Université catholique de Louvain

École Polytechnique de Louvain

Institut des Technologies de l'Information et de la Communication, Électronique et Mathématiques Appliquées (ICTEAM)

Place du Levant 2
B - 1348 Louvain-la-Neuve
Belgique

Protein Surface Properties
using
Signal Processing and Statistical Tools

Joachim Giard

Thèse présentée en vue de l'obtention du grade de
Docteur en Sciences de l'Ingénieur

Examination committee:

Benoît MACQ (UCL/ICTEAM) - Supervisor
Jean-Luc GALA (UCL/IREC)
Patrice RONDÃO ALFACE (Alcatel-Lucent Bell NV.)
Patrick BAS (CNRS/LAGIS - France)
Christophe DE VLEESCHOUWER (UCL/ICTEAM) - President

June 2010

Remerciements

Je voudrais tout d'abord exprimer ma gratitude envers mon superviseur, Benoît Macq, qui m'a fait confiance et fait profiter d'un environnement de travail bouillonnant tout en me laissant une grande liberté d'action.

Je remercie Jean-Luc Gala grâce à qui la rencontre avec le monde des protéines n'a pas été trop brutale.

Je suis également très reconnaissant envers Christophe De Vleeschouwer et Patrick Bas dont les remarques pertinentes ont permis d'améliorer la qualité et la clarté de ce travail.

Patrice Rondão Alface a joué tour à tour les rôles de coach personnel, de collaborateur et de relecteur avec une sympathie et un investissement sans égal. Je l'en remercie vivement.

Je remercie sincèrement Jérôme Ambroise avec qui le travail au jour le jour s'est passé à merveille, tant au point de vue scientifique qu'au point de vue humain.

Merci à tous ceux qui rendent l'ambiance du labo TELE accueillante et conviviale, que nos intérêts communs soient les tartes, la musique, la piscine ou même le travail. Pour ce dernier point, je remercie plus spécifiquement les membres du cluster "imagerie médicale". Cette convivialité est aussi assurée par Jean, Patrcia, Marie-Hélène, François, Christian, sans oublier Robert. Je tiens à les remercier pour leur disponibilité et leur gentillesse.

Pour finir, je voudrais remercier mes amis et ma famille pour leur soutien matériel ou immatériel, conscient ou inconscient. Merci à Claudine pour ses mots d'anglais et sa compagnie rayonnante. Merci à mes parents d'avoir toujours stimulé notre curiosité et notre imagination sans jamais nous contraindre. Merci à Thibault pour sa complicité.

Cette recherche a été financée par la Région Wallonne dans le cadre du projet NANOTIC/TSARINE.

Contents

List of Figures

Notations, Acronyms and Remarks

Mathematical Notations

a	a point with the coordinates (x_a, y_a, z_a).
$d_a(b)$	the geodesic distance between a and b.
$\|\overrightarrow{ab}\|$	the Euclidean distance between a and b defined as follows:

$$\|\overrightarrow{ab}\| = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2}.$$

$\delta_a(b)$	a distance between a and b. Depending on the context, it can be the geodesic distance on a decimated mesh, the reference distance in an unfolded plane, or the travel depth.

Acronyms

vdW	van der Waals
vdWS	van der Waals Surface
SAS	Solvent Accessible Surface
SES	Solvent Excluded Surface

Remarks

- The coordinates of an atom refer to the coordinates of its center. Similarly, the distance between atoms refer to the distance between their centers.
- When computation times for methods developed in this work are mentioned, it refers to tests performed on a AMD®Athlon™64 X2 Dual Core Processor 3800+, 2 Gb RAM or on an Intel®Core™2 CPU T7200 @ 2.00GHz, 2 Gb RAM. The algorithms were implemented in C++ with the VTK (Visual ToolKit)[1].

[1] http://www.vtk.org

Introduction

1

The understanding of life processes has always been evolving. Recently, the knowledge about genomics and proteomics has considerably increased. This improvement is joined to the birth of bioinformatics, i.e. the application of information technology to biological issues, in the 70's.

Genomics is the study of the entirety of the genes of an organism, the genome, containing all the information about its heredity. The major recent progresses in genomics include the human genome sequencing. Indeed, by 2003, the atomic composition of all the human genes was declared to be known, with only some errors. This kind of information can be very useful for diagnosis or therapeutic applications.

Proteomics is the study of the entirety of the proteins of an organism. It is closely related to genomics because the proteins compositions are encoded in the genes. Proteins are long chains of atoms folded into a globular form. They are the physical expression of the genes and rule most cellular functions, and therefore, almost every life process. Some well known examples are transport proteins such as hemoglobin, catalyzing proteins (named enzymes), structural proteins such as keratin.

Proteins fulfill their function by assembling with other molecules, statically or temporarily. Studying these interactions and modeling them is called *protein docking*. One of the main application of protein docking, is the drug design. When the way two proteins interact and the effect of this interaction are perfectly known, it is possible to design an artificial molecule inhibiting or promoting this interaction. This kind of molecules are medicines and the possibility of designing them with less experiments allows researchers to save much time and money. Instead of designing a new molecule ab nihilo, it can be advantageous to search in data bases for existing molecules (natural or artificial) with the same properties. The available material in the data bases containing information about proteins is exponentially increasing, so that there is a need for efficient and rapid methods for executing the different steps of a docking process.

One of the major problem concerning protein docking is the huge number of degrees of freedom, that makes impossible the explicit research of all potential interactions. Indeed, protein are long folded chains of atoms and may be deformed with

very small amounts of energy. Thus the number of possibilities of fitting such objects together is countless.

A common way to proceed to protein docking is to study the properties of the *binding site*, i.e. the part of the molecular surface which plays a role in the binding with other components [1, 2]. The docking research can be limited to this part of the protein in order to reduce the computation times. It has been pointed out that binding sites share common properties. For example, a binding site often lies in a hollow with local properties such as high hydrophobicity [3, 4] and good sequence conservation in terms of evolution [5, 6].

The main theme of this thesis is the use of a 3D surface modeling to approach protein docking and, in particular, binding sites detection. 3D surface modeling is used in many domains, such as animation movies, construction engineering or medical imaging. The available tools in 3D modeling and visualization are transposed to the world of molecules. The algorithms are adapted to meet the speed objectives but keeping or even improving the results precision. In this work, proteins surfaces are modeled as polygonal meshes, which are collections of points linked by polygonal faces and lines.

1.1 Contributions

The major contributions of this work concern:

- Distances calculation on surface meshes (travel depth and geodesic distance).
- The detection of sites of interest on protein surfaces (binding sites and epitopes).

The common Euclidean straight distance is not always realistic in applications using 3D objects. Indeed, it does not take into account that the objects are sometimes solid and then impenetrable. In the case of proteins, a situation where the Euclidean distance is not relevant is the pocket detection. It is interesting to locate places where other molecules may be stuck and interact with the studied protein. These places are called *pockets* and can be found by measuring the depth on the surface. In this context, Coleman and Sharp [7] defined the **travel depth** as the shortest distance from the convex hull without going through the inner part of the molecule. With this definition, the bottom of a bent pocket can be close to the convex hull in an Euclidean point of view but far from it according to the travel depth definition.

Coleman and Sharp also proposed an algorithm to compute the travel depth. However, this algorithm is quite slow because it is volume based. The **first contribution** of this thesis is to propose a surface based algorithm to compute the **travel depth rapidly and with high precision**.

Sometimes, it may be useful to know the distance between two points along the surface (the geodesic distance). The travel depth calculation is an example which involves Euclidean distances as well as geodesic distances computations. The geodesic

distance can also be used, for instance, to define zones of the surface that are not too much affected by volume deformations like those that animate proteins, or to compute surface properties such as the curvature.

Computing geodesic distances on 3D meshes may be time consuming, especially in applications requiring *"all sources to all targets"* computations, in other word, when computing this distance between each pair of points of the mesh. **The second contribution** of this thesis is the design of algorithms to **approximate rapidly geodesic distances** on a surface. Two approaches are developed. In the first one, the number of points of the mesh is reduced and the geodesic distances are computed on the simplified mesh and stored. These distances are later reused for the computation of distances on the complete mesh. In the second one, the mesh is unfolded into the plane around reference points. Then, the geodesic distances are estimated as Euclidean distances in those reference unfolding planes.

The final aim of computing surface properties is the localization of sites of interest on proteins. **The third contribution** of this thesis is a **prediction method for binding sites** using surface patches and regression or classification models, as well as a comparison between these different statistical models. This method is also applied to the special case of antigens epitopes, i.e. the parts of antigens recognized by antibodies, and it gives better results than existing methods.

There are some other minor contributions in this thesis in the domain of

- The construction of meshes representing the protein surface.
- The estimation of electronic density taking atom movements into account.
- The study of spectral properties of the 3D shape of proteins.

1.2 Organization of the Thesis

Chapter 1 is a general introduction of the thesis. Chapter 2 is a presentation of the biological context, which introduces the necessary background to understand the applications, from genes to protein folding. It is followed by a chapter about bioinformatics in this domain (Chapter 3), where the protein docking and binding sites detection are introduced. Chapter 3 covers a larger basis than the one needed to address the issues explored in this work. This additional information (global protein docking, molecular dynamics, etc) is provided in order to clarify the context and to be able to understand future work perspectives. In this work, the analysis of proteins is made using a surface based approach. In Chapter 4, basic concepts about 3D modeling and methods to model protein surfaces as polygonal meshes are presented. Chapter 5 is dedicated to geodesic distances, i.e. lengths of paths lying on the surface mesh, and to algorithms computing them. Another type of distance, the travel depth, is studied in Chapter 6. Chapter 7 is a compilation of all the proteins properties analyzed in this work. A method for detecting binding sites is presented in Chapter 8, which also presents results on several data sets, containing various binding sites or antigen epitopes.

Biological Introduction

2

Proteins are macromolecules taking part in almost every life process [8]. They are made of amino acid residues organized in a linear chain and linked by covalent bonds, called peptide bonds. The use of the term *"residue"* is due to the fact that amino acids lose two hydrogen and one oxygen atoms to join each other. The sequences are written in the genetic code and are constructed from a set of 20 amino acids. The long chain of residues is folded into a 3D conformation which is generally unique for a given sequence. It is interesting to notice that the function and the 3D structure of a protein are closely related, even if there are other parameters determining the function, such as the surface atoms partial charge.

In this chapter, the structural elements of a protein are described, from the bonds and forces existing between atoms (Section 2.1), to amino acids constitution (Section 2.2), to the structure of the whole protein (Section 2.3), itself divided into four levels of organization. In Section 2.4, the way a protein is synthesized from the genetic code is addressed. When parts of the genetic code are modified (genetic mutations), the effects are directly propagated to proteins with more or less serious damages or in exceptional cases, efficiency improvements (Section 2.5). Then, in Section 2.6, different functions of proteins are presented. The case of enzymes and antigens-antibodies are developed as examples. The practical application of drug design necessitating an advanced knowledge of proteins functioning is introduced with the same two examples in Section 2.7.

2.1 Bonds and Forces

Proteins are molecules, which are electrically neutral sets of atoms. These atoms are linked together by bonds of different natures either inside a molecule, inside a crystal, inside a metallic solid, or between atoms from separated molecules. They are resulting from electromagnetic forces and are generally classified into two categories: covalent (or strong) and non-covalent (or weak) bonds.

2.1.1 Covalent Bonds

The covalent bond is the local and directional link between two atoms sharing one or several pair(s) of valence electrons, coming from one or both atoms involved in the bond. When more than one pair is shared, the bond is called multiple bond. Double and triple bonds are the commonest, but bonds with four, five or six shared pairs of electrons exist. The lengths and the angles of a covalent bond are strongly constrained and depend on the nature of the atoms and on the order of the bond (single, double, etc). The bond is non-polar when the atoms have the same electronegativity, and polar otherwise.

2.1.2 Non-Covalent Bonds

Weak bonds link atoms from separated molecules or different regions of same macromolecules. Isolated, this kind of bond is insignificant in comparison with covalent bonds but collectively, they have a very significant influence on the 3D structure of proteins, nucleic acids, polysaccharides, and membrane lipids. The weak bonds of different natures are described.

The ionic bond is a non-local and non-directional link between atoms with a large electrostatic difference, that is, a positively charged atom and a negatively charged atom. There is no real difference between a polar covalent bond and an ionic bond: the more the bond is polar, the more it is ionic. An electronegativity difference of 1.7 is conventional threshold to consider that the electron transfer is total. The ionic interaction can also be repulsive when the distance between atoms is too small, even for atoms with an opposite electronegativity.

The hydrogen bond is a dipole-dipole link between an electronegative atom (the hydrogen *donor*) and a hydrogen atom covalently bonded with another electronegative atom carrying a lone pair of electrons (the hydrogen *acceptor*). Its length is around 2 Å and it is about twenty times weaker than a covalent bond. It plays an important role in the protein life, from the structure stability to the binding affinity with other molecules. The hydrogen bond is local and directional, it means that the bond is stronger in the direction of the hydrogen covalent bond axis (for the donors) and in the direction of the nonbonding electron orbitals (for the acceptors). The electron orbitals are distributed around the atom in such a way that the pair of electrons and the covalent bond are almost uniformly distributed. Indeed, the orbitals are crowded by electrons and consequently, they take up more place than covalent bonds. Thus, for example, the oxygen atom of a water molecule is a double donor because it is bonded to two hydrogen atoms and a double acceptor because it carries two free electron pairs (see Figure 2.1). The ensemble forms an irregular tetrahedron with a 104.5° angle between both hydrogen atoms. A nitrogen atom with a two covalent bonds, a simple one and a double one, and a free electron pair is a simple

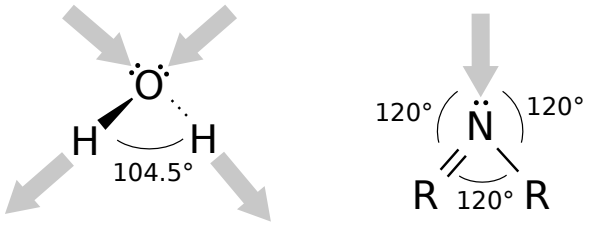

Figure 2.1 Two examples of hydrogen bonds configuration. A water molecule (left) is a double donor because of the hydrogen atoms and a double acceptor because it carries two free electron pairs. The potential hydrogen bonds and their preferred direction are represented by thick gray arrows. In this case, they form a rough tetrahedron. A nitrogen atom (right) with a two covalent bonds, a simple one and a double one, and a free electron pair is a simple acceptor. The potential hydrogen bonds are in the plane and are separated from each other by 120° angles.

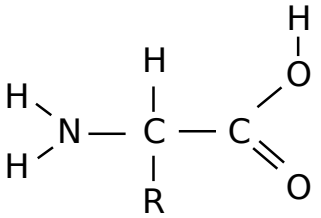

Figure 2.2 The structure of an amino acid. "R" is the R group characterizing the amino acid.

acceptor (see Figure 2.1). The orbital and the covalent bonds are in the plane and are separated from each other by 120° angles.

The van der Waals bond is a weak intensity interaction between all the atoms that are close enough. It is ten times weaker than the hydrogen bond.

The hydrophobic interactions are the forces that hold the non-polar region together in an aqueous environment.

2.2 Amino Acids

Amino acids are organic molecules carrying two functions around a carbon skeleton: an amine (NH_2) and a carboxyl ($COOH$). Both groups are linked to a central carbon atom (the alpha carbon, C_α), which is also linked to a R group characterizing the amino acid (see Figure 2.2).

