# Intersurf: dynamic interface between proteins

Nicolas Ray, Xavier Cavin, Jean-Claude Paul, Bernard Maigret

# Dynamic interface between proteins ⋆

Nicolas Ray [a], Xavier Cavin [a,*], Jean-Claude Paul [a], Bernard Maigret [b]

[a] *Project Isa, Inria Lorraine, 615 rue du Jardin Botanique, BP 101, 54602 Villers-les-Nancy Cedex, France*

[b] *Project Edam, UMR CNRS/UHP 7565, Universite Henri Poincare - Nancy 1, BP 239, 54506 Vandoeuvre-les-Nancy Cedex, France*

**Abstract**

Protein docking is a fundamental biological process that links two proteins. This link is typically defined by an interaction between two large zones of the protein boundaries. Visualizing such an interface is useful to understand the process thanks to 3D protein structures, to estimate the quality of docking simulation results, and to classify interactions in order to predict docking affinity between classes of interacting zones. Since the interface may be defined by a surface that separates the two proteins, it is possible to create a map of interaction that allows comparisons to be performed in 2D. This paper presents a very fast algorithm that extracts an interface surface and creates a valid and low-distorted interaction map. Another benefit of our approach is that a pre-computed part of the algorithm enables the surface to be updated in real-time while residues are moved.

*Key words:*
Protein docking, interface extraction, interaction map, visualization

## Introduction

Molecular recognition is involved in all biological processes and is one of the key feature for the communications between and within cells. Detailed ener-

⋆ Supported by ARC Docking (Cooperative Research Initiative) Inria grant.

* Corresponding author.
*Email addresses:* Nicolas.Ray@loria.fr (Nicolas Ray), Xavier.Cavin@loria.fr (Xavier Cavin), Jean-Claude.Paul@loria.fr (Jean-Claude Paul), Bernard.Maigret@edam.uhp-nancy.fr (Bernard Maigret).

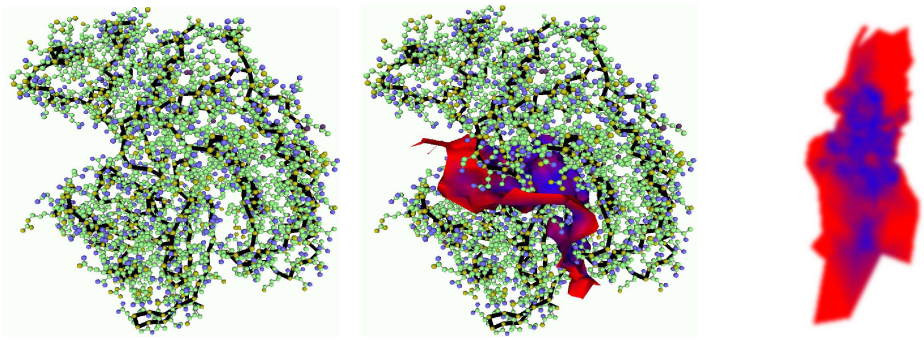

Fig. 1. Left: a pair of proteins docked together. Middle: visualization of this interaction using our algorithm. Right: corresponding map showing Van der Walls interactions.

getic and structural knowledge of interactions between biomolecules is fundamental to understand the complex regulatory and metabolic pathways that occur in living organisms and also to design drugs for blocking or modifying these interactions. In the present post-genomic era, research increasingly focuses on proteomics and protein-protein interaction networks as protein-protein interactions play a central role in numerous processes in the cell and are the key for controlling most of the cellular regulation pathways. The knowledge of the structures and properties of contact surfaces, forces involved in protein-protein interactions, kinetic and thermodynamic parameters of these reactions are therefore of crucial interest. The properties of protein contact surfaces depend on their function so they represent prospective targets for a new generation of drugs. Experimental and computational efforts are therefore devoted to large-scale generation and analysis of information derived from 3D structures and dynamics of proteins, with the goal of scientific and commercial breakthrough in drug discovery.

A large number of protein structures has been experimentally determined and deposited into the Protein Data Bank (Berman et al., 2000), and this number will grow rapidly with the development of new high-throughput technologies in structural proteomics. However, only a small fraction of numerous protein-protein complexes has been experimentally characterized so far. In this context, theoretical prediction of protein-protein complexes is becoming critically important in structural biology. Computational generation of protein structures *via* modelling by homology and threading, and by *ab initio* prediction, and docking of a protein structure with potential interacting partners are two related steps in what is now considered as computational proteomics. In recent years, several groups have developed a variety of tools in an attempt to solve the so-called protein-protein docking problem, that is, the prediction of the geometry of a complex from the atom co-ordinates of its uncomplexed constituents. Early protein-protein docking algorithms used exclusively a geometric criterion based on the shape complementarity, as it is well known that molecular shape plays a critical role in the binding of two proteins. Over the

years, the notion of shape complementarity has been confirmed by inspection of a large number of complexed structures in the protein data bank. Consequently, shape complementarity has been used as a prime consideration in docking approaches that take into account entire molecular surfaces rather than strictly active site regions.

