance (the distance between atoms separated by two covalent bonds), bond angle, dihedral angle, and improper torsion angle. The symbols with subscript zero represent the equilibrium values. n is the multiplicity of the rotor (e.g., 3 for a methyl group) and δ is the phase angle. The above terms are referred to as the internal parameters. The Coulomb and Lennard-Jones (LJ) terms are the external or nonbonded interactions. ε_{ij} is the LJ well depth and $R_{\min_{ij}}$ is the mininum interaction radius. The dielectric constant ε is 1 in the energy function. Different versions of CHARMM have different optimization strategies for the parameters K_b, K_{UB}, K_θ, K_χ, K_{imp}, b_0, S_0, θ_0, n, δ, ϕ_0, q_i, ε_{ij}, and $R_{\min_{ij}}$. CHARMM 19 [**?**] treats polar hydrogens (i.e. H atoms on N and O) explicitly and hydrogens bonded to S and C are treated as parts of the extended atom (e.g., CH_3 is treated as a single atom), while CHARMM 22 [**?**] and CHARMM 27 [**?**] include all the atoms explicitly. Comparing CHARMM 22 and CHARMM 27, the latter yields a better representation of the equilibrium between the A and B forms of DNA and the A form of RNA.

The potential energy function of AMBER [**?**] has the form

$$\begin{aligned} E = & \sum_{\text{bonds}} K_b(b-b_0)^2 + \sum_{\text{angle}} K_\theta(\theta-\theta_0)^2 + \sum_{\text{dihedrals}} \frac{V_n}{2}(1+\cos(n\phi-\delta)) \\ & + \sum_{\text{nonbond}} \left[\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} + \frac{q_i q_j}{\varepsilon r_{ij}}\right] \end{aligned} \tag{0.2}$$

The major difference between the energy function of CHARMM and AMBER force field is that AMBER omits the Urey-Bradley terms. Amber 99 [**?**] is developed to minimize the number of torsion energies and extend the additive force field

described by (0.2) to a nonadditive model by adding the polarization energy of the form

$$E_{pol} = -\frac{1}{2}\sum_i \mu_i E_i^0 \tag{0.3}$$

$$\mu_i = \alpha_i E_i \tag{0.4}$$

$$E_i = E_i^0 + \sum_{j\neq i} T_{ij}\mu_j \tag{0.5}$$

$$E_i^0 = \sum_{j\neq i} q_j \frac{\mathbf{r}_{ij}}{r_{ij}^2} \tag{0.6}$$

$$T_{ij} = \frac{2}{r_{ij}^3} \tag{0.7}$$

The improvement of AMBER 03 is that it can deal with condense phase simulations of proteins [**?**].

The strategies to assign partial charges of CHARMM and AMBER are also different. For CHARMM, the partial atomic charges are optimized by minimizing the interaction energies and geometries between a water molecule and the chemical groups in a variety of orientation [**?**]. For AMBER, the partial charges are assigned using a restrained electrostatic potential fit (RESP) model [**?**] which imposes symmetry on the hydrogens and constrains the charge on the central iron. CHARMM force field is optimized for molecular dynamics simulations with the TIP3P water model (explicit solvent) [**?**], while AMBER force field is optimized for the model with continuum solvent [**?**]. According to [**?**], simulation with an explicit representation of solvent and counterions, as well as periodic boundary conditions, have been the predominant method of applying MD simulation to nucleic acid.

In Molecular dynamics (MD) simulations, atomic trajectories are computed by solving equations of motion numerically using the force fields [**?**].

7.1 Energetic derivatives

7.2 Area, Volume Derivatives

Related Work

Relevant Mathematics

Ideals

Higher order quadratures

Commutative Algebra

Bibliography

Chapter 8

Structural Similarity

Related Work

Relevant Mathematics

Chapter 9

Docking

9.1 Affinitiy Functions

9.2 Search and Scoring

9.3 Filters

9.4 Re-Ranking

Related Work

Relevant Mathematics

Chapter 10

Molecular Machines

10.1 Virus

Viruses are one of the smallest parasitic nano-objects that are agents of human disease [39]. They have no systems for translating RNA, ATP generation, or protein, nucleic acid synthesis, and therefore need the subsystems of a host cell to sustain and replicate [39]. It would be natural to classify these parasites according to their eukaryotic or prokaryotic cellular hosts (e.g. plant, animal, bacteria, fungi, etc.), however there do exist viruses which have more than one sustaining host species [39]. Currently, viruses are classified simultaneously via the host species(Algae, Archae, Bacteria, Fungi, Invetebrates, Mycoplasma, Plants, Protozoa, Spiroplasma, Vetebrates), the host tissues that are infected, the method of virial transmission, the genetic organization of the virus (single or double stranded, linear or circular, RNA or DNA), the protein arrangement of the protective closed coats housing the genome (helical, icosahedral symmetric nucleo-capsids), and whether the virus capsids additionally have a further outer envelope covering (the complete virion)[39]. Table 1 summarizes a small yet diverse collection of viruses and virions [1]. The focus of this article is on the computational geometric modeling and visualization of the nucleo-capsid ultrastructure of plant and animal viruses exhibiting the diversity and geometric elegance of the multiple protein arrangements. Additionally, one computes a regression relationship between surface area v.s. enclosed volume for spherical viruses with icosahedral symmetric protein arrangements. The computer modeling and quantitative techniques for virus capsid shells ultra-structure that we review here are applicable for atomistic, high resolution (less than 4 Å) model data, as well as medium (5 Å to 15 Å) resolution map data reconstructed from cryo-electron microscopy.

10.1.1 The Morphology of Virus Structures

Minimally viruses consist of a single nucleocapsid made of proteins for protecting their genome, as well as in facilitating cell attachment and entry. The capsid proteins magically self-assemble, into often a helical or icosahedral symmetric shell (henceforth referred to as capsid shells). There do exist several examples of capsid shells which do not exhibit any global symmetry [1], however we focus on only the symmetric capsid shells in the remainder of this article.

Different virus morphologies that are known, (a small sampling included in Table 10.1) are distinguished by optional additional outer capsid shells, the presence or lack of a surrounding envelope for these capsid shells (derived often from the host cell's organelle membranes), as well as additional proteins within these optional capsids and envelopes, that are necessary for the virus lifecycle. The complete package of proteins, nucleic acids and envelopes is often termed a virion.

The asymmetric structural subunit of a symmetric capsid shell may be further decomposable into simpler and smaller protein structure units termed protomers. Protomers could be a single protein in monomeric form (example TMV), or form homogeneous dimeric or trimeric structure units (example RDV). These structure units also often combine to form symmetric clusters, called capsomers, and are predominantly distinguishable in visualizations at even medium and low resolution virus structures. The capsomers and/or protomeric structure units pack to create the capsid shell in the form of either helical or icosahedral symmetric arrangements.

Name	Family	Host	NA	Symmetry (T)	Shell (E?)	Modality (res in A)(pdbid)
Tobacco mosaic [8, 29]	Tobamoviridae	P	sR (L)	He	1(n)	X(2.45) (1ei7)
Ebola [38]	Filoviridae	V	sR (L)	He	1(E)	X(3) (1ebo)
Vaccinia [12]	Poxviridae	V	dD (L)	He	1(E)	X(1.8) (1luz)
Rabies [25]	Rhabdoviridae	V	sR(L)	He	1(E)	X(1.5) (1vyi)
Satellite tobacco necrosis [23]	Tombusviridae	P	sR (L)	Ic (1)	1(n)	X(2.5) (2stv)
L-A (Saccharomyces cerevisiae) [27]	Totiviridae	F	dR (L)	Ic (1)	1(n)	X(3.6) (1m1c)
Canine parvovirus-Fab complex [40]	Parvoviridae	V	sD (L)	Ic (1)	1(n)	X(3.3) (2cas)
T1L reovirus core [32]	Reoviridae	V	dR (L)	Ic (1,1)	2(n)	X(3.6) (1ej6)
T3D reovirus core [36]	Reoviridae	V	dR (L)	Ic (1,1)	2(n)	X(2.5) (1muk)
P4 (Ustilago maydis) [21]	Totiviridae	P	dR (L)	Ic (1)	1(n)	X(1.8) (1kp6)
Tomato bushy stunt [19]	Tombusviridae	P	sR (L)	Ic (3)	1(n)	X(2.9) (2tbv)
Cowpea Chlorotic Mosaic [34]	Bromoviridae	P	sR (L)	Ic (3)	1(n)	X(3.2) (1cwp)
Cucumber mosaic [34]	Bromoviridae	P	sR (L)	Ic (3)	1(n)	X(3.2) (1f15)
Norwalk [31]	Caliciviridae	V	sR (L)	Ic (3)	1(n)	X(3.4) (1ihm)
Rabbit hemorrhagic disease complex [30]	Caliciviridae	V	sR (L)	Ic (3)	1(n)	X(2.5) (1khv)
Galleria mellonella denso[33]	Parvoviridae	I	sD (L)	Ic (1)	1(n)	X(3.7) (1dnv)
Semiliki Forest [24]	Togaviridae	I,V	sR	Ic (4,1)	2(E)	C(9) (1dyl)
Polyoma [11]	Papovaviridae	V	dD (C)	Ic (7D)	1(n)	X(2.2) (1cn3)
Simian [35]	Papovaviridae	V	dD (C)	Ic (7D)	1(n)	X(3.1) (1sva)
Papillomavirus Initiation Complex [15]	Papovaviridae	V	dD (C)	Ic (7D)	1(n)	X(3.2) (1ksx)
Blue Tongue [16]	Reoviridae	V	dR (L)	Ic (1,13L)	2(n)	X(3.5) (2btv)
Rice dwarf [28]	Reoviridae	P	dR (L)	Ic (1,13L)	2(n)	X(3.5) (1uf2)
T1L reovirus virion [22]	Reoviridae	V	dR (L)	Ic (1,13L)	2(n)	X(2.8) (1jmu)
Simian rotavirus (SA11-4F) TLP [17]	Reoviridae	V	dR (L)	Ic (1,13L)	2(n)	X(2.38) (1lj2)
Rhesus rotavirus [13]	Reoviridae	V	dR (L)	Ic (1,13L)	2(n)	X(1.4) (1kqr)
Reovirus [44]	Reoviridae	V	dR (L)	Ic(1,13L)	2(n)	C(7.6)
Nudaurelia capensis w [18]	Tetraviridae	I	sR (L)	Ic (4)	1(n)	X(2.8) (1ohf)
Herpes Simplex [9]	Herpesviridae	V	dD (L)	Ic (7L)	1(E)	X(2.65) (1jma)
Chilo Iridescent [42]	Iridoviridae	I	dD (C)	Ic(147)	1(E)	C(13)
Paramecium Bursaria Chlorella [42]	Phycodnaviridae	P	dD (L)	Ic(169D)	1(E)	C(8)
HepBc (human liver) (nHBc) [41]	Hepadnaviridae	V	dD (C)	Ic(4)	1(E)	X(3.3) (1qgt)

Table 10.1: Helical and Icosahedral Viruses and Viral subunit structures: (1) Name and structure reference are given in square brackets (2) Family nomencleature from the ICTV database (3) Host types are P for Plant, V for Vertebrate, I for Invertebrate, F for Fungi (4) Virus Nucleic Acid (NA) type is single stranded RNA (sR) or DNA (sD), double stranded RNA (dR) or DNA (dD) and linear (L) or circular (C) (5) Capsid symmetry is Helix (He) or Icosahedral (Ic) with the triangulation number of each capsid shell in parenthesis (6) The number of capsid shells and whether enveloped (E) or not (n) (7) The acquisition modality X-ray, feature resolution and PDB id in parenthesis.

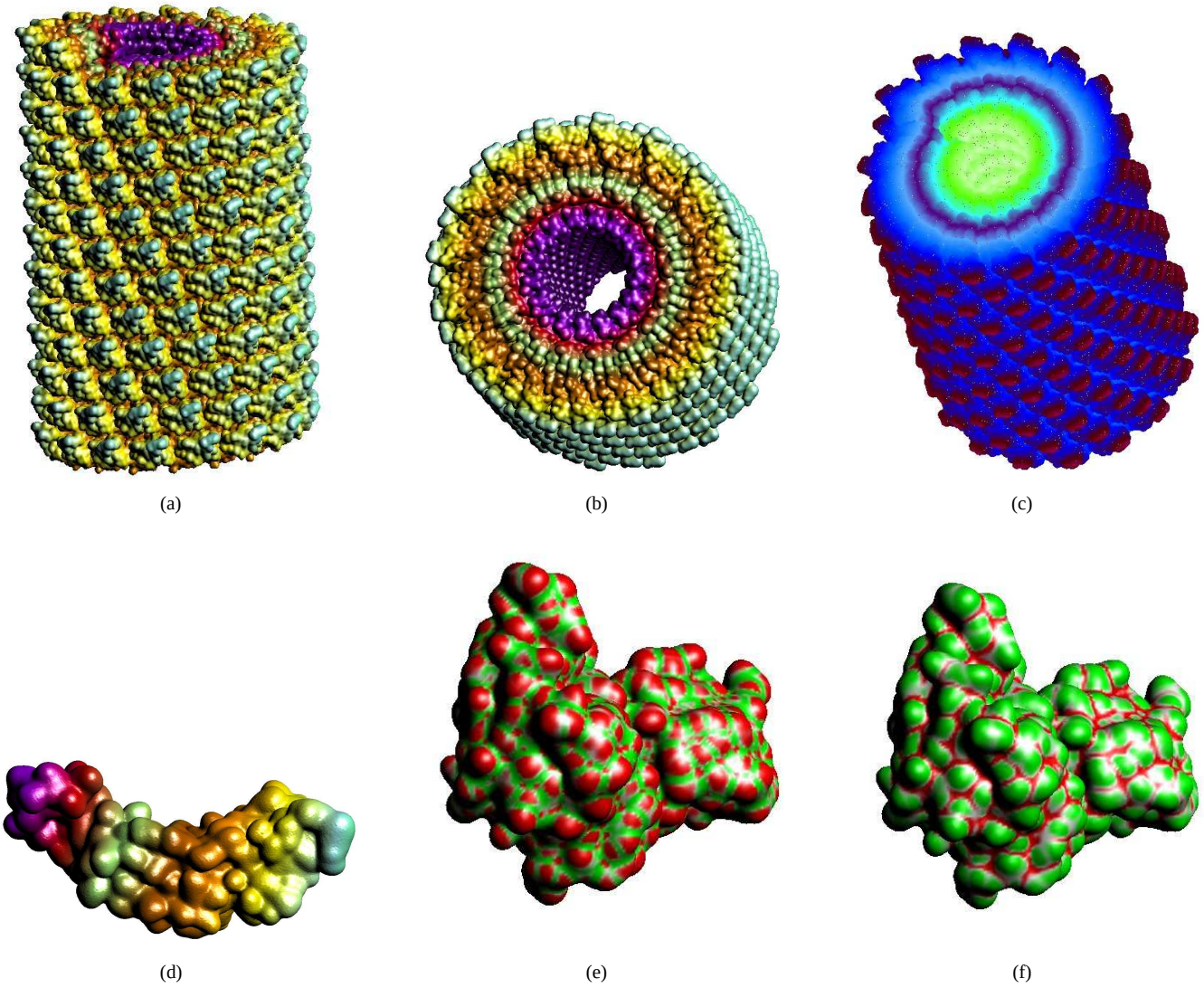


Figure 10.1: Organization of the Tobacco Mosaic Virus (1EI7) with its helical nucleo-capsid shown in (A), (B) and (C). (A) and (B) are surface rendered, while (C) is volume rendered. The asymmetric protomeric structure unit is visualized in (C) as an implicit solvation molecular surface colored by distance from the helix symmetry axis (D) with a transparent molecular surface and the protein backbone showing helix secondary structures. (E) molecular surface of protomer with the mean curvature function with red showing positive mean curvature and green with negative mean curvature (F) Gaussian curvature function on the protomer molecular surface, with green showing positive Gaussian curvature and the red signifying negative Gaussian curvature, displayed on the molecular surface.

The subsequent sub-sections dwell on the geometry of the individual protomers, and capsomers, as part of a hierarchical arrangement of symmetric capsid shells.

The Geometry of Helical Capsid Shells

Helical symmetry can be captured by a 4 x 4 matrix transformation $\mathbf{H}_{(\mathbf{a},\theta,L)}$ parameterized by $\mathbf{a} = (a_x, a_y, a_z)$, a unit vector along the helical axis, by θ, an angle in the plane of rotation, and by the pitch L, the axial rise for a complete circular turn.