Human beings proteins are formed by 20 standard amino acids, differentiated by their R group. For example, the R group of the Glycine is simply an hydrogen atom and the R group of the Lysine is $(CH_2)_4NH_2$. The list of the standard amino

Amino acid	three-letter	one-letter
Alanine	Ala	A
Arginine	Arg	R
Asparagine	Asn	N
Aspartic acid	Asp	D
Cysteine	Cys	C
Glutamic acid	Glu	E
Glutamine	Gln	Q
Glycine	Gly	G
Histidine	His	H
Isoleucine	Ile	I
Leucine	Leu	L
Lysine	Lys	K
Methionine	Met	M
Phenylalanine	Phe	F
Proline	Pro	P
Serine	Ser	S
Threonine	Thr	T
Tryptophan	Trp	W
Tyrosine	Tyr	Y
Valine	Val	V

Table 2.1 List of the 20 standard amino acids, their 3-letter code, and their 1-letter code.

acids is shown in Table 2.1 with the corresponding commonly used three-letter and one-letter codes.

Depending on the properties of the R groups, the amino acids belong to different categories. These properties have a capital importance in the protein structure and function. The main categories of R groups are:

Small [A, N, C, D, G, P, S, T, V]: with a few number of atoms.

Tiny [A,C,G,S]: a subset of small.

Hydrophobic [F,Y,W,A,P,V,L,I,M]: repelled from water, generally because they are apolar or/and they have no predispositions to create hydrogen bonds.

Aromatic [F,Y,W]: containing an aromatic cycle, i.e. a planar cycle with $4n + 2$ electrons in a delocalized cloud. The cycle is apolar and then hydrophobic. In the case of amino acids, the cycles contain 6 carbon atoms and 6 delocalized electrons.

Aliphatic [G,A,P,V,L,I,M]: containing only carbon and hydrogen atoms and no aromatic cycle. However, it is convenient to consider methionine in this category.

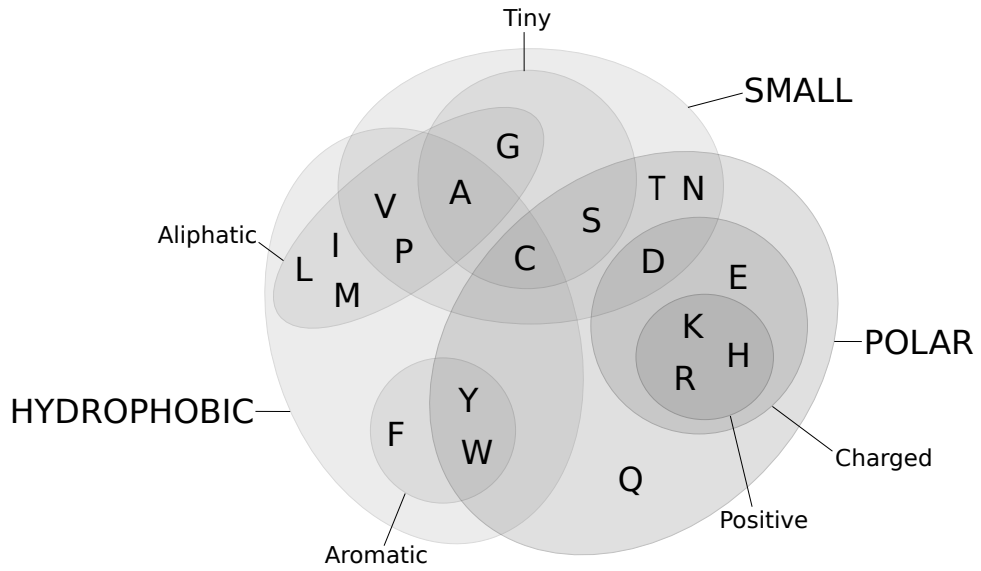

Figure 2.3 Classification of the 20 standard amino acids according to physicochemical properties.

Although its side-chain contains a sulfur atom, it is largely non-reactive, meaning that methionine effectively substitutes well with the true aliphatic amino acids [9].

Polar [S,T,C,N,Q, K,R,H,D,E,Y,W]: with an asymmetrical charge distribution, and then more hydrophilic.

Charged [K,R,H,D,E]: carrying a positive (K,R,H) or negative (D,E) charge.

This classification can be different depending on the source because the behavior of the amino acids varies from one environment to another and because the properties scales are generally continuous, hence the classification require the application of thresholds. The classification shown in the description above-mentioned and in Figure 2.3 is inspired from [8].

2.3 Protein Structure

The structure of a protein can be seen at four different levels (see Figure 2.4) named the primary, secondary, tertiary and quaternary structure.

2.3.1 Primary Structure

The primary structure is the sequence of amino acid residues. This sequence depends directly on the genetic code and define the order of the residues in the protein chain. In a protein, the amino acids are linked by the peptide bond, a covalent bond

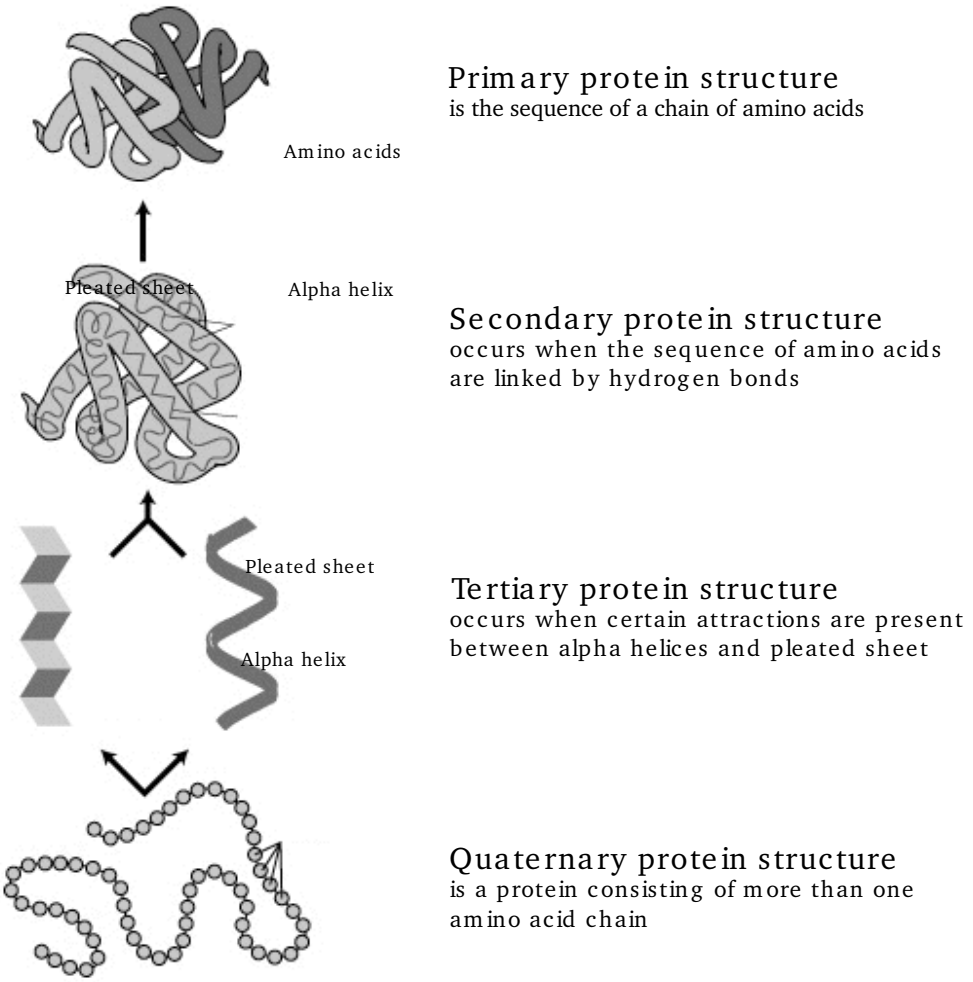

Figure 2.4 The four structure levels of a protein. Image courtesy: National Human Genome Research Institute.

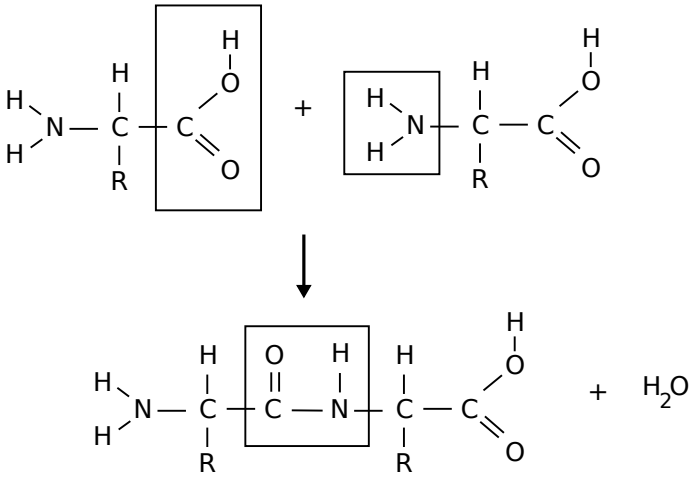

Figure 2.5 Scheme of the reaction of peptide bond formation. The products are a polypeptide and water.

between the carbon of the carboxyl group and the nitrate of the amine group of another amino acid (see Figure 2.5). Each amino acid is linked by its both ends (except the first one and the last one) to form a long chain, called a polypeptide chain. The chain formed by the alpha carbon, the carboxyl carbon and the amine nitrate linked by the peptides bonds is called the backbone of the protein.

2.3.2 Secondary Structure

The secondary structure is the organization of some residues into stable structures such as an α helix or a β sheet. The α helix is an helical structure in which the backbone is tightly wound around an imaginary axis. This structure is formed in particular cases of local sequences and is stabilized by hydrogen bonds between the residue n and the residue $n + 4$. An helix turn is 3.6 residues long. β sheets are materialized by zigzags in the backbone. The pleated polypeptide chain segments can be arranged side by side to form a sheet. Such as for the α helix, it is formed in particular cases of local sequences and is stabilized by hydrogen bonds.

2.3.3 Tertiary Structure

The tertiary structure is the arrangement of the protein atoms, including those organized in a secondary structure, in the three dimensional space. It is also called the protein folding or the protein 3D conformation. The tertiary structure is stabilized by weak interactions, such as ionic bonds or hydrogen bonds but also by covalent bonds. One of the most frequent covalent bond (additionally to the peptide bonds) necessary for the stability of certain proteins is the disulfide bridge: a covalent bond between two sulfur atoms of different residues.

2.3.4 Quaternary Structure

The quaternary structure is the spatial arrangement of several chains. The suffix ***-mer*** is used to designate a single subunit. It is preceded by a prefix giving the number of subunits composing the structure. A quaternary structure composed of one chain is called a *monomer* and composed of several chains, an *oligomer*. Structures of two, three, and four chains are called respectively *dimers*, *trimers*, and *tetramer* (and so on, this number is not limited to four). The ***Hetero-*** and ***Homo-*** prefixes are used to specify if the subunits are different (*Hetero*) or are all the same (*Homo*).

The interaction between several chains can be necessary for the subunits to exist. In this case, the interaction is said to be **obligate**. A **transient** interaction is an interaction between subunits that also exist in the monomeric form.

2.4 Proteins Synthesis

Proteins are chains of amino acid residues. The sequence of amino acids is encoded in the DNA (deoxyribonucleic acid) and is specific to a particular protein. The process of protein synthesis is quite complex and only an overview of this process for eukaryotes is described in this work (see Figure 2.6). For further information, see [8].

Genes are parts of the DNA constituting the chromosomes that is present in the nucleus of every living cell. The DNA is a macromolecule composed of two nucleotides chains organized as a double helix. It exists four different nucleotides in the DNA: the adenine (A), the thymine (T), the cytosine (C) and the guanine (G).

The first step of the protein synthesis is the transcription. In this step, the two DNA strands are separated and are transcripted into mRNA (messenger ribonucleic acid). It is a copy of the DNA strand except that the thymine is replaced by the uracil (U) because it is easier to produce. The mRNA leaves the cell nucleus and is translated into a polypeptide chain, which folds into a protein. The translation is effectuated by the ribosomes with the help of tRNA (transfer ribonucleic acid), translating each triplet of nucleotides, or codon, into an amino acid. There are 64 (4^3) possible triplets for only 20 amino acids to code, thus several combinations of three nucleotides may code for the same amino acid. Some codons have a particular role: the start-codons, indicating the beginning of the gene (generally AUG), and the stop-codons, making the translation stop (generally UAG, UAA or UGA).

2.5 Mutations and Evolution

Mutations are irreversible modifications in the genetic code resulting from errors in the copy of a gene or from exposure to mutagen agents. The different types of mutations are

- The substitution: a nucleotide is replaced by another one.
- The deletion: a nucleotide is deleted.

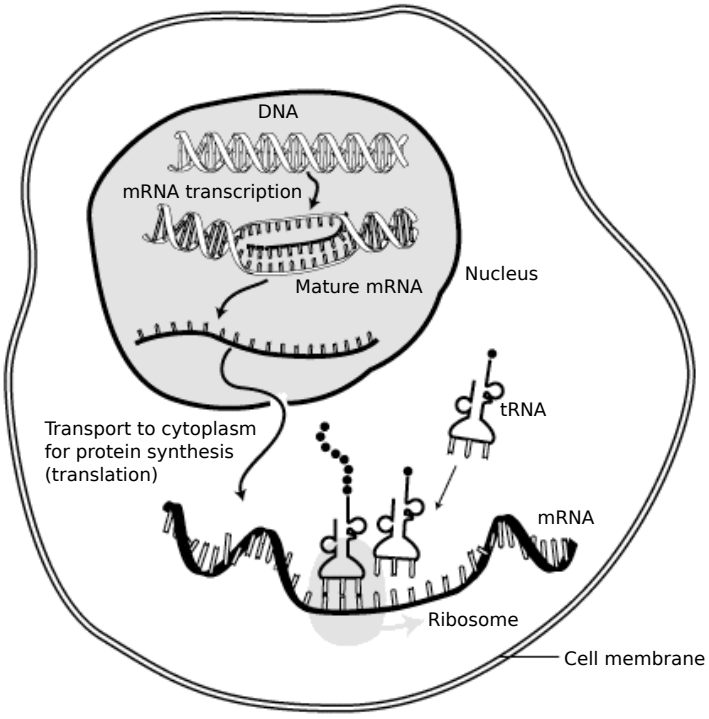

Figure 2.6 A scheme of a protein synthesis. DNA is transcripted into mRNA in the cell nucleus. Then the mRNA leaves the nucleus and is translated into an amino acid chain thanks to a ribosome and to tRNA. Image courtesy: National Human Genome Research Institute.

- The insertion: a nucleotide is inserted in the sequence.

The effect of a mutation can be more or less important depending on which nucleotide is affected and how it is affected. In general, the effects of insertion and deletion are more serious than substitution because it induces a reading frame shift: if the number of nucleotides inserted/deleted is not a multiple of three, the reading frame for the translation is shifted from the mutation spot to the end of the mRNA strand. Thus, the number of affected amino acids is consequently higher. The possible type of mutations classified by their effects on the protein function are:

- The silent mutation: the mutant codons are encoding the same amino acids as the original ones and then, the mutation has no effect. It is an advantage of the redundancy of the genetic code.
- The neutral mutation: the mutation has an effect on the protein sequence but not on the protein function because no important part of the protein is modified or because the mutant amino acids have the same physicochemical properties as the original amino acids.
- The nonsense mutation: a mutant codon is a stop-codon, consequently, the protein is truncated and often nonfunctional. This mutations may lead to severe genetic diseases or even to death.
- The missense mutation: the mutation has an effect on the protein sequence and on the protein function because the mutant amino acids do not have the same properties as the original ones and it may lead to a wrong folding. The consequences are the same as for the nonsense mutation.

Most of the time, the effect of a mutation on the protein function is negative if the protein become too active or not active enough. In some very rare cases, however, the mutation might bring an improvement to the organism carrying it. It is the driving force behind evolution.

