University of West Bohemia in Pilsen
Department of Computer Science and Engineering
Univerzitni 8
30614 Pilsen
Czech Republic

Regular Triangulation and Tunnels in Proteins

Michal Zemek Ivana Kolingerová Petr Medek Jiří Sochor

Technical Report No. DCSE/TR-2007-15
December, 2007

Distribution: public

Technical Report No. DCSE/TR-2007-15
December, 2007

Regular Triangulation and Tunnels in Proteins

Michal Zemek Ivana Kolingerová Petr Medek Jiří Sochor

Abstract

Proteins are organic compounds and are contained in every living cell. Protein engineering studies proteins and tries to modify them. For this purpose it uses (among others) an analysis of tunnels – paths, which lead from an interesting place inside a protein to its surface. In this paper the geometric approximation of tunnels in protein molecules and the application of regular triangulation for its computation are described. Two new types of tunnel approximations are introduced and compared with the existing method of the tunnel computation.

Keywords

Regular triangulation, tunnel, protein molecule.

This work was supported by The Ministry of Education of The Czech Republic, project no. LC 06008.

Copies of this report are available on
http://www.kiv.zcu.cz/publications/
or by surface mail on request sent to the following address:

University of West Bohemia in Pilsen
Department of Computer Science and Engineering
Univerzitni 8
30614 Pilsen
Czech Republic

Table of Contents

1. Introduction

A protein consists of hundreds or thousands of amino acids, an amino acid consists of a few types of atom. In a protein, amino acids are connected in a linear chain. An order of acids determines a primary structure of a protein and has an essential influence on protein chemical properties. A small change in primary structure can cause significant changes of these properties. This fact is used by a protein engineering, which tries to create "improved proteins". These improved proteins are used in designs of new drugs and agrochemicals or for detection and neutralization of dangerous chemicals [DAM].

A place, where a protein should be modified, is called an active place. During a protein modification it is necessary to transport some certain molecule to the active place, and this molecule must not affect the protein structure. Therefore an existence of a tunnel (leading from the active place to a protein surface) and its size is an important question. In this paper the geometric approximation of tunnels in protein molecules and the application of regular triangulation for its computation are described. Two new types of tunnel approximations are introduced and compared with the existing method of the tunnel computation.

Contents of the paper are as follows. Section 2 describes state of the art. Regular triangulation is described in section 3. Sections 4 and 5 are dedicated to the tunnel definition and to algorithm for the tunnel construction. We give and discuss results of our algorithm in the section 6.

2. State of the Art

Computer simulation and visualization of proteins is intensively used in biochemical research, so many papers with this topic have been published. Most of them deal with protein visualization or with computing of a molecule surface ([B01, R05]). As far as we know, only two papers are focused on the tunnel computation. The first is based on a space discretization [P06] – the space with protein is divided into a regular grid and each cell of this grid is evaluated by its distance from the nearest atom. The tunnel is constructed as a set of cells using Dijkstra's algorithm.

The second approach is described in [M07]. It is based on a space subdivision using Voronoi diagram and its dual structure – Delaunay triangulation. At first, a Voronoi diagram is created, positions of protein atoms are used as Voronoi generators. Second, the Voronoi diagram is traversed using Dijkstra's algorithm and a tunnel is searched as a sequence of Voronoi edges. The Voronoi edges are evaluated by their distance from the nearest atom.

Our approach is based on the second solution. We use power diagrams and regular triangulations instead of Voronoi diagrams and Delaunay triangulations and we use a different evaluation of diagram edges.

3. Regular Triangulation

Regular triangulation (RT) is a generalization of well-known Delaunay triangulation (DT). Each point in the regular triangulation is associated with a real number – a weight of the point. If weights of all points are equal, then regular triangulation and Delaunay triangulation of this set of points are identical. If weights of points are different, then Delaunay and regular triangulation can be different (see Fig. 3.1). We are interested in a 3D triangulation (and in 3D tunnels), but we often use figures of 2D triangulations (and 2D tunnels) for a simplicity.

In this section we define the regular triangulation accordingly to the definition in [E92] and we describe an algorithm for its construction [E92, F93].

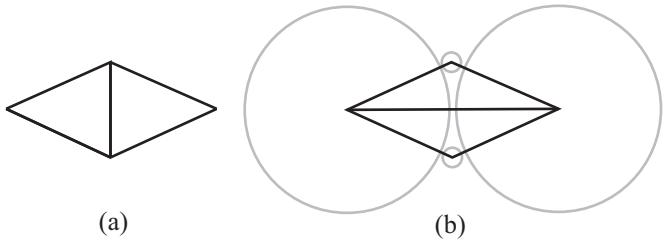


Fig. 3.1: (a) Delaunay triangulation (in 2D), (b) Regular triangulation (in 2D). The circles represent weights of he points.

3.1 Definition

At first we need a few basic definitions, further we give the exact definition of the regular triangulation and also two lemmas, which are useful for a construction of the regular triangulation:

Triangulation: Triangulation $T(S)$ of a finite set S, $S \subset E^3, |S| = n$, is a subdivision of the convex hull of S into a set of non-intersecting tetrahedra, whose vertices are points from S.

Weighted points: $S \subset E^3, |S| = n$. Each point $p \in S$ has an associated weight $w_p \in R$. The weight w_p can be negative, but usually only non-negative weights are used. The weighted point can be interpreted as a sphere centered at the point p with a radius $\sqrt{w_p}$.

Power distance: For each weighted point p its power distance from an unweighted point $x \in E^3$ is defined as $\pi_p(x) = |xp|^2 - w_p$ ($|xp|$ denotes the Euclidean distance between points p and x). Power distance can be interpreted as a square of length of a tangent from the point x to a sphere with the centre p and with the radius $\sqrt{w_p}$ (if x lies outside the sphere), see Fig. 3.2a.

Orthogonal points: Two weighted points p and q are orthogonal if $|pq|^2 = w_p + w_q$, see Fig. 3.2b.

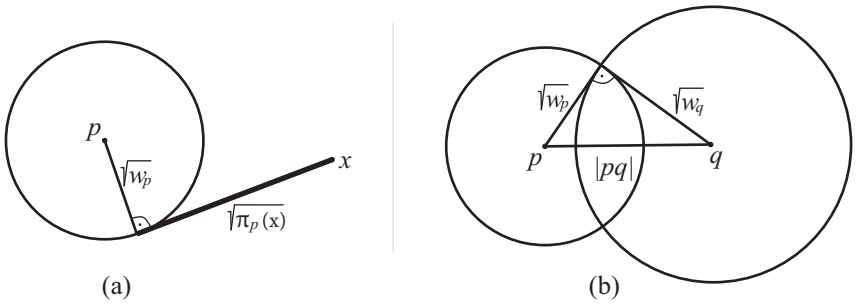


Fig. 3.2: (a) Power distance, (b) Two orthogonal weighted points.