If $\mathbf{P}$ is the center of any atom of the protomer, then $\mathbf{P}'$ is the transformed center, and $\mathbf{P}' = \mathbf{H} * \mathbf{P}$. Repeatedly applying this

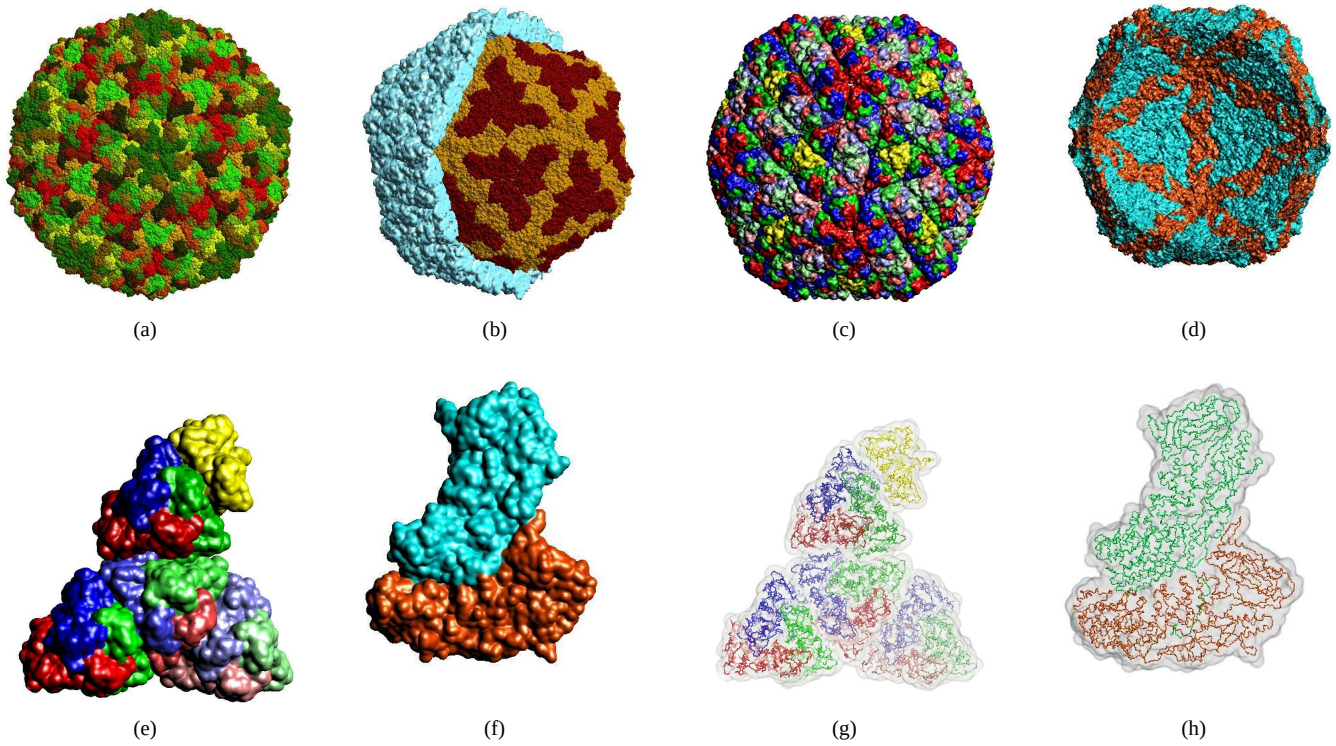


Figure 10.2: Organization of Rice Dwarf Virus (1UF2) with icosahedral capsid shells. (A) 2D texture based visualization of the outer capsid shell showing a single sphere per non-hydrogen atom, and colored to distinguish individual proteins subunits (B) the outer capsid shell shown as a smooth analytic molecular surface while the inner capsid surface is displayed using 2D texture maps of a union of spheres and colored (C) shows the outer capsid (D) displays the inner capsid (E) shows the icosahedral asymmetric structure unit of the outer unit (F) displays the icosahedral asymmetric structure unit of the inner unit (G) shows the protein backbone of the structure unit shown in (E) and (H) shows the protein backbone of the structure unit show in (F).

$$\mathbf{H}_{(\mathbf{a},\theta,L)} = \begin{vmatrix} a_x^2(1-\cos\theta)+\cos\theta & a_xa_y(1-\cos\theta)-a_z\sin\theta & a_xa_z(1-\cos\theta)+a_y\sin\theta & \frac{a_xL\theta}{2\pi} \\ a_xa_y(1-\cos\theta)-a_z\sin\theta & a_y^2(1-\cos\theta)+\cos\theta & a_ya_z(1-\cos\theta)-a_y\sin\theta & \frac{a_yL\theta}{2\pi} \\ a_xa_z(1-\cos\theta)+a_y\sin\theta & a_ya_z(1-\cos\theta)-a_x\sin\theta & a_z^2(1-\cos\theta)+\cos\theta & \frac{a_zL\theta}{2\pi} \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

transformation to all atoms in a protomer yields a helical stack of protomeric units. The desired length of the helical nucleocapsid shell is typically determined by the length of the enclosed nucleic acids. The capsid shell of the tobacco mosaic virus (TMV) exhibits helical symmetry (Fig. 10.1, and 10.3), with the asymmetric protein structure unit or the protomer consisting of a single protein (pdb id 1EI7)

The Geometry of Icosahedral Capsid Shells

In numerous cases the virus structure is icosahedrally symmetric. The advantage over the helical symmetry structure is the efficient construction of a capsid of a given size using the smallest protein subunits. An icosahedron has 12 vertices, 20 equilateral triangular faces, and 30 edges, and exhibits 5:3:2 symmetry. A 5-fold symmetry axis passes through each vertex, a 3-fold symmetry axis through the center of each face, and a 2-fold axis through the midpoint of each edge (see Fig. 10.4).

A rotation transformation around an axis $\mathbf{a} = (a_x, a_y, a_z)$ by an angle θ is described by the 4x4 matrix

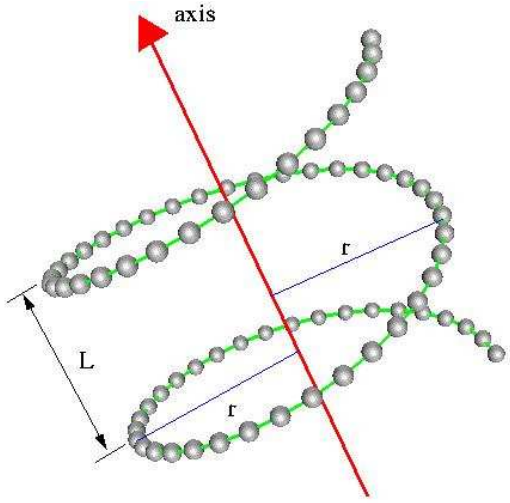


Figure 10.3: Helical Symmetry Axis.

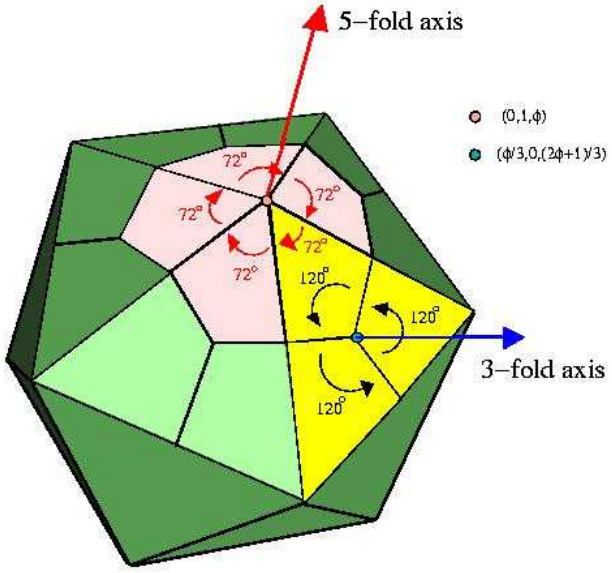


Figure 10.4: Icosahedral Transformations showing 5-fold and 3-fold Symmetry Axis.

$$\mathbf{R}_{(\mathbf{a},\phi)} = \begin{vmatrix} a_x^2(1-\cos\theta)+\cos\theta & a_x a_y(1-\cos\theta)-a_z\sin\theta & a_x a_z(1-\cos\theta)+a_y\sin\theta & 0 \\ a_x a_y(1-\cos\theta)-a_z\sin\theta & a_y^2(1-\cos\theta)+\cos\theta & a_y a_z(1-\cos\theta)-a_y\sin\theta & 0 \\ a_x a_z(1-\cos\theta)+a_y\sin\theta & a_y a_z(1-\cos\theta)-a_x\sin\theta & a_z^2(1-\cos\theta)+\cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

The vertices of a canonical icosahedron are given by $(0,\pm1,\pm\phi)$, $(\pm1,\pm\phi,0)$, $(0,\pm\phi,\pm1)$, where $\phi = (1+\sqrt{5})/2$ is the golden ratio. For a 5-fold symmetry transformation around the vertex $(0,\pm1,\pm\phi)$ the normalized axis of rotation is $\mathbf{a} = (0, 0.52573, 0.85064)$ and the angle of rotation is $\theta = \frac{2\pi}{5}$, yielding a five fold symmetry transformation matrix

Similarly, one is able to construct five fold symmetry transformation matrices for the other icosahedron vertices. Using the

$$\mathbf{R}_{(5-fold)} = \begin{vmatrix} 0.30902 & -0.80902 & 0.5000 & 0 \\ 0.80902 & 0.5000 & 0.30902 & 0 \\ -0.5000 & 0.30902 & 0.80902 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

generic rotational transformation matrix $\mathbf{R}_{(\mathbf{a},\phi)}$, one is able to construct the three fold transformation matrices via the rotation axis passes through the centroid of the triangular faces of the icosahedron and an angle of rotation of $\theta = \frac{2\pi}{5}$. Consider the triangular face with corners at $(0,1,\phi)$, $(1,\phi,0)$ and $(0,\phi,1)$. The centroid is at $\phi/3, 0, (2\phi+1)/3$ and the normalized axis of rotation is $\mathbf{a} = (0.356822, 0, 0.934172)$ and the transformation matrix

$$\mathbf{R}_{(5-fold)} = \begin{vmatrix} 0.30902 & -0.80902 & 0.5000 & 0 \\ 0.80902 & -0.5000 & -0.30902 & 0 \\ 0.5000 & 0.30902 & 0.80902 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

A polyhedron with faces that are all equilateral triangles is called a deltahedron. Deltahedra with icosahedral symmetry are classified as icosadeltahedra. Any icosadeltahedron has $20T$ facets, where T is the triangulation number given by $T = pf^2$, where, $P = h^2 + hk + k^2$ for all pairs of integers h and k which do not have a common factor, and f is any integer [Caspar and Klug 1962]. The possible values of P are $1, 3, 7, 13, 19, 21, 31, 37, \ldots$. In Fig. 10.5(A) we display triangles with different triangulation numbers, for icosahedral virus structures.

With a fixed size asymmetric unit the greater the T number, the larger the size of the virus capsid. Each triangular portion of the icosahedral virus capsid is easily subdivided into its three asymmetric units, with each unit containing some combination of protein structure units (protomers). In total an icosahedral virus capsid has 60T asymmetric units with numerous protein structures inter-twined to form a spherical mosaic. In Fig. 10.5 we see that when T=1, each vertex is at the center of a pentagon, and the capsid proteins are in equivalent environments, i.e. five neighbors cluster at a common vertex. However, for icosadeltahedra with larger triangulation numbers, e.g. $T = 13$, there are pentagons and hexagons in the capsid mosaic (Fig. 10.5). Therefore, even though the capsid proteins (protomers) may be chemically identical, some cluster into a 5-fold neighborhood and the others into a 6-fold neighborhood. Such locally symmetric clusterings of protomers are alternatively termed capsomers. In these situations, the proteins are no longer global symmetrically equivalent, but only quasi-equivalent [10].

10.1.2 Surface and Volumetric Modeling and Visualization

Atomistic Resolution Model Structures

Numerous schemes have been used to model and visualize bio-molecules and their properties [45, 2, 4, 20]. All these different visual representation are often derived from an underlying geometric model constructed from the positions of atoms, bonds, chains, and residues information deposited as part of an atomic resolution structure of the protein or nucleic acid in the Protein Data Bank (PDB). Hence, structural models are designed to represent the primary (sequence), secondary (e.g. α-helices, β-sheets), tertiary (eg. $\alpha - \beta$ barrels) sub-parts, and quaternary structures of the entire protein or nucleic acid.

An early approach to molecular modeling is to consider atoms as hard spheres, and their union as an attempt to capture shape properties as well as spatial occupancy of the molecule. This is similar to our perception of surfaces and volume occupancy of macroscopic objects. The top two pictures in Figure 10.2 shows hard-sphere model visualizations of the twin Rice Dwarf capsid shells, with individual proteins colored differently. Solvated versions of these molecular surfaces have been proposed by Lee - Richards, Connolly, et al. for use in computational biochemistry and biophysics. Much of the preliminary work, along with later extensions focused on finding fast methods of triangulating this molecular surface (or as sometimes referred to as the solvent contact surface). Two prominent obstacles in modeling are the correct handling of surface self-intersections (singularities) and the high communication bandwidth needed when sending tessellated surfaces to the graphics hardware.

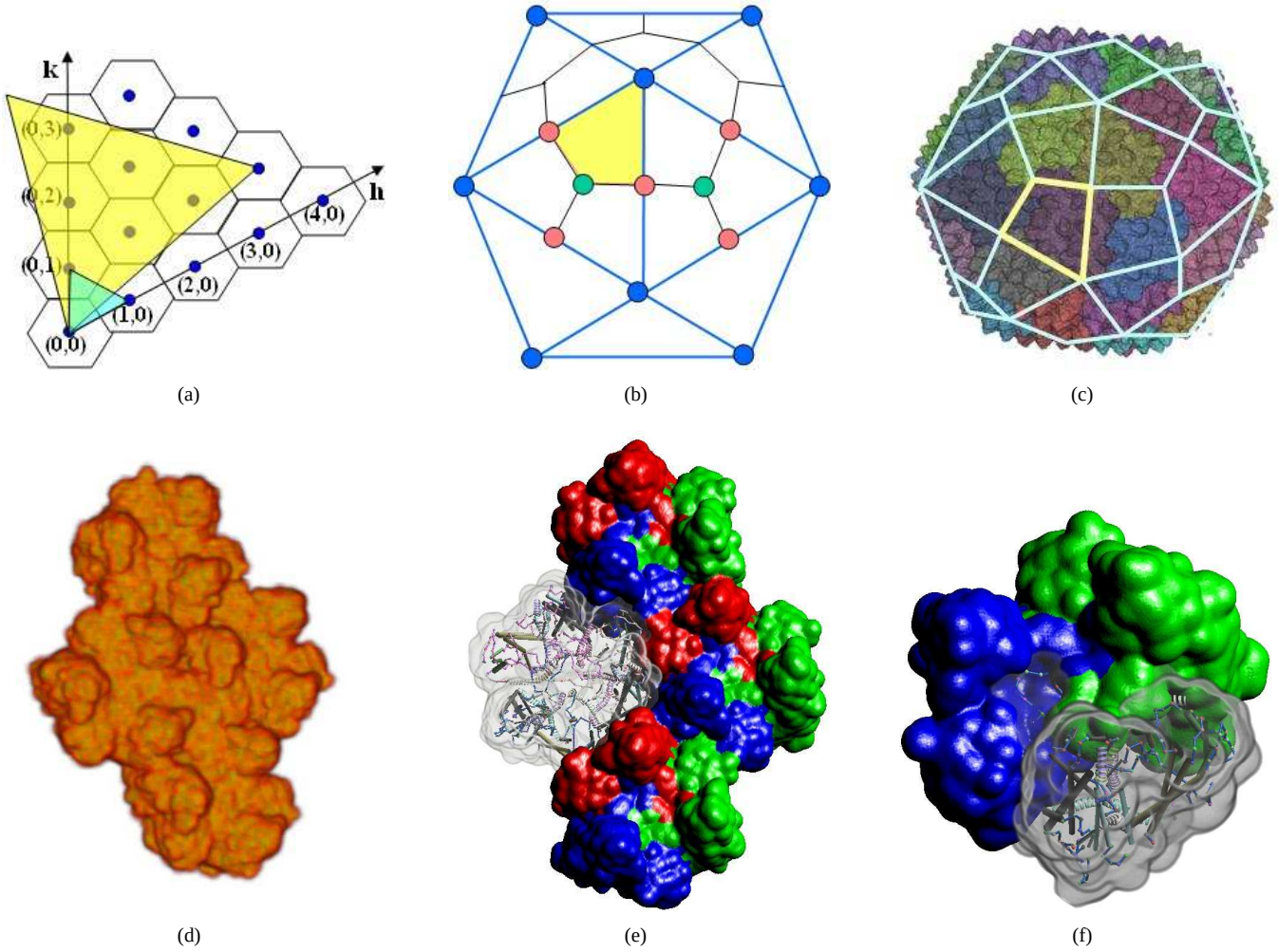
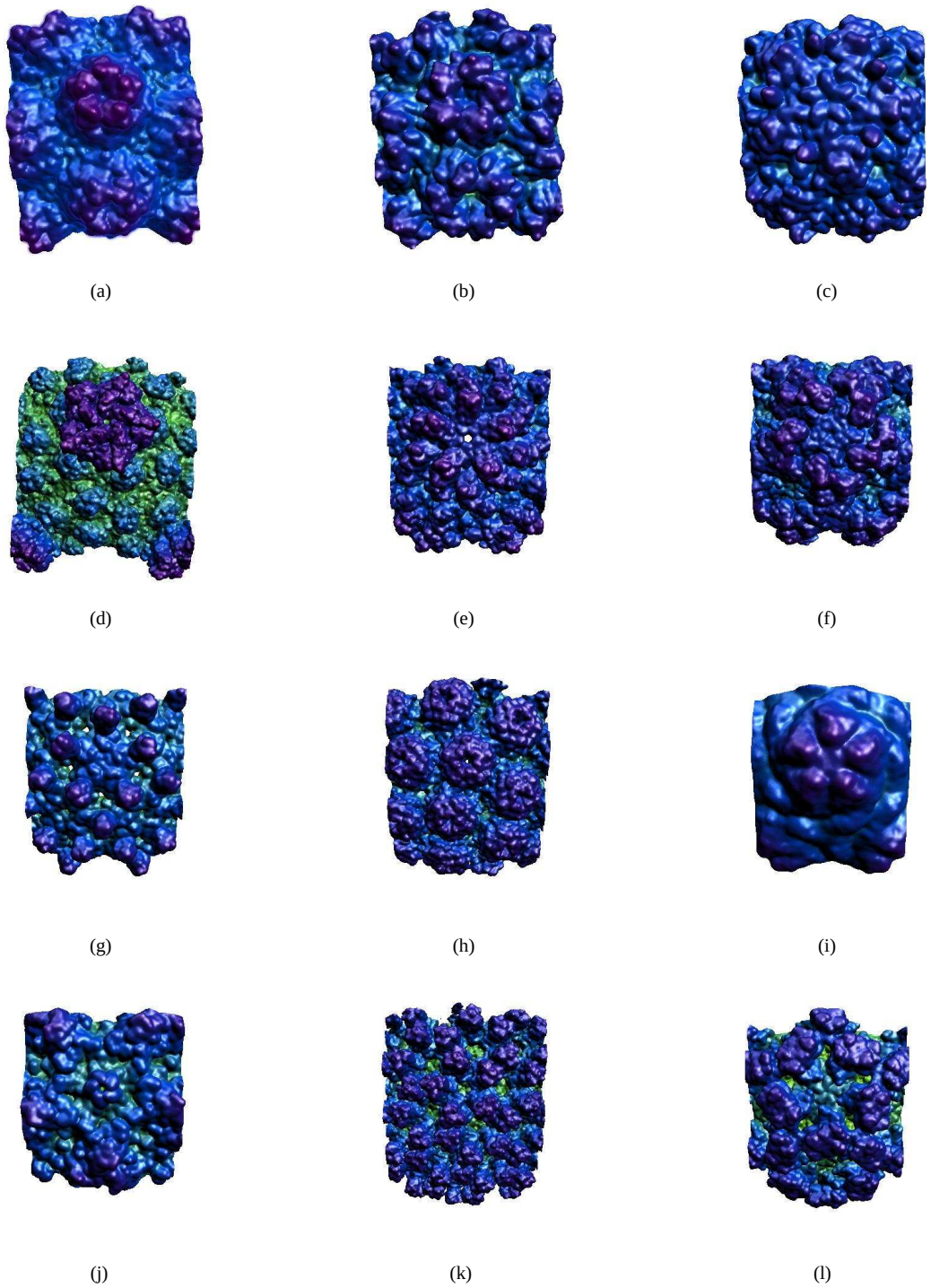