2.6 Protein Functions

Proteins exhibit a large variety of biological functions and are present in all living cells. Some well known examples are transport proteins such as hemoglobin, catalyzing proteins (named enzymes), and structural proteins such as keratin. We have seen that proteins fold into a determined three-dimensional stable conformation. However, this conformation is not rigid, small side chain vibrations as well as larger domains movements can appear. Indeed, proteins need to interact with other molecules to fulfill their function. Therefore, they need to be geometrically compatible with the molecule they interact with, and the flexibility is sometimes necessary to make the molecules complementary. The need for geometrical compatibility explains why the three-dimensional conformation is directly linked to the function. Nonetheless, we should not forget that the compatibility has to be as good as possible for physiochemical properties as well. The cases of the enzyme and of antigens, developed below, are examples of protein functions analyzed in this work.

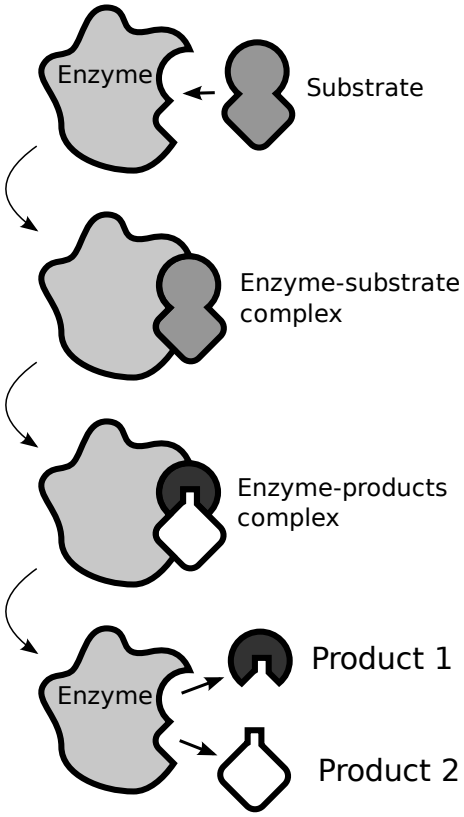

Figure 2.7 Scheme of the functioning of an enzyme. In this illustration, the substrate is a single molecule but it is not always the case. First, the substrate and the enzyme approach each other. Second, they form a complex thanks to weak bonds. Third, products are formed from the substrate by a reaction catalyzed by the enzyme. Fourthly, the products are separated from the enzyme. The enzyme stays unchanged and can be used again.

2.6.1 Enzymes

Enzymes are catalyzing proteins, i.e. proteins lowering the activation energy and accelerating a reaction without being changed by this reaction (see Figure 2.7).

2.6.2 Antibodies-Antigens

Antibodies are proteins of the immune system recognizing foreign elements such as viruses or bacteria. The molecules present in the foreign elements that are recognized by the antibodies are called antigens and are generally proteins, polysaccharides or their lipidic derivatives. When the antigen is recognized by the antibody, an immune response is activated. The antibody is made of two identical heavy chains and two identical light chains, bonded by disulfide bridges and forming a Y shape (Figure 2.8). The upper extremities of the Y, the Fab (fragment antigen binding) regions, are partially variable regions and are dedicated to the specific recognition of small parts of the antigen, the epitopes.

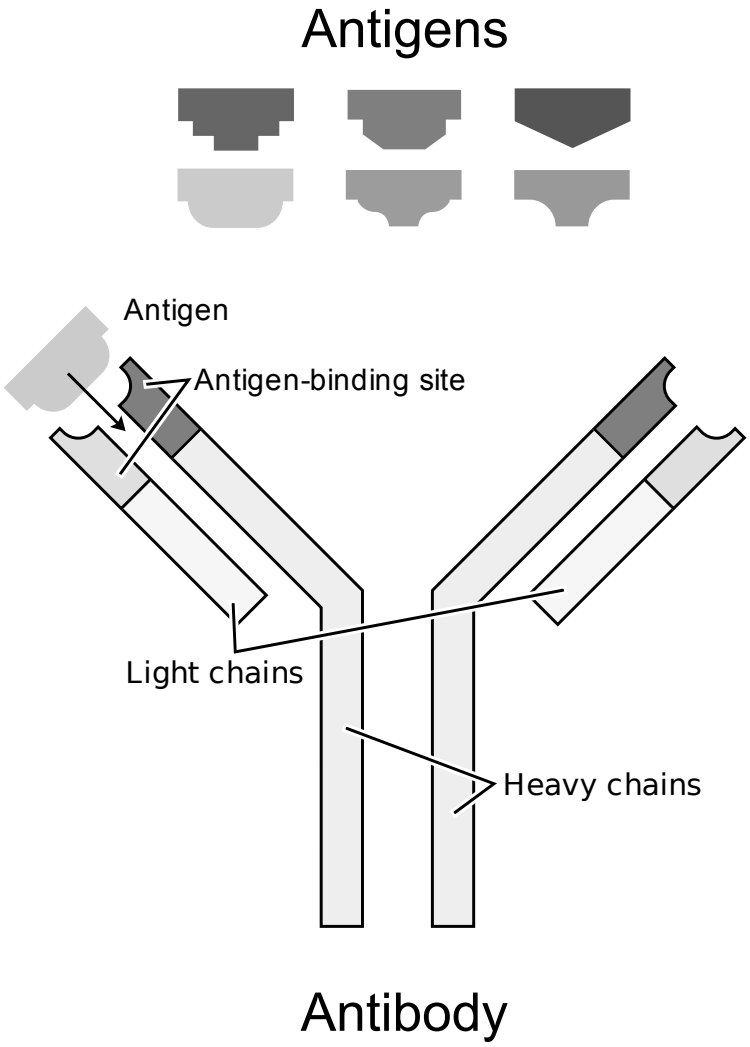

Figure 2.8 Scheme of the structure of an antibody-antigen complex. The antibody is composed of two identical heavy chains and two identical light chains forming a Y shape. The tips of this Y are the antigen-binding sites and are specific to a particular antigen.

2.7 Drug Design

Drug design [10, 11] is the process of creating new medications. Drugs are artificial molecules used to influence a biochemical process.

The aptamers are an example of such molecules; they are short oligonucleotides selected from a large combinatorial pool of sequences for their ability to bind to a specific target molecule [12, 13]. Thanks to their cheap and easy synthesis and to their wide scope of applications (from protein inhibition to antibody substitution, and including protein detection for rapid diagnosis), aptamers are of great interest in the world of biotechnologies.

In this section, examples of context for drug design are described for enzymes and antibodies-antigens.

2.7.1 Catalytic Site (Enzymes)

The particular spot of the enzyme structure where the reaction occurs is called the catalytic site, or active site. It is generally a small cavity (also called a pocket) where the substrates are bound with weak bonds. The reaction occurs in this site and converts the substrates into products (see Figure 2.7). Some natural mechanisms decreasing the activity of enzymes allow the regulation of biological processes such as the metabolic pathway. There are two possible ways to inhibit an enzyme activity: by blocking the access to the active site, taking the place of the substrate, or by provoking the deformation of the active site (allosteric inhibitor). This second kind of inhibitor binds to another site of the enzyme, the allosteric site, and induces a conformation change of the whole protein making the active site incompatible with the substrate. For some proteins, the active site has to be modified by an allosteric system to be compatible with the substrate. In this case, the small molecule inducing the conformational change is called an activator. The study of such mechanism allows researchers to create artificial molecules taking the role of inhibitor or activator in case of malfunctioning.

2.7.2 Epitopes (Antibodies-Antigens)

The epitope is the region of an antigen which is recognized by an antibody, leading to an immune response. It is possible to trigger the immune response even without the presence of the antigen by creating artificial molecules similar to the epitopes. These mimicking molecules are a class of vaccines. They can stimulate the immune system or prepare it to react faster for future attacks of the organism. This kind of studies are included in the vaccine design process.

2.8 Conclusion

In this chapter, the biological context of the thesis was introduced. This work is about protein structure and surface analysis for binding site prediction and for protein docking in general. Thus, there is a need to understand the mechanisms of pro-

teins functioning from the elementary interactions between atoms to the quaternary structure of the molecule (Sections 2.1 to 2.3). The sequences of proteins are encoded in the DNA which is subject to mutations with different consequences (Sections 2.4 and 2.5). This notion was discussed in this chapter for two reasons:

- The mutations are the motor of evolution and a particular protein can be the common ancestor of a family of proteins with similar functions. The important parts of proteins are generally conserved through the successive mutations, otherwise, in most cases, the mutations are lethal. The identification of these important parts with the help of the conservation score, introduced in the next chapter, is later used in Chapter 8 to predict sites of interest on the protein surface.
- Mutations may induce 3D conformation changes and prevent the protein from correctly fulfilling its functions. Studying the effects of mutations on the protein interactions, using, for example, molecular dynamic simulations (Section 3.7) and machine learning tools, would be an interesting perspective.

Proteins are active in almost every life process (Section 2.6) and, in case of dysfunction, artificial solutions can be found. A branch of these solutions, drug design (Section 2.7), is one the major concrete long-term application of this work.

Given the huge amount of data, which is continuously increasing, most of biological issues in the domain of proteins require the use of computers. *Bioinformatics* is the topic which is explored in the next chapter.

Bioinformatics Applied to Genes and Proteins

3

Bioinformatics is the application of information technology to biological issues. The domain of genes and proteins is a typical subject for bioinformatics because of its complexity and of the amount of data to treat.

In this chapter, different aspects of bioinformatics applied to genes and proteins are introduced. The available techniques for sequence alignment, i.e. comparisons between primary structures are presented in Section 3.1. These techniques are quite different from the structure alignment techniques (Section 3.2) because the objects which have to be compared are different: successions of letters in the first case and organizations of points in a 3D space in the second case. However, since a protein with a particular sequence has almost always the same 3D structure, both applications can be linked. Methods for predicting the 3D structures of proteins from their sequences are introduced in Section 3.3. These techniques are still quite inefficient and incidentally, protein structure prediction constitutes one of the major challenges in the protein research field. Fortunately, data banks containing structures of proteins extracted from natural environments exist. The most famous is the Protein Data Bank (Section 3.4). In Section 3.5, different ways to visualize proteins are described. Another active challenging research concerns protein docking (Section 3.6), that is, the study of protein interactions. Proteins are continuously in movement, so that dynamic models (Section 3.7) are useful for protein docking in order to predict how proteins are animated.

3.1 Sequence Alignment

Sequence alignment is a family of techniques used to identify similitudes between several sequences (DNA, RNA or protein). The most frequent applications are the identification of binding sites, the prediction of protein functions, the prediction of the secondary of tertiary structure, or the establishment a phylogeny, that is, the construction an evolution tree. In this last example, the ancestors of the currently existing genes or proteins are deduced from the known existing sequences, their similitudes and the most probable mutation that could occur during the evolution.

In most sequence alignment techniques, an alignment score has to be optimized. This score introduces a penalty when an element of the sequence is replaced by another one in another sequence, and when elements were inserted or deleted. The score which has to be maximize for each pair of residues is generally taken in a substitution matrix, such as a BLOSUM matrix [14]. An element of the substitution matrix is large if the amino acids corresponding to the row and the column of this element are similar, and that they can substitute to each other, for example, because they have the same physicochemical properties. The BLOSUM matrices were constructed by comparing sequences from databases and by statistically computing the probability that a particular amino acid is replaced by another particular amino acid. The number following the name BLOSUM is a reference to the data set used to compute the scores. For example, the default substitution matrix for the BLAST [15] algorithm is BLOSUM62 (Table 3.1).

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
A	**4**																			
C	0	**9**																		
D	-2	-3	**6**																	
E	-1	-4	2	**5**																
F	-2	-2	-3	-3	**6**															
G	0	-3	-1	-2	-3	**6**														
H	-2	-3	1	0	-1	-2	**8**													
I	-1	-1	-3	-3	0	-4	-3	**4**												
K	-1	-3	-1	1	-3	-2	-1	-3	**5**											
L	-1	-1	-4	-3	0	-4	-3	2	-2	**4**										
M	-1	1	-3	-2	0	-3	-2	1	-1	2	**5**									
N	-2	-3	1	0	-3	0	-1	-3	0	-3	-2	**6**								
P	-1	-3	-1	-1	-4	-2	-2	-3	-1	-3	-2	-2	**7**							
Q	-1	-3	0	2	-3	-2	0	-3	1	-2	0	0	-1	**5**						
R	-1	-3	-2	0	-3	-2	0	-3	2	-2	-1	0	-2	1	**5**					
S	1	-1	0	0	-2	0	-1	-2	0	-2	-1	1	-1	0	-1	**4**				
T	-1	-1	1	0	-2	1	0	-2	0	-2	-1	0	1	0	-1	1	**4**			
V	0	-1	-3	-2	-1	-3	-3	3	-2	1	1	-3	-2	-2	-3	-2	-2	**4**		
W	-3	-2	-4	-3	1	-2	-2	-3	-3	-2	-1	-4	-4	-2	-3	-3	-3	-3	**11**	
Y	-2	-2	-3	-2	3	-3	2	-1	-2	-1	-1	-2	-3	-1	-2	-2	-2	-1	2	**7**

Table 3.1 BLOSUM62 substitution Matrix. This matrix is symmetrical. Bold numbers are on the diagonal. Framed numbers are related to the example of Table 3.2.

K	P	Y	E	N	C	N	Q	-	-	-	C	G	K	A
K	M	Y	M	N	-	-	Q	V	I	N	M	V	K	A
+5	-2	+7	-2	+6	-4	-1	+5	-4	-1	-1	-1	-3	+5	+4

Table 3.2 Example of alignment.

The penalty for insertion and deletion (gaps) consists in a fixed cost per gap and a variable cost per inserted/deleted amino acid. For example, for the alignment of Table 3.2, the score is 11 when using the BLOSUM62 matrix, a fixed gap penalty of -4 and a variable gap penalty of -1 per inserted/deleted residue.

The alignment can be pairwise, i.e. between two sequences or multiple, i.e. including more than two sequences. Here are some examples of software for pairwise sequences alignment: BLAST [15], FASTA [16], PSI-BLAST [17]; and for multiple sequence alignments: ClustalW [18], MUSCLE [19], T-Coffee [20].

3.2 Structure Alignment

Structure alignment is the quantitative comparison between 3D structures of proteins. 3D structure is strongly related to function and sometimes, the vital parts of a protein are conserved in term of structure despite several sequences mutations. Thus classifications can be made by detecting similarities between proteins structures instead of sequences. Structure alignment is also used to compare two 3D conformations of the same protein after a dynamic simulation in order to quantify the effects of the protein agitation.

The most simple index to compare two structures is the root mean square deviation (RMSD) computation. The global RMSD is only possible between structures with broadly similar sequences because it is based on a pairwise comparison between the 3D position of the elements (atoms of residues). The 3D position of the i^{th} element of the molecule A is denoted a_i. The RMSD between the structures $A = (a_1, \ldots, a_N)$ and $B = (b_1, \ldots, b_N)$.

$$RMSD_{AB} = \sqrt{\frac{\sum_{i=1}^{N} \|\overrightarrow{a_i b_i}\|^2}{N}},$$

where $\|\overrightarrow{a_i b_i}\|$ is the Euclidean distance between the elements a_i and b_i.