Therefore, several protein-protein docking projects are concerned with the surface matching problem, in order to determine the most favourable interface (contact surface) between two macro-molecules, and many algorithms have been proposed in this respect (for a review, see Halperin et al. (2002)). In most of these algorithms a necessary preliminary step is to delineate and to characterize the interface between the two partners. The problem is here to depict conveniently the properties of a surface which can be very complicated. For that purpose, 2D projections and maps are useful, because they can show the entire surface and the distribution of many properties simultaneously. Many parts of the surface that can not be seen in 3D from a given point of view may be seen easily in a 2D projection. On the other hand, any projection of a 3D surface onto a 2D map introduces distortions of the 3D object, sometimes making such 2D maps useless.

A well-known program for computing a 2D projection of a protein-protein interface is ADSI, the interface-mapping module of ADS (Gabdoulline and Wade, 1996), whose output can be visualized by MolSurfer (Gabdoulline et al., 1999). MolSurfer was designed to display the location of interfacial voids and "hot-spot" patches of surface properties of interest and the identification of patterns in the distribution of interaction properties. However, despite its high interest, the surface quality may be very poor and severe distortions occur on the 2D representation when the interface is highly distorted, due to the greedy algorithm used to recover both the surface and the map.

We present in this paper a new algorithm which is able to perform such tasks in a very fast and efficient way, as illustrated by Figure 1.

## 1 Interface extraction

The surface used to represent the interaction may be defined as the set of points that are equidistant to each protein. This definition ensures that the surface will separate the two proteins, and that the distance between a given point of the surface and each protein will be the same. Using this definition, finding the interface is equivalent to extract the iso-0 surface of the function:

$$f(x) = dist(x, protein_A) - dist(x, protein_B)$$

where $dist(x, p)$ is the minimal distance between the point $x$ and the protein $p$.

This problem is usually solved using a marching cube algorithm but, since only one surface has to be found, greedy algorithms may be used to speed-up the process. Gabdoulline and Wade (1996) propose to start from a seed and to expand the surface around. This solution is very efficient when the surface is flat enough (example: docking based on hydrophobic links), but fails to deal with complex surfaces (example: a cavity filled by a residue).

We propose a faster and more robust approach using a non-structured 3D mesh, namely a Delaunay tetrahedralization (if you are not familiar with Delaunay, a good review has been written by Bern and Eppstein (1995)). The first step of our algorithm is a 3D Delaunay tetrahedralization computation using atoms as vertices. This volume representation enables both a fast detection of atoms that are close to each other and a decomposition of the volume into tetrahedra that are easier to manipulate. This tetrahedralization is then used to detect the volume between proteins (as a set of tetrahedra) and to extract an iso-surface (the interface) in this volume.

The remaining of this section is organized as follows: a description of the 3D Delaunay tetrahedralization is given, followed by the marching tetrahedra algorithm used to extract iso-surfaces. The dynamic extraction (allowing residues to be moved) of interfaces using these algorithms is finally discussed.

### *1.1 Tetrahedralization*

**Definition.** The Delaunay tetrahedralization is defined as the dual (switch faces/edges and vertices/cells) of a 3D Voronoi diagram. Given a set of seeds (in our case, seeds are atoms), the Voronoi diagram is a segmentation of space into cells such that all points included in the same cell are sharing the same nearest seed.

**Property.** In a Delaunay tetrahedralization all spheres having an edge of the tetrahedralization as a diameter will contain no vertices. This property ensures that the shape of each cell (tetrahedron) is as good as possible (no face corner angle can be more than $\pi/2$). This makes it suitable to represent a volume defined by a set of points.

In our application, a Delaunay tetrahedralization is created using atoms positions as vertices, so interface cells will be very easy to detect. Figure 2 shows an example of such a tetrahedralization.

**Complexity.** The construction of a Delaunay tetrahedralization is an iterative process. Inserting a new vertex requires to find the cell containing the new point, to tetrahedralize it, and to apply a set of local operations to ensure that the tetrahedralization remains a Delaunay tetrahedralization. The most time