Figure 10.5: Architecture of Icosahedral Viruses: (A) Caspar-Klug Triangulation Number (T) via a hexagonal lattice. Green triangle has T =1 while yellow represents T = 13 (B) shows the asymmetric unit of an icosahedron, (C) asymmetric structure units of the capsid shell (D) a single asymmetric structure unit (E) asymmetric unit colored by protein as well as showing protein backbone. (F) a capsomere consisting of three proteins. (VIRUS PDB: 1GW8).

A more analytic and smooth description of molecular surfaces (without singularities) is provided by a suitable level set of the electron density representation of the molecule. Isotropic Gaussian kernels have been traditionally used to describe atomic electron density due to their ability to approximate electron orbitals. The electron density of a molecule with M atoms, centered at $\mathbf{x}_j, j \in 1,\ldots,M$, can thus be written as $F_{elec\_dens}(\mathbf{x}) = \sum_{j=1}^{M} \gamma_j K(\mathbf{x}-\mathbf{x}_j)$ where γ_j and K are typically chosen from a quadratic exponential description of atomic electron density

$$Atom(\mathbf{x}) = e^{-\frac{d}{r^2}((\mathbf{x}-\mathbf{y})^2-r^2)} = e^d e^{-\frac{d}{r^2}(\mathbf{x}-\mathbf{y})^2} = AK^q(\mathbf{x}-\mathbf{y})\gamma_{elec\_dens}(\mathbf{x}) = e^{-\frac{d}{r^2}((\mathbf{x}-\mathbf{x}_j)^2-r^2)} = \gamma_j K(\mathbf{x}-\mathbf{x}_j)$$

The atomic electron density kernels are affected by the radius r of individual atoms and the decay parameter d. Smooth and molecular surface models for individual proteins, structure units, as well as entire capsid shells can be easily constructed as a fixed level set of $F_{elec\_dens}(\mathbf{x}) = \sum_{j=1}^{M} \gamma_j K(\mathbf{x}-\mathbf{x}_j)$. An array of such structural molecular model visualizations are shown as Figures 10.1-10.5 as well as Figure 10.6. Some of them use transparency on the solvated molecular surface and show the protein back-bone structure (folded chains of α-helices and *beta*-sheets).



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figure 10.6: Portions of Capsid Shells of Icosahedral Viruses showing a significant portion of the capsid which properly includes the asymmetric subnit. Note the isosurface is selected to provide a good capsid surface approximation, while maintaining topological equivalence to a sphere. This makes the surface area and enclosed volume computation directly amenable to the calculations reported in the contour spectrum paper.

Structure Elucidation from 3D Maps

Electron Microscopy (EM) and in particular single particle reconstruction using cryo-EM, has rapidly advanced over recent years, such that many virus structures can be resolved routinely at low resolution (10-20 Å) and in some cases at sub-nanometer (intermediate) resolution (7-10 Å) [6, 7].

Symmetries within the virus capsid shells are exploited both in the 3D map reconstructions from raw 2D EM images, as well as in structure elucidation in the 3D map. In many cases, the 3D maps are of spherical viruses, with protein capsid shells exhibiting icosahedral symmetry. In these cases, the global symmetry detection can be simplified to computing the location of the 5-fold rotational symmetry axes, passing through the twelve vertices of the icosahedron, from which the 3-fold symmetry axes for the twenty icosahedron faces and the 2-fold symmetry axes for the thirty icosahedron edges can be easily derived. However determining the local symmetries of the capsomers (structure units) is more complicated, as they exhibit varied k-fold symmetry, and their detection requires a modified correlation based search algorithm [43]. Volumetric segmentation methods are additionally utilized to partition, color and thereby obtain a clearer view into the macromolecules architectural organization. Furthermore, electronically dissecting the local structure units from a 3D Map allows for further structural interpretation (tertiary and secondary folds). Visualizations from the afore-mentioned local symmetry detection and automatic segmentation, applied to a 3D volumetric Map of the Turnip Yellow Mosaic virus (pdbid 1AUY), are shown in Figure 10.7.

10.1.3 Quantitative Visualization

The geometric modeling of virus capsids and the individual virus structure units, can be further augmented by the computation of several global and local shape metrics [5]. While integral, topological and combinatorial metrics capture global shape properties, differential measures such as mean and Gaussian curvatures have also proved useful to an enhanced understanding and quantitative visualization of macromolecular structures.

Integral Properties

Integral shape metrics include the area of the molecular capsid surface defining the capsid, the volume enclosed by closed capsid shells, and the gradient integral on the molecular capsid surface. Given our smooth analytic level set definition of the molecular surface from Section 10.1.2, $F_{elec\_dens}(\mathbf{x}) = \sum_{j=1}^{M} \gamma_j K(\mathbf{x} - \mathbf{x}_j) = const$, for all the atoms that make up either an individual structure unit, or the entire virus capsid, an efficient and accurate integration computation for these metrics is given by the contour spectrum [[3]. The surface integrations can be performed by adaptively sampling the capsid surface using a technique known as contouring [3]. Contouring is often performed by first decomposing (meshing) the space surrounding the capsid surface into either a rectilinear Cartesian grid mesh, a tetrahedral or a hexahedral mesh. For a tetrahedral mesh, the surface area for the portion of the level set inside a tetrahedron can be represented by a quadratic polynomial B-spline [3]. Summing these B-splines over all of the tetrahedra containing the capsid surface yields the capsid surface area. The volume enclosed by a closed capsid surface is determined by the definite integration of the surface area polynomial B-splines.

In Figure 10.8 we display the results of surface area and volume calculations, and a regression relationship between the two, for a selection of spherical icosahedral capsids for virus structures summarized in Table 10.2. The analytic molecular surfaces were first computed, and then surface area and enclosed volume were estimated through B-spline evaluation as stated above.

Differential Properties

The gradient function of our smooth analytic capsid surface is simply $\bigtriangledown F_{elec\_dens}(\mathbf{x}) = \sum_{j=1}^{M} \gamma_j \bigtriangledown K(\mathbf{x} - \mathbf{x}_j)$, the summation of the vector of first derivatives of the atomic electron density function. This gradient function is non-zero everywhere on the virus capsid surface (i.e. no singularity). The second derivatives of the molecular surface capture additional differential shape properties and provide suitable metrics. Popular metrics are the magnitudes of *Mean Curvature H* and the *Gaussian curvature G*. These are given directly as $H = \frac{1}{2}(k_{min} + k_{max})$ and $G = k_{min}k_{max}$, and are respectively the average and the product of the twin principal curvatures, namely, k_{min} and k_{max}, also sometimes known as the minimum and maximum curvatures at a point

Virus	Surface Area	Volume	ln(Surface Area)	ln(Volume)
Satellite tobacco necrosis	17401.22	24419.51	9.76	10.1
L-A (Saccharomyces cerevisiae)	99643.6	155223.15	11.51	11.95
Canine parvovirus-Fab complex	48482.09	66028.46	10.79	11.1
T1L reovirus core	412654.67	517093.8	12.93	13.16
T3D reovirus core	99627.14	161424.33	11.51	11.99
P4 (Ustilago maydis)	7362.92	11269.59	8.9	9.33
Tomato Bushy Stunt	69600.33	98169.33	11.15	11.49
Cowpea Chlorotic Mosaic	42523.74	56607.48	10.66	10.94
Cucumber Mosaic	43317.17	61885.43	10.68	11.03
Norwalk	116674.31	170940.17	11.67	12.05
Rabbit hemorrhagic disease VLP-MAb-E3 complex	80585.54	121611.94	11.3	11.71
Galleria mellonella densovirus	33251.61	46216.49	10.41	10.74
Human Rhino	67337.7	99964.3	11.12	11.51
HepBc (human liver) (nHBc)	41669.23	65963.21	10.64	11.1
Nudaurelia capensis w	170957.88	278225.27	12.05	12.54
Semiliki Forest	47586.6	68392.18	10.77	11.13
Polyoma	104897.53	171532.31	11.56	12.05
Simian	177557.44	246603.02	12.09	12.42
Herpes Simplex Virus Glyco-Protein	29035.25	43098.51	10.28	10.67
Blue Tongue	590265.25	711692.59	13.29	13.48
Rice Dwarf	727228.58	820906.09	13.5	13.62
T1L reovirus virion	412654.67	517093.8	12.36	12.81
Simian rotavirus (SA11-4F) TLP	26451.69	35311.17	10.18	10.47
Rhesus rotavirus	15093.03	23469.18	9.62	10.06

Table 10.2: Icosahedral Viruses and Viral Components: Area (units are square Angstrom), Volume (units are cube Angstrom) and Logarithm entries displayed below are showing pictorially in Figure 10.8

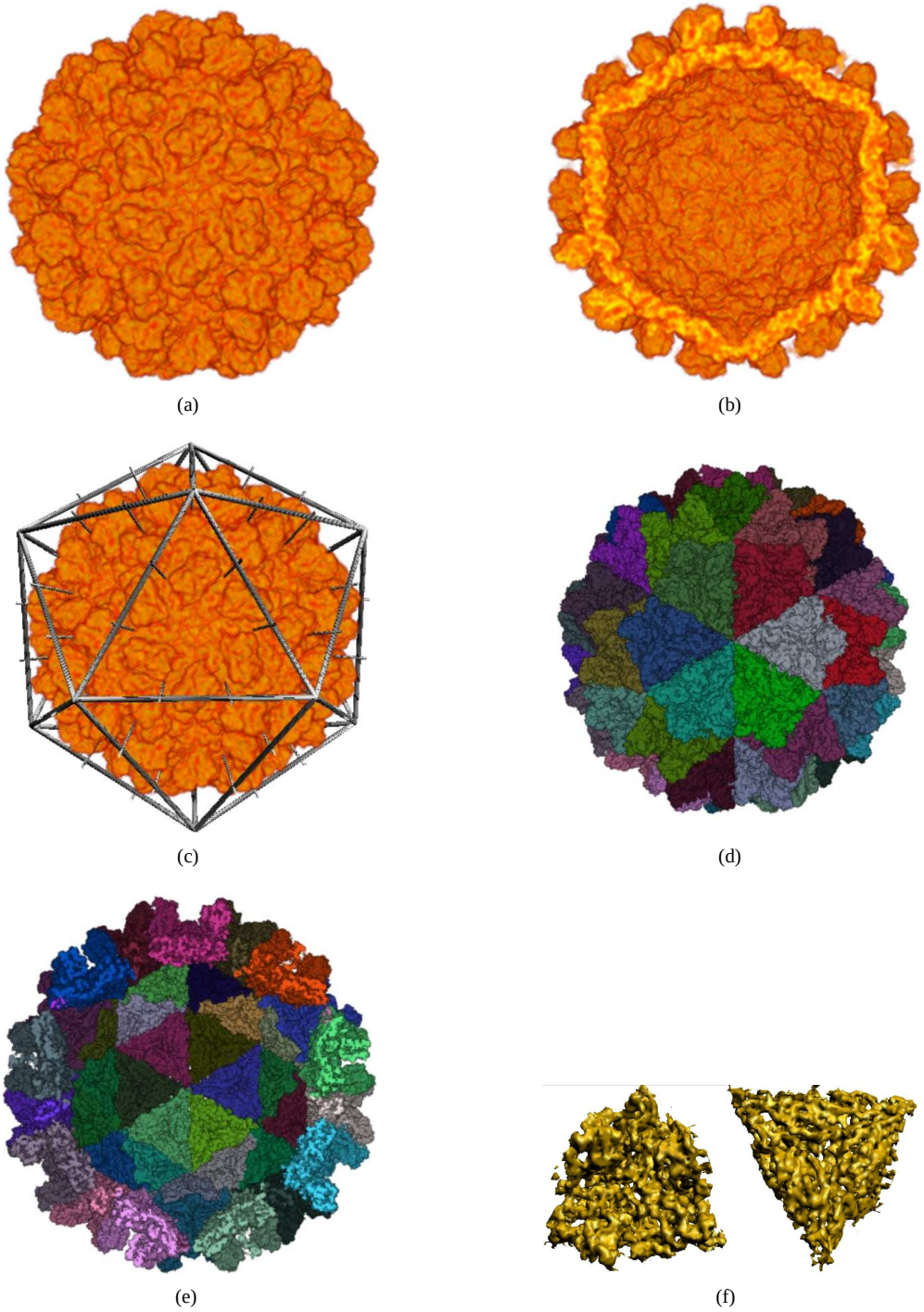
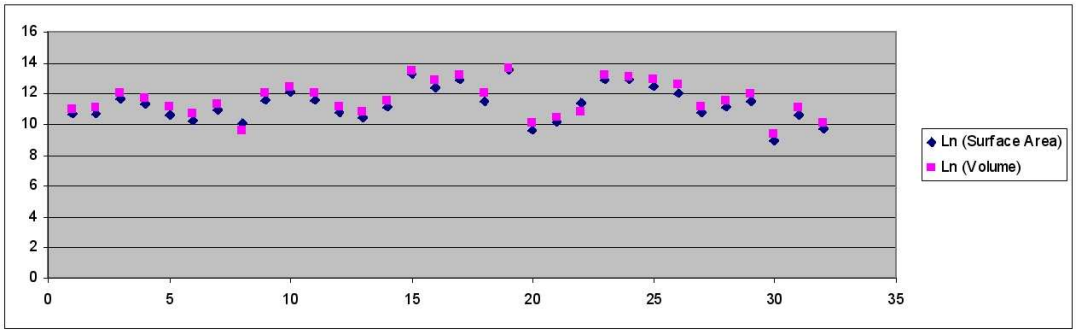
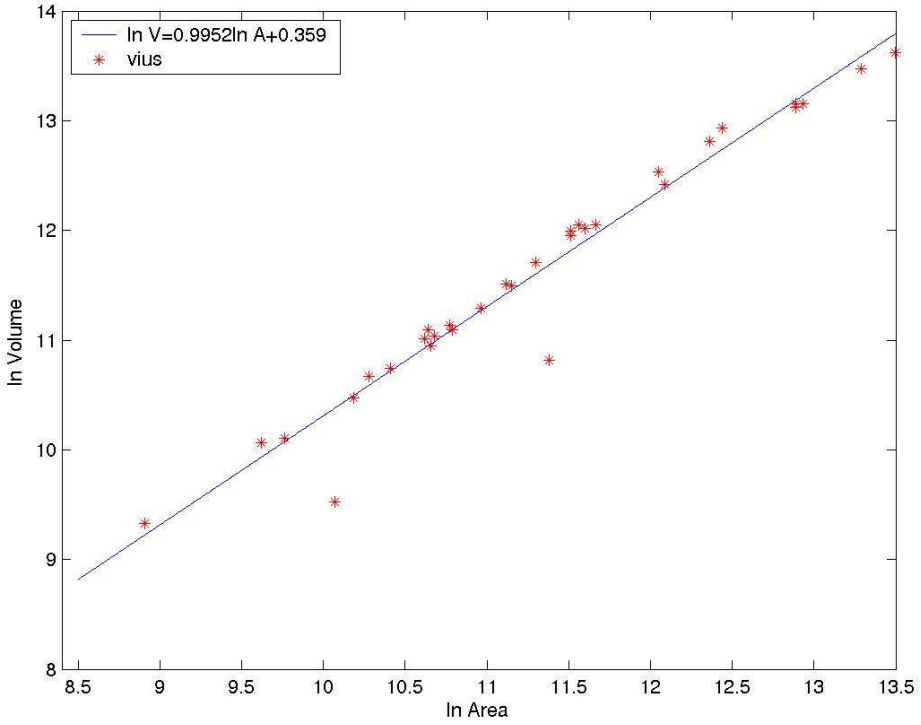