In the case of molecular dynamic modeling, the fluctuation of position of a particular element i can be expressed by the root mean square fluctuation of i ($RMSF_i$):

$$RMSF_i = \sqrt{\frac{\sum_{t=1}^{T} \|\overrightarrow{a_i(t)\tilde{a}_i}\|^2}{T}},$$

where $a_i(t)$ is the position of the element i at the time step t, T is the number of different conformations considered, and $\tilde{a}_i$ is the reference position. The *B-Factor* β_i

or *temperature factor* is generally defined as

$$\beta_i = \frac{8\pi^2}{3} RMSF_i.$$

In order to recognize similar patterns or to be more robust to angular movement of large domains, the sequences can be divided into subsequences that are subjects to pairwise comparison [21]. Another frequent technique to measure structure similarity is to construct a distance matrix between all the elements and to compare small patterns (submatrices) [22, 23].

3.3 Structure Prediction and Acquisition

The prediction of the tertiary or quaternary structures of proteins from their sequences is one of the main challenges in bioinformatics [24]. The problem is very difficult to solve because of the number of degrees of freedom. Moreover, the phenomenon is not fully understood yet and no theory can explain why a protein always folds in the same 3D conformation. The main way to predict protein structures are:

- The ab initio modeling [25], from the amino acids sequence and based on physical interactions.
- The comparative modeling [26], based on the comparison with already solved structures.

The ground truth structures may generally be acquired by X-ray crystallography or NMR (Nuclear Magnetic Resonance) spectroscopy.

In X-ray crystallography, identical proteins are artificially organized as a crystal. The atoms are localized by analyzing the X-ray diffraction pattern. Since diffracted X-ray beams can not be focused to create an image, the precision is increased thanks to the high structural order and repetition. The main advantage of this technique, which is the most used one, is its high precision.

In NMR spectroscopy, the molecules are placed in a strong magnetic field and probed with radio waves. The observed resonances analysis gives a list of atoms that are close from each other, making the reconstruction of the whole structure possible. Since the protein is not immobilized in a crystal, it is possible to observe its flexibility.

Acquired and fewer predicted 3D protein structures are made available in public data banks such as the Protein Data Bank.

3.4 The Protein Data Bank (PDB)

The Protein Data Bank (PDB) [27] is a collection of information about 3D structures of macromolecules: proteins and nucleic acids. Around 65000 structures (March 2009) are contained in the data bank and 6000 to 7000 new structures are added each year. PDB also refers to the name of the file format used to encode the information, mainly, the name of the molecule, the 3D position of the atom centers,

Record Name	Description
HEADER	First line of the entry, contains PDB ID code, classification, and date of deposition.
AUTHOR	List of contributors.
COMPND	Description of macromolecular contents of the entry.
EXPDTA	Experimental technique used for the structure determination.
KEYWDS	List of keywords describing the macromolecule.
SOURCE	Biological source of macromolecules in the entry.
TITLE	Description of the experiment represented in the entry.
ATOM	Atomic coordinate records for standard groups.
CONECT	Connectivity records.
HELIX	Identification of helical substructures.
SEQRES	Primary sequence of backbone residues.
SHEET	Identification of sheet substructures.
SITE	Identification of groups comprising important sites.
MODEL	Specification of model number for multiple structures in a single coordinate entry.
JRNL	Literature citation that defines the coordinate set.
REMARK	General remarks, some are structured and some are free form.

Table 3.3 Some important information available in a PDB file

the secondary structure and the non-covalent connectivity. All the connectivity information, however, is not contained in the CONECT record. Some connectivity is implicitly contained in the name of the atom and of the residue. For example, in the side chain of a valine, there are three carbon atoms, labeled respectively CB, CG1 and CG2 in the ATOM record. The program reading the molecule would have to be able to connect CB to both CG1 and CG2 [28]. A list of some important information available in a PDB file appears in Table 3.3. An entry has a particular code composed of 4 characters. The same molecule can be described in several entry, for instance, complexed with another molecule, mutated, or acquired with another experimental technique. The main techniques to acquire atom positions are the X-ray crystallography and the NMR spectroscopy. The X-ray crystallography is a method based on the measure of diffraction of the x-rays on the atoms of crystallized proteins.

3.5 Visualization of Proteins

There are several ways to represent and visualize a protein that can be used alternatively, depending on the user's preferences. In this section, the commonest representations are shortly described and illustrated. When the covalent bonds have to be visualized, they are represented as sticks and the atoms (sometimes represented as

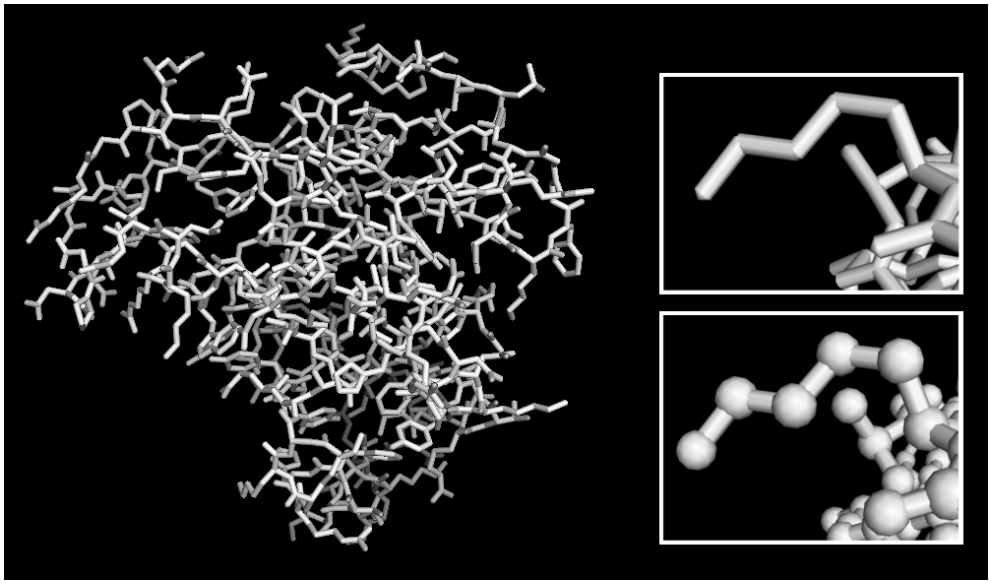

Figure 3.1 A protein (PDB code: 1I2M_A) with the covalent bonds represented as sticks. On the right, two detailed views: without (top) and with (bottom) balls representing the atoms. These images have been generated by PyMOL using the sticks option.

small balls) are at the intersection of these sticks (see Figure 3.1). A color is generally affected to a specific element. The atoms of hydrogen are generally not displayed. When the aim is to see the folding of the protein without being interested in the exact position of the side chains, it can be represented as a ribbon linking the alpha carbons or all the atoms of the backbone ($... - C - C_\alpha - N - ...$). This visualization can be enhanced by the introduction of secondary structure elements, representing helices as coiled ribbons and sheets as big flat arrows (see Figure 3.2). Finally, the protein can be represented as the surface defining the limit between accessible places and unaccessible places (see Figure 3.3). This kind of representations is the most widely used in this work and is more detailed in Section 4.3.

3.6 Protein Docking

Molecular docking is a field of research in which interaction between molecules are studied. Protein docking is restricted to proteins.

Within the protein docking, a differentiation can be made between protein-protein docking and protein-ligand docking. In the second case, the interaction between a protein and a ligand (small molecule binding to the protein with a biological purpose) is studied. The main difference between a ligand and a bigger molecule is the number of degrees of freedom. Thus considering a ligand as a rigid body does not lead to important errors, whereas the same kind of approximations for a protein may hide some possibilities of interaction.

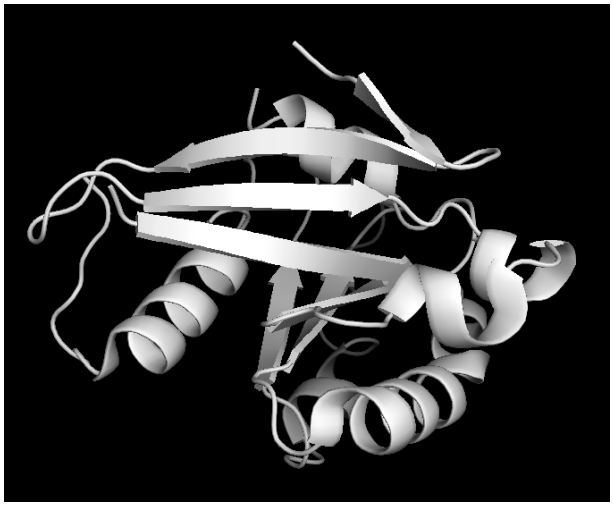

Figure 3.2 The backbone of a protein (PDB code: 1I2M_A) with the helices represented as coils and the sheets represented as big flat arrows. This image has been generated by PyMOL using the cartoon option.

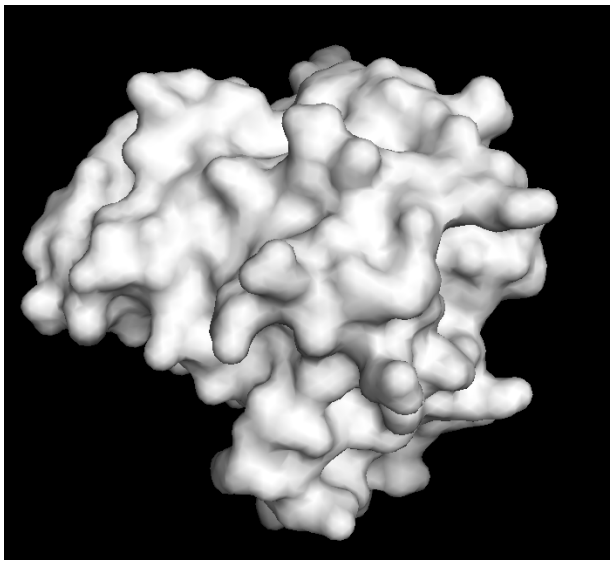

Figure 3.3 An image of the surface of a protein (PDB code: 1I2M_A). This image has been generated by PyMOL using the surface option.

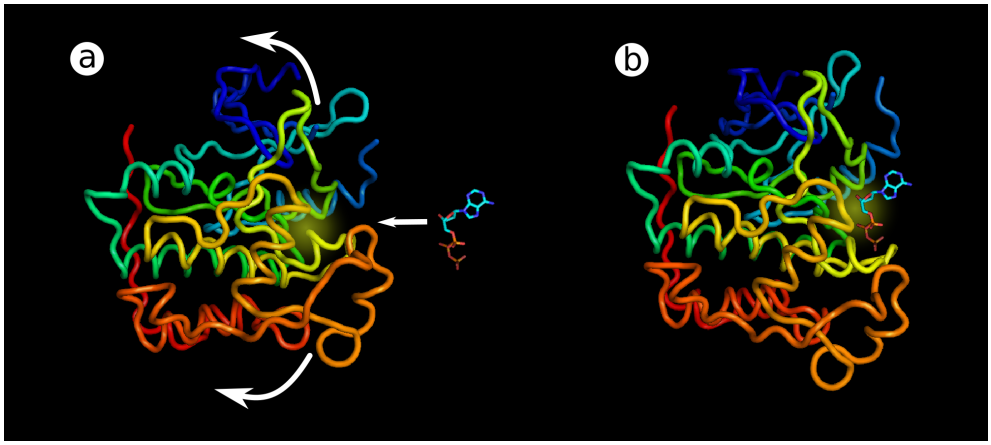

Figure 3.4 Illustration of a conformational change that open an active site. a) The active site (yellow halo) is unaccessible to the small ligand. The protein has to be deformed to make the interaction possible (white arrows). b) The active site is open and the interaction happens.

Protein docking already has a long academic history [29–31], and numerous algorithms (AutoDock, DOCK, FlexX, Gold, etc) are used each day in the academic world as well as in the pharmaceutical industry. Many of these algorithmic or software solutions, however, are exhaustive searches or assume that the affinity between both components to dock is high.

Most approaches to docking consist in two steps: rigid docking, producing some potential configurations considering only rotations and translations and a refinement step, locally improving the docking configuration, adding, for example, a limited degree of flexibility.

A distinction must be made between **bound** and **unbound** docking. In the first situation, the protein structures to dock are in their *bound* conformation, having a perfect steric, i.e. related to the spatial arrangement, complementarity with the target molecule. In the second situation, the protein structures are in their *unbound* conformation. It means that the interaction with the target molecule may require a conformational change, which is not easy to predict (see Section 3.6.3). The unbound docking is more interesting to study because since proteins have a continuously changing conformation, assuming that they are found naturally in their bound conformation is a strong hypothesis. Very often, proteins have to be deformed in order to interact with another molecule (Figure 3.4). Bound docking has interest, e.g. when information about the binding site topology is required, when no other structural information is available, or to test subparts of docking methods on easier examples.

3.6.1 Rigid Docking

In rigid docking, many possible docking sites are found, often in a simplified version of the problem. These approaches are mainly based on geometric complementarity [32]. A way to accelerate the computations is a uniform discretization of the rotations and translations spaces (rigid transformations) and to browse them exhaustively [32]. This approach can be accelerated using Fast Fourier Transform (FFT [33]) [34], which is the basis of the following docking programs: FTDock [35], 3D-Dock [36], GRAMM [37] and ZDock [38]. Other methods consist in discretization in a non-uniform way the rigid transformation space by matching critical points detected on both studied molecular surfaces [39,40]. This idea is a result of Connolly's proposition [41] to use minima and maxima of a function of the mean curvature, also known as the "Connolly function". An example of this method has been described by Fisher et al. [42] who use a geometric hashing technique [43] to match critical points.

3.6.2 Optimization for the Rigid Docking

A second step can be the research of an exact solution from the docking sites found in the first step. Both essential components of this step are a score function and an optimization algorithm. The score function discriminates configurations that seem realizable from incorrect solutions. This function is generally based on the thermodynamical aspects of the interaction. Then an optimization algorithm such as a genetic algorithm [44] or a Monte Carlo method [45] has to be used to find an optimum of this score function, so that, the molecular docking is modeled as the research of the minimum in an energy landscape resembling a funnel [46].

3.6.3 Flexible Docking

In a molecular docking process, it is also possible to introduce structural flexibility. In this case, 3D conformations of the molecules can be modified before or during the docking [47]. As mentioned in Section 2.6, this flexibility reflects a reality: molecules, and more specifically proteins, move and change their conformation when interacting with other molecules. Generally, conformation changes are realized by rotation of the structure around acyclic covalent bonds. Thus these changes modify the surface geometry as well as the energy score function.

Obviously, the difficulty is increased with the research space dimension or, equivalently, with the number of degrees of liberty which can be huge in this kind of problems [48]. Camacho et al. [30] proposed a protocol of progressive and hierarchical refinement, generally converging to an almost native configurations (natural functional configuration). This protocol does not take into account movements of the backbone, but only movements of the side chains. The study of conformational changes including the flexibility of the backbone does not seem to produce better results [49]. Since every refinement step is costly, it is essential that the set of possible configurations generated by the rigid docking is small and that the potential

solutions are not too far from the native configuration. Generally, current solutions to protein docking only respect one of these two constraints.

Another way to approach flexible docking is the soft docking. With this technique, the molecule is considered to be rigid but with a fuzzy boundary. Thus the interpenetration between two molecules is allowed but is compensated by a score to minimize and reflecting the intersection volume. [50]

3.6.4 Binding Site Detection

A way to reduce the number of degrees of freedom is to find potential binding sites, on the proteins without knowing the other molecule to dock with. Indeed, it has been pointed out that the binding sites share common geometric, physicochemical and evolutionary properties [51].

Chapter 8 is completely devoted to this issue.

3.7 Dynamic Modeling

As mentioned above, proteins in natural environment are not static. They are continuously animated in consequence of various forces and because the bonds between atoms have degrees of freedom (mainly angular). Modeling such animation is not an easy task because the atoms are close to each other, thus they can be influenced by the force fields of several other atoms. Moreover, the description of the involved forces can be complicated [28]. The description of the dynamics can be based on classical or quantum mechanics [52].