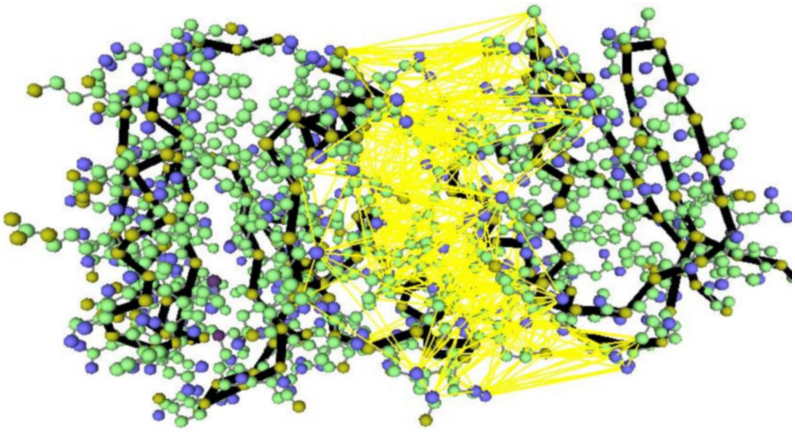

Fig. 2. Delaunay tetrahedralization of the empty space between two proteins.

consuming step in large datasets is known to be the search of the containing cell, which has a complexity of $O(\log(n))$, where $n$ is the number of vertices. The whole algorithm complexity is then $O(n \log(n) + k)$ ($n$ insertions are required), where $k$ is the complexity of the result, that is $O(n)$ for proteins (thanks to homogeneous repartition of seeds). However, in our application, atoms are inserted in the order of the peptidic chain, so starting the searching step from the last inserted vertex bounds this step complexity by the maximum number of cells between atoms of two adjacent residues (which is constant). This remark makes the complexity of the tetrahedralization equal to $O(n)$.

### *1.2 Surface extraction*

The interface surface is extracted in two steps. The first step determines the volume between the two proteins. This operation is straightforward using the Delaunay tetrahedralization: all tetrahedra having vertices in both proteins belong to the interface volume. The second step extracts a surface from this volume using a marching tetrahedra algorithm. This algorithm works as follows: for each cell, a local surface (composed by one or two triangles) is extracted depending on the cell configuration. The cell configuration is defined by the 4 atoms and the protein they belong to. So, the number of possible configurations is $2^4 = 16$ (4 vertices and 2 possible proteins for each); it is then possible to pre-compute the triangles to be extracted for each configuration (see Figure 3 for an example).

By construction, any vertex of the resulting surface is constrained to be placed on an edge of the tetrahedralization; however, it can be be placed anywhere on this edge. As shown by Figure 3, our algorithm sets vertices in the middle of the segment bounded by Van Der Wall balls. It is also possible to move the surface vertices along the tetrahedralization edges to smoothly slide the surface between the molecular surfaces of each proteins (see Figure 4).

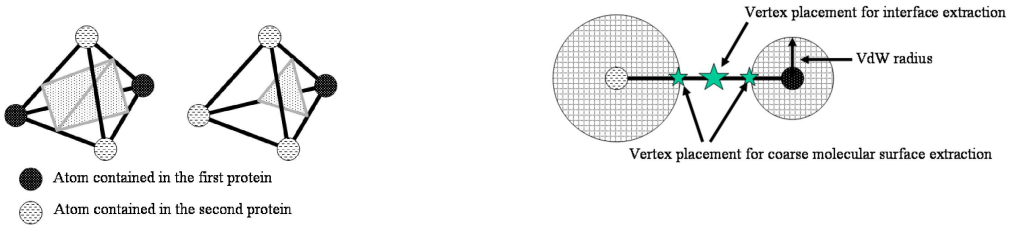


Fig. 3. Left: two configurations and their associated triangulation. Right: location of the surface vertices.

The proposed algorithm enables to extract a set of triangles at the interface of the proteins. To be manipulated as a continuous surface, access to the neighboring triangles of a given triangle has to be made easy. This information is added by an algorithm that retrieves topological information from the geometry. This operation complexity is known to be $O(n\log(n))$, where $n$ is the number of triangles. However, it is possible in our context to remove the $\log(n)$ factor needed to match vertices sharing the same position: to do so, vertices are created from the tetrahedralization before launching the marching tetrahedra algorithm.

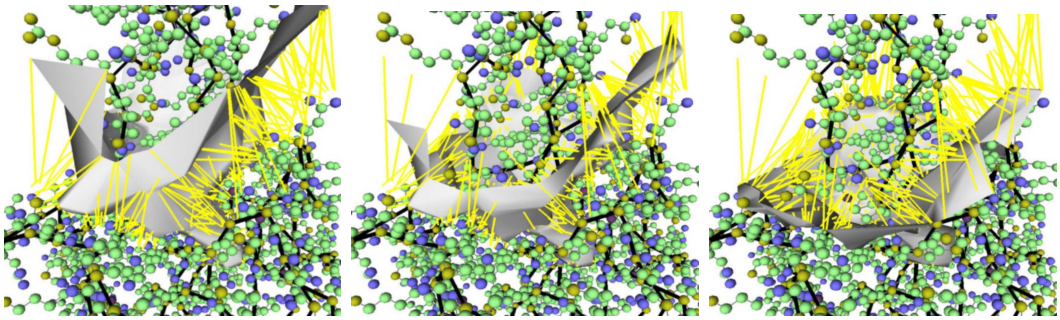

Fig. 4. Left: the surface is snaped to the first protein. Middle: the surface is equidistant to both protein VdW surfaces. Right: the surface is snaped to the first protein.