Figure 10.7: (PDB-ID = 1AUY. Size: $256\hat{3}$. Resolution: 4Å). (A) Gaussian blurred map (outside view) from the non-hydrogen atom locations given in the PDB. (B) Gaussian blurred map (inside view). (C) Symmetry detected, including global and local 3-fold symmetry axes. (D) Segmented trimers (outside view), with randomly assigned colors. (E) Segmented trimers (inside view). (F) One of the segmented trimers (left-bottom: outside view; right-top: inside view).



(a)



(b)

Figure 10.8: (Area, Volume Relationship for Icosahedral Viruses given in Table 1. The area and volume units are Square Angstrom and Cube Angstrom respectively.

on the surface. Again for our level set based analytic molecular surface $F_{elec\_dens}(\mathbf{x}) = const = f$, the twin curvatures H and K can be evaluated as $H = \frac{\sum f_x^2(f_{yy}+f_{zz})-2\sum f_x f_y f_{xy}}{2(\sum f_x^2)^{1.5}}$ and $G = \frac{2\sum f_x f_y(f_{xz}f_{yz}-f_{xy}f_{zz})}{(\sum f_x^2)^2}$ where $\sum$ represents a cyclic summation over x, y and z, and where additionally f_x, etc., denotes partial differentiation with respect to those variables.

Displaying the magnitude of the gradient function and its variation, as expressed by the mean and Gaussian curvature functions over a molecular surface helps quantitatively visualize the bumpiness or lack thereof of an individual protomer, a structure unit or the entire viral capsid. In Figure 10.1 the bottom two pictures display the mean and Gaussian curvature functions of the Tobacco Mosaic virus asymmetric protomer surface, exhibiting and enhancing the bumpiness of the surface.

Topological and Combinatorial Properties

Affine invariant topological structures of volumetric functions, such as our smooth analytic electron density function of Section 10.1.2, include the Morse complex [14, 26] and the contour tree (*CT*) [37]. Both the Morse complex and contour tree are related to the critical points of the volumetric function f, i.e., those points in the domain M where the function gradient vanishes. The functional range of f is the interval between the minimum and maximum values of the function $f : [f_{min}, f_{max}]$. For a scalar value $w \in [f_{min}, f_{max}]$, the level set of the field f at the value w is the subset of points $L(w) \subset M$ such that $f(x) = w, \forall_x \in L(w)$.

A level set may have several connected components, called contours. The topology of the level set $L(w)$ changes only at the critical points in M, whose corresponding functional values are called critical values. A contour class is a maximal set of continuous contours which have the same topology and do not contain critical points. Without loss of generality, the critical points are assumed to be non-degenerate, i.e. only isolated critical points. This assumption can be enforced by small perturbations of the function values. If the critical points are non-degenerate, then the Hessian $H(a)$ at a critical point a has non-zero real eigenvalues. The index of the critical point a is the number of negative eigenvalues of $H(a)$. For a 3D volumetric function, there are four types of critical points: index 0 (minima), indices 1 and 2 (saddle points), and index 3 (maxima).

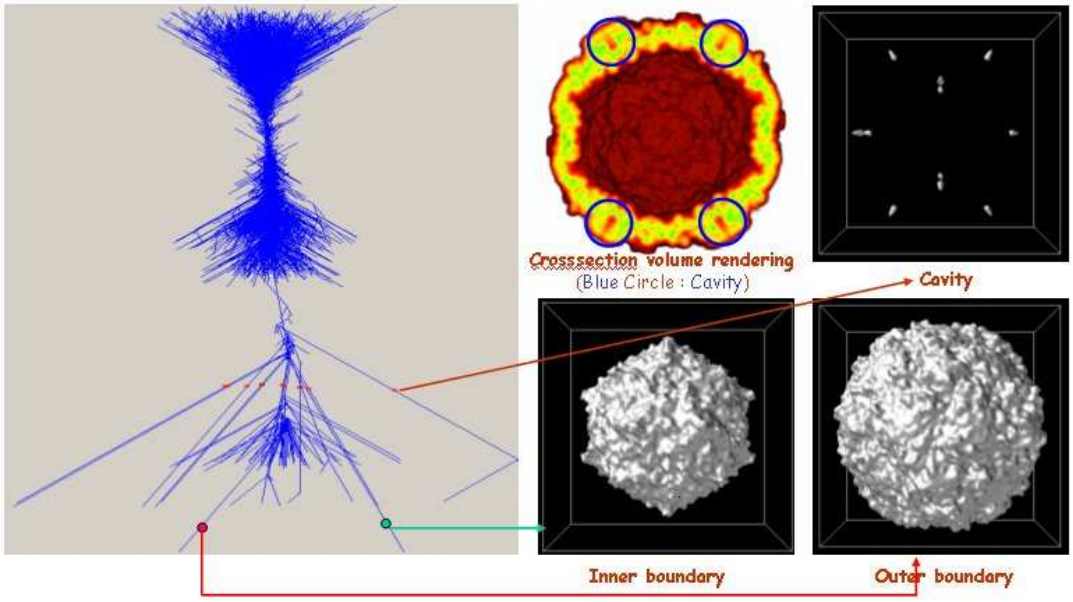


Figure 10.9: (The contour tree (upper left) and the contour spectrum (bottom) for the Human Rhinovirus serotype 2 (pdbid: 1 FPN).The red color in the spectrum curve is the graph of molecular surface area, while the blue and green curves are the excluded and enclosed volume by the various level surfaces of the volumetric density map. The horizontal axis of the plot above is map density, while the vertical axis is spectrum function value.

The contour tree (*CT*) was introduced by Kreveld et al. [37] to find the connected components of level sets for contour generation. The *CT* captures the topological changes of the level sets for the entire functional range $[f_{min}, f_{max}]$ of f; each node of the tree corresponds to a critical point and each arc corresponds to a contour class connecting two critical points. As an example, the contour tree for a virus capsid is shown in Fig. 10.9. Each leaf node of the *CT* represents the creation or deletion of a component at a local minimum or maximum and each interior node represents the joining and/or splitting of two or more components or topology changes at the saddle points. A cut on an arc of the tree (v_1, v_2) Î T by an isovalue $v_1 \leq w \leq v_2$ represents a contour of the level set $L(w)$. Therefore, the number of connected components for the level set $L(w)$ is equal to the number of cuts to the *CT* at the value w. The *CT* can be enhanced by tagging arcs with topological information such as the Betti numbers of the corresponding contour classes [37]. Betti numbers β_k, $(k = 0, 1, \ldots)$ intuitively measure the number of k-dimensional holes of a virus capsid surface or of any individual structure unit. Only the first three Betti numbers $\beta_0, \beta_1, \beta_2$ of a smooth surface are non-zero: β_0 corresponds to the number of connected components; β_1 corresponds to the number of independent tunnels; β_2 represents the number of voids enclosed by the surface. For example, a sphere has the Betti numbers

$(\beta_0, \beta_1, \beta_2) = (1, 0, 1)$ while a torus has $(\beta_0, \beta_1, \beta_2) = (1, 2, 1)$. Betti number computations for virus capsid surfaces provide useful topological and combinatorial structural information.

10.2 Ribosome

Related Work

Bibliography

[1] ICTV database. http://www.ncbi.nlm.nih.gov/ICTVdb/Ictv/index.htm.

[2] C. Bajaj, P. Djeu, V. Siddavanahalli, and A. Thane. Texmol: Interactive visual exploration of large flexible multi-component molecular complexes. In *Proceedings of the conference on Visualization'04*, page 250. IEEE Computer Society, 2004.

[3] C. Bajaj, V. Pascucci, and D. Schikore. The contour spectrum. In *Proceedings of the 8th conference on Visualization'97*. IEEE Computer Society Press, 1997.

[4] C. Bajaj, V. Pascucci, A. Shamir, R. Holt, and A. Netravali. Dynamic maintenance and visualization of molecular surfaces* 1. *Discrete Applied Mathematics*, 127(1):23–51, 2003.

[5] C. Bajaj and Z. Yu. Geometric processing of reconstructed 3d maps of molecular complexes. In S. Aluru, editor, *Handbook of Computational Molecular Biology*, Computer and Information Science Series. Chapman and Hall, CRC Press, 2005.

[6] T. Baker, N. Olson, and S. Fuller. Adding the third dimension to virus life cycles: three-dimensional reconstruction of icosahedral viruses from cryo-electron micrographs. *Microbiology and Molecular Biology Reviews*, 63(4):862–922, 1999.

[7] D. Belnap, A. Kumar, J. Folk, T. Smith, and T. Baker. Low-Resolution Density Maps from Atomic Models: How Stepping" Back" Can Be a Step" Forward". *Journal of structural biology*, 125:166–175, 1999.

[8] B. Bhyravbhatla, S. Watowich, and D. Caspar. Refined atomic model of the four-layer aggregate of the Tobacco Mosaic Virus coat protein at 2.4- Å resolution. *Biophysical journal*, 74(1):604–615, 1998.

[9] A. Carfí, S. Willis, J. Whitbeck, C. Krummenacher, G. Cohen, R. Eisenberg, and D. Wiley. Herpes simplex virus glycoprotein D bound to the human receptor HveA. *Molecular cell*, 8(1):169–179, 2001.

[10] D. Caspar and A. Klug. Physical principles in the construction of regular viruses. In *Cold Spring Harbor Symposia on Quantitative Biology*, volume 27, pages 1–24. Cold Spring Harbor Laboratory Press, 1962.

[11] X. Chen, T. Stehle, and S. Harrison. Interaction of polyomavirus internal protein VP2 with the major capsid protein VP1 and implications for participation of VP2 in viral entry. *The EMBO journal*, 17(12):3233–3240, 1998.

[12] A. Dar and F. Sicheri. X-ray crystal structure and functional analysis of vaccinia virus K3L reveals molecular determinants for PKR subversion and substrate recognition. *Molecular cell*, 10(2):295–305, 2002.

[13] P. Dormitzer, Z. Sun, G. Wagner, and S. Harrison. The rhesus rotavirus VP4 sialic acid binding domain has a galectin fold with a novel carbohydrate binding site. *The EMBO Journal*, 21(5):885–897, 2002.

[14] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical MorseÑ Smale Complexes for Piecewise Linear 2-Manifolds. *Discrete and Computational Geometry*, 30(1):87–107, 2003.

[15] E. Enemark, A. Stenlund, and L. Joshua-Tor. Crystal structures of two intermediates in the assembly of the papillomavirus replication initiation complex. *The EMBO Journal*, 21(6):1487–1496, 2002.

[16] J. Grimes, J. Jakana, M. Ghosh, A. Basak, P. Roy, W. Chiu, D. Stuart, and B. Prasad. An atomic model of the outer layer of the bluetongue virus core derived from X-ray crystallography and electron cryomicroscopy. *Structure*, 5(7):885–893, 1997.

[17] C. Groft and S. Burley. Recognition of eIF4G by rotavirus NSP3 reveals a basis for mRNA circularization. *Molecular cell*, 9(6):1273–1283, 2002.

[18] C. Helgstrand, S. Munshi, J. Johnson, and L. Liljas. The refined structure of Nudaurelia capensis [omega] Virus reveals control elements for a T= 4 capsid maturation. *Virology*, 318(1):192–203, 2004.

[19] P. Hopper, S. Harrison, and R. Sauer. Structure of tomato bushy stunt virus. V. Coat protein sequence determination and its structural implications. *Journal of molecular biology*, 177(4):701–713, 1984.

[20] B. Kang, Y. Devedjiev, U. Derewenda, and Z. Derewenda. The PDZ2 domain of syntenin at ultra- high resolution: bridging the gap between macromolecular and small molecule crystallography. *Journal of molecular biology*, 338(3):483–493, 2004.

[21] N. Li, M. Erman, W. Pangborn, W. Duax, C. Park, J. Bruenn, and D. Ghosh. PROTEIN CHEMISTRY AND STRUCTURE- Structure of Ustilago maydis killer toxin KP6 a-subunit. A multimeric assembly with a central pore. *Journal of Biological Chemistry*, 274(29):20425–20431, 1999.

[22] S. Liemann, K. Chandran, T. Baker, M. Nibert, and S. Harrison. Structure of the reovirus membrane-penetration protein,[mu] 1, in a complex with its protector protein,[sigma] 3. *Cell*, 108(2):283–295, 2002.

[23] L. Liljas and B. Strandberg. The structure of satellite tobacco necrosis virus. *Biological macromolecules and assemblies (USA)*, 1984.

[24] E. Mancini, M. Clarke, B. Gowen, T. Rutten, and S. Fuller. Cryo-electron microscopy reveals the functional organization

of an enveloped virus, Semliki Forest virus. *Molecular Cell*, 5(2):255–266, 2000.

[25] M. Mavrakis, A. McCarthy, S. Roche, D. Blondel, and R. Ruigrok. Structure and function of the C-terminal domain of the polymerase cofactor of rabies virus. *Journal of molecular biology*, 343(4):819–831, 2004.

[26] J. Milnor. *Morse theory*. Princeton Univ Pr, 1963.

[27] H. Naitow, J. Tang, M. Canady, R. Wickner, and J. Johnson. LA virus at 3.4 Å resolution reveals particle architecture and mRNA decapping mechanism. *Nature Structural & Molecular Biology*, 9(10):725–728, 2002.

[28] A. Nakagawa, N. Miyazaki, J. Taka, H. Naitow, A. Ogawa, Z. Fujimoto, H. Mizuno, T. Higashi, Y. Watanabe, T. Omura, et al. The atomic structure of rice dwarf virus reveals the self-assembly mechanism of component proteins. *Structure*, 11(10):1227–1238, 2003.

[29] K. Namba, R. Pattanayek, and G. Stubbs. Visualization of protein- nucleic acid interactions in a virus* 1:: Refined structure of intact tobacco mosaic virus at 2.9 Å resolution by X-ray fiber diffraction. *Journal of molecular biology*, 208(2):307–325, 1989.

[30] K. Ng, M. Cherney, A. Vázquez, Á. Machı?n, J. Alonso, F. Parra, and M. James. Crystal structures of active and inactive conformations of a caliciviral RNA-dependent RNA polymerase. *Journal of Biological Chemistry*, 277(2):1381, 2002.

[31] B. Prasad, M. Hardy, T. Dokland, J. Bella, M. Rossmann, and M. Estes. X-ray crystallographic structure of the Norwalk virus capsid. *Science*, 286(5438):287–290, 1999.

[32] K. Reinisch, M. Nibert, and S. Harrison. Structure of the reovirus core at 3.6 Å resolution. *Nature*, 404(6781):960–967, 2000.