In this section, tools that are implementable and widely used in practice are presented: the force fields and the normal modes analysis.

3.7.1 Force Fields

In this approach of classical mechanics, atoms are considered to be punctual masses interacting with each other through various forces. When the global energy E_{tot} of the system is minimized, the atoms are considered to be in a stable position [28]. The global energy can be expressed as:

$$E_{tot} = E_{bond} + E_{angle} + E_{dihedral} + E_{vdW} + E_{coulomb}. \tag{3.1}$$

The bond, angle and dihedral energies are related to the covalent bonds (see Section 2.1). The van der Waals (vdW) and the Coulomb energies are related to non bonded atoms that are close enough to interact. This force field, whom different components are illustrated in Figure 3.5, is frequently used but there are other way to express the global energy, using different energy terms or different definitions of these terms [53–57]. The description of the energy terms comes from [28].

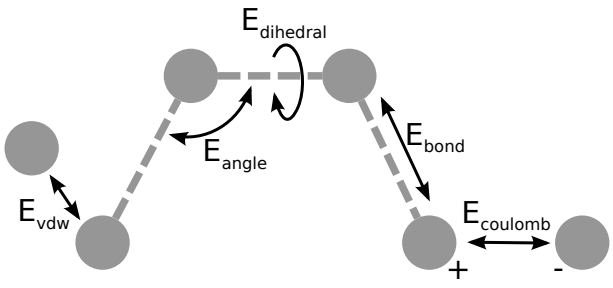

Figure 3.5 The energies taken into account in the force field of equation 3.1. The energies are each represented once but can appear at several places. For example, E_{bond} exists between the atoms of each bonded pair and E_{vdW} exists between each pair of close enough atoms.

Bond Energy The bond energy E_{bond} is the sum of the energies of covalent bonds. A covalent bond is modeled by a spring (Hooke's law). Then,

$$E_{bond} = \sum_{(i,j)\in B} K_{ij}^{b}(\| \overrightarrow{ij} \| - \| \overrightarrow{ij} \|^{*})^2, \tag{3.2}$$

where B is the set of pairs (i,j) such that the atom i is bonded to the atom j, $\| \overrightarrow{ij} \|$ is the distance between the atoms i and j, $\| \overrightarrow{ij} \|^{*}$ is the equilibrium distance and K_{ij}^{b} is a stretch force constant. $\| \overrightarrow{ij} \|^{*}$ and K_{ij}^{b} depend both on the nature of the bond ij.

Angle Energy The angle energy E_{angle} is the sum of the covalent bond-angle bending energies. It is modeled as a torsion spring. Then,

$$E_{angle} = \sum_{(i,j,k)\in A} K_{ijk}^{a}(\theta_{ijk} - \bar{\theta}_{ijk})^2, \tag{3.3}$$

where A is the set of atoms triplets (i,j,k) such that the atoms i and k are bonded to j, θ_{ijk} is the $\widehat{ijk}$ angle, $\bar{\theta}_{ijk}$ is the equilibrium angle and K_{ijk}^{a} is a torsion force constant.

Dihedral Energy The dihedral energy $E_{dihedral}$ is the sum of the covalent dihedral angle bending energies. It is modeled as follows:

$$E_{dihedral} = \sum_{(i,j,k,l)\in D} \frac{K_{ijkl}^{d}}{2}[1 + \cos(n_{ijkl}\phi_{ijkl} - \bar{\phi}_{ijkl})], \tag{3.4}$$

where D is the set of atoms quadruplets (i,j,k,l) with three covalent bonds, ϕ_{ijkl} is the angle between the planes ijk and jkl, n_{ijkl} is the periodicity of the dihedral angle ($n = 2$ for planar centers and $n = 3$ for tetrahedral centers), $\bar{\phi}_{ijkl}$ is the equilibrium angle and K_{ijkl}^{d} is a torsion force constant.

Van der Waals Energy The vdW energy E_{vdW} is the sum of the van der Waals interaction energies between each pair of atoms. However, the contribution for atoms

connected by less than three consecutive bonds is included in the first three energies, thus it is not included in this energy term.

$$E_{vdW} = \sum_{j \in F} \sum_{i \in N_j} \left[\frac{A_{ij}}{\| \overrightarrow{ij} \|^{12}} - \frac{B_{ij}}{\| \overrightarrow{ij} \|^{6}} \right], \tag{3.5}$$

where F is the set of all atom indices, N_j is the set of atoms from another molecule or atoms that are separated from j by at least three consecutive bonds and such that $i < j$, A_{ij} and B_{ij} are the Lennard-Jones 12-6 parameters [58] and r_{ij} is the distance between the atom i and the atom j.

Coulomb Energy The Coulomb energy $E_{coulomb}$ is the sum of the electrostatic energies existing between each pair of charged atoms. As for the E_{vdW}, the contribution for atoms connected by less than three consecutive bonds is not included in this energy term.

$$E_{coulomb} = \sum_{j \in F} \sum_{i \in N_j} \frac{q_i q_j}{\epsilon \| \overrightarrow{ij} \|}, \tag{3.6}$$

where F is the set of all atom indices, N_j is the set of atoms from another molecule or atoms that are separated from j by at least three consecutive bonds and such that $i < j$, q_i and q_j are the atoms partial charges, ϵ is the dielectric constant for the environment and r_{ij} is the distance between the atom i and the atom j.

3.7.1.1 Computer Simulations

The implementation of force fields could seem easy and direct but some factors make the computation much more complicated. First, the number of atoms and of bonds in a protein is quite important. Generally, there are between 1000 and 10000 atoms in a protein and this number can be increased when simulating interactions between several molecules. Moreover, reactions and interactions take place in a solvent, which is also composed of small molecules and makes the atom count increase. The complexity is still increased because finding the energy minimum given the number of dimensions of the problem is not explicitly possible [46]. Numerical solutions are computed for billions of short time steps, each requiring a new evaluation of the potential energy. Sometimes, in order to find other local minima, simulated annealing is performed, extending the computation time. In this technique, the system is excited by making the energy grow during short phases, followed by minimum energy conformation search again.

Various solutions to simplify the calculation exist. For example, the bonds lengths and bonds angles may be considered to be constant, reducing the global energy term to $E_{tot} = E_{dihedral} + E_{vdW} + E_{coulomb} + K$, where K is a constant. The vdW and Coulomb energies can also be evaluated on a small neighborhood instead of all the atoms, because these terms become negligible as $\| \overrightarrow{ij} \|$ increases (equations 3.5 and 3.6). Another frequent simplification is to consider that the solvent is not

composed of molecules but is continuous. This approach is referred as continuum modeling or implicit solvent modeling. In this case, the dielectric constant ϵ in equation 3.6 is replaced by a function simulating the effect of the solvent molecules [59].

Some famous force fields and set of parameters are AMBER, CHARMm, GROMOS and OPLS. AMBER, CHARMm and GROMOS are also the name of the corresponding packages of programs for molecular dynamics simulations. NAMD [60] is an open source program using the CHARMM force field for simulations.

3.7.2 Spectral Analysis

Studying the spectral properties of the relations between atoms in a large molecule is a way to estimate the movements that this molecule is likely to have. In the normal mode analysis, the eigenvectors of the second derivative of the potential energy matrix K are studied in order to find low frequency modes that are the expression of low energy conformation movement [61]. The computation times are largely reduced in comparison with the classical force field computations and the solutions are more global. The main assumption is that the potential energy can be expressed as a quadratic function of this form:

$$E_p = \sum_{(i,j)\in P} K_{ij}(\| \overrightarrow{ij} \| - \| \overrightarrow{ij} \|^*)^2,$$

where P is the set of all atoms pairs, K_{ij} is an energy coefficient, $\| \overrightarrow{ij} \|$ is the distance between the atoms i and j and $\| \overrightarrow{ij} \|^*$ is the distance between the atoms i and j in the given crystallized structure. The coefficients K_{ij} have to be chosen in order to approximate at best all the energy terms of equation 3.1. Since the interactions are considered to be null if $\| \overrightarrow{ij} \|$ is too large, the matrix K can be sparsified by restricting the set P to pairs of atoms $\{(i,j) | \| \overrightarrow{ij} \| < d_t\}$, with d_t a threshold distance, typically 8 Å [62].

3.8 Conclusion

This chapter focused on common tools of information technology related to protein docking (Section 3.6). In this context, this thesis mainly focuses on surface and structure features extraction and on the detection of sites of interest on protein surfaces. Details about protein surfaces and other protein representations were presented in Section 3.5. The representation which is used most of the time is a polygonal mesh (see next chapter) of the Solvent Excluded Surface (SES) of which the construction is described in Section 4.3. All the studies about proteins compositions and 3D conformations are realized on structures coming from the PDB (Section 3.4). For applications requiring statistical tests or statistical model training, data sets of structures are constructed. In order to avoid redundancy, similarity between proteins is measured through sequence alignment (Section 3.1), which is also used to

compute the conservation score, helpful for binding site prediction (Chapter 8). Indeed, residues belonging to sites of interest may not be mutated without collateral damages and are consequently more conserved. The prediction of the effect of mutations on the structure and on the function of proteins is of great interest for diagnosis of genetic diseases and it would be a good perspective for future works. Since a 3D conformation can not be easily predicted given a sequence (Section 3.3), it could be interesting to modify a residue in a known structure in silico and to predict conformational changes after dynamic simulations (Section 3.7). Molecular dynamics simulations could also be integrated in a global docking process and, in particular, in binding sites prediction methods. In order to consider the molecules flexibility in this work without too much increasing computation times, an index of flexibility (Section 7.1.3) reflecting the mobility of the atoms during a dynamic simulation is computed. The mobility of the atoms is measured by structure alignment (Section 3.2).

Surface Based Approach

4

The work of this chapter has been published in [63].

In this chapter, the different possibilities of representing 3D objects numerically are presented (Section 4.1). Polygonal meshes are collections of points, linked by edges and faces representing a surface in a 3D environment (Section 4.2). In this work, protein properties are mainly analyzed on the Solvent Excluded Surface (SES), which is modeled as a polygonal mesh. Methods generating a mesh representing the SES from the atoms positions are detailed in Section 4.3.

4.1 Solid or Boundaries Representation

There are several ways to represent 3D objects and to visualize them in a computer environment. Two categories can be distinguished: the solid representations and the boundaries representations. The solid representations are descriptions of the 3D occupancy of the considered object. In that case, a value is associated to each point of the 3D space. An example is the voxels (equivalent to 3D pixels) representation, in which the space is divided into cubes corresponding to a value, e.g. a grayscale intensity. The boundaries representations are descriptions of the frontiers of the considered object. These representations can be analytical with some control points, like the Bezier surfaces, the B-Splines, and the NURBS, or discrete like the polygonal meshes. Only the theory about polygonal meshes is developed in this work because it is the chosen solution to represent protein surfaces. This choice is justified in the next section.

4.2 Polygonal Meshes

Polygonal meshes are collections of points linked by polygonal faces and/or lines in order to form a surface approximating the frontier of the object. The main advantage of a mesh is the low amount of information used to describe the objects. It leads to a small memory need and to computationally efficient processing algorithms. The main drawback of this kind of representation is the absence of information about the inner part of the volume. However, the boundaries of a molecules can be considered

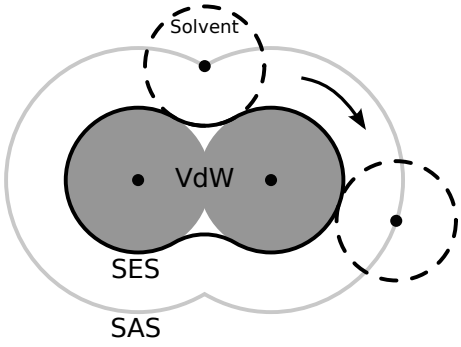

Figure 4.1 A cutaway view of a small molecule. The most frequent molecular surface representations are detailed. Gray discs depict the van der Waals volumes, the gray outer line depicts the Solvent Accessible Surface (SAS) and the continuous black line depicts the Solvent Excluded Surface (SES). The SAS (*resp.* SES) is the limit surface for the solvent molecule center (*resp.* external surface). The solvent molecule is represented as a black dashed circle.

to be sharp and the inner part to be impenetrable because of the profile of the van der Waals forces (see Section 3.7.1), hence the polygonal mesh approximation seems appropriate. The meshes considered in this work are manifolds, in other words, they can be considered locally as 2D Euclidean spaces (planes).

4.3 Protein Surface Modeling

At first, the external surface of a molecule has to be defined. Indeed, molecules are made of atoms which have no real surface. The most frequent molecular surface representations are the van der Waals Surface (vdWS), the Solvent Accessible Surface (SAS), and the Solvent Excluded Surface (SES) [29, 64]. In the case of the vdWS, the electron clouds around atoms are approximated by rigid spheres of different radii which correspond to the van der Waals (vdW) radii of the atoms. The SAS (*resp.* SES) is the inner surface of the volume filled by the possible positions of the center (*resp.* exterior surface) of a ball representing a molecule of solvent, e.g. water (see Figure 4.1).

Polygonal meshes are efficient tools to represent such surfaces. In the last few years, many methods have been developed for the generation of molecular surface meshes. In 1983, Connolly [41] proposed an analytical algorithm in which points were strategically placed around the molecule with a specific analytical role (maximum, minimum or saddle point) depending on the number of atoms present in the neighborhood. In 2003, Bajaj et al. [65] introduced another analytical method based on NURBS that offers the advantage to be parameterizable without recalculation. In 2002, Laug and Borouchaki [66] used a parametric representation of intersecting spheres to create the surface mesh. MSMS, developed by Sanner [67] is based on alpha-shapes [68] of molecules. This algorithm is widely used because it is time

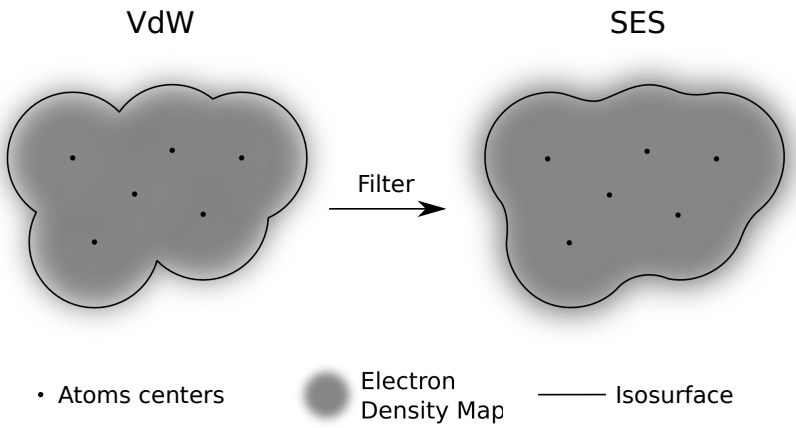

Figure 4.2 Overview of the surface generation process. An electron density cloud (blurred gray cloud) is place around each atom center (black dot). The isosurface of the union of clouds is the vdW surface before filtering, and the SES after filtering.

efficient, but the generated mesh is not a manifold and is composed of very irregular triangles. The beta-shapes [69] are a generalization of the alpha-shapes and were used by Ryu [70] in 2007 to design a similar algorithm. Another vertex based method was used by Cheng and Shi [71]. In this method, molecular surfaces are generated with the help of restricted union of balls. Finally, some methods based on volumetric computation exist, such as the one of Zhang et al. [72] in which the solvent accessible surface is seen as the isosurface of Gaussian shaped electron density maps, and the algorithm of Can et al. [73] (the LSMS) which is based on a front propagation from atom center and on level-sets.