### *1.3 Dynamic modifications*

The surface extraction algorithm is very fast (it takes 0.5 to 2 seconds on a standard PC), but not enough to enable interactive surface extraction. We present here a way to interactively modify the surface when a part (typically a residue, as shown by Figure 5) of the protein is moved. To do this we will see how to speed-up the most time consuming step of the algorithm, based on the following considerations:

- The tetrahedralization has a complexity of $O(n)$.

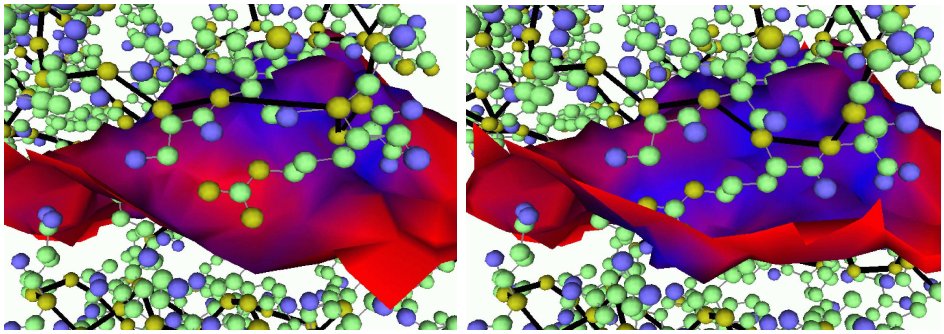

Fig. 5. Left : the original configuration. Right: the surface is locally updated while a residue is moved to maximize the VdW forces.

- The surface extraction complexity is proportional to the number of tetrahedra of the interface. Since these tetrahedra are on the surface (an area) of the protein (a volume), their number is proportional to $n^2/3$. Note that adding the topology information to the surface has the same complexity ($O(n^2/3)$).

The most time consuming step is the tetrahedralization. Fortunately, it is possible to update a tetrahedralization while moving some vertices in real-time. To do this, a vertex to be moved is first removed from the tetrahedralization, and then inserted again with its new position. Adding and removing a vertex are two fast local operations, so it is possible to update the surface in real-time. Notice that it is possible to update the surface only in new or modified tetrahedra, this solution should be a bit faster but much harder to implement.

## 2 Mapping attributes on the surface

The interface surface between two proteins is very useful to visualize the interaction. However, the interaction is not completely defined by the geometry of this surface. To improve the information given by the surface, it is possible to "paint" it with several attributes that characterize the potential interactions both qualitatively and quantitatively, as illustrated by Figure 6.

To bind attributes on the surface, remember that each vertex of the interface fits on a segment (an edge of the tetrahedralization) defined by an atom of each protein as extremities. The Delaunay tetrahedralization ensures that these atoms are the nearest ones of the interface vertex. This property makes it very easy to extract local informations about docking possibilities around each vertex of the interface. This local information is used as an attribute, that is binded as a color to the interface surface. In our experiments, we have tested:

- A quantitative attribute: the distance of the surface to the proteins. On each vertex of the interface, a color scale from red (far) to blue (close) is

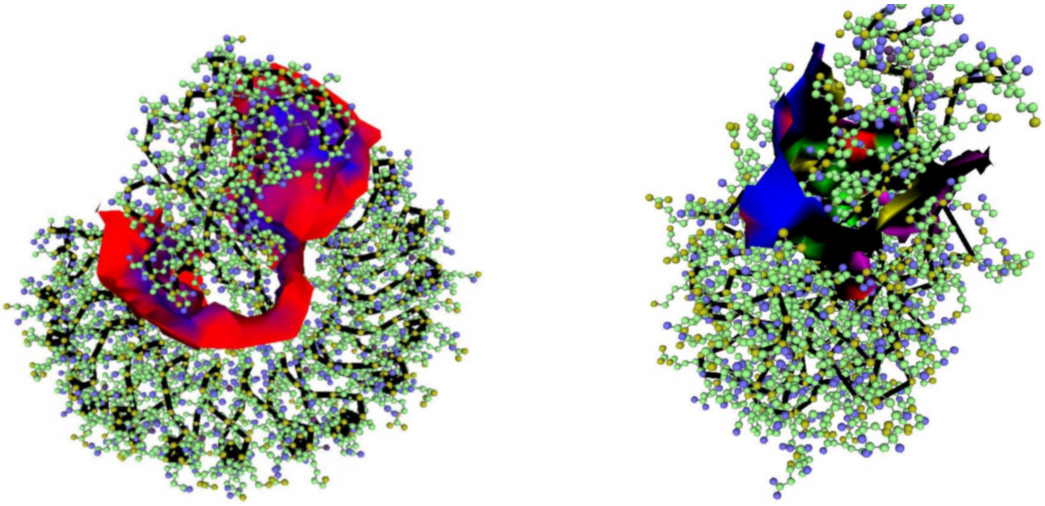

Fig. 6. Two interfaces with attributes as color. Left: "distance to proteins" attribute. Right: "kind of possible interactions between residues" attribute.