[33] A. Simpson, P. Chipman, T. Baker, P. Tijssen, and M. Rossmann. The structure of an insect parvovirus (Galleria mellonella densovirus) at 3.7 Å resolution. *Structure*, 6(11):1355–1367, 1998.

[34] T. Smith, E. Chase, T. Schmidt, and K. Perry. The structure of cucumber mosaic virus and comparison to cowpea chlorotic mottle virus. *Journal of Virology*, 74(16):7578–7586, 2000.

[35] T. Stehle, S. Gamblin, Y. Yan, and S. Harrison. The structure of simian virus 40 refined at 3.1 Å resolution. *Structure*, 4(2):165–182, 1996.

[36] Y. Tao, D. Farsetta, M. Nibert, and S. Harrison. RNA Synthesis in a Cage–Structural Studies of Reovirus Polymerase [lambda] 3. *Cell*, 111(5):733–745, 2002.

[37] M. Van Kreveld, R. Van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In *Proceedings of the thirteenth annual symposium on Computational geometry*, page 220. ACM, 1997.

[38] W. Weissenhorn, A. Carfí, K. Lee, J. Skehel, and D. Wiley. Crystal structure of the Ebola virus membrane fusion subunit, GP2, from the envelope glycoprotein ectodomain. *Molecular Cell*, 2(5):605–616, 1998.

[39] D. White and F. Fenner. Herpesviridae. *Medical virology, 4th ed. Academic Press, San Diego, CA*, 1994.

[40] H. Wu and M. Rossmann. The canine parvovirus empty capsid structure. *Journal of molecular biology*, 233(2):231– 244, 1993.

[41] S. Wynne, R. Crowther, and A. Leslie. The crystal structure of the human hepatitis B virus capsid. *Molecular Cell*, 3(6):771–780, 1999.

[42] X. Yan, N. Olson, J. Van Etten, M. Bergoin, M. Rossmann, and T. Baker. Structure and assembly of large lipid-containing dsDNA viruses. *Nature Structural & Molecular Biology*, 7(2):101–103, 2000.

[43] Z. Yu and C. Bajaj. Automatic ultra-structure segmentation of reconstructed cryo-em maps of icosahedral viruses. *IEEE Transactions on Image Processing: Special Issue on Molecular and Cellular Bioimaging*, 14(9):1324–1337, 2005.

[44] X. Zhang, S. Walker, P. Chipman, M. Nibert, and T. Baker. Reovirus polymerase $\lambda 3$ localized by cryo-electron microscopy of virions at a resolution of 7.6 Å. *Nature Structural & Molecular Biology*, 10(12):1011–1018, 2003.

[45] Y. Zhang, G. Xu, and C. Bajaj. Quality meshing of implicit solvation models of biomolecular structures. *Computer aided geometric design*, 23(6):510–530, 2006.

Chapter 11

Conclusions

Something to conclude

Related Work

Appendix A

Molecules

A.1 Internal Coordinates

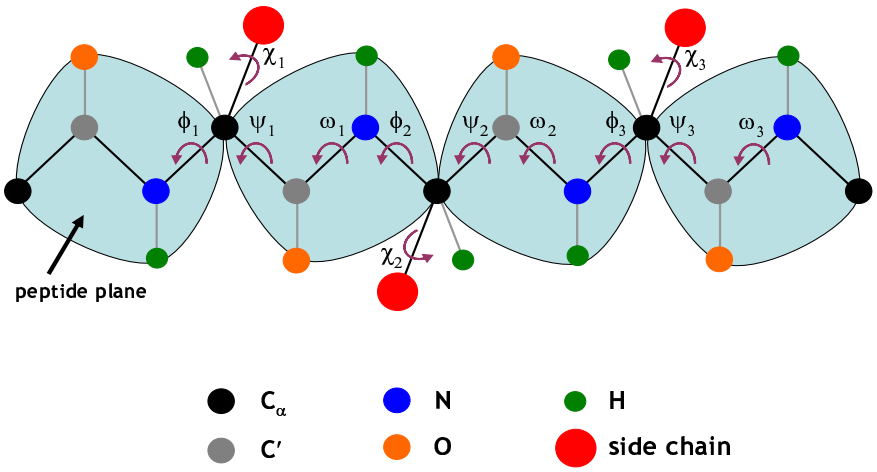


Figure A.1: A polypeptide chain with backbone dihedrals (ψ, ϕ, ω) and side-chain dihedrals (χ) shown.

Proteins have a naturally occurring backbone consisting of $-NH-C(H)R-CO-$ sequences, where R is some functional group defined for 20 different amino acids. These functional groups appear as side-chains connected to the backbone. As in all organic molecules, each type of bond formed in a protein conforms to the characteristic bond length and bond angles for that type. Hence, the conformation of a protein can be approximately defined by a set of *dihedral angles* (or *torsional angles*) that determine the orientation of different chemical groups along and around the backbone.

The following three dihedral angles determine the conformation of the backbone (see Figure A.1).

ϕ_i. This is the angle between the planes $C_{i-1}-N_i-C_{\alpha_i}$ and $N_i-C_{\alpha_i}-C'_i$, i.e., the angle of rotation ($-180^\circ \leq \phi_i < +180^\circ$) around the $N_i-C_{\alpha_i}$ bond. A positive change in the ϕ_i value occurs by counter-clockwise rotation of the $C_{i-1}-N_i-C_{\alpha_i}$ plane around the $N_i-C_{\alpha_i}$ bond.

ψ_i. This is the angle of rotation ($-180^\circ \leq \psi_i < +180^\circ$) around the $C_{\alpha_i}-C'i$ bond, and is determined by the angle between the $N_i-C_{\alpha_i}-C'i$ and $C_{\alpha_i}-C'i-N_{i+1}$ planes.

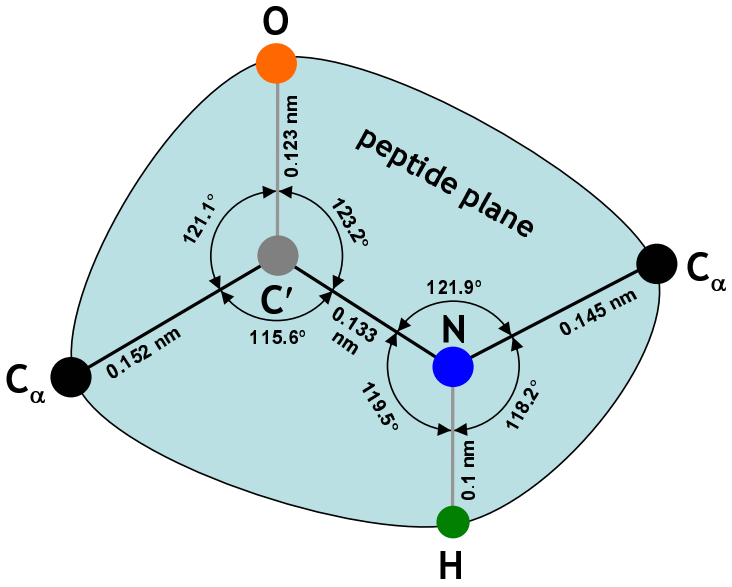


Figure A.2: A peptide plane with all bond lengths and bond angles shown [9].

ω_i. This is the angle of rotation around the peptide bond ($C'_{i-1} - N_i$), and is given by the dihedral angle between the $C_{\alpha_{i-1}} - C'_{i-1} - N_i$ and $C'_{i-1} - N_i - C_{\alpha_i}$ planes. The partial (40 %) double-bond character of the peptide bond and the steric interactions between adjacent side-chains causes the amide group (N_i, C_{α_i}, H_i, C'_{i-1}, O_{i-1} and $C_{\alpha_{i-1}}$) to be almost planar with the distance between $C_{\alpha_{i-1}}$ and C_{α_i} as large as possible (see Figure A.1 for bond lengths and bond angles on this plane). Therefore, almost always $\omega_i \approx 180^\circ$ (for *trans*-peptides), or $\omega_i \approx 0^\circ$ (for *cis*-peptides).

More than than 99.9% of all residues (except proline) are *trans*-peptides, and hence have $\omega_i \approx 180^\circ$. Approximately 5% of all proline peptide bonds have $\omega_i \approx 0^\circ$.

The side chains change conformation through torsional changes in the χ_i angles.

χ_i. Depending on the amino acid type of the side chain there can be up to 4 such successive angles per side chain: $\chi_{i,1}$, $\chi_{i,2}$, $\chi_{i,3}$ and $\chi_{i,4}$. However, for *Glycine* side chain which consists of only one hydrogen atom, and *Alanine* whose side chain is only a single methyl group, these angles are undefined. For all other side chains $\chi_{i,1}$ is defined as the dihedral angle between the planes $N - C_\alpha - C_\beta$ and $C_\alpha - C_\beta - X$, where X is either C_γ, or C_{γ_1} (Val, Ile), O_γ (Ser), O_{γ_1} (Thr), or S_γ (Cys). All side chain dihedrals have values clustered near three conformers known as *gauche*$^+$ or g^+ ($+60^\circ$), *trans* or *t* (180°), and *gauche*$^-$ or g^- (-60°).

Figure A.3 shows the side-chain dihedrals of all amino acids except Glycine and Alanine. Table A.1 shows that about 90% of all side-chains in proteins can be completely described with three dihedral angles (i.e., $\chi_{1,1}$, $\chi_{1,2}$ and $\chi_{1,3}$), and only two dihedral angles (i.e., $\chi_{1,1}$ and $\chi_{1,2}$) are necessary to completely specify more than two-thirds of them.

number of dihedrals (d)	frequency (%)
$d \leq 4$	100.00
$d \leq 3$	89.48
$d \leq 2$	70.64
$d \leq 1$	23.46

Table A.1: Amino acid frequencies in proteins based on the number of (side-chain) dihedrals they have (based on data in [26]).

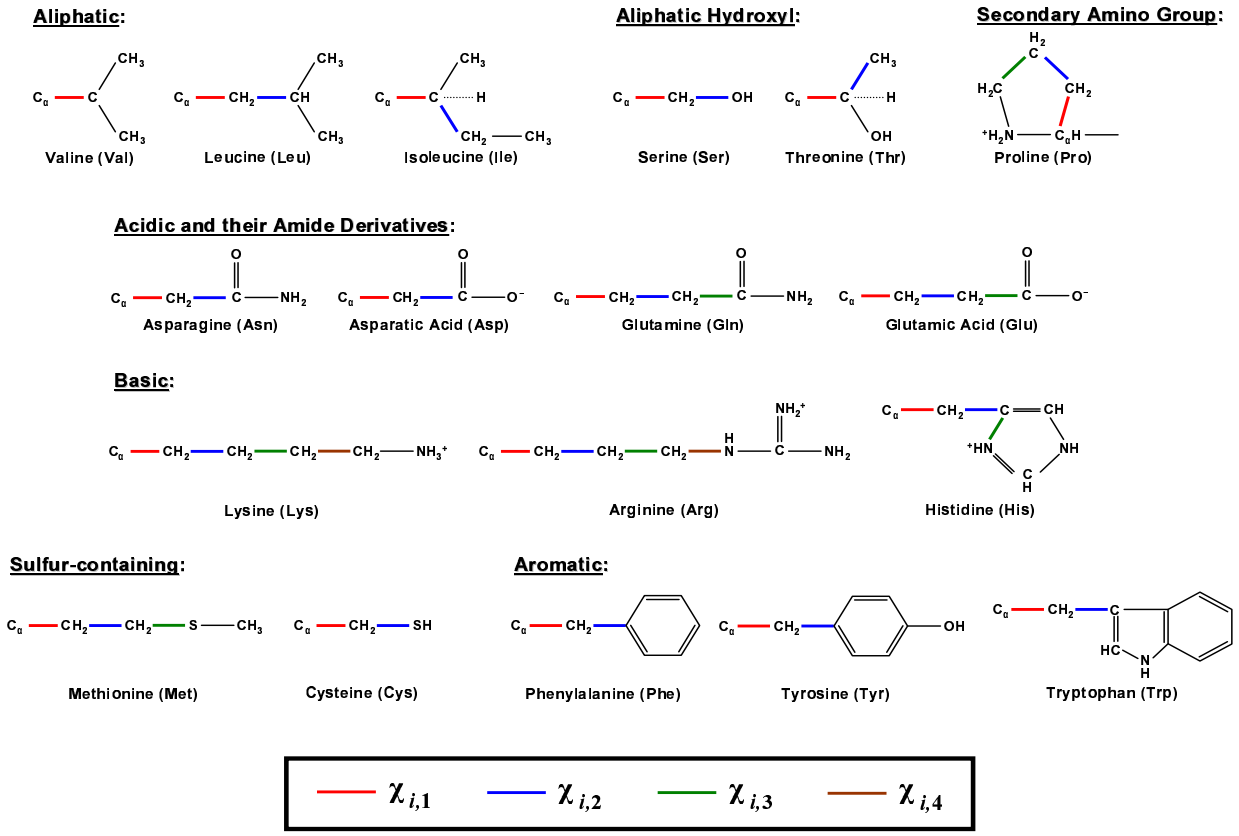


Figure A.3: Side-chain dihedrals ($\chi_{i,1}$, $\chi_{i,2}$, $\chi_{i,3}$, $\chi_{i,4}$) are shown for 18 of the 20 amino acids. The remaining two, i.e., Glycine (Gly) and Alanine (Ala), do not have any side-chain dihedrals. Adapted from [28].

A.2 LEG (Labelled Embedded Graph) Representations

The **LEG** representation of a molecule is simply an annotated graph representation of the chemical structure of the molecule, in which each node represents an atom and each edge a chemical bond. Each atom may be annotated by its symbol and the **vdW** radius, each edge may be annotated by the length of the corresponding chemical bond and possibly a dihedral angle, and each pair of consecutive edges by a bond angle.

In Figure A.3 we show the chemical structures of various amino acids, and in Tables A.2, A.3 and A.4 we list all possible **vdW** radii, bond lengths and bond angles, respectively, that appear in these chemical structures. Using these information, it is straight-forward to construct the required **LEG** representations of the amino acids.

Since secondary structures (e.g., α-helices and β-sheets) are composed of primary structures (i.e., amino acids), the **LEG** representation of secondary structures can also be constructed from the information in Figure A.3 and Tables A.2, A.3 and A.4. However, the (ϕ, ψ) dihedral angles of the residues in α-helices and β-sheets lie in fairly restricted ranges: $(-45°, -60°)$ for α-helices, $(-120°, 115°)$ for parallel β-sheets, and about $(-140°, 135°)$ for anti-parallel β-sheets. The bond lengths and bond angles may also change slightly.

We can use geometric properties of α-helices and β-sheets in order to extract them from the **LEG** representation L of the given protein P.

Extracting α-helices from L. We traverse L along the peptide backbone of P, and using the internal coordinates (i.e., bond lengths, bond angles, dihedral angles, etc.), bond types and atom types specified in L, we detect and output all maximal contiguous segments of this backbone (along with side chains) that satisfy the following properties of α-helices.

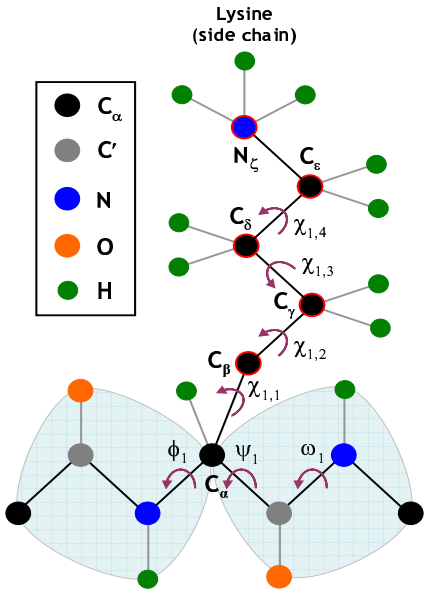


Figure A.4: A Lysine side-chain with side-chain dihedrals ($\chi_{1,1}$, $\chi_{1,2}$, $\chi_{1,3}$, $\chi_{1,4}$).

- The amino acids in an α-helix are arranged in a right-handed helical structure with each amino acid corresponding to a 100° turn in the helix and a 1.5 Å translation along the helical axis. Thus there are 13 atoms and 3.6 amino acid residues per turn, and each turn is 5.4 Å wide (see Figure A.5).
- The *C*=*O* group of residue i forms a hydrogen bond with the *N*-*H* group of residue $i+4$.
- Amino acid residues in an α-helix typically have dihedral angles $\phi \approx -45^\circ$ and $\psi \approx -60^\circ$.

Extracting β-sheets from L. We scan the peptide backbone of P given in L, and detect and output all maximal contiguous segments of this backbone (along with side chains) that satisfy the following properties of β-sheets.