Orthogonal centre: Weighted point z is an orthogonal centre of tetrahedron *abcd* if z is orthogonal to the points a, b, c, d.

Regularity (global): Let us suppose that a weighted point z is the orthogonal centre of tetrahedron *abcd*. Tetrahedron *abcd* is globally regular if $\pi_z(p) > w_p$ for each point $p \in S - \{a,b,c,d\}$ (w_p is the weight of a point p).

Regular triangulation: Triangulation $RT(S)$ is regular if each tetrahedron in $RT(S)$ is globally regular.

Redundant point: A point $p \in S$ is called a redundant point, if no globally regular tetrahedron *pabc* exists in *RT*(*S*). The redundant point is not a vertex of any tetrahedron, therefore, the vertex set of *RT*(*S*) is generally only a subset of *S*.

Lemma (local regularity): *abcd* and *abce* are two tetrahedra with common face *abc* and *z* is the orthogonal centre of the tetrahedron *abcd*. The common face *abc* is locally regular if $\pi_z(e) > w_e$.

Lemma (regular triangulation): If the vertex set of *T*(*S*) contains all non-redundant points from *S* and all faces of *T*(*S*) are locally regular then the triangulation *T*(*S*) is regular[1].

3.2 The Construction Algorithm

We use an incremental flipping algorithm (published in [E92] and [F93]) for a construction of regular triangulation. The algorithm creates a triangulation of a given set of points $S = \{p_0, p_1, ..., p_n\}$ incrementally – in each step one point is inserted into the triangulation. At first tetrahedra, which contain the new point[2], are located. Then the new point is connected into these tetrahedra – they are splitted and new tetrahedra are created. Some faces of the new tetrahedra can be non-regular – these have to be repaired using flips[3] before the next point is inserted. Further we will describe the particular steps of the flipping algorithm.

The Starting Tetrahedron

The algorithm needs a starting triangulation, in which the points from *S* will be inserted. One possibility is to create the convex hull of *S*. Simpler solution is to use an auxiliary tetrahedron *t*. The positions of vertices $p_{-4}, p_{-3}, p_{-2}, p_{-1}$ of the tetrahedron *t* have to be chosen so that all points from *S* are contained in *t*. When the triangulation is finished, all tetrahedra with a vertex from $\{p_{-4}, p_{-3}, p_{-2}, p_{-1}\}$ have to be removed.

The typical positions of the points $p_{-4}, p_{-3}, p_{-2}, p_{-1}$ are as follows:

$$p_{-4} = (s_x + K \cdot d, s_y, s_z),$$
$$p_{-3} = (s_x, s_y + K \cdot d, s_z),$$
$$p_{-2} = (s_x, s_y, s_z + K \cdot d),$$
$$p_{-1} = (s_x - K \cdot d, s_y - K \cdot d, s_z - K \cdot d),$$

where $s = (s_x, s_y, s_z)$ is the centre of a min-max box of *S*, *d* is the longest edge of the min-max box and *K* is a constant. The choice of *K* is problematic – when we use small *K*, the triangulation do not have to be convex after we remove the auxiliary tetrahedra. If we use a big *K*, numeric precision suffers. The used value of *K* is between 10 and 20.

Point Location

Before a point *p* can be inserted into a triangulation *T*, a tetrahedron *t*, which contains *p*, has to be found. For this purpose we use a simple visibility walk algorithm. The expected complexity of the walking algorithm in 3D is $O\left(n^{1/4}\right)$ for one point location. The group

[1] Due to the lemmas, we can check the regularity of a tetrahedron or of a whole triangulation using local tests.
[2] Typically a point is contained only in one tetrahedron.
[3] Later we will describe what the mentioned term "flip" means.

of walking algorithms is described in [D01] in detail. We describe only the basic visibility walking strategy. Our goal is to find the tetrahedron *t*, which contains the point *p*:

```
Visibility walk algorithm                                          (3.1)
   1. t := a random tetrahedron.
   2. While t does not contain the point p do
   3.    Find a face f of t, so that the plane supporting f
         separates t and p.
   4.    t := the neighbor of t through f.
   5. Return t.
```

Flips

A flip is an operation, which replaces a few neighboring tetrahedra with other tetrahedra, which fill the same space. We use six types of flips in 3D. The former three flips are used for a point connection into a triangulation and the later three flips are used for reparations of the non-regular faces. We denote the flips according to the number of tetrahedra before and after the flip.

Insertion Flips

Let us suppose that we are inserting a point *p* into the triangulation *T*. Three situations may appear:

- flip 1→4: The inserted point *p* lies inside a tetrahedron *abcd*. The point *p* is just connected with the points *a, b, c, d* – four new tetrahedra are created from one tetrahedron (see Fig. 3.3a).
- flip 2→6: The inserted point *p* lies in a face *abc*, two tetrahedra *abcd* and *abce* share this face. The point *p* is connected with the points *a, b, c, d, e*, six new tetrahedra are created from two tetrahedra (see Fig. 3.3b)
- flip N→2N: The inserted point *p* lies on an edge *ab*, which is contained in N tetrahedra. All these tetrahedra have to be found, and each of them is split in two (for example, a tetrahedron *abcd* is divided into tetrahedra *acdp* and *bcdp*, see Fig. 3.4).

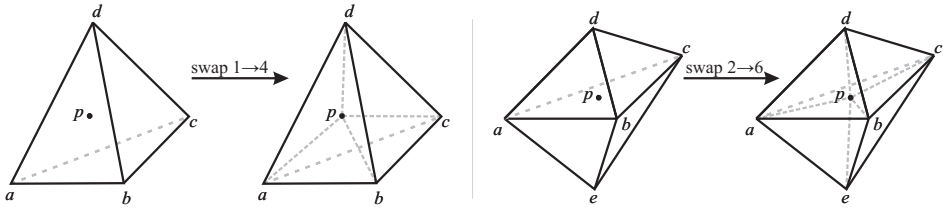


Fig. 3.3: (a) Flip 1→4, (b) Flip 2→6.

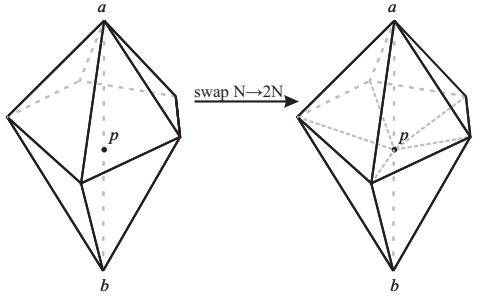


Fig. 3.4: Flip N→2N.