used to represent the distance between the two nearest atoms. During an interactive docking session, it might also be intersting to prevent proteins overlaps by adding a third symbolic color for negative distances.

- A qualitative attribute: the kind of residues interaction that can occur. On each vertex of the interface, a symbolic color is used to represent the kind of interaction that is likely to occur between the two nearest residues. [1] The color coding that has been used in this paper is the following: hydrogen link (light blue), hydrophobic link (green), Pi...X (purple), Pi...Pi (yellow), same charge (blue), opposite charges (red), none (black).

Moreover, as in the MolSurfer application, electrostatic potential or hydrophilic links can also be used as attributes. Finally, to enable to compare two surfaces only through their 2D maps (see next Section), it is also interesting to add attributes that represent the geometry of the surface. For instance, mapping the surface curvature or average geodesic distance to other points of the surface enables cavities to be detected.

## 3 Map generation

The interface is represented as a 3D surface with attributes mapped on it. To make this information easier to be automatically manipulated by algorithms, it is better to use a 2D map of the surface representing the attributes. In this context, real-time dynamic updates of the 2D map are not of a crucial

[1] Our implementation makes the assumption that the most important contacts are represented by segments in the tetrahedralization. It would be possible to take into account more distant interactions using a neighborhood of the atoms.

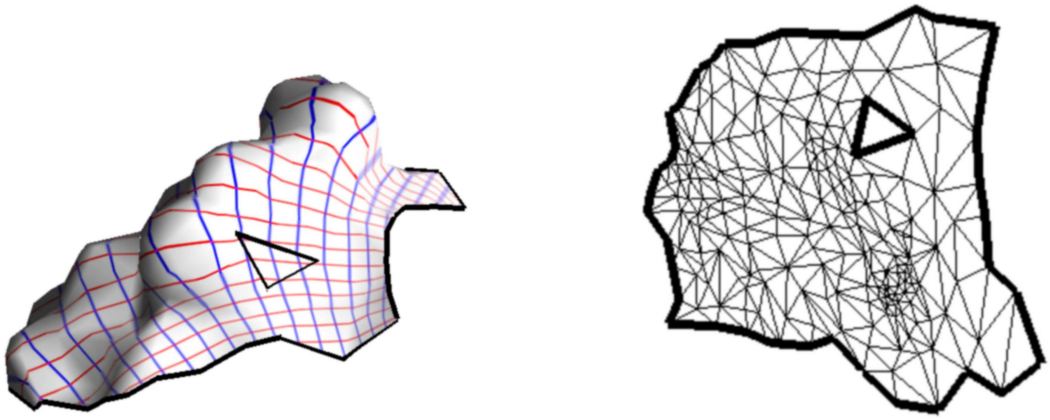

Fig. 7. Left: the surface with blue and red lines as iso-$u$ and iso-$v$. Right: the corresponding map. The 2D/3D correspondance of triangles is illustrated by the highlighted triangle.

importance, and therefore we will not concentrate on this problem.

To do this, the first step is to create a one to one function that will associate the surface to a 2D domain. The second step it to use this function to create an attibute map. The creation of such a mapping function for triangulated surface is know as parameterization in the computer graphics field. The function is piecewise linear on each triangle and is completely defined by 2D coordinates (called texture coordinates) associated to each vertex (see Figure 7). We have choosen the LSCM method of Levy et al. (2002), that is fast, does not require to fix the borders, and generates almost conformal maps. The attribute map is generated using this piecewise linear function. A hardware accelerated way to do this is presented at the end of this Section.

### *3.1 Parameterization: Least Squares Conformal Maps*

The LSCM parameterization method generates maps that are as conformal as possible. Due to the piecewise linear nature of the map (*i.e.* defined by texture coordinates), a real conformal map can not be found in the general case. So, the goal is then to find a map that is almost conformal.

A conformal map respects the Cauchy-Riemann equation:

$$\frac{\partial \mathcal{U}(s)}{\partial x} + i\frac{\partial \mathcal{U}(s)}{\partial y} = 0$$

where $\mathcal{U} = \mathcal{X}^{-1}$ is the function that links the surface to the parametric space, $x$ and $y$ denotes a local coordinate system on the surface and $s$ is a complex number representing the position in the map.