- Each β-strand can be viewed as a helical structure with two residues per turn. The distance between two such consecutive residues is 3.47 Å in anti-parallel β-sheets and 3.25 Å in parallel β-sheets.
- Unlike α-helices the *C*=*O* groups in the backbone of a β-strand form hydrogen bonds with the *N*-*H* groups in the backbone of adjacent strands.
  - In parallel β-sheets all *N*-termini of adjacent strands are oriented in the same direction (see Figure A.7(b)). If the C_α atoms of residues i and j of two different strands are adjacent, they do not hydrogen bond to each other, rather rasidue i may form hydrogen bonds to residues $j-1$ or $j+1$ of the other strand.
  - In anti-parallel β-sheets the *N*-terminus of one strand is adjacent to the *C*-terminus of the next strand (see Figure A.7(a)). If a pair of C_α atoms from two successive β-strands are adjacent, then unlike in parallel β-sheets they form hydrogen bonds to each other's flanking peptide groups.
- The (ϕ, ψ) dihedrals are about $(-120^\circ, 115^\circ)$ in parallel β-sheets, and about $(-140^\circ, 135^\circ)$ in anti-parallel β-sheets.
- Unlike in α-helices, peptide carbonyl groups in successive residues point in alternating directions.

m or Group	Symbol	R_{vdW} (Å)	Notes
>CHR	CA	1.90	Main-chain α-carbon (excluding α-carbon of Gly)
>C=O	C	1.75	Main-chain carbonyl carbon
>CH—	CH	2.01	Side-chain aliphatic carbon with one hydrogen (C^{β} of Ile, C^{γ} of Leu, C^{β} of Thr, C^{β} of Val)
			Side-chain aliphatic carbon with two hydrogens, except those at β-position and those next to a charged group (C^{γ} of Arg, $C^{\gamma 1}$ of Ile, C^{γ} and C^{δ} of Lys, C^{γ} of Met, C^{γ} and C^{δ}

Table A.2: List of van der Waals radii for 25 protein atoms [21].

tom type	Description
	Carbonyl C atom of the peptide back
5W	Tryptophan C^{γ}
W	Tryptophan $C^{\delta 2}$, $C^{\epsilon 2}$
F	Phenylalanine C^{γ}
Y	Tyrosine C^{γ}
Y2	Tyrosine C^{ζ}
5	Histidine C^{γ}
N	Neutral carboxylic acid group C ato
H1E	Tetrahedral C atom with one H atom
H2E	Tetrahedral C atom with two H atom CH2G)
H2P	Proline C^{γ}, C^{δ}
H2G	Glycine C^{α}
H3E	Tetrahedral C atom with three H ato
R1E	Aromatic ring C atom with one H at CR1W, CRH, CRHH, CR1H)
R1W	Tryptophan $C^{\zeta 2}$, $C^{\eta 2}$
RH	Neutral histidine $C^{\epsilon 1}$
RHH	Charged histidine $C^{\epsilon 1}$

ond type	Bond length (Å)	Bond type	Bo
5W–CW	1·433	CH1E–CH1E	
W–CW	1·409	CH1E–CH2E	
–CH1E	1·525	CH1E–CH3E	
5–CH2E	1·497	CH1E–N	
5W–CH2E	1·498	CH1E–NH1	
F–CH2E	1·502	CH1E–NH3	
Y–CH2E	1·512	CH1E–OH1	
–CH2E	1·516	CH2E–CH2E	
N–CH2E	1·503	CH2P–CH2E	
–CH2G	1·516	CH2P–CH2P	
5W–CR1E	1·365	CH2E–CH3E	
W–CR1E	1·398	CH2P–N	
W–CR1W	1·394	CH2G–NH1	
F–CR1E	1·384	CH2E–NH1	
Y–CR1E	1·389	CH3E–NH1	

Table A.3: Bond lengths in proteins [7].

5W–CW–CR1E	133·9	CH3E–C
W–CW–CR1E	118·8	C–CH2E
W–CW–CR1W	122·4	C5–CH2
W–C5W–CR1E	106·3	CF–CH2
W–CW–NH1	107·4	C5W–CE
H1E–C–N	116·9	CY–CH2
H1E–C–NH1	116·2	C–CH2E
H1E–C–O	120·8	C–CH2C
H1E–C–OC	117·0	C–CH2C
H2E–C5–CR1E	129·1	CH1E–C
H2E–C5–CR1H	131·2	CH1E–C
H2E–CF–CR1E	120·7	CH1E–C
H2E–C5W–CR1E	126·9	CH1E–C
H2E–CY–CR1E	120·8	CH1E–C
H2E–C–N	118·2	CH1E–C
H2G–C–N	118·2	CH1E–C
H2E–C5–NH1	122·7	CH2E–C
H2E–C–NH1	116·5	CH2E–C
H2G–C–NH1	116·4	CH2P–C
H2E–C–NH2	116·4	CH2E–C
H2E–C5–NR	121·6	CH2E–C
H2E–C–O	120·8	CH2E–C
H2G–C–O	120·8	CY2–CE
H2E–C–OC	118·4	CW–CR
H2G–C–OC	118·4	CW–CR
R1E–CY2–CR1E	120·3	CF–CR1
R1E–CY–CR1E	118·1	CY–CR
R1E–CF–CR1E	118·6	C5–CR1
R1W–CW–NH1	130·1	C5–CR1
R1E–C5–NH1	105·2	C5W–CE
R1H–C5–NH1	106·1	C5–CR1
R1E–CY2–OH1	119·9	CR1E–C
–C–O	122·0	CR1W–C
C2–C–NC2	119·7	CR1E–C
C2–C–NH1	120·0	NH1–CE
H1–C–O	123·0	NH1–CE
H2–C–O	122·6	C–N–CE

Table A.4: Bond angles in proteins [7].

A.3 FCC (Flexible Chain Complex) Representations

Complex biomolecules have a naturally occurring backbone, forming chains which flex through their torsion angles. This *nerve* is biochemically well defined, and described by a labeled complex. Structural (shape) and functional properties of a biomolecule can be described as a labeled *sheath* around the central *nerve*. This combined representation (Flexible Chain Complex, or FCC) of a *nerve* and a *sheath* describe a flexible biomolecule.

The nerve of the FCC. The chain complex consists of the following elements.

- *Vertices*: Atom or pseudo atom positions. Atom positions are obtained typically from the PDB files. For pseudo atoms, we use the centers of a set of enclosing spheres which represent the finer level using some error norm like the Hausdorff error.

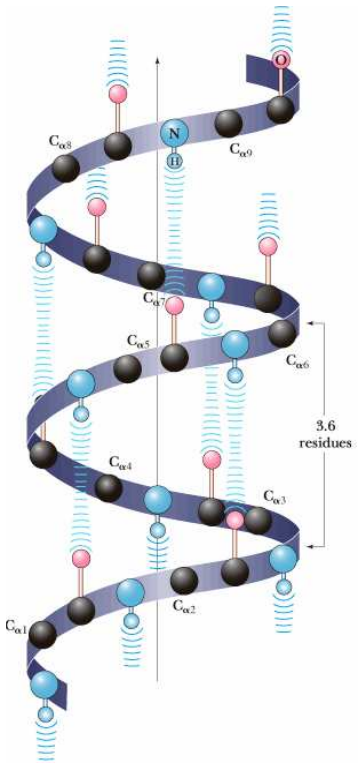


Figure A.5: Geometric structure of an α-helix [9].

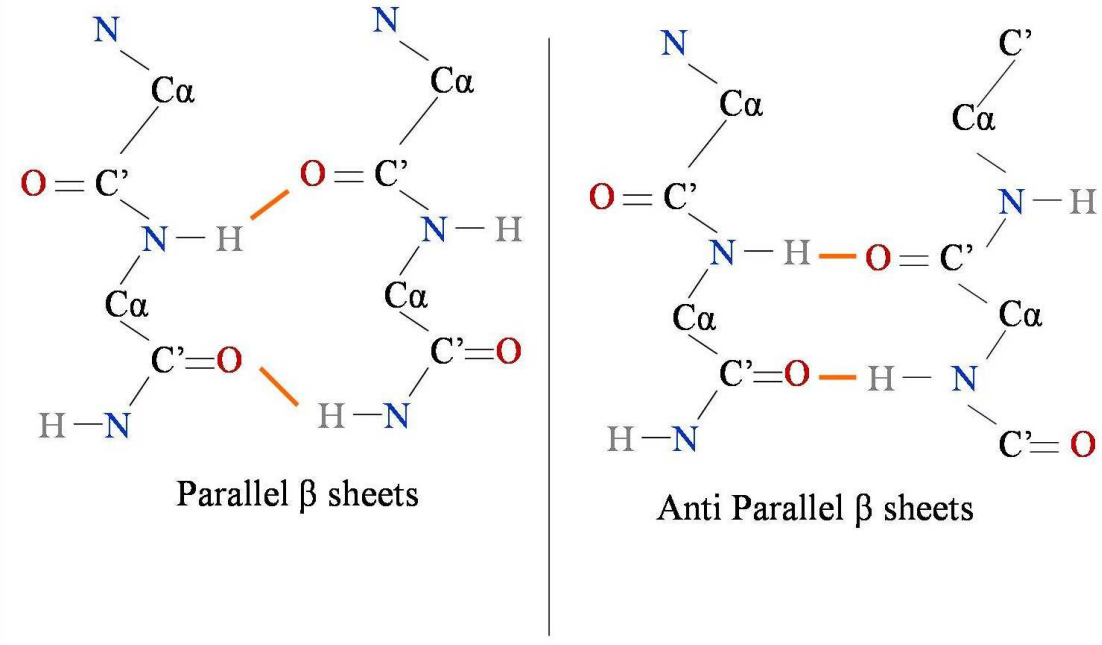


Figure A.6: Geometric structure of a β-sheet [9].

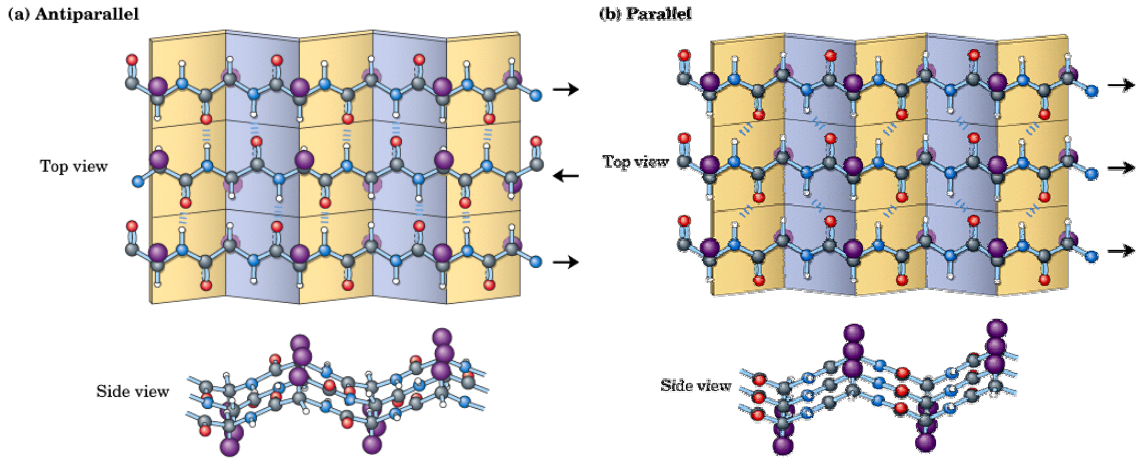


Figure A.7: Two types of β-sheets: (a) anti-parallel, and (b) parallel [13].

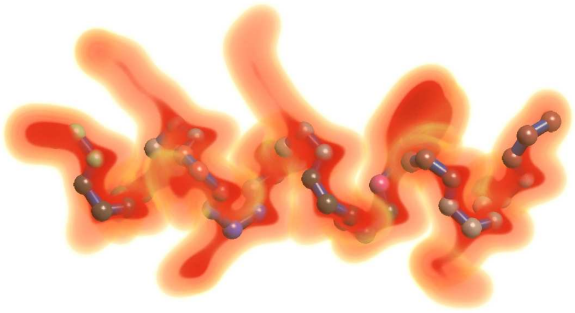


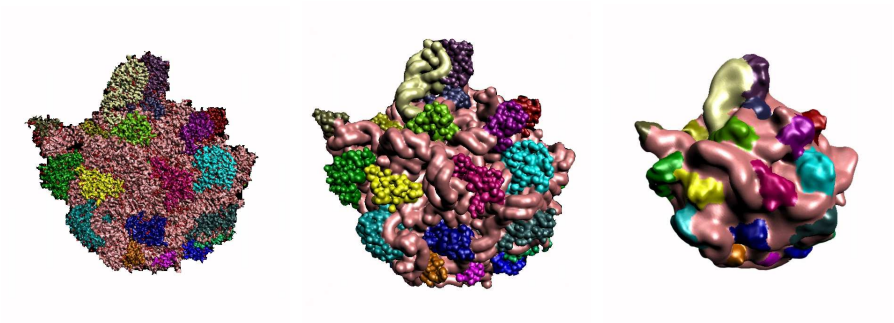
Figure A.8: Flexible Chain Complex: Combined volume (through hardware accelerated 3D texture mapping based volume rendering) and imposter rendering, showing the chain together with the high density volumetric regions formed by the functional groups protruding outwards from the chain.

- *Edges*: Bonds or pseudo bonds. This is again from the PDB or from the hierarchical complex formed by clustering the finer resolutions to a DAG.
- *Faces*: Residues, bases or pseudo structures.

These elements are labeled with the following attributes.

- Position, length, areas.
- Ranges for flexible angles, lengths.
- Sub structural markers.
- Field attributes.

We allow the molecules to flex around their torsion angles as it is widely accepted that bond angles and bond lengths do not have much flexibility. In protein chains, the ϕ and ψ angle variations are obtained and stored in the complex attributes. For RNA, we have 8 different torsion angles along the backbone. The ranges for these atoms are obtained either from molecular dynamics simulations or from NMR analysis for certain structures.



(a) Atom level (b) Residue level with a low $G_{dropoff}$ factor (c) Residue level with a high $G_{dropoff}$ factor

Figure A.9: LOD volume rendering of a large ribosomal subunit (1JJ2.pdb). The parameter $G_{dropoff}$ controls the spread of the density around a pseudo atom when blurring the chain complex

The sheath of the FCC. The surrounding volume, sub volumes and surfaces of a biomolecule are used to represent shape, volumetric properties (like electrostatics, hydrophobicity) and surface properties (like curvatures). These representations enjoy a dual implicit and explicit representation.

- *Implicit volumetric representation* In this representation, we have a vector containing of (a). A set of centers of expansion points, (b) A parameter referred to as the blobbiness parameter which is useful to represent the van der Waals forces in a continuous and hierarchical fashion, and (c), a set of radii. These parameters are necessary and sufficient to define the electron density function of a molecule. For functions like hydrophobicity and electrostatics, charges at each center of expansion is required.

- *Explicit volumetric representation* There are three representations which can be used for explicitly describing a volumetric function.

  - *Simplicial representation*: The data is described over a simplex like a surface grid at the vertices.
  - *Tensor product*: An explicit grid is used to represent the functions. The size of such a representation can be very large. Hence it is useful to develop compression based algorithms to represent and visualize such a representation.
  - *Multipole summations*: Since our data set consists of a set of vertices and functions which are summations of functions defined over this limited set, Multi-Pole type summations can be used efficiently to represent the data sets.

A.3.1 Hierarchical Representation

Both the skeletal and the volumetric features are represented in a hierarchical fashion. We have a biochemical based static hierarchy of the molecules, with atoms at the finest resolution. Groups of atoms are collapsed to form residues and residues form secondary structures. Chains consist of a set of these secondary structures. A dynamic hierarchy, which could be more useful for interactive dynamic level of detail rendering and manipulation is also performed as outlined in [1].

Once a flexible chain complex hierarchy is rebuilt due to dynamic changes in the molecule, the implicitly defined volumetric and surface properties can be quickly updated. Explicit volumes can also be extracted in a hierarchical fashion.

When we have a hierarchical representation of a FCC skeleton, we implicitly have a hierarchical representation of the surrounding differentiable sheath. In figure A.9, we show the large ribosomal subunit at three different levels of a hierarchy.

A.3.2 Flexibility representation

The paper [38] describes how to store the flexibility information in a structure. More specifically, they describe existing and new methods to obtain new atom positions when rotations are performed. Three schemes for storing and manipulating rotation matrices are given below.

Simple rotations scheme. A tree is constructed from the molecule by taking any atom as the root, and bonds in the molecule as bonds in the tree. Rings in a protein are simply taken as a single atom. When a torsional angle changes at a node, then all the nodes below it are rotated to new positions. This rotation update involves a matrix multiplication. The update has to be from the node to the leaves and numerical errors can occur due to manipulating positions of atoms down a chain for each rotation.