The computational time, however, remains generally high for quality meshes, and it may be a problem when there is a great amount of data to treat. In this section, we introduce a fast and parameterizable method to generate smooth molecular surface meshes. The generated mesh is the isosurface of a filtered electron density map. Two tools can be used to filter the density map: a morphological closing or an ideal low pass filter on the 3D image spectrum. The two versions of the method are respectively called Closed Density Map (CDM) and Fourier Density Map (FDM).

4.3.1 Method

The CDM and the FDM methods are based on volumetric electron density and a filtering. Each atom is seen as a Gaussian electron cloud, the dimensions of which are depending on the vdW radius (Figure 4.2). Then, this density map is filtered in order to smooth the 3D image and to fill regions that are unaccessible to solvent molecules. Finally, a marching cubes [74] algorithm is used on the reverse Fourier transform to

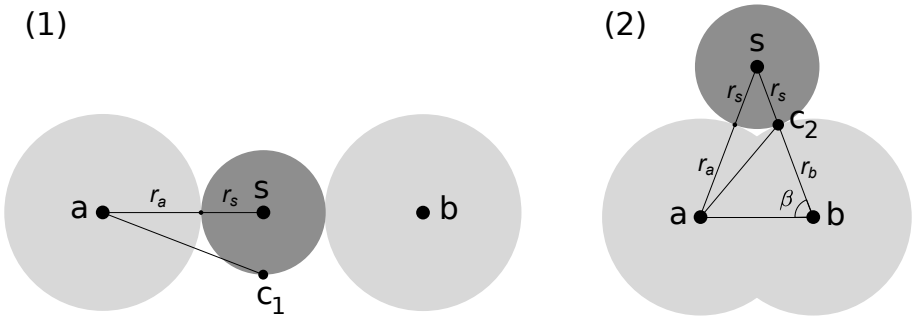

Figure 4.3 Two situations in which the Gaussian functions would have contradictory properties. (1) Two atoms of the molecule, a and b, are just close enough to block the way to a solvent molecule s. The point c_1 is on the SES and must be influenced by the filed generated by a, G_a. (2) a and b are close. The point c_2 is on the SES but also on the vdWS, thus it should not be influenced by G_a. It is possible to show that in some (frequent) configurations, $\|\overrightarrow{ac_1}\| > \|\overrightarrow{ac_2}\|$.

find an isosurface. A refinement of the final mesh constitutes an optional step of the method.

4.3.1.1 Electron Density Map

A Gaussian function is constructed around each atom. The value of this function at a point i for an atom a is:

$$G_a(i) = te^{-\left(\frac{r_a^2 - \|\overrightarrow{ai}\|^2}{r^2}\right)}, \tag{4.1}$$

where t is a threshold parameter, r is a radius parameter, $\|\overrightarrow{ai}\|$ is the Euclidean distance between the center of a and the point i, r_a is the vdW radius of the atom a. Thus the isosurface for the threshold t is the vdW sphere because if i is located on the vdWS of a, $\|\overrightarrow{ai}\| = r_a$ and $G_a(i) = t$. In this work, r is set to 3Å because this value is suitable for an ideal low-pass filter (see Section 4.3.1.2).

For the implementation, the three-dimensional space is divided into voxels. The spacing (T_e) between voxels is an adaptable parameter. The more T_e is small, the more the surface approximation is fine.

The density map of the whole molecule for a point in the space is defined as the maximal value of all the Gaussian functions at this point. The maximum of the Gaussian functions is chosen instead of the summation because it is not possible to evaluate the SES using the isosurface of a summation of Gaussian functions. It can be shown by the following counterexample, in which the Gaussian affected to the atoms a and b must have contradictory properties depending on the situation.

In the first situation, the space between a and b is just small enough to block the way to a solvent molecule (see Figure 4.3(1)). The other atoms are considered to be too far to have an influence. Thus the SES has a concave shape at this place and the

value of the density map at the "center" of the concavity c_1 must be influenced by the fields of a and b: $G_a(c_1) > 0$, $G_b(c_1) > 0$. We can state that $\|\overrightarrow{ac_1}\| > r_a + r_s$ because s, the center of the solvent molecule, can be very close to the ab axis.

In the second situation, $\|\overrightarrow{ab}\| < r_a + \frac{r_s}{2}$, which is often the case for covalent bonds. Let c_2 be a point belonging both to the SES and to the vdWS of b with the necessary condition $\|\overrightarrow{bc_2}\| = r_b$ (see Figure 4.3(2)). The point c_2 should not be influenced by the field of a, so $G_a(c_2) = 0$. If $\|\overrightarrow{ac_2}\| > \|\overrightarrow{ac_1}\|$, this condition is in contradiction with $G_a(c_1) > 0$, because the Gaussian function is strictly decreasing in the positive domain. Using the Al-Kashi theorem, we know that:

$$\|\overrightarrow{ac_2}\|^2 = r_b^2 + \|\overrightarrow{ab}\|^2 - 2r_b\|\overrightarrow{ab}\|\cos\beta,$$

and

$$\|\overrightarrow{ac_1}\|^2 > (r_a + r_s)^2 = (r_b + r_s)^2 + \|\overrightarrow{ab}\|^2 - 2(r_b + r_s)\|\overrightarrow{ab}\|\cos\beta,$$

where β is the $\widehat{abs}$ angle. Thus $\|\overrightarrow{ac_1}\| > \|\overrightarrow{ac_2}\|$ if

$$r_s^2 + 2r_br_s - 2r_s\|\overrightarrow{ab}\|\cos\beta > 0,$$

what is verified by the hypothesis: $\|\overrightarrow{ab}\| < r_b + \frac{r_s}{2}$ because $\cos\beta \leq 1$.

In order to avoid interferences, the maximum is preferred to the summation of Gaussian functions. Isosurfacing this density map returns the vdWS. This surface is not smooth and in order to compute the SES, the density map must be filtered.

4.3.1.2 Map Filtering

Isosurfacing the electron density map provides the vdWS. In order to obtain the SAS or the SES, the map has to be processed. Two ways of processing are presented in this section. First, the morphological operations with the advantage that it follows exactly the definition of the surfaces, and second, the filtering in the frequency domain with the advantage that it is less time consuming while providing a result with an equivalent visual aspect.

Morphological Operations Morphological operations modify a scalar field with the help of a structuring element. The two basic morphological operations are the dilatation and the erosion. There are many definition of such operations in the case of non-binary scalar fields [75]. In this work, the dilatation ($\oplus$) and erosion ($\ominus$) are defined as followed:

$$(G \oplus W)(i) = \max_{j \in W(i)} G(j),$$

$$(G \ominus W)(i) = \min_{j \in W(i)} G(j),$$

where, G(i) is the value of the scalar field at the point i, W is the structuring element and $W(i)$ is the zone contained in the structuring element when centered around i. It

is important to notice that these operations are not commutative. The morphological closing (Figure 4.4) is a succession of a dilatation and an erosion $((G \oplus W) \ominus W)$ and a morphological opening is a succession of an erosion and a dilation $((G \ominus W) \oplus W)$. For this application, the electron density map can be filtered by a morphological closing with a structural element representing a solvent molecule. Normally, this solvent is water and generally, it is approximated by a sphere of radius 1.4 Å. Intuitively, it returns the field that is accessible by making a solvent molecule rolling on the existing field. In accordance to the definition, taking an isosurface of this filtered field gives the SES. The SAS can be obtained similarly after a dilatation only.

Fourier Transform and Frequency Filtering Another way to filter the original density map is a frequency filtering. The Fourier transform of this electron density map is computed using the FFT algorithm [33]. The frequency representation of the function is filtered by an ideal low pass filter in order to eliminate frequencies corresponding to elements smaller than a solvent molecule, e.g. inflexion points between two vdW spheres.The cutoff frequency is $f_c = \frac{1}{4r_s}$, where r_s is the radius of the sphere approximating the solvent molecule (typically 1.4Å for water). The wave length must be four times longer than r_s because a molecule solvent diameter has to fit in a half wave length (see Figure 4.5).

Gaussian functions are preferred to balls in the spatial domain because an ideal low pass filter makes the Gibb's phenomenon appear on sharp edges. An ideal filter is used because the cutoff frequency is exactly known and because it is numerically possible. An ideal low-pass filter in the frequency domain is equivalent to a convolution product with a *sinc* function in the space domain. Let $X = (x, y, z)$ be the space variable in $\mathbb{R}^3$ and $\mathcal{M}(X)$ be the initial electron density map. Then, the filtered density map is:

$$\widetilde{\mathcal{M}}(X) = \big(\mathcal{M}(\bullet) * 2f_c sinc(2f_c \bullet)\big)(X),$$

where $\bullet$ represents the variable. The parameter r of the Gaussian functions in (4.1) is related to the width of the function. To keep the isosurface at the same place, a wider function has a smaller maximum. If r is too small, the maximum is high and the secondary ripples of the *sinc* function take too much importance when they are in phase with this maximum. It makes oscillations appear in the final density map, which may lead to the apparition of unwanted surfaces after isosurfacing. Simulations with several 1D and 2D functions were performed to verify the effect of r. The three main conditions to verify are:

$$\begin{cases} \widetilde{\mathcal{M}}(X_{c_1}) = t & \text{for } c_1 \text{ lying on the SES and on the vdWS,} \\ \widetilde{\mathcal{M}}(X_{c_2}) = t & \text{for } c_2 \text{ lying on the SES but not on the vdWS,} \\ \widetilde{\mathcal{M}}(X_{c_3}) < t & \text{for } c_3 \text{ lying outside the molecule.} \end{cases} \tag{4.2}$$

Here is a 2D example:

$$\mathcal{M}(x) = \max\left(G_a(x), G_b(x)\right),$$

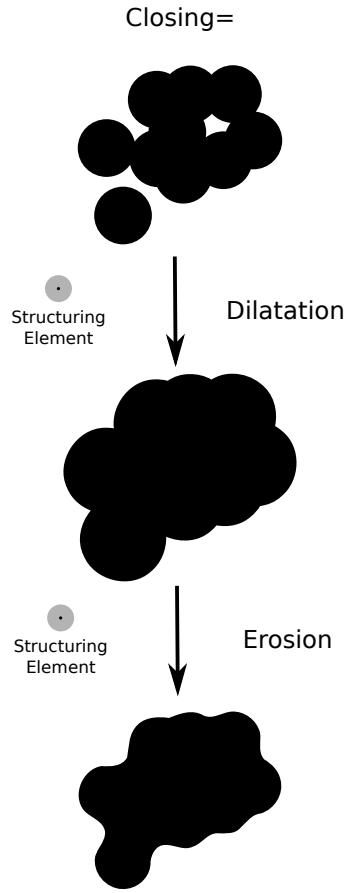

Figure 4.4 A 2D representation of a morphological closing operation. The closing is a succession of a dilatation and an erosion. The structuring element determines the extension and the shape of the operations. In the case of molecular surface modeling, the structuring element represents a molecule of solvent and the closed image marks the frontier between accessible and non-accessible zones for this molecule.

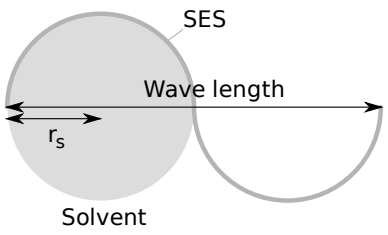

Figure 4.5 Minimal wave length allowed on the SES. It corresponds to four times the solvent radius r_s and determines the cutoff frequency for the filtering.

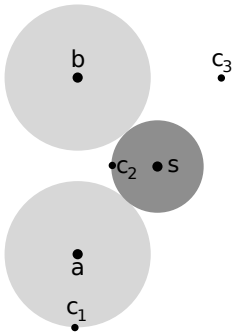

Figure 4.6 In this example, the atoms a and b are 5Å far from each other and have both a vdW radius of 1.8Å. The radius of the solvent molecule s is 1.4Å. The conditions to verify for the density map before ($\mathcal{M}$) and after ($\widetilde{\mathcal{M}}$) filtering are: $\mathcal{M}(c_1) = \widetilde{\mathcal{M}}(c_1) = t$, $\mathcal{M}(c_2) < \widetilde{\mathcal{M}}(c_2) = t$, and $\mathcal{M}(c_3) < t \quad and \quad \widetilde{\mathcal{M}}(c_3) < t$, with t, the threshold value for the isosurfacing.

with the atom a centered in $(x_a, y_a) = (0, -2.5)$ and the atom b in $(x_b, y_b) = (0, 2.5)$, the threshold $t = 1$, and $r_a = r_b = 1.8$ (Figure 4.6). The value of the density maps $\mathcal{M}(X)$ and $\widetilde{\mathcal{M}}(X)$ for $c_1 = (0, x_a - r_a)$, $c_2 = (\sqrt{x_a^2 + (r_a + r_s)^2} - r_s, 0)$ and $c_3 = (x_3, y_3)$ are plotted as a function of r in Figure 4.7. (x_3, y_3) is the position of the maximal value of the density map outside the "molecule" for $r = 1$. When the parameter $r = 3$, the conditions (4.2) are verified and the Gaussian functions are not too wide, what leads to shorter execution times. The 2D density maps $\mathcal{M}(X)$ and $\widetilde{\mathcal{M}}(X)$ with $r = 3$ as well as $\widetilde{\mathcal{M}}(X)$ with $r = 1$ are shown in Figure 4.8. The isocontours, representing the vdWS or the SES are depicted in white and we can see the artifacts appearing for too small values of r. It is important to notice that if the spacing (T_e) for the spatial sampling is too large, there would be no filtering. Normally, the sampling frequency, $f_e = \frac{1}{T_e}$, has to verify the Nyquist-Shannon theorem: $f_e > 2f_{max}$, where f_{max} is the higher frequency with a non-zero coefficient in the original signal. For this application, however, the errors resulting from a subsampling are not too important and the sampling frequency is chosen such that $f_e > 2f_c$, which is equivalent to $T_e < 2r_s$. In this situation, the filtering is always possible.

4.3.1.3 Isosurfacing

The final triangular mesh is an approximation of the isosurface of the modified electron density map. The most popular technique to extract an isosurface from a 3D image is the marching cubes algorithm [74]. In this algorithm, the voxels are screened by group of eight sharing the same point. Mesh vertices, faces and edges are added depending on the value of these eight voxels. There are 256 (2^8) possibilities that can be reduced to 15 situations thanks to symmetries and complementarities.