The algorithm finds texture coordinates that approximate a conformal map in the least squares sense. Since the equation has to be respected for all points of the surface, the energy to minimize is an integral of squares defined as follows:

$$\mathcal{E}_C(Surface) = \int_{Surface} \left| \frac{\partial \mathcal{U}(s)}{\partial x} + i \frac{\partial \mathcal{U}(s)}{\partial y} \right|^2 ds$$

where $\mathcal{E}_C()$ is the conformal energy. The notation $|z|$ denotes the modulus of the complex number $z$.

This energy is integrated over each triangle:

$$\mathcal{E}_C(T) = \frac{1}{A_T} \cdot \| \begin{pmatrix} W_1 & W_2 & W_3 \end{pmatrix} \cdot \begin{pmatrix} U_1 & U_2 & U_3 \end{pmatrix}^T \|^2$$

where $T$ is the triangle, $U_i$ are the complex numbers that represent the parametric space vector of each edge, and $W_i$ are the complex numbers that represent the edge vectors in a local base of the surface triangle. Since the sum of these energies can be represented as a quadratic form, the minimization can be efficiently performed using a conjugate gradient algorithm. The choice of this algorithm is motivated by the conformal goal that ensures to often find a valid parameterization and that avoids texture swimming when used on a multiresolution structure such as Progressive Meshes. Other benefits are the natural border extrapolation and the overall performance.

### *3.2 Attribute map generation*

The attribute maps are generated using the following idea: the 2D map is rasterized using the attributes. For each triangle of the surface, the corresponding triangle in 2D (defined by the texture coordinates of its vertices) is rasterized and the vertices attributes are interpolated. This operation is performed very quickly by modern graphics hardware. Examples of maps are given in Figure 8.

## 4 Results and experimentations

Our algorithm has been implemented as a plugin of the VMD (Visual Molecular Dynamics) software of Humphrey et al. (1996) and should be distributed (as the Intersurf Plugin Version 1.0) along with the official release by the time of publication.

Our algorithm has been tested against the ZDOCK benchmark (Chen et al., 2003), composed of 58 co-crystallized pairs of proteins. Generating a map for

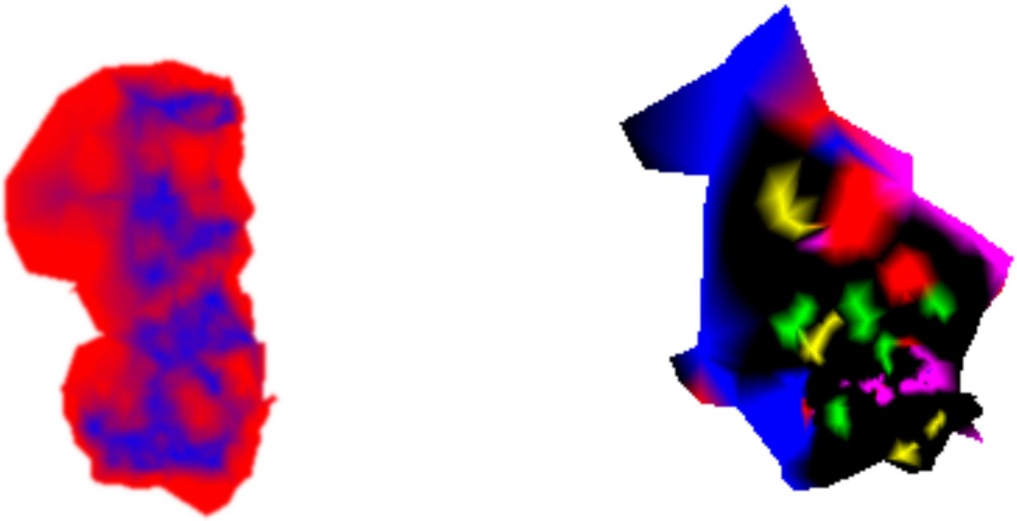

Fig. 8. Examples of attribute maps. Left: "distance to protein". Right: "kind of interactions between residues".

all of these examples has been done in three minutes on a standard PC, *i.e.* the average time required to generate a map is about three seconds. It is decomposed as follows: extracting the interface is typically performed in one second, creating the mapping function takes approximatively another second, and the remaining second is spent creating the topology of the map. Table 1 shows detailed statistics for the fastest (1AVZ_C) and the longest (1GOT_C) runs. The process is fast enough to be used in a molecular modelling software or in an automatic pipeline. However, it is too slow to extract the surface in real-time while moving a whole protein. Compared to previous work of Gabdoulline and Wade (1996), our algorithm is a bit faster (about three times faster) when ADSI is launched with the default settings; their mesh needs to be denser than ours (about two times denser) in order to capture the geometry with their regular connectivity mesh.

| | **1AVZ_C** | **1GOT_C** |
|---|---|---|
| **# Atoms** | $461 + 873$ | $3082 + 2627$ |
| **# Triangles** | 868 | 2115 |
| **# Tetrahedra** | 8119 | 22461 |
| **Interface area** | 1000 $\mathring{A}^2$ | 3100 $\mathring{A}^2$ |
| **Total extraction time** | 1.5 $s$ | 5 $s$ |

Table 1
ZDOCK benchmark: detailed statistics for the fastest (1AVZ_C) and the longest (1GOT_C) runs.