Consider a bond b_i rotated by angle θ_i. Let **v** be a vector along the bond and T be the translation matrix formed by the i^{th} atoms position. Then the update matrix is

$$T \begin{pmatrix} v_x^2+(1-v_x^2)cos\theta_i & v_xv_y(1-cos\theta_i)+v_zsin\theta_i & v_zv_x(1-cos\theta_i)+v_ysin\theta_i & 0 \\ v_xv_y(1-cos\theta_i)+v_zsin\theta_i & v_y^2+(1-v_y^2)cos\theta_i & v_yv_z(1-cos\theta_i)-v_xsin\theta_i & 0 \\ v_zv_x(1-cos\theta_i)-v_ysin\theta_i & v_yv_z(1-cos\theta_i)+v_xsin\theta_i & v_z^2+(1-v_z^2)cos\theta_i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} T^{-1} \tag{3.1}$$

A.3.3 Denavit-Hartenberg scheme

In this scheme, we again maintain a tree, with matrices and update from a root to the leaf. But now, the matrices no longer need the information on the current position of the atom, but only the rotations it underwent as a single matrix. Hence this is numerically stable.

To construct the matrix, we first define a local frame at each node. The origin and the vectors are the node position and

- **w** the bond from the node to its parent
- **u** a vector perpendicular to the previous vector and the bond containing this atom and a child. This means that a frame is to be defined for each child.
- **v** a vector perpendicular to the above two.

The matrix which takes a point from one frame defined at a node to the frame of the parent of that node is defined as

$$\begin{pmatrix} cos\theta_i & -sin\theta_i & 0 & 0 \\ sin\theta_i cos\phi_{i-1} & cos\theta_i cos\phi_{i-1} & -sin\phi_{i-1} & -l_i sin\phi_{i-1} \\ sin\theta_i sin\phi_{i-1} & cos\theta_i sin\phi_{i-1} & cos\phi_{i-1} & -l_i cos\phi_{i-1} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.2}$$

θ_i is the torsional angle of bond b_i
ϕ_{i-1} is the bond angle between bonds b_{i-1} and b_i

Atomgroup scheme. This scheme eliminates the requirement for multiple frames and frames where the bond does not rotate. It simply aggregates the tree into a new tree where sets of vertices (atoms) which do not have rotatable bonds are collapsed into a new vertex. Here, we define the local frame as the atomgroup origin and the vectors

- $\mathbf{w}_i$ as a vector along the bond to atomgroup $i-1$
- $\mathbf{u}_i$ as any vector perpendicular to $\mathbf{w}_i$
- $\mathbf{v}_i$ as any vector perpendicular to the above two.

Let the frames after and before rotation be $[\mathbf{x}_i, \mathbf{u}_i, \mathbf{v}_i, \mathbf{w}_i]$ and $[\mathbf{x}'_i, \mathbf{u}'_i, \mathbf{v}'_i, \mathbf{w}'_i]$. In this case the transformation matrix, which takes a point in frame i to local frame at $i-1$ (rotated by θ around the connecting bond) is defined as the product

$$\begin{pmatrix} \mathbf{u}_{i-1}.\mathbf{u}'_i & \mathbf{u}_{i-1}.\mathbf{v}'_i & \mathbf{u}_{i-1}.\mathbf{w}_i & \mathbf{u}_{i-1}.(\mathbf{x}_i - \mathbf{x}_{i-1} \\ \mathbf{v}_{i-1}.\mathbf{u}'_i & \mathbf{v}_{i-1}.\mathbf{v}'_i & \mathbf{v}_{i-1}.\mathbf{w}_i & \mathbf{v}_{i-1}.(\mathbf{x}_i - \mathbf{x}_{i-1} \\ \mathbf{w}_{i-1}.\mathbf{u}'_i & \mathbf{w}_{i-1}.\mathbf{v}'_i & \mathbf{w}_{i-1}.\mathbf{w}_i & \mathbf{w}_{i-1}.(\mathbf{x}_i - \mathbf{x}_{i-1} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos\theta_i & -sin\theta_i & 0 & 0 \\ sin\theta_i & cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.3}$$

The concatenation of such matrices until the root gives the global position of the atomgroup.

A.3.4 Flexibility analysis in molecules - creation of flexible models

One classification of flexibility analysis methods in the biomolecular area is given by [14] as

- Molecular dynamics

  Molecular dynamics involves simulation of the protein in a solvent environment and saving the conformation state at regular time intervals. Since this simulation is often at very small time scales, (pico or nano seconds), large conformational changes (which occur over micro or milli seconds) will not be recorded. Hence obtaining flexibility analysis through molecular dynamics is limited. An adaptive solver is given in [17]. By allowing users to interact with the system, conformational changes can be forced and observed [20], [32]. A multiple grid method for solving the electrostatics efficiently [31]. Compact structural domains were computed in [12] using simple force calculations in a protein structure.

- Xray Crystallography and Nuclear Magnetic Resonance (NMR)

  Xray Crystallography is used to obtain high resolution images of proteins, upto the atomic level. Most structure in the PDB are generated using this method.

  NMR techniques have been used to obtain dynamic conformations of proteins. The basic idea behind NMR is that atoms have an intrinsic property spin, which determines its behavior when exposed to magnetic fields. Different atoms are seen to emit different frequencies of light, providing an image of the underlying protein as a signature. NMR imaging yields lower resolution results than xray crystallography.

  Given the large number of states which could be obtained from molecular dynamics, NMR and xray crystallography, the following methods generate certain important conformal states by reducing the number of degrees of freedom in the protein.

- Comparison of conformal states

  Protein dynamics give rise to a large number of conformations. Analyzing these conformations for any problem, including flexible protein docking is not computationally feasible. Hence many methods are used to reduce these conformations to a new basis, where the principal basis gave the large fluctuations efficiently. Many authors [35], have shown that the main conformational changes of a protein is mostly captured by using only a few bases and projection vectors, [34]. Normal mode analysis and principal component analysis are two methods to reduce the dimensionality of the problem.

  Singular Value Decomposition (SVD) is commonly used to find basis vectors to reduce the dimensionality of a set of vectors. An equivalent formulation using Principal Component Analysis (PCA) is also done. Consider the column vectors of a matrix A as the zero mean weighted atomic displacement positions. Usually, this vector is also aligned with a given conformation, so that the displacements are relative. The SVD of a matrix is

  $$SVD(A) = U \sum V^T \tag{3.4}$$

  U, V are orthonormal matrices
  $\sum$ is a diagonal matrix
  The diagonal matrix has entries are all non negative and decreasing, called the singular values.

  In this decomposition, the set of left column vectors of U are the basis set for A, and the vectors in V^T are the projections along these basis vectors with magnitudes given by the singular values. Hence, we have an ordering on the influence of the basis vectors for the matrix.

To apply the PCA algorithm, a matrix A is defined with elements a_{ij} as follows

$$a_{ij} = ((x_i - x_{i,avg})(x_j - x_{j,avg})) \tag{3.5}$$

The eigenvector problem $AW = W\zeta$ is solved to get the axis vectors and the corresponding fluctuations in the eigenvectors and eigenvalues [19].

In [10], a theorem relating the atom displacements to the frequencies of vibrations is presented. In this paper, the authors prove that if a large molecule only flexes around a certain minimal energy state, approximated by a multidimensional parabola, then the average displacements of the atom positions is the sum of the contributions from each normal mode, which is proportional to the inverse square of the frequency [19]. For Normal Mode Analysis (NMA), the moment matrix diagonalized is

$$A = k_B T F^{-1} \tag{3.6}$$

k_B is the Boltzmann constant,
T is the absolute temperature,
F is a matrix of the second derivatives of the potential energy at a minimum point.

Successful modeling of the Chaperonin GroEL was performed using NMA in [22]. To avoid the computations on a large matrix, [33] compute a blocked version of NMA by grouping residues.

Gaussian Network Models (GNM) are used in [18]. In this model, the correlation matrix is formed as

$$(3kT/\gamma[T^{-1}]_{ij}) \tag{3.7}$$

k is the boltzmann constant,
T is the absolute temperature,
γ is a harmonic potential,
T is a nearness matrix, called Kirchoff matrix

The kirchoff matrix inverse can only be approximated since its determinant is 0.

- Deriving flexibility through a single structure.

  Non polar regions in protein tend to lie in the interior and this hydrophobic effect folds the protein. In [37], the authors describe how to capture this information into rigid domains of the protein. Their assumption is that rigid domains folded by the hydrophobic effect behave as a *compact unit* during conformational changes. To quantify this, they hierarchically grouped residues in a protein to form a tree, using a coefficient of compactness Z given by

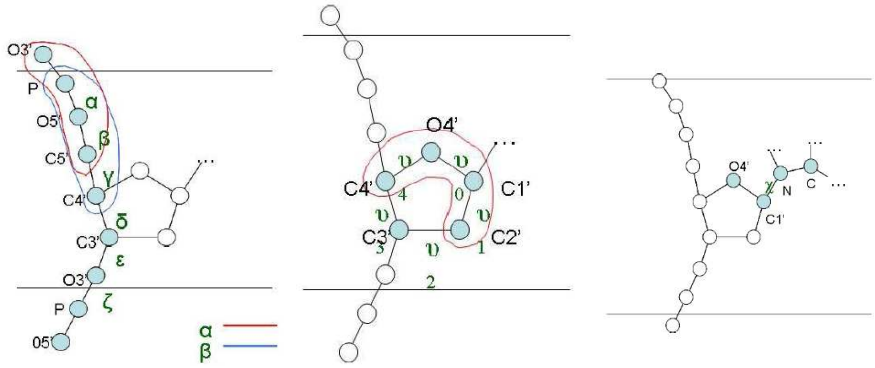
$$Z = \frac{\text{accessible surface area of segment}}{\text{surface area of sphere of equal volume}} \tag{3.8}$$

  Static core or the backbone of molecules and their associated rigid domains were computed in [3] using two different conformations of a given protein. α helices, β strands and loops were segmented. Similar pairs of segments were clustered in a tree-like fashion using a rmsd calculation. Domains or compact units of a protein were also computed by [30]. The heuristic they used was that the amount of internal contact a domain had was larger than the amount of contact it had with the rest of the protein. Hence by choosing suitable split planes along the sequence, they form compact sequences. Extending this idea, a Monte Carlo sampling in internal coordinates using relevant torsion angles was performed in [23]. They obtained a set of low energy conformations for any given protein structure as a representation of its flexibility. Using graph theoretical algorithms, [14] obtained flexible and rigid domains in a protein.

A.4 Flexibility in RNA

Flexibility in RNA is given by three sets of angles

- The backbone torsion angles.



(a) The six backbone torsion angles for a RNA

(b) The torsion angles defined along the sugar ring

(c) Nucleotides can rotate about the χ torsion angle

Figure A.10: The torsion angles around which a RNA can flex.

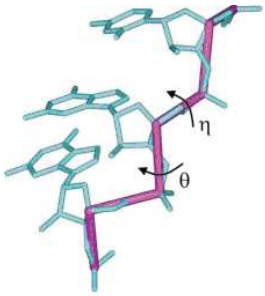


Figure A.11: The backbone torsion angles are represented by just two pseudo rotation angles η and θ.

- The angles on the sugar ring, also defined by amplitude and a phase.
- An angle about which the residue can flex.

The angles are shown in figure A.10. Due to the large number of angles, people have studied and proposed various means to reduce the conformational space.

A.4.1 Reduced conformation space

Due to the large number of angles defining the flexibility of nucleotides, it is useful to find fewer pseudo torsion angles to represent the other angles.

Reduction to two angles. Duarte et al. have reduced the number of torsion angles necessary to describe an RNA molecule to two, η and θ [5], [4]. Figure A.11 gives the relative positions of these angles and the specific atoms of the backbone involved.

η is the torsion angle resulting from $C4'_{i-1} - P_i - C4'_i - P_{i+1}$. The atoms connected $P_i - C4'_i - P_{i+1} - C4'_{i+1}$ create θ [5]. In their most recent publication, Duarte et al. combined the η and θ data with position information to describe the overall structure of the RNA molecule. Using PRIMOS [6] to create an "RNA worm" - a sequential description of the angle data - allows for analysis of the structure on a nucleotide by nucleotide basis.

After all η and θ angles have been calculated from the PDB [8], NDB [2], and RNABASE [36] data, PRIMOS creates an RNA worm file which gets deposited into a database. The two angles are plotted and a 3^{rd} dimension, sequence, is added to the graph to form a 3D representation of structure. See Figure A.12.

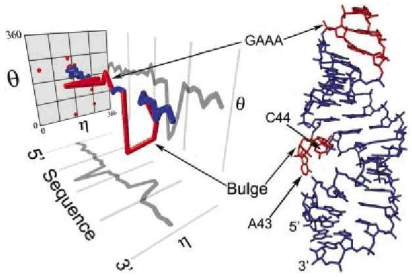


Figure A.12: Example of 3D representation of RNA structure. Plotting the RNA chains using only two angles per residue in a 3D plot shows similar structures along the *worms*

Angle	Bin 1	Bin 2	Bin 3	Bin 4
α	40 - 90	135 - 190	260 - 330	other
γ	35 - 75	150 - 200	260 - 320	other
δ	68 - 93	130 - 165	other	
ζ	255 - 325	other		

Table A.5: Classification of angles:discrete ranges of angles or "bins" as defined by Hershkovitz.

In this plot, A-helices (the most common form of RNA; represented in blue) travel in relatively straight lines, whereas the motifs/other features of the RNA show large deviations from the straight line (shown in red).

To compare RNA worm representations, and thus conformational variations between molecules, it is necessary to find the difference between the η and θ values in the two molecules. Simply put:

$$\Delta(\eta,\theta)_i = \sqrt{(\eta^A - \eta_i^B)^2 + (\theta^A - \theta_i^B)^2} \tag{4.9}$$

The larger the value of $\Delta(\eta,\theta)_i$ the more extreme the disparity between the two RNA fragments, chains, or molecules.

Further, Duarte et al. use this method to compare ribosomal complexes, search for existing motifs, identify new motifs, and characterize two different types of the same motifs. To compare ribosomal complexes, Duarte et al use PRIMOS to calculate differences in h and q when the ribosome is in different conformational states. For example, the conformational state of the ribosome is altered during antibiotic binding or during different stages of translation. The same method can be used to compare conformational states of ribosomes from different species.

To find existing motifs in RNA structures, they used PRIMOS to create another RNA worm database. From this database, a fragment of RNA that contained the motif of interest was selected and compared to every other fragment of the same size within the database and given a score according to equation 4.10.

$$\overline{\Delta(\eta,\theta)} = \frac{\sum_{i=1}^{n} \Delta(\eta,\theta)_i}{n} \tag{4.10}$$

The scores were sorted in increasing order. The smaller scores indicate a closer match.

Reduction to four angles and binning. Hershkovitz et al. [11] suggest a more complex, yet complementary, method to that of Duarte [5]. This method involves calculating four torsion angles, α, γ, δ and ζ, and binning these angles into allowable ranges. "Binning" is a term used to describe the technique used by Hershkovitz to classify various RNA configurations into discrete bins. For example, nucleotides in the A-form helix, the most common conformation of RNA, have a bin number of 3111 where each number represents which "bin", or range, the torsion angles belong to (i.e. α is in bin 3, or 260° - 320° and γ, δ and ζ and are in bin 1, or 35° - 75°). See Table A.5.

The bin number combination 3111 is then assigned an ASCII character, "a". All combinations of bin numbers are assigned a unique ASCII character, enabling the entire RNA chain to be described by a sequence of letters that represent the structure of the molecule. Their goals were to recognize and catalogue all the RNA conformational states, eliminate any unnecessary angle information, and to assess the validity of their binning model by comparing it to a torsion-matching model. The torsion-matching method for RNA motif searching is a brute force method. So while it is highly accurate, it is computationally expensive as it involves calculating all backbone angles including a ribopseudorotation phase angle, P, for each residue and comparing each set of angles to all other sets of angles in the molecule.

After using the binning method for all RNA fragments and molecules in their database, Hershkovitz et al found 37 distinct conformational states of RNA. Table A.13 lists the assigned bin numbers, the corresponding ASCII symbols, and the observed frequency of these 37 conformational states.

Because this method allows the three dimensional structure of an RNA molecule to be displayed as a sequence of characters, it facilitates motif searching. Without computational aides, one could see that a string of repeating letters (other than "a") represents a possible motif.

Hershkovitz et al suggest an alternative to the Ramachandran plots traditionally used for representing angle distributions. The tree diagram in figure A.14 is a natural progression from the four integer code, or bin. Here the widths of the line correspond to the log of the number of residues in each bin.

A.4.2 Classification of RNA using clustering

Nucleotides from the large ribosomal subunit (1JJ2.pdb) were clustered into commonly occurring structures by Schneider et al. [29]. They classified the non A-type nucleotides separately (830 of them). Eighteen distinct non A-type conformations and fourteen A-type conformations were reported. They report that a large number of the RNA were very close (in a RMSE sense) to the clusters. The authors also say that their results agree with those from Murray et al. [24].

The steps used in obtaining the conformations were as follows.