Repairing Flips

Let us suppose that a face *abc* is shared by tetrahedra *abcd* and *abce* and that the face *abc* is non-regular. We have to replace the tetrahedra *abcd* and *abce* (and sometimes one or two neighboring tetrahedra) with two, three, or four new tetrahedra, which do not contain the face *abc* and fill the same space as the original tetrahedra. Again, we have to distinguish three cases:

- flip 2→3: The edge *de* goes through the inside of the face *abc*. Then the tetrahedra *abcd* and *abce* are replaced with three new tetrahedra *abde*, *acde*, *bcde* (see. Fig. 3.5a).
- flip 3→2: The edge *de* does not go through the face *abc*. If a tetrahedron exists, which together with *abcd* and *abce* fulfills the convex hull of points *a, b, c, d, e*, then these three tetrahedra are replaced with two new tetrahedra *abde* and *acde* (see Fig. 3.5b). This operation is inverse to flip 2→3.
- flip 4→4: The edge *de* goes through some edge of face *abc*. Let us suppose that *de* goes through *ab*. If there exist two neighboring tetrahedra *abdx* and *abex*, then the four tetrahedra *abcd*, *abce, abdx* and *abex* are replaced with new four tetrahedra *acde*, *bcde*, *adex* and *bdex* (see Fig. 3.6).

If one of these cases occurs, we call the face *abc* **flippable** and use the proper flip to remove the non-regular face. In any other case the face *abc* cannot be repaired immediately (we say that the face is not flippable). Other faces are repaired and during this process the face *abc* will be repaired too.

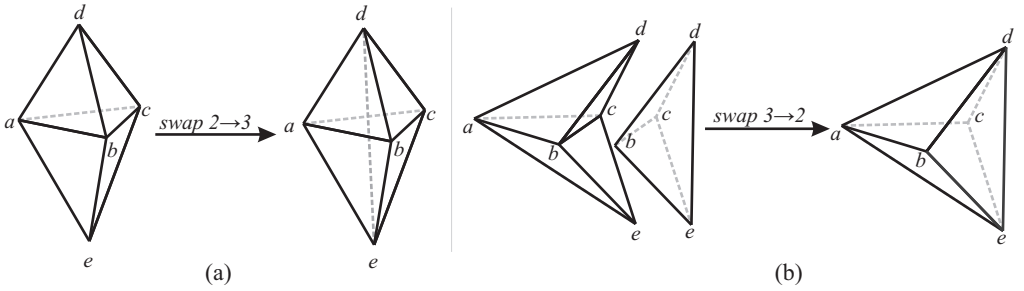


Fig. 3.5: (a) Flip 2→3, (b) Flip 3→2

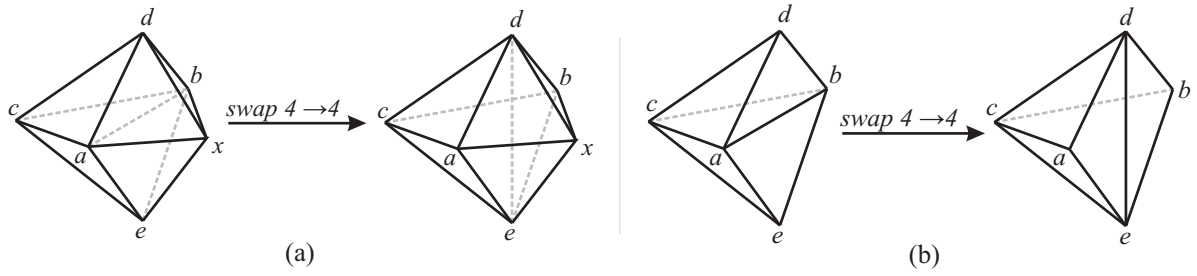


Fig. 3.6: (a) Flip 4→4, (b) Flip 4→4, left part only

Orientation and Regularity Test

The incremental flipping algorithm uses two basic tests – an orientation test and a regularity test. The orientation test of four points *a, b, c, d* tells, on which side of plane passing through points *a, b, c* the point *d* lies. The regularity test of five points *a, b, c, d, e* tells, whether the face *abc* in the tetrahedron *abcd* is locally regular with respect to the point *e*.

Both these tests can be realized as a determinant computation (x_a, y_a, z_a denote the *x, y, z* coordinates of the point *a* and w_a is the weight of *a*).

$$orient(a,b,c,d) = \begin{vmatrix} x_a & y_a & z_a & 1 \\ x_b & y_b & z_b & 1 \\ x_c & y_c & z_c & 1 \\ x_d & y_d & z_d & 1 \end{vmatrix} \tag{3.2}$$

If the result of the test *orient*(*a, b, c, d*) is positive or negative, respectively, the point *d* lies above or bellow, respectively, the plane ρ passing through points *a, b, c*. If the result is equal to zero, *d* lies on the plane ρ.

$$regular(a,b,c,d,e) = orient(a,b,c,d) \cdot \begin{vmatrix} x_a & y_a & z_a & x_a^2+y_a^2+z_a^2-w_a & 1 \\ x_b & y_b & z_b & x_b^2+y_b^2+z_b^2-w_b & 1 \\ x_c & y_c & z_c & x_c^2+y_c^2+z_c^2-w_c & 1 \\ x_d & y_d & z_d & x_d^2+y_d^2+z_d^2-w_d & 1 \\ x_e & y_e & z_e & x_e^2+y_e^2+z_e^2-w_e & 1 \end{vmatrix} \tag{3.3}$$

If the result of the test *regular*(*a, b, c, d, e*) is negative, the face *abc* is locally regular. Otherwise is non-regular.

Redundant points

During a point insertion it is easy to test if the new point is redundant or not. If a point *p* should be inserted into a tetrahedron *abcd* and the test *regular*(*a, b, c, d, p*) is negative then the point *p* is redundant and must not be added into the triangulation.

The situation is much more complex, if due to insertion of the point *p* some other points become redundant. This problem has two parts. First, the points must be recognized as redundant. Second, the redundant points have to be removed from the triangulation. Neither [F93] nor [E92] describe this problem. Fortunately, if we construct the regular triangulation of protein molecule, the redundant points do not appear[1], therefore, we do not need to solve the problem of redundant points.

Pseudocode

The described flipping algorithm is summarized in the following pseudocode.

```
Flipping algorithm                                                    (3.4)
   1. Create the starting tetrahedron
   2. For each point p in S do
   3.    t := tetrahedron, which contains the point p.
   4.    If p is not redundant, connect p into the tetrahedron t
         (using flip 1→4, or 2→6, or N→2N).
   5.    Put the outer faces of new tetrahedra (faces, which do not contain
         the point p) into the list L.
   6.    While L is not empty do
   7.      Remove the face f from the list L.
   8.      If f is not regular and is flippable, use a proper flip
           on f. Put outer faces of new tetrahedra into the list L.
```

[1] If weighted points are interpreted as spheres, then the sphere of a redundant point is totally included in a union of the surrounding spheres. This does not happen with atoms in protein molecules.

Time complexity

A time complexity of the incremental flipping algorithm in 3D is $O\left(n^2\right)$ in the worst case (proof in [E92]), the expected complexity is $O\left(n\sqrt[4]{n}\right)$ (if a walking algorithm is used for a point location).