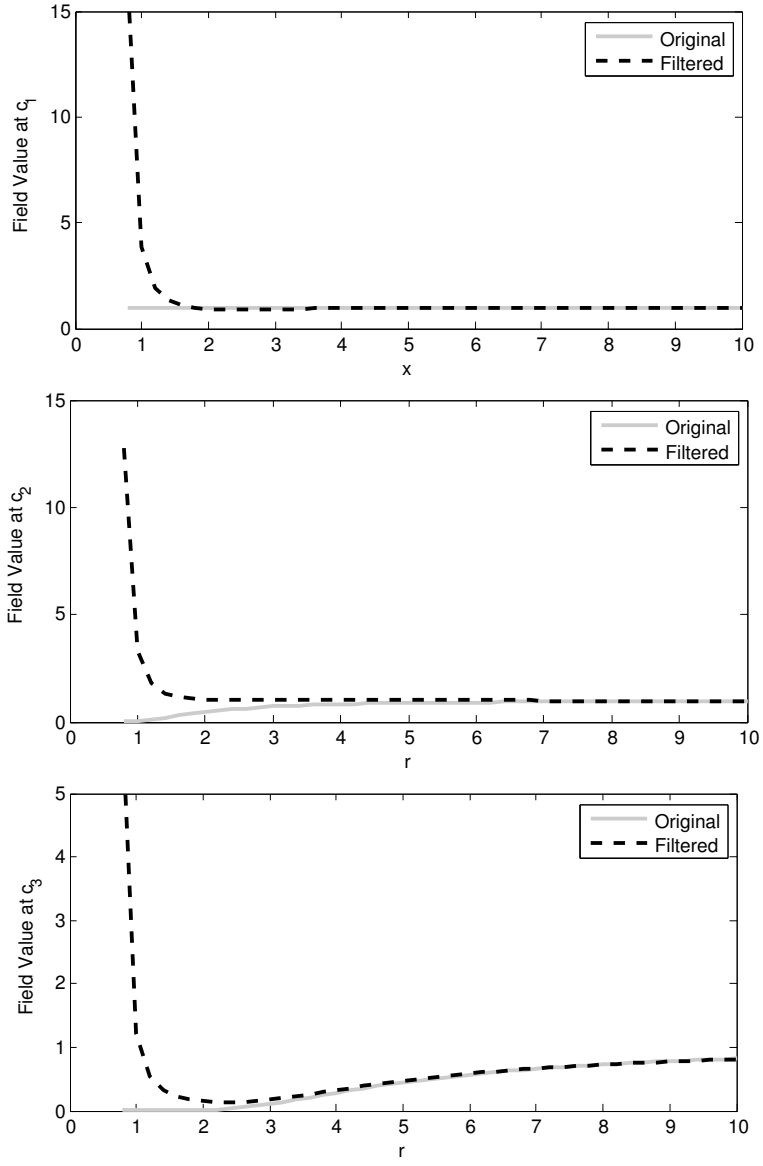

Figure 4.7 Value of the density map vs. the parameter r of equation 4.1 before (continuous gray line) and after (dashed black line) filtering at a point belonging to both the SES and the vdWS, c_1 (top); a point belonging to the SES but not the vdWS, c_2 (middle); and a point outside the molecule, c_1 (bottom) (see Figure 4.6). $t = 1$, hence, when $r = 3$, the conditions (4.2) are satisfied.

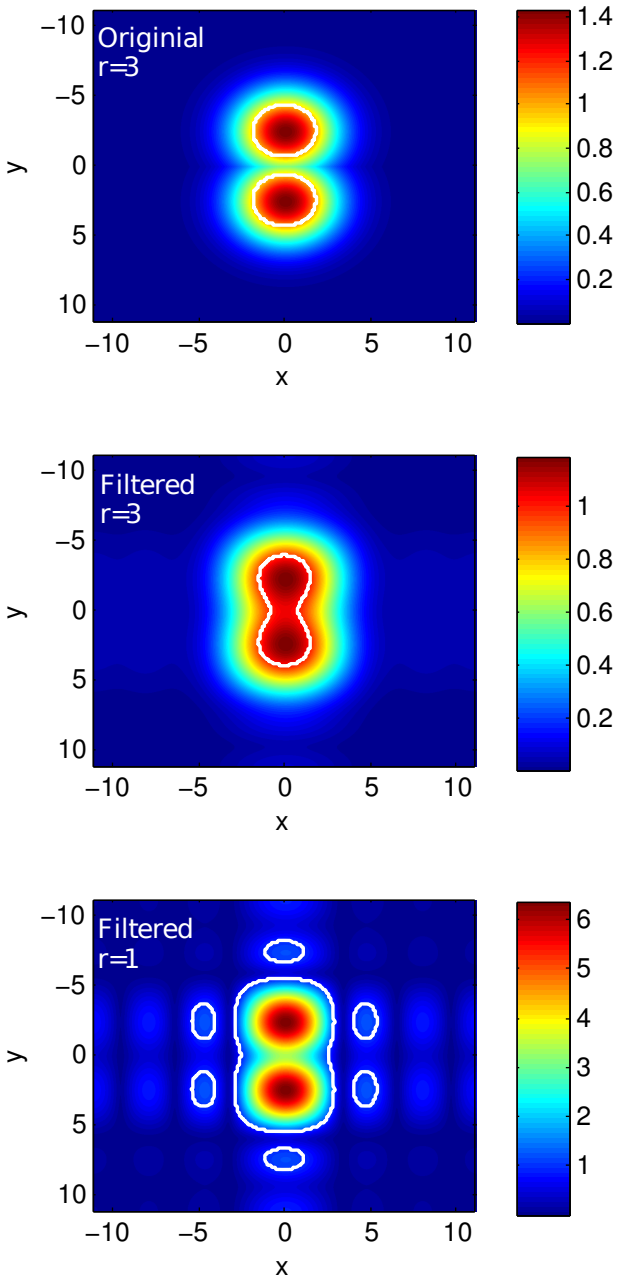

Figure 4.8 Value of the density map for the example of the Figure 4.6. x and y are the two spatial dimensions (Å) and the colors represent the value of the density map before filtering and with $r = 3$ (top), after filtering and with $r = 3$ (middle) and after filtering with $r = 1$ (bottom). The contour for $t = 1$ is depicted with a white line. It represents the vdWS (top), the SES (middle) and the SES with artifacts (bottom).

4.3.1.4 Refinement

The visual appearance of the final mesh can be improved by magnifying the number of vertices. The number of vertices is increased using a smooth interpolation scheme such as the piecewise smooth surface reconstruction of Hoppe et al. [76], or the algorithm based on the butterfly scheme proposed by Zorin et al. [77].

4.3.2 Results

Some numerical results pointing out advantages and drawbacks of the CDM and the FDM are shown in this section. The main characteristics to be observed are the computation time and the quality of the generated mesh. The section is divided into two parts: the analysis of the effects of the different parameters and the computation time comparisons with other existing methods.

4.3.2.1 Parameters

There are three main parameters modifiable by the user. First, the spatial spacing T_e, i.e. the distance between two neighbor voxels center, which determines the total number of voxels. Second, the size of the structuring element r_p (for the CDM), which determines the size of the hole that are filled during the morphological closing or the cutoff frequency f_c (for the FDM), which determines the smoothness of the final mesh. And third, the refinement rate k_r, i.e. the number of new points in a triangle for the final mesh magnification. In this section, the effect of these parameters on the visual quality, on the computation time and on the memory space are discussed.

Spatial Spacing With a small spatial spacing, it is possible to represent fine details. However, it drastically increases the memory space needed as well as the computation time. Indeed, reducing T_e by a factor α increases the number of voxels by α^3. The parts of the method depending on the number of voxels are the creation of the density map (time: $\mathcal{O}(n_v)$ and space: $\mathcal{O}(n_v)$, with n_v the number of voxels) and the Fast Fourier transform (time: $\mathcal{O}(n_v \log n_v)$ and space: $\mathcal{O}(n_v)$). A visual comparison between meshes generated with $T_e = 1.9r_s$, $T_e = r_s$ and $T_e = \frac{r_s}{3}$ is shown in Figure 4.9. In these examples, $f_c = \frac{1}{4r_s}$ and $k_r = 0$, then, the meshes represent the SES without final refinement.

Structuring Element Size/Cutoff Frequency In order to generate a mesh representing the SES, r_p is set to 1.4 Å or f_c is set to $\frac{1}{4r_s}$ (see Section 4.3.1.2). Depending on the application, however, the surface could be other than the SES. For example, if $r_p = 0$ or $f_c > \frac{f_e}{2}$, there is no actual filtering, and thus the generated mesh represents the vdWS. On the other hand, to obtain a smooth approximation of the molecule shape, r_p can be enlarged or f_c can be reduced. A larger structuring element radius increases the computation time and memory space needed whereas changing f_c does

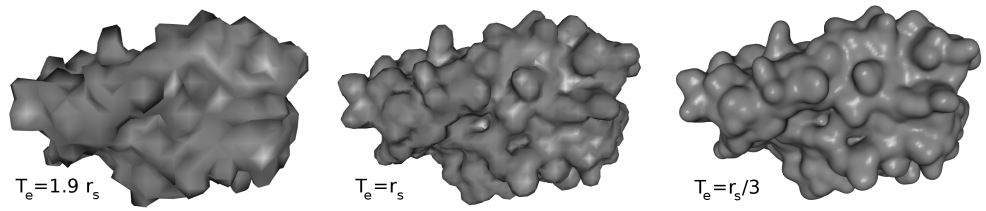

Figure 4.9 Meshes generated from electron density map at a spatial spacing of $T_e = 1.9r_s$ (left), $T_e = r_s$ (center) and $T_e = \frac{r_s}{3}$ (right). (PDB code: 3EBZ)

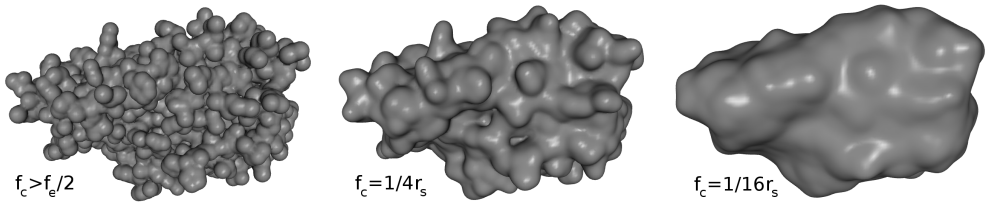

Figure 4.10 Meshes generated from electron density map filtered at a cutoff frequency of $f_c > \frac{f_e}{2}$ (left) to give the vdWS, $f_c = \frac{1}{4r_s}$ (center) to give the SES and $f_c = \frac{1}{16r_s}$ (right) to give an approximation of the general shape. (PDB code: 3EBZ)

neither have any effect on the computation time nor on the memory space needed. A visual comparison between meshes generated with $f_c > \frac{f_e}{2}$ (vdWS), $f_c = \frac{1}{4r_s}$ with $r_s = 1.4$ Å (water SES), and $f_c = \frac{1}{16r_s}$ with $r_s = 1.4$ Å (shape approximation) is shown in Figure 4.10. In these examples, $T_e = \frac{r_s}{3}$ and $k_r = 0$, then, a solvent molecule diameter takes 6 voxels and there is no final refinement. Another visual comparison between meshes generated with $r_s = 0$ Å (vdWS), $r_s = 1.4$ Å (water SES), and $r_s = 5.6$ Å (shape approximation) is shown in Figure 4.11. In these examples, $T_e = \frac{1.4}{3}$ and $k_r = 0$, then, a water molecule diameter takes 6 voxels and there is no final refinement.

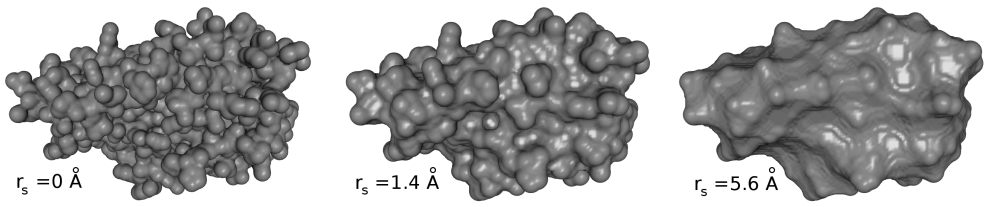

Figure 4.11 Meshes generated from electron density map morphologically closed with a spherical structuring element with a radius of $r_s = 0$ Å (left) to give the vdWS, $r_s = 1.4$ Å (center) to give the SES and $r_s = 5.6$ Å (right) to give an approximation of the general shape. (PDB code: 3EBZ)

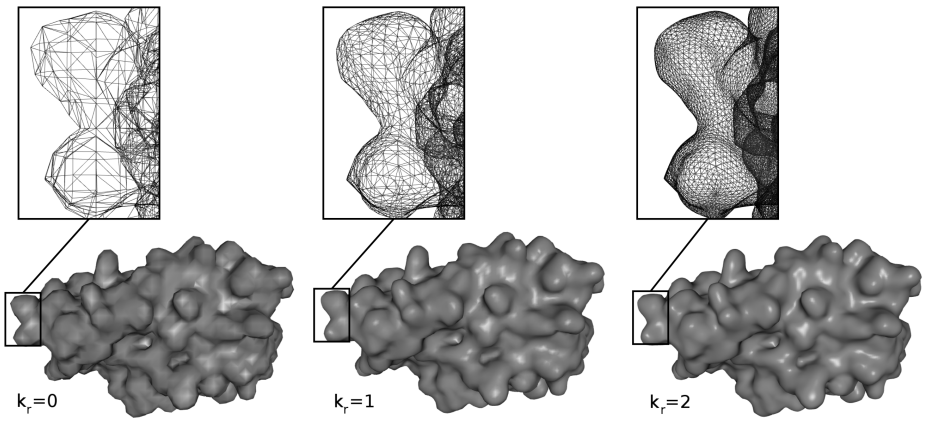

Figure 4.12 Meshes refined with a factor of $k_r = 0$ (left), $k_r = 1$ (center) and $k_r = 2$ (right). (PDB code: 3EBZ)

Refinement Factor The final mesh refinement gives foremost an aesthetic advantage. The needed memory space does not considerably increase because the number of voxels remains the same. Only the size of the mesh changes and this is negligible in comparison with the space needed by the voxels representation. The computation time slightly increases but, when $k_r = 1$ (which gives a good visual improving), this is negligible in comparison with the voxels operations computation time. A visual comparison between meshes generated with $k_r = 0$, $k_r = 1$ and $k_r = 2$ is shown in Figure 4.12. In these examples, $T_e = r_s$ and $k_r = 0$, then, a solvent molecule diameter takes 2 voxels and there is no final refinement.

4.3.2.2 Time Comparisons

In this section, computation times are compared between the CDM algorithm, the FDM algorithm and algorithms found in the literature for equivalent qualities. When available, the algorithm were run on the same computer, when not, the computation times were those announced in the original paper. Can et al. made a comparison of their method computation time with three molecular visualization tools: UCSF Chimera [78], Swiss-PDBViewer [79] and PyMol [80]. We added MSMS [67] to this set of methods. These programs, as well as the one of Can (LSMS), are available for free, thus the computation time could be measured on the same computer than for our method, except for Chimera that was not supported by the system. Thus the computation times mentioned here for Chimera are the one announced in the paper of Can [73]. For the LSMS method, the grid size is set to 256x256x256. In this condition, $T_e \simeq 1$ with the tested molecules. Other programs are run with defaults settings ($T_e \simeq 1$). Two tests are made for the FDM method. In the first one, the parameters are set to $T_e = 1.9r_s$ and $k_r = 0$, in order to be as fast as possible while keeping a correct solution. In the second one, the parameters are set to $T_e = \frac{4}{3}r_s$ and

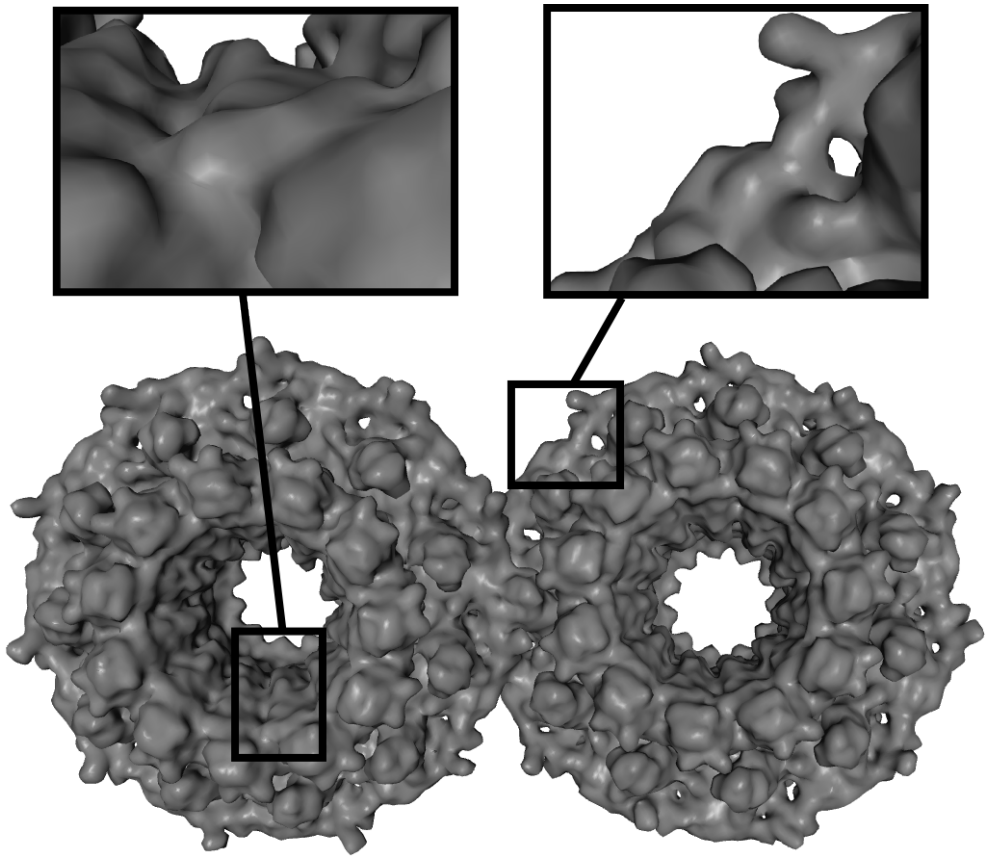

Figure 4.13 Mesh of the SES of a molecule generated with the FDM method with $T_e = \frac{4}{3} r_s$ and $k_r = 1$. (PDB Code: 1H2I)

$k_r = 1$, which gives a correct mesh with a good appearance (see Figure 4.13). The computation times are shown in Table 4.1. They only include the mesh generation time, thus it does not take the loading time into account.