In terms of quality, our approach ensures that the triangulation is valid and composed of nice shaped triangles. The generated surface achieves its goal to

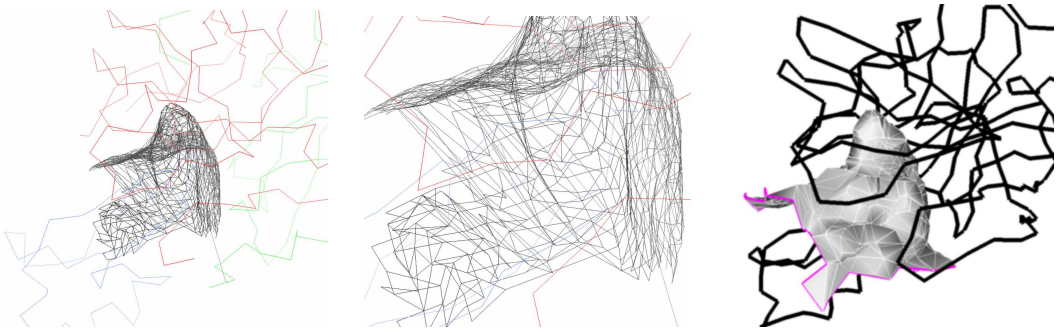

Fig. 9. Left: a screenshot of an interface created by MolSurfer. Middle: a close-up of this surface shows that the mesh was not correctly extracted. Right: our approach does not suffer from these problems.

make it easy to understand the way two proteins are docked. Notice that if the triangulation is too coarse for high quality rendering, it is possible to smooth it using a mesh subdivision algorithm such as (Dyn et al., 1990). Previous work of Gabdoulline et al. (1999) gives similar results if the surface is easy to unwrap, but fails to find a valid triangulation if the surface is spiky. Since proteins are often linked together by a residue of the first protein fitting a cavity of the second one, many interfaces are hard to unwrap. Figure 9 shows the benefits of our approach for these kinds of surfaces.

Our parameterization step enables to generate valid and low distorded maps for most cases (see Figure 10). However, a complex geometry or a bad topology can make the map unsuitable for vizualization. In our benchmark, the map extraction often (about 95% of the times) provides the expected result, and the failed cases either present high distortions (due to complex geometry) or are not valid (due to incompatible topology).

## Conclusion

This paper introduces an efficient way to represent the interaction between two proteins. This representation is very useful for visualization as well as for automatic comparison between several interactions. The algorithm is fast enough (always less than five seconds) to be integrated into an automatic process that compares a large set of maps in order to classify the types of interaction. The Delaunay tetrahedralization approach allows to fastly estimate attributes and to interactively remove and add new vertices, making it possible to update the interface in real-time (less than 0.1 second) while locally modifying the protein geometry.

**Acknowledgments**

We thank the members of the Theoretical and Computational Biophysics Group of the Beckmann Institute at the University of Illinois; special thanks to Professor Klaus Schulten for making the cooperation on VMD possible and to John Stone for the great technical support on the software. We also want to thank to the anonymous reviewers for their helpful comments. This work is supported by grants from the Inria (ARC Docking Cooperative Research Initiative) and the Region Lorraine (Pôle de Recherche Scientifique et Technologique "Intelligence Logicielle" / CRVHP).

**References**

Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The Protein Data Bank. In Nucleic Acids Research, 28 (1):235-242 (2000).

Inbal Halperin, Buyong Ma, Haim Wolfson, and Ruth Nussinov. Principles of Docking: An Overview of Search Algorithms and a Guide to Scoring Functions. In PROTEINS: Structure, Function, and Genetics; 47:409-443 (2002).

Razif R. Gabdoulline, and Rebecca C. Wade. Analytically defined surfaces to analyze molecular interaction properties. In Journal of Molecular Graphics; 14 (6): 341-353 (1996).

Razif R. Gabdoulline, Rebecca C. Wade, and Dirk Walther. MolSurfer: two dimensional maps for navigating three-dimensional structures of proteins. In Trends Biochem. Sci., 24, 285-287 (1999).

Marshall Bern and David Eppstein. Mesh generation and optimal triangulation, 2nd edition. In Computing in Euclidean Geometry; 4: 47-123 (1995).