4. Geometric Tunnel

A protein molecule is simplified to a set of spheres. For many protein molecules the atom coordinates are known – they have been measured using methods as X-Ray diffraction and nuclear magnetic resonance (files with atom coordinates can be downloaded from internet). A radius of a sphere is dependent on a type of atom and it is given by Van der Walls radius.

Because a penetrating molecule is approximated with a sphere, we search tunnels with a circular cross-section. In fact, the tunnel R can be defined by a union of spheres, which do not intersect any atom of protein. Centers of spheres form a **centerline** C_R of the tunnel R (see Fig. 4.1[1]). The sphere with minimal radius is called the **bottleneck sphere** of the tunnel R, the centre of this sphere is called the **bottleneck point** of R and the radius of this sphere is called the **bottleneck radius**.

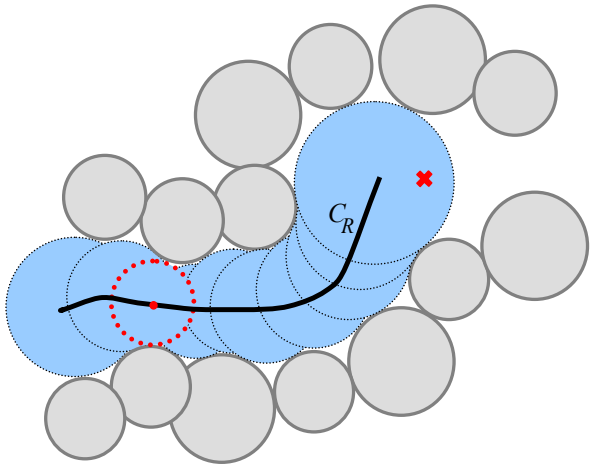


Fig. 4.1: Atoms (gray circles), tunnel (blue circles), tunnel centerline C_R (thick black line), bottleneck circle and bottleneck point (red dotted circle and its center), active place (red cross).

Let us say the tunnel R leads from the point p to the point q. We call the tunnel R **ideal**, if no other tunnel with a bigger bottleneck radius exists. Our approach to tunnel searching is based on the following lemma [M07]:

Lemma: Let S be a set of spheres in 3D. Consider an ideal tunnel R with a centerline from a point p to a point q. The bottleneck point of R is situated on some Voronoi edge of Euclidean Voronoi[2] diagram V(S) or in the point p or q.

Due to this lemma, we can search a tunnel centerline as a sequence of edges in a Euclidean Voronoi diagram – we are sure that there cannot be any tunnel with bigger bottleneck radius and also we are sure that we cannot miss any bottleneck point of the final tunnel.

[1] We are interested in 3D tunnels, but we use figures of 2D tunnels for a simplicity.
[2] The Euclidean Voronoi diagram is also called the Voronoi diagram of spheres or the Weighted Voronoi diagram.

The construction algorithm of Euclidean Voronoi diagram (EVD) is time consuming and hard to implement. For example the time complexity for EVD algorithm described in [Y04] is $O\left(n^3\right)$, where n is the number of spheres.

The differences between atom radii in proteins are not very big (the smallest and the biggest atoms in protein are hydrogen with radius 1,2A and sulfur with radius 1,85A), hence some approximation of EVD should be sufficient. In [M07] Voronoi diagram of points is used. This type of diagram ignores the atom radii completely. Our idea was to try a power diagram (PD). The power diagram reflects weights of points, it can be constructed effectively and it is not so hard to implement as EVD.

5. Tunnel Approximation

In fact, we use rather the dual structure to the power diagram – the regular triangulation. Consider the duality between the regular triangulation and the power diagram:

- vertex in PD ⇔ tetrahedron in RT
- edge in PD ⇔ face in RT
- face in PD ⇔ edge in RT
- cell in PD ⇔ vertex in RT

As mentioned above, an ideal tunnel in EVD can be searched as a sequence of edges in EVD. We approximate EVD with PD and so we search an approximation of the ideal tunnel as a sequence of edges in PD. Because an edge in power diagram is dual with a face in regular triangulation, a sequence of edges in PD corresponds to a sequence of faces in RT. Every two following faces in the sequence determine one tetrahedron, so in a triangulation we can search the tunnel as a sequence of tetrahedra.

5.1 Bottleneck sphere of a face in RT

We have already defined the bottleneck sphere of a tunnel (and also the bottleneck point and bottleneck radius of a tunnel). The bottleneck sphere **of an edge** h (an edge in the power diagram) is the minimal sphere in the tunnel, whose centre lies on the edge h. The bottleneck sphere **of a face** f (a face in the triangulation) is the bottleneck sphere of an edge h, which is dual to the face f (see Fig. 5.1). The meaning of terms "bottleneck point of face" and "bottleneck radius of face" is evident.

In the Euclidean Voronoi diagram the bottleneck point of an edge h has to be located in the point of intersection of the edge h and the plane ρ passing through the three points, which determine the edge h. Or, if this intersection does not exist, the bottleneck point lies in one of the endpoint of the edge h (see Fig. 5.2). We can construct a bottleneck sphere s with a center in the bottleneck point of edge h, such a sphere s is tangent to three or more nearest spheres (atoms, generators of EVD).

However, in the power diagram the bottleneck point of the edge h can be located in one endpoint of the edge h (vertex of PD) even if the intersection of edge h and plane ρ exists (see Fig. 5.3).