- Separate the A-type from the non A-type nucleotides.
- Plot the histogram for the backbone ($\alpha, \beta, \gamma, \delta, \varepsilon, \zeta$) and the base ($\chi$) angles.
  - α and γ were seen to have tri-modal distributions.
  - β has a wide gaussian with 180 as its center.
  - ε has values greater than 180 due to the ring, and lacked a gaussian shape.
  - *delta* also was constrained by the ring, and had a sharp bimodal distribution due to the C3'-endo and C2'-endo ribose puckers.
  - The base χ angle was largely bimodal, due to the two main configuration of bases (anti and syn).
  - There was a wide distribution of ζ.
- Plot 2D scatter plots for the following angle pairs : [α, ζ], [β, ζ], [ε, ζ], [γ, α], [χ, ζ] and [χ, δ].
  - The reason for choosing the above sets were not given.
  - Clusters were found in the pairs [ζ, α], [α, γ] and [χ, δ].
  - The lack of clusters in other plots led to clustering of 3 tuples of angles.
- From the features and distributions seen in the 1D and 2D plots, the authors choose six 3D plots to base their clusters on to classify the structure of nucleotides.
  - The following six 3tuples were chosen for clustering: [ζ_i, α_{i+1}, δ_i], [ζ_i, α_{i+1}, γ_{i+1}], [α_i, γ_i, δ_i], [ζ_i, α_{i+1}, χ_i], [ζ_i, α_{i+1}, ε_i] and [ζ_i, δ_i, χ_i].
  - The clusters in the 3D plots were assigned peaks and labeled.

Ascii letter[a]	Bin number	Frequency
a	3111	1709
e	3112	169
r	3122	124
i	2211	103
o	2111	58
t	4111	48
n	1111	37
s	2122	34
l	1211	31
c	3121	30
u	4211	28
d	1121	26
p	4122	21
m	1122	21
h	3411	18
g	1322	18
b	1112	14
f	3211	14
y	4112	13
w	2212	11
k	4121	11
v	3212	10
x	3222	10
z	1331	9
j	4222	9
q	3321	8
1	1212	8
2	3422	8
3	4311	8
4	4411	8
5	2121	7
6	3322	7
7	2222	7
8	2411	7
9	1311	7
0	1221	7
+	3311	6

[a]The assignment of characters to configuration classes was made by frequency of observation. The choice of letter assignment was taken from http://www.askoxford.com/asktheexperts/faq/aboutwords/frequency. All bins with less than five residues are denoted by * and are omitted from this table.

Figure A.13: Classification into 37 clusters through binning

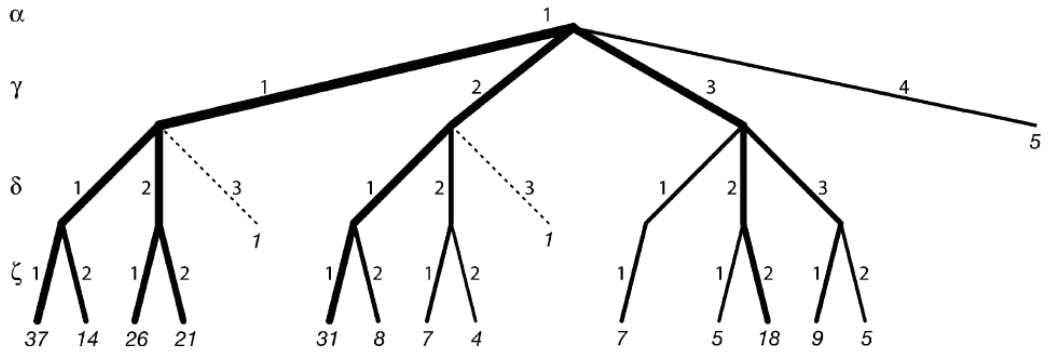


Figure A.14: This tree represents the case where α is 1. There are three others; one for each possible value of α.

- Each nucleotide was assigned the corresponding label from each plot, if any, or simply a '-'.

- Each nucleotides 6 letter classification was clustered using lexicographic clustering. The authors do not mention why this method was used.

- From this clustering, eighteen distinct non A-type conformations and fourteen A-type conformations were reported.

A.4.3 Division of RNA backbone by *suites*

Murray et al. [24] identify several problems associated with the methods of Murthy [25], Hershkovitz [11], and Duarte [5]. While these methods are excellent at finding and comparing RNA motifs in a large nucleic acid sample, they oversimplify the problem of determining RNA backbone structure. As a result, Murray et al. propose to analyze the folding structure of RNA molecules on a more detailed level, correct the artifacts created in the data structures (sometimes caused by NMR or X-ray crystallography), produce *low-noise data distributions*, and create a list of the resulting, distinct RNA backbone conformers.

The traditional nine angles of the RNA backbone and its bases (i.e. $\alpha, \beta, \gamma, \delta, \varepsilon, \zeta, \chi$, and the 2 puckering angles of the sugars) were reduced to six. χ was not included in the model. The two puckering angles were combined and represented as δ, where δ was bimodal - either C3' endo or C2' endo. This allowed the six remaining angles two be divided into 2 sets of 3D distributions, α, β, γ and $\delta, \varepsilon, \zeta$. Dividing the RNA backbone into *heminucleotides*, a term coined by Malathi and Yathinda [27], in this manner provided some advantage to the traditional phosphate - phosphate division in that it reduced the dimension of the problem and made visualization more feasible. In other words, two 3D plots can be created using α, β, γ data and $\delta, \varepsilon, \zeta$ data respectively. See figure A.15

The methods of Murray et.al were fairly straightforward. They obtained the sequence and structure data samples from the Protein Database and/or the Nucleic Acid Database. From these samples they calculated all the dihedral angles and added hydrogens with REDUCE [15]. The backbone steric hindrances were calculated with PROBE and CLASHLIST [16]. A clash was noted when the overlap between two atoms was greater than 0.4Å. The angles, quality, resolution, base id, highest crystallographic B factor, and d-e-z values were entered into excel. Images were created using the software PREKIN and MAGE from the same authors. For each of the seven peaks created in the $\delta, \varepsilon, \zeta$ distributions, the α, β, γ set was plotted. Finally, a quality filter was applied to rule out nucleotides with greater than 2.4Å resolution.

210 potential RNA conformers were determined from which 146 had an acceptable (low) amount of steric hindrance. 42 conformers had actual cluster points from the data.

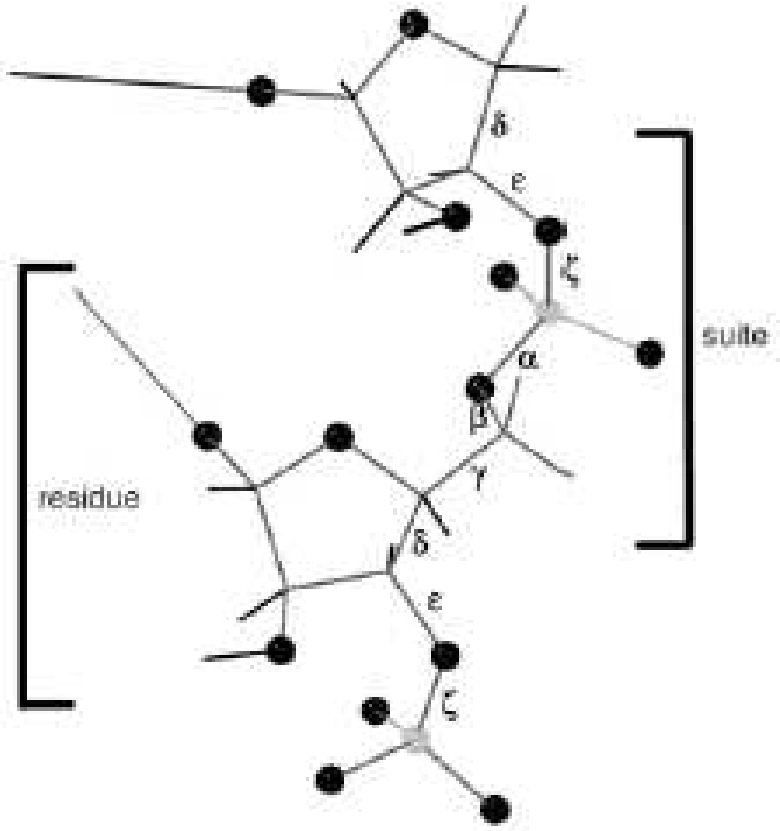


Figure A.15: Division of angles into residue and "suite" data

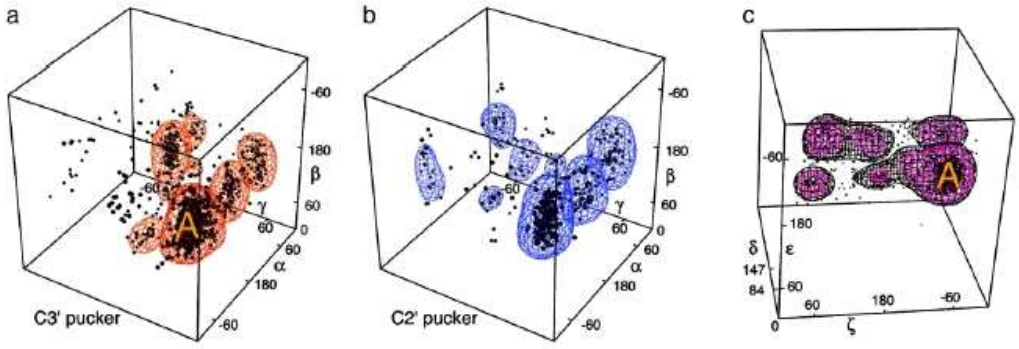


Figure A.16: 3D visualization of clusters

Bibliography

[1] C. Bajaj, V. Pascucci, A. Shamir, R. Holt, and A. Netravali. Dynamic maintenance and visualization of molecular surfaces. *Dis. App. Math.*, 127(1):23–51, 2003.

[2] H. M. Berman, W. K. Olson, D. L. Beveridge, J. Westbrook, A. Gelbin, T. Demeny, S.-H. Hsieh, A. R. Srinivasan, and B. Schneider. The nucleic acid database: A comprehensive relational database of three-dimensional structures of nucleic acids. *Biophys. J.*, 63:751–759, 1992.

[3] N. Boutonnet, M. Rooman, and S. Wodak. Automatic analysis of protein conformational changes of by multiple linkage clustering. *Journal of Molecular Biology*, 253(4):633–647, 1995.

[4] D. CM and P. AM. Stepping through an rna structure: a novel approach to conformational analysis. *Journal of Molecular Biology*, 284:1465–1478, 1998.

[5] D. CM, W. LM, and P. AM. Rna structure comparison, motif search and discovery using a reduced representation of rna conformational space. *Nucleic Acids Research*, 31:4755–4761, 2003.

[6] C. Duartes. Primos. software:www.pylelab.org.

[7] R. A. Engh and R. Huber. Accurate bond and angle parameters for x-ray protein structure refinement. *Acta Crystallographica Section A*, 47(4):392–400, July 1991.

[8] F.C.Bernstein, T.F.Koetzle, G.J.B.Williams, E. Jr, M.D.Brice, J.R.Rodgers, O.Kennard, T.Shimanouchi, and M.Tasumi. The protein data bank: a computer-based archival file for macromolecular structures. *J. Mol. Biol.*, 112:535–542, 1977.

[9] R. Garrett and C. Grisham. *Biochemistry*. Saunders Collge Publishing, New York, 2nd edition, 1999.

[10] N. Go. A theorem on amplitudes of thermal atomic fluctuations in large molecules assuming specific conformations calculated by normal mode analysis. *Biophysical Chemistry*, 35(1):105–112, January 1990.

[11] E. Hershkovitz, E. Tannenbaum, S. B. Howerton, A. Sheth, A. Tannenbaum, and L. D. Williams. Automated identification of rna conformational motifs: theory and application to the hm lsu 23s rrna. *Nucleic Acids Res.*, 31(21):6249–6257, November 2003.

[12] L. Holm and C. Sander. Parser for protein folding units. *Proteins*, 19(3):256–268, July 1994.

[13] H. R. Horton, L. A. Moran, R. S. Ochs, D. J. Rawn, and K. G. Scrimgeour. *Principles of Biochemistry*. Prentice Hall, 3rd edition, July 2002.

[14] D. J. Jacobs, A. J. Rader, L. A. Kuhn, and M. F. Thorpe. Protein flexibility predictions using graph theory. *Proteins: Structure, Function, and Genetics*, 44:150–165, 2001.

[15] W. JM, L. SC, R. JS, and R. DC. Asparagine and glutamine: using hydrogen atom contacts in the choice of side-chain amide orientation. *J Mol Biol.*, 284(4):1735–1747, January 1999.

[16] W. JM, L. SC, L. TH, T. HC, Z. ME, P. BK, R. JS, and R. DC. Visualizing and quantifying molecular goodness-of-fit: small-probe contact dots with explicit hydrogen atoms. *J Mol Biol.*, 285(4):1711–1733, January 1999.

[17] L. KalÃ©, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, and K. Schulten. Namd2: greater scalability for parallel molecular dynamics. *J. Comput. Phys.*, 151(1):283–312, 1999.

[18] O. Keskin, R. L. Jernigan, and I. Bahar. Proteins with similar architecture exhibit similar large-scale dynamic behavior. *Biophysical Journal*, 78(4):2093–2106, April 2000.

[19] A. Kitao and N. Go. Investigating protein dynamics in collective coordinate space. *Current Opinion in Structural Biology*, 9(2):164–169, 1999.

[20] J. Leech, J. Prins, and J. Hermans. Smd: Visual steering of molecular dynamics for protein design. *IEEE Computational Science and Engineering*, 3(4):38–45, 1996.

[21] A.-J. Li and R. Nussinov. A set of van der waals and coulombic radii of protein atoms for molecular and solvent-accessible surface calculation, packing evaluation, and docking. *Proteins: Structure, Function, and Genetics*, 32(1):111–127, 1998.

[22] J. Ma and M. Karplus. The allosteric mechanism of the chaperonin groel: A dynamic analysis. *Proceedings of the National Academy of Sciences USA.*, 95(15):8502–8507, July 1998.

[23] V. N. Maiorov and R. A. Abagyan. A new method for modeling large-scale rearrangements of protein domains. *Proteins*, 27:410–424, 1997.

[24] L. J. Murray, W. B. A. III, D. C. Richardson, and J. S. Richardson. Rna backbone is rotameric. *Proceedings of the National Academy of Sciences U S A*, 100(24):13904–13909, November 2003.

[25] V. L. Murthy, R. Srinivasan, D. E. Draper, and G. D. Rose. A complete conformational map for rna. *Journal of Molecular Biology*, 291(2):313–327, August 1999.

[26] J. Park, S. Teichmann, T. Hubbard, and C. Chothia. Intermediate sequences increase the detection of homology between

sequences. *Journal of Molecular Biology*, 273(6):349–354, 1997.

[27] M. R and Y. N. Backbone conformation in nucleic acids: an analysis of local helicity through heminucleotide scheme and a proposal for a unified conformational plot. *Journal Biomolecular Structural Dynamics*, 3(1):127–144, August 1985.

[28] T. Schlick. *Molecular Modeling and Simulation: An Interdisciplinary Guide*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[29] B. Schneider, Z. MorÃ¡vek1, and H. M. Berman. Rna conformational classes. *Nucleic Acids Research*, 32(5):1666–1677, 2004.

[30] A. S. Siddiqui and G. J. Barton. Continuous and discontinuous domains: an algorithm for the automatic generation of reliable protein domain definitions. *Protein Science*, 4(5):872–884, May 1995.

[31] R. D. Skeel, I. Tezcan, and D. J. Hardy. Multiple grid methods for classical molecular dynamics. *Journal of Computatioanl Chemistry*, 23(6):673–684, 2002.

[32] J. E. Stone, J. Gullingsrud, and K. Schulten. A system for interactive molecular dynamics simulation. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 191–194. ACM Press, 2001.

[33] F. Tama, F. X. Gadea, O. Marques, and Y. H. Sanejouand. Building-block approach for determining low-frequency normal modes of macromolecules. *Proteins*, 41(1):1–7, October 2000.

[34] M. Teodoro, J. G. N. Phillips, and L. E. Kavraki. Molecular docking: A problem with thousands of degrees of freedom. In *Proc. of the 2001 IEEE International Conference on Robotics and Automation (ICRA 2001)*, pages 960–966, Seoul, Korea, May 2001. IEEE press.

[35] M. L. Teodoro, G. N. Phillips, Jr., and L. E. Kavraki. A dimensionality reduction approach to modeling protein flexibility. In *Proceedings of the sixth annual international conference on Computational biology*, pages 299–308. ACM Press, 2002.

[36] M. VL and R. GD. Rnabase: an annotated database of rna structures. *Nucleic Acids Research*, 31(1):502–504, 2003.

[37] M. H. Zehfus and G. D. Rose. Compact units in proteins. *Biochemistry*, 25(19):5759–5765, September 1986.

[38] M. Zhang and L. E. Kavraki. A new method for fast and accurate derivation of molecular conformations. *Journal of Chemical Information and Computer Sciences*, 42(1):64–70, 2002.