Bruno Levy, Sylvain Petitjean, Nicolas Ray and Jerome Maillot. Least squares conformal maps for automatic texture atlas generation. In Proceedings of the 29th annual conference on computer graphics and interactive techniques, ACM Press (2002).

William Humphrey, Andrew Dalke, and Klaus Schulten VMD - Visual Molecular Dynamics. In Journal of Molecular Graphics; 14: 33-38 (1996).

Rong Chen, Julian Mintseris, Joe Janin, and Zhiping Weng. A Protein-Protein Docking Benchmark. In PROTEINS: Structure, Function,and Genetics 52: 88-91 (2003).

Nira Dyn, David Levine, and John A. Gregory. A butterfly subdivision scheme for surface interpolation with tension In ACM Transactions on Graphics, ACM Press (1990).

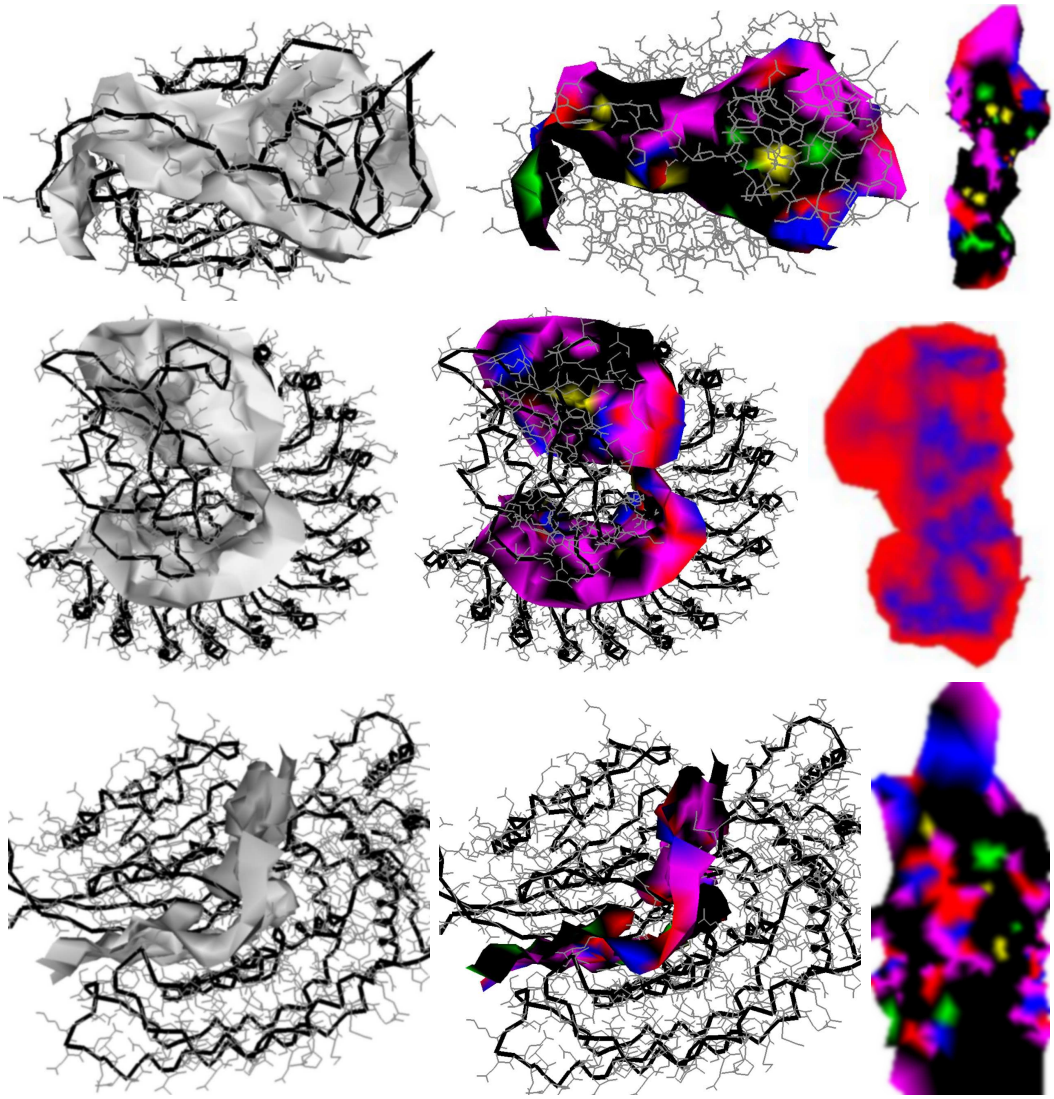

Fig. 10. Examples from the ZDOCK benchmark. Left: “distance to protein”. Right: “kind of interactions between residues”.