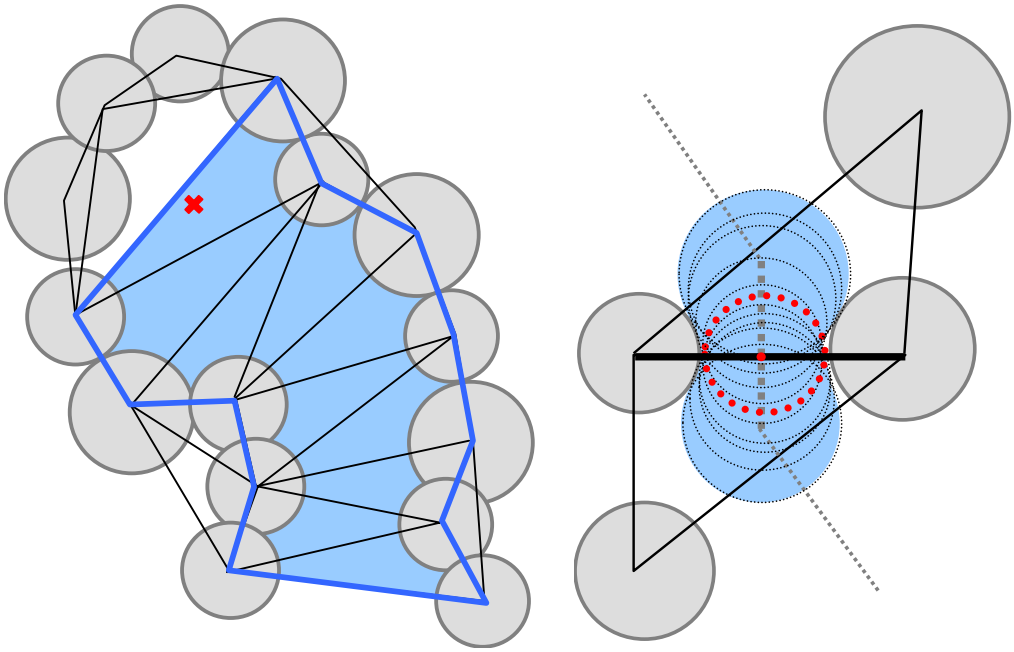


Fig. 5.1: (a) Tunnel in regular triangulation in 2D. The red cross marks the active place, the tunnel is given by the shaded tetrahedra. (b) A bottleneck sphere (thick dotted circle) of a face (thick solid black line) in 2D. Gray dotted lines are edges of power diagram. The centers of blue circles are placed on the thick edge of power diagram.

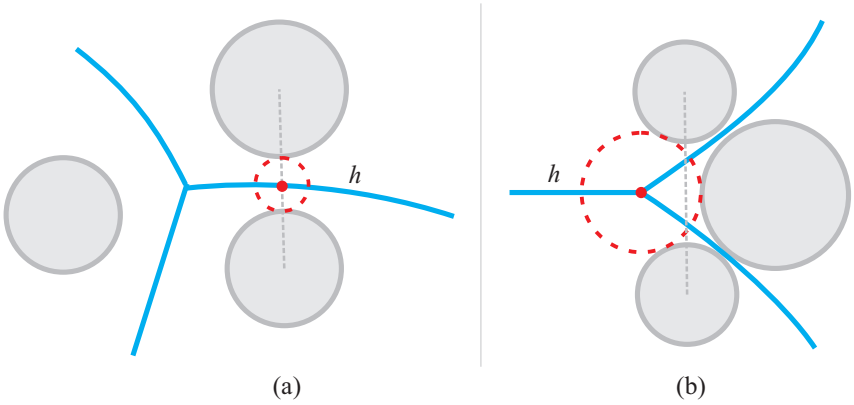


Fig. 5.2: (a) The bottleneck point (dashed circle) of *h* lies in the point of intersection of the edge *h* and the plane passing through neighboring atoms, (b) The bottleneck point in the endpoint of *h*.

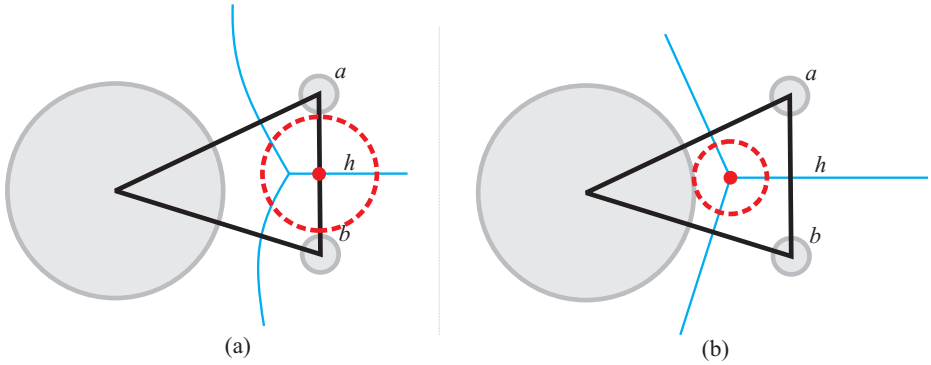


Fig. 5.3: Location of the bottleneck circle (a) In EVD, (b) In power diagram.

Because we use regular triangulation instead of power diagram, we do not know the power diagram vertices. But the vertices can be easily computed – a vertex in PD is an orthogonal center of tetrahedron in RT.

Let us have two tetrahedra *abcd* and *bcde* with a common face *bcd*. The bottleneck point and the weight of the face *bcd* are computed as follows:

Bottleneck computation (5.1)

1. Compute the orthogonal sphere s_1 of the tetrahedron *abcd*.
2. Compute the orthogonal sphere s_2 of the tetrahedron *bcde*.
3. If the face *bcd* separates centers of spheres s_1 and s_2, compute sphere s_3 with its centre in the intersection of the plane passing through the points *b, c, d* and the line joining centers of s_1 and s_2. Else set the radius of s_3 to infinity and the centre of s_3 to an arbitrary position.
4. Set the radii r_{s1}, r_{s2} and r_{s3} of spheres s_1, s_2, s_3 as follows[1]:

$$r_{s1} = \min\{|as_1| - r_a, |bs_1| - r_b, |cs_1| - r_c, |ds_1| - r_d\},$$
$$r_{s2} = \min\{|bs_2| - r_b, |cs_2| - r_c, |ds_2| - r_d, |es_2| - r_e\},$$
$$r_{s3} = \min\{|bs_3| - r_b, |cs_3| - r_c, |ds_3| - r_d\},$$

5. The bottleneck sphere of the face *bcd* is the smallest sphere of s_1, s_2 and s_3. We use the radius of the bottleneck sphere as a weight of the face *bcd*.

However, if we construct the bottleneck sphere of the face *bcd* in the regular triangulation in this way, the bottleneck sphere can still have intersections with some atoms (see Fig. 5.4). We call this intersection a **tunnel error**.

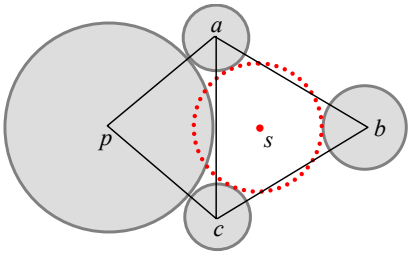


Fig. 5.4: The center of the circle *s* lies in the orthogonal center of a triangle *abc*. The circle *s* does not have any intersection with the atoms *a, b, c*, but it still intersects the atom *p*.

Therefore we distinguish two types of tunnels:

- a pessimistic tunnel – the tunnel must not contain any intersection with any atom.
- an optimistic tunnel – the tunnel can contain small intersections with some atoms.

5.2 Pessimistic Tunnel

In the pessimistic tunnel computation we follow the bottleneck computation algorithm, but we add one more step between the steps 4 and 5. After we have computed the centers of spheres s_1, s_2 and s_3 and we have set their radii (step 4), we compute an upper estimation of the tunnel error for each sphere s_1, s_2 and s_3 and reduce the radii of these spheres by the error estimate. Then we continue with step 5. Now we briefly describe the error estimate computation (a full derivation is in appendix A).

[1] A sphere with a center in a vertex of power diagram and tangent to the nearest three or more generators of PD does not have to exist in power diagram. r_a denotes the radius of the atom *a*, i.e. $r_a = \sqrt{w_a}$.