PDB code	# atoms	$FDM_{1.9r_s,0}$	$FDM_{\frac{4}{3}r_s,1}$	$CDM_{1.9r_s,0}$	$CDM_{\frac{4}{3}r_s,1}$	LSMS	PyMol	Swiss-PDBV	Chimera	MSMS
1A8R	26400	**0.65**	2.61	1.79	5.69	5.56	10.52	6.38	16.36^b	0.95
1H2I	32318	**0.72**	1.83	2.28	8.25	6.50	11.37	5.25	40.04^b	3.03
1GTP	34740	**0.51**	2.20	2.47	10.14	6.98	13.15	4.75	67.04^b	9.02
1FKA	34977	**0.75**	2.77	3.01	12.41	7.89	26.29	7.36	77.25^b	4.50
1GT7	42700	**0.95**	2.43	2.92	10.90	7.32	16.10	6.50	54.39^b	3.32
1GAV	43335	**0.55**	2.54	3.16	13.67	7.05	28.86	7.71	78.35^b	4.22
1G3I	45528	**0.54**	2.85	3.33	12.68	8.18	19.45	6.21	-	7.67
1PMA	45892	**0.40**	1.97	3.21	13.34	8.23	18.67	6.72	-	12.90
1FJG	51995	**0.71**	2.88	3.39	14.72	8.01	25.18	8.05	-	15.11
1AON	58870	**0.62**	2.64	4.61	17.63	8.83	26.36	8.91	-	10.87
1J0B	60144	**0.69**	3.07	4.12	14.84	6.87	32.66	7.92	-	5.61
1OTZ	68620	**0.67**	2.28	4.36	16.68	8.46	30.14	9.56	-	9.03
1IR2	77088	**0.65**	2.88	4.83	18.51	7.09	29.31	9.55	93.87^b	9.49

Table 4.1 Computation times (in s) for different methods. (^{b} from [73])

In addition, for three molecules, computation times for the method of Cheng et al. [71] are reported from their paper. The computation times comparison is shown in Table 4.2. The computer in the cited papers were more or less equivalent to the

PDB code	# atoms	$\simeq$ # triangles	$\text{FDM}_{\frac{r_s}{2},1}$	$\text{CDM}_{\frac{r_s}{2},1}$	Cheng	MSMS
200D	232	65k	0.55	0.82	1.35^a	**0.33**
1FG1	873	100k	0.85	2.94	2.41^a	**0.65**
3EBZ	1651	200k	1.37	1.95	15.43^a	**0.97**

Table 4.2 Computation times (in s) for different methods. (^{a} from [71])

one used in this study.

It appears in Tables 4.1 and 4.2 that all the molecule surfaces in this data set are computed faster with the FDM than with any other method and for different values of the parameters.

4.3.3 Discussion

In this section, we introduced two similar methods to compute molecular surface meshes (the CDM and the FDM algorithm). The molecular surface is constructed as an isosurface of a filtered electron density map. The FDM, using a Fourier transform and an ideal low-pass filter is faster than other algorithm tested in equivalent conditions. It is slower than the MSMS algorithm for small molecules ($<$ 30000 atoms) but, in the manner of the CDM, it returns a smooth manifold surface, which is not the case with MSMS. It makes possible to compute a precise representation of the surface with a limited number of voxels, so that the computation time and the memory space needed are reduced. Moreover, both methods are parameterizable on the spatial resolution, the refinement of the final mesh, and the size of the solvent molecule. Thus the spatial resolution can be improved for a finer result but with an important computation time increase. Similarly, a smoother result can be obtained with a final refinement with a small influence on the computation time but with less precise results than reducing the spacing. Finally, the solvent molecule size can be chosen without influence on the computation time for the FDM and with a small time increase for the CDM.

The refinement could be improved to be specific to molecular surface. It would enable coarse meshes to be generated rapidly and to be improved by a priori knowledge about local geometry of molecule surfaces, such that the curvature deduced from the closest atom radius.

Geodesic Distance

5

The work of this chapter has been published and presented in [81] and [82].

The Euclidean distance is not always appropriate in applications using 3D objects. Indeed, it does not take into account that the objects are sometimes solid and then impenetrable. The geodesic distance, i.e. distance along the surface, is a base for most surface features, but this distance can also be used, as it is, for processes like remeshing [83, 84], segmentation [85] or animation effects (fire [86], water [87], etc).

In this work, it is used in the algorithm computing the travel depth (Chapter 6), to smooth surface properties such as the curvature (Section 7.1.2), and to define patches in the frame of the binding site detection methods (Chapter 8).

Computing these distances on 3D meshes may be time consuming, especially in applications requiring *"all sources to all targets"* computations, i.e. when the distance between each pair of points has to be computed. Mitchell et al. [88] introduced an exact *"single source to all targets"* (i.e. between one point and all other points) algorithm running in $O(n^2 \log n)$, where n is the number of vertices in the mesh. This algorithm is close to the Dijkstra algorithm [89] finding shortest paths in graphs. In 1998, Kimmel and Sethian [90] presented the fast marching algorithm running in $O(n \log n)$. In this method, the shortest path is propagated from face to face by locally resolving a quadratic equation. The algorithm of Surazhsky et al. based on facet windowing [91] is efficient to compute geodesic paths that cross faces.

In the methods exposed above, *"single source to single target"* or *"single source to all targets"* problems are treated, but often *"all sources to all targets"* problems have to be considered. It is, for instance, the case when geodesic distance is used as a basis for discrete differential-geometry operators [92] such as curvature estimators, integrators or surface field filters. Using *"single source to all targets"* algorithms to compute such distances, without taking into account redundant information, multiplies the complexity and the effective computation time by n.

In this chapter, the front propagation method is introduced and two ideas to approximate the geodesic distance as fast and as correctly as possible are proposed.

The first idea is to use mesh decimation and front propagation. Front propagation is a technique to compute geodesic distances from a source vertex to every vertices

of the mesh. The idea of this algorithm is to decimate the original mesh, compute the geodesic distance between each pair of vertices on the decimated mesh and store it in an array. The distance between two vertices of the original mesh can be retrieved using connections of the decimated mesh as shortcuts.

The second idea is to parameterize the mesh by unfolding it around references vertices and to compute distances in the plane. For source vertices which are not references, the parameterization of the closest reference vertex is used as if the mesh has been unfolded around it. The idea of unfolding the mesh in a plane has already been explored by Kimmel and Sethian [90] and by Surazhsky [91]. These methods differ from the one proposed in this chapter because they use local faces unfolding instead of global edges unfolding. Moreover, they do not use the unfolding in order to approximate the distances from other neighboring source points.

The general principle of both ideas is to keep in memory information which can be used for several different geodesics computations.

The different methods are described in Section 5.1. The advantages and drawbacks of the three methods are discussed in Section 5.2. The chapter is concluded by Section 5.3.

5.1 Methods

5.1.1 Front Propagation

The front propagation algorithm is a method for the estimation of *"single source to all targets"* geodesic distances inspired from the Dijkstra algorithm [89]. Let s be the starting point and t the target point. $d_s(t)$ denotes the geodesic distance between s and t. First, the distance vector is filled with infinite distances except at the starting vertex s:

$$d_s(t) = \begin{cases} \infty & \text{if } t \neq s \\ 0 & \text{if } t = s. \end{cases}$$

Then, the active vertex p is chosen as the non-visited vertex with the smallest distance. The distances to vertices q connected to the active vertex p are updated:

$$d_s(q) = \min\left(d_s(q), d_s(p) + \|\overrightarrow{pq}\|\right) \qquad \forall q \in C(p),$$

where $C(p)$ is the set of vertices connected to p and $\|\overrightarrow{pq}\|$ is the Euclidean distance between p and q. At the first step, $p = s$ because it is the only vertex to have a non-infinite distance. After each step, the vertex p is labeled as *visited* and the non-visited vertex with the smallest distance value, considering the updates, becomes the new p. The algorithm stops when all the vertices are visited.

The drawback of this method, in comparison, for instance, with the one of Kimmel and Sethian [90], is that it does not consider that the geodesics can cross faces. However, it is easy to implement, quite fast and does not introduce much error if the polygons of the mesh are sufficiently small.

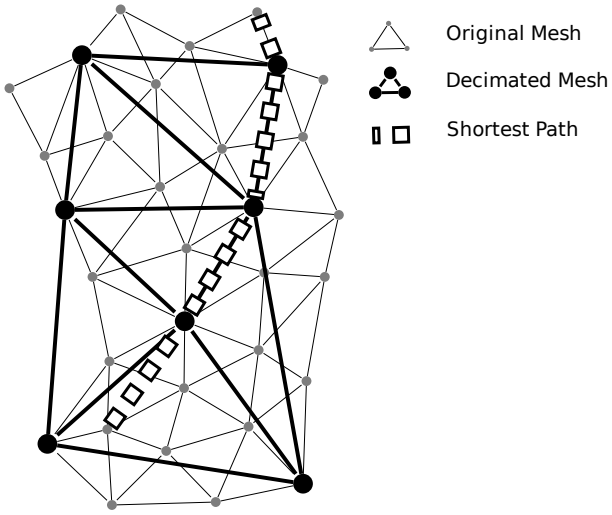

Figure 5.1 An illustration of the MDFP algorithm. The original mesh is depicted with small gray vertices and lines. The mesh is decimated and the resulting mesh is depicted with thick black vertices and lines. The shortest path between two selected vertices of the original is approximated by the big black and white squares path going through the decimated mesh. The parts of the path joining the original mesh and the decimated mesh are Euclidean.

5.1.2 Combining Mesh Decimation and Front Propagation (MDFP)

The algorithm described in this section is an algorithm for fast geodesic distance approximation using mesh decimation and front propagation (MDFP). It is separated into a preparation phase and an estimation phase. During the preparation phase, distances are computed on a decimated mesh using a front propagation and they are stored. These distances are used during the estimation phase, each time a geodesic distance is required for a final application. This technique (illustrated in Figure 5.1) is appropriated when numerous distances have to be computed or even if the same computations must be done several times. The algorithm is a solution for both low-memory and low-time consummation.

5.1.2.1 Mesh Decimation

The decimation or simplification of a mesh is a subsampling of the original mesh through reduction of the number of vertices. The objectives of this operation are, for instance, a reduction of the computation time for the algorithm processed on the mesh or a reduction of the memory space occupied. Mesh simplification can be done by successive vertices or edges decimation. In this kind of method, vertices or edges are ordered using an error metric and are successively deleted until a stop condition is fulfilled. The stop condition is generally a percentage of remaining vertices or topological constraints. The error metric can be, for example, the distance to the average plane or the average edge, depending on local geometry [93]; or the sum of

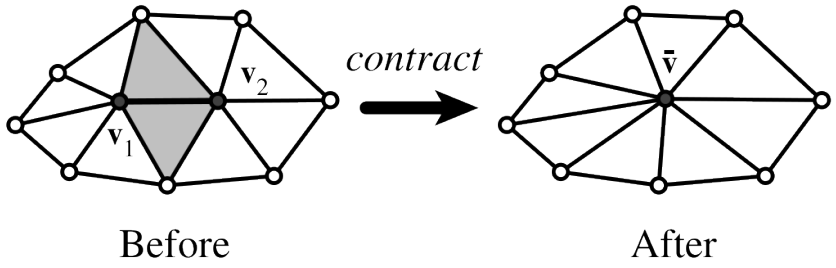

Figure 5.2 Edge deletion in a mesh simplification. The edge V_1V_2 is replaced by the vertex $\bar{V}$ which is connected to all the neighbors of V_1 and V_2. Image courtesy [94].

Figure 5.3 Example of a mesh simplification at different rates. The original mesh, on the left, contains 5,804 faces. The approximations to the right have 994, 532, 248, and 64 faces respectively. The algorithm is designed to keep important features, so that it is possible to recognize a cow even after many strong simplifications. Details such as horns begin to disappear at very low level of details. Image courtesy [94].

the squared distances to the average plane [94]. The elements can also be removed in the order that modifies the global volume or local curvature as little as possible [95]. Each time an element is removed, the local geometry and connectivity are modified, for example, by replacing an edge by its midpoint and connecting this new vertex to neighbors of the extremities of the deleted edge (Figure 5.2). An example of a mesh simplified at different rates is shown in Figure 5.3.

5.1.2.2 Preparation Phase

This phase has to be performed only once and prepares ground information to use during the estimation phase. The preparation phase is illustrated in figure 5.4. First, the *original mesh* ($\mathcal{OM}$) is decimated. The result of the decimation is the *decimated mesh* ($\mathcal{DM}$). Second, a front propagation is performed on each point i of $\mathcal{DM}$. The distances are stored in a vector δ_i, which constitutes the i^{th} column of a reference array D. Finally, for each point of $\mathcal{OM}$, the N Euclidean closest vertices of $\mathcal{DM}$ are stored in a vector $\gamma(s)$ in order to retrieve them faster during the estimation phase. Generally, $\mathcal{DM}$ is triangulated, so N=3 is a good choice because vertices of $\mathcal{OM}$ can be projected on triangular faces of $\mathcal{DM}$. If the polygons of the mesh are not triangles, they can be triangulated in linear time [96].

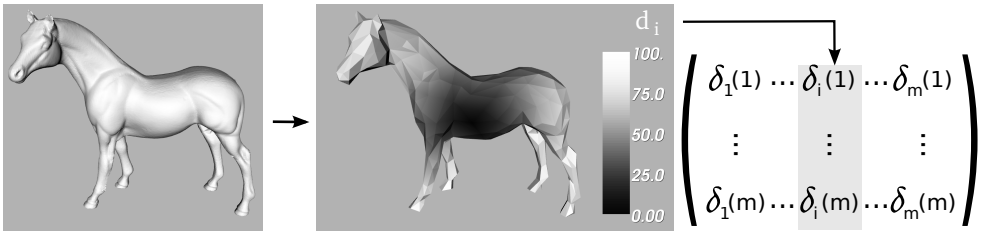

Figure 5.4 The original mesh (left), $\mathcal{OM}$, containing n vertices is decimated, resulting in a decimated mesh (center), $\mathcal{DM}$, containing m vertices, with $m < n$. The geodesic distance between each pair of vertices of