Let us have a tetrahedron *abcd*, a sphere *s* with a center in the orthogonal center of *abcd* and with a radius, which is set accordingly to the step 4 in the algorithm (5.1). Further we assume that the weight of the point *a* is greater then or equal to the weights of points *b, c, d*. Then the maximal possible tunnel error of sphere *s* fulfills an inequality (5.2)

$$possible\_error \le r_{\max} + r_s - \sqrt{r_s^2 + 2r_a r_s + r_{\max}^2} = F\left(r_s, r_a, r_{\max}\right), \qquad (5.2)$$

where r_s is the radius of the sphere *s*, r_a is the radius of the point *a* and $r_{\max}$ is the radius of the point with the biggest weight in the triangulation. If we reduce the radius of the sphere *s* by $F\left(r_s, r_a, r_{\max}\right)$, the resulting sphere cannot intersect any point in the triangulation (atom in protein).

Let us note that the tunnel error estimation is dependent on the radii $r_s, r_a, r_{\max}$. In other words, the error estimate is dependent on the local geometry, therefore, it is evaluated for each tunnel sphere again. The time complexity of one error estimate calculation is clearly $O(1)$.

5.3 Optimistic Tunnel

In the optimistic tunnel computation we just follow the algorithm (5.1) without any extra correction of the tunnel spheres radii and so the resulting tunnel can intersect some points in a triangulation. But an analyses of the error function *F* proves that the size of the maximal intersection between a tunnel and any point of a triangulation has to be equal to or smaller then $r_{\max} - r_{\min}$, where $r_{\max}$, $r_{\min}$, respectively, is the radius of the biggest, smallest point, respectively, in a triangulation. So the upper bound of tunnel error is independent on the protein geometry. Let us note that we do not reduce the tunnel spheres radii by $r_{\max} - r_{\min}$.

5.4 Delaunay Tunnel

In section 6 the comparison of our tunnels and tunnels defined in [M07] is given. We call this type of tunnels the Delaunay tunnel and we describe it in this section.

Medek et al. use simple Delaunay triangulation of a set of points for tunnel computation. The main difference between Medek's and our approach is the computation of a weight of a face. The weight of a face *abc* (shared by tetrahedra *abcd* and *bcde*) is computed as follows:

Delaunay bottleneck computation (5.3)

1. Compute a circumscribed sphere s_1 of the tetrahedron *abcd*.
2. Compute a circumscribed sphere s_2 of the tetrahedron *bcde*.
3. If the face *bcd* separates the centers of spheres s_1 and s_2, compute the sphere s_3 tangent to the spheres *b, c, d*. The centre of s_3 lies in the plane passing through points *b, c, d*.
   Else set the radius of s_3 to infinity and the centre of s_3 to an arbitrary position.
4. Set radii r_{s1}, r_{s2} and r_{s3} of the spheres s_1, s_2, s_3, so that s_1, s_2 and s_3 do not intersect any sphere in vertices of tetrahedron *abcd*, *bcde* and their neighboring tetrahedra.
5. Bottleneck sphere of the face *bcd* is the smallest sphere of s_1, s_2 and s_3. The radius of the bottleneck sphere is used as a weight of the face *bcd*.

Let us note that this tunnel type is independent on a triangulation type. In the Delaunay triangulation of a set of spheres it produces ideal tunnels, in other triangulations (Delaunay triangulation of a set of points, regular triangulation, etc.) it produces only approximation of ideal tunnels.

5.5 Algorithm for Tunnel Searching

Our current interest is a tunnel radius – i.e. we search the tunnel with the maximal bottleneck radius. Therefore we can use a simple greedy algorithm.

The algorithm distinguishes two types of tetrahedra – unknown tetrahedra and expanded tetrahedra. An unknown tetrahedron is a tetrahedron, to which the ideal tunnel is not known yet. The expanded tetrahedron is a tetrahedron, to which an ideal tunnel is already known. The triangulation is traversed from the tetrahedron containing the active place to the triangulation border. In each step the algorithm goes throw a face with the biggest weight from an expanded tetrahedron to an unknown tetrahedron. The algorithm stops, when it finds a tetrahedron, which lies on the triangulation border. The algorithm is described by the following pseudocode.

```
Tunnel computation                                                    (5.3)
   1. t := tetrahedron, which contains the active place.
   2. Mark t as expanded, for each unknown neighbor n of t do
   3.      Insert a triplet (target tetrahedron n, previous tetrahedron t,
           a weight w of the face between n and t) into the heap H
           according to the weight w.
   4. Remove a triplet Q from the top of H. t := Q.target_tetrahedron.
   5.      If t has less then 4 neighbors, continue with step 8.
   6.      Else if t is already expanded, continue with step 4.
   7.      Else if t is unknown, remember the Q.previous_tetrahedron for t
           and continue with step 2.
   8. Reconstruct the tunnel - traverse the triangulation back from
      the last tetrahedron (on the triangulation border) to the starting
      tetrahedron (with the active place) using the saved information about
      the previous tetrahedron.
```

6. Experimental Results

In this section we give a comparison of various tunnel types. Also we show the typical number of tetrahedra in the regular triangulations of protein molecules and time for their computation. A few pictures of tunnels are showed in appendix B.

6.1 Comparison of Bottleneck Radii

Three different proteins were used for tests – their basic characteristics are given in the Table 6.1.

Protein	# atoms	# atoms according to type H, N, S, C
awj	1250	600, 132, 106, 1, 407
dbja	2332	0, 418, 412, 7, 1491
dhaa	4710	2319, 426, 401, 9, 1551

Tab. 6.1: The basic characteristics of the three tested proteins.

For each protein, two hundreds pseudo-active places were generated. We have got two requests on a pseudo-active place location – it must not lie inside an atom and it should not lie near the protein surface.

Four different types of tunnel were tested: the optimistic and pessimistic tunnel in regular triangulation (see the section 5.2 and 5.3) and the Delaunay tunnel in regular and Delaunay triangulation (see the section 5.3). These four tunnels were searched for each pseudo-active place. We observed a few simple statistics on the resulting tunnels.

In Tab. 6.2 we show the average bottleneck radius. The Delaunay tunnels in Delaunay triangulation achieve the best result, but the differences are only hundredths of angstrom[1].

	Average bottleneck radius [A]			
Protein	Optimistic in RT	Pessimistic in RT	Delaunay in RT	Delaunay in DT
1awj	1,048	0,927	1,073	1,080
1dbja	0,875	0,828	0,877	0,879
dhaa	0,899	0,788	0,928	0,942

Tab. 6.2: Average bottleneck radii of four different tunnel types in three proteins.

Further we focus on comparison of couples of tunnels (of different types) to the same active-place. The Table 6.3 shows the average ratios of tunnels bottleneck radii, standard deviations are in the brackets. The ratios of optimistic and Delaunay tunnels are near to 1 and their standard deviations are only a few tenths, so the radii of the corresponding tunnels are very close. The bottleneck radii of pessimistic tunnels are slightly narrower.

	Average ratio of tunnels bottleneck radii, (standard deviation)		
Protein	Optimistic in RT/ Delaunay in RT	Optimistic in RT / Delaunay in DT	Delaunay in RT/ Delaunay in DT
1awj	1,022 (0,36)	1,059 (0,73)	1,020 (0,22)
1dbja	0,999 (0,11)	0,996 (0,09)	0,999 (0,04)
dhaa	0,993 (0,2)	1,003 (0,3)	0,985 (0,35)
	Pessimistic in RT/ Optimistic in RT	Pessimistic in RT/ Delaunay in RT	Pessimistic in RT/ Delaunay in DT
1awj	0,884 (0,06)	0,940 (0,69)	0,940 (0,69)
1dbja	0,943 (0,02)	0,939 (0,09)	0,939 (0,09)
dhaa	0,875 (0,07)	0,869 (0,2)	0,881 (0,28)

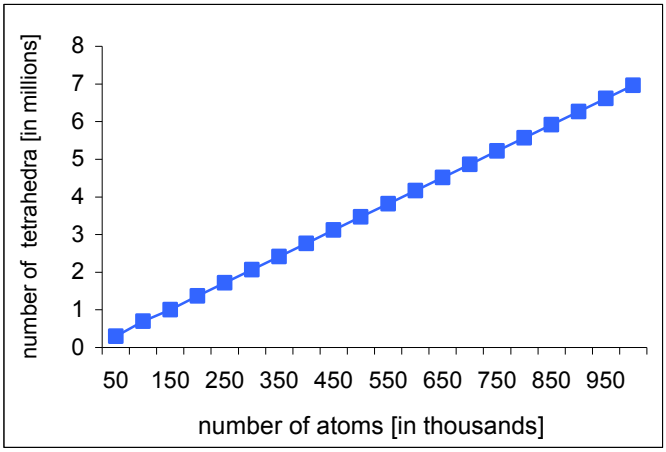
Tab. 6.3: Average ratio and standard deviation of tunnels bottleneck radii.

Although the bottleneck radii are often very similar, sometimes are the differences bigger. In the protein dhaa, 10% of optimistic tunnels are 1,25 times wider then the corresponding Delaunay tunnels. We might use this fact and combine both strategies. We can compute the regular triangulation of protein, then search the optimistic and Delaunay tunnels and simply choose the one with the bigger bottleneck radius.

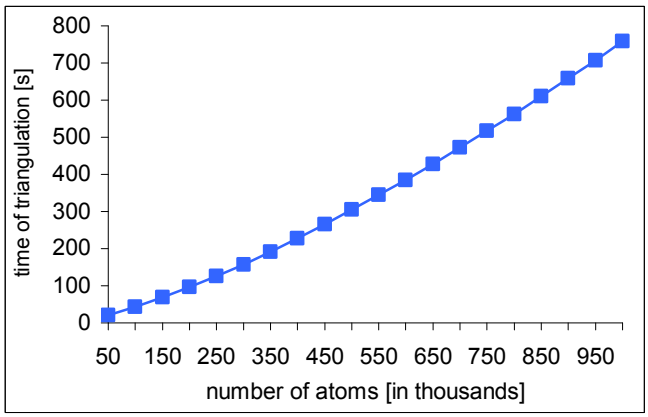
6.2 Time and Space Complexity

In this section we show the typical space complexity of triangulation of protein molecules and also the time complexity of the flipping algorithm (see section 3) for the triangulation of protein. The graph 6.4 shows the dependence of number of tetrahedra on the number of atoms – the dependence is linear for a typical protein. The time complexity of the flipping algorithm (see graph 6.5) is greater then $O(n)$ but smaller then $O\left(n^{4/3}\right)$, where n denotes the number of atoms in the protein.

[1] 1 angstrom = 1A = 1.0×10^{-10} meters



Graph 6.3: The dependence of the number of tetrahedra on the number of atoms.



Graph 6.4: The time complexity of the flipping algorithm.

7. Conclusion

In this paper we have described the regular triangulation and the known algorithm for its computation. Further we have designed two methods for tunnel computation in regular triangulation: the pessimistic tunnels, which cannot contain any error, and optimistic tunnels, which may contain small errors but are wider than the pessimistic tunnels. The results of our solution and the comparison with the existing method are described. The proposed solution does not bring a big improvement, but it can be used in combination with the existing method.

References

[A91] Aurenhammer, F.: Voronoi Diagrams – A Survey of a Fundamental Geometric Data Structure. ACM Computing Surveys 23, 345-405, 1991.

[B01] Bajaj, C.L., Pascucci, V., Shamir, A., Holt, R.J., and Netravali, A.N.: Dynamic maintenance and visualization of molecular surfaces. Discrete Applied Mathematics 127, 23–51, 2001.

[D01] O. Devillers, S. Pion, M. Teillaud: Walking in a Triangulation. ACM Annual Symposium on Computational Geometry 17, 106-114, 2001.

[DAM] Damborský, J.: Proteinové inženýrství: od laboratorních testů k biotechnologiím. http://www.otevrena-veda.cz/ov/users/Image/default/C2Seminare/MultiObSem/106.pdf (in czech only).

[E92] Edelsbrunner, H., Shah, N. R.,: Incremental Topological Flipping Works for Regular Triangulations. ACM Annual Symposium on Computational Geometry 8, 43-52, 1992.

[M07] Medek, P., Beneš, P., Sochor, J.: Computation of tunnels in protein molecules using Delaunay triangulation. Journal of WSCG 15, 107-114, 2007.

[P06] Petřek, M., Otyepka, M., Banáš, P., Košinová P., Koča, J., Damborský J.: CAVER: a new tool to explore routes from protein clefts, pockets and cavities. BMC Bioinformatics, 2006.

[R05] Ryu, J., Kim, D., Cho, Y., Park, R. Kim, D.-S: Computation of molecular surface using Euclidean Voronoi Diagram. Computer-Aided Design & Applications 2, 439-448, 2005.

Appendix A

Lemma

A set of weighted points *S*, a power diagram PD(*S*) and a regular triangulation RT(*S*) of the set *S* are given. Let us have tetrahedra *abcd* and $bcde \in RT(S)$ with the common face *bcd* and an edge $h \in PD(S)$ such that the edge *h* is dual to the face *bcd*. Further a sphere *s* is given, the center of the sphere *s* lies on the edge *h*, and the radius of *s* is $r_s = \min\{|as|-r_a, |bs|-r_b, |cs|-r_c, |ds|-r_d, |es|-r_e\}$ (where r_a^2 is the weight of the point *a* and |*as*| denotes the Euclidean distance between points *a* and *s*). We assume that the weight of the point *b* is greater then or to the weights of points *a, c, d, e*. Then the $error = \max\{r_s + r_p - |sp|; p \in S\}$ fulfils the inequality

$$error \le r_{\max} + r_s - \sqrt{r_s^2 + 2r_b r_s + r_{\max}^2} \qquad \text{(A1)}$$

where r_{max} is the radius of the point $\in S$ with the biggest weight.

Proof

The edge *h* is the intersection of the three bisectors of points *b, c, d*, which means that the power distance of any point $x \in h$ has the same power distance from the points *b*, *c*, *d*.

$$\pi_b(x) = \pi_c(x) = \pi_d(x) = w \qquad \text{(A2)}$$

Further it holds that the power distance of $x \in h$ from any point $p \in S - \{b,c,d\}$ is greater than or equal to *w*:

$$\pi_p(x) \ge w = \pi_b(x) = \pi_c(x) = \pi_d(x) \qquad \text{(A.3)}$$

The center of *s* lies on the edge *h*:

$$\pi_p(s) \geq w = \pi_b(s) = \pi_c(s) = \pi_d(s) \tag{A.4}$$

We assume that the weight of the point *b* is greater then or equal to the weights of points *c, d*. Then we determine the radius of sphere *s* as:

$$r_s = |bs| - r_b \tag{A.4}$$

Now the sphere *s* cannot intersect the spheres *a, b, c, d, e*. We are interested, how "deeply" can *s* penetrate into any $p \in S$:

$$error_p = r_p + r_s - |ps| \tag{A.5}$$

The equation (A.3) implies:

$$\pi_p(s) = |ps|^2 - r_p^2 \geq |bs|^2 - r_b^2 = \pi_b(s)$$

$$|ps|^2 \geq |bs|^2 - r_b^2 + r_p^2$$

$$|ps| \geq \sqrt{|bs|^2 - r_b^2 + r_p^2} \tag{A.6}$$

If we replace $|bs|$ in (A.6) with (A.4), we obtain:

$$|ps| \geq \sqrt{|bs|^2 - r_b^2 + r_p^2} = \sqrt{(r_s + r_b)^2 - r_b^2 + r_p^2} = \sqrt{r_s^2 + 2r_b r_s + r_p^2} \tag{A.7}$$

We substitute (A.7) to (A.5) and so get the upper estimation of the $error_p$:

$$error_p = r_p + r_s - |ps| \leq r_p + r_s - \sqrt{r_s^2 + 2r_b r_s + r_p^2} \tag{A.8}$$

In expression (A.8) an unknown point *p* still remains. If we replace the unknown r_p by radius r_{max} (radius of the point $\in S$ with the biggest weight) in the right side of expression, we get:

$$error_p = r_p + r_s - |ps| \leq r_{\max} + r_s - \sqrt{r_s^2 + 2r_b r_s + r_{\max}^2} \tag{A.9}$$

The inequality (A.1) is proved.

Appendix B

The pictures show the protein 1lin, its triangulation and the tunnel in triangulation.

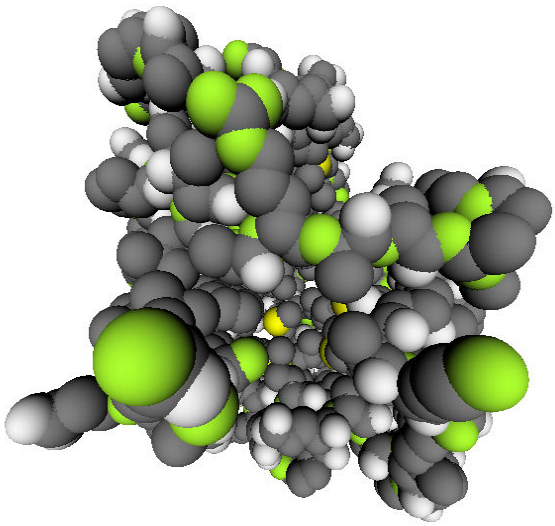


Fig. B.1: Protein 1lin.

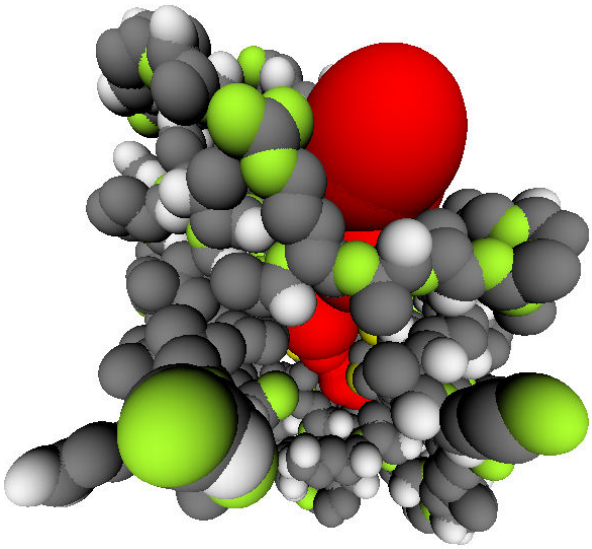


Fig. B.2: Protein and the tunnel (red).

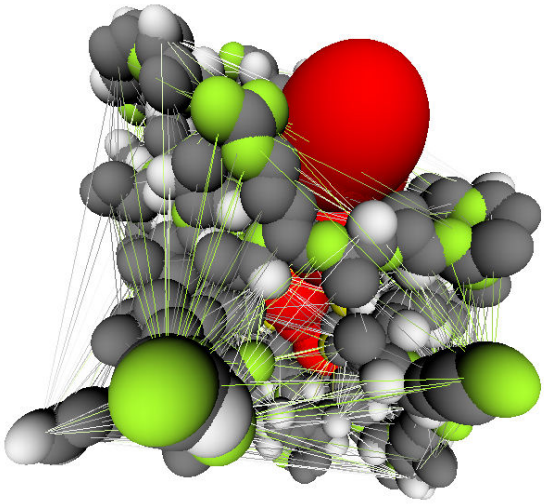


Fig. B.3: Protein, its triangulation and the tunnel.

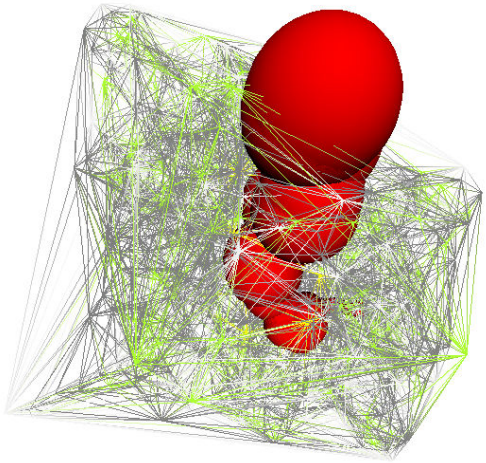


Fig. B.4: The triangulation and the tunnel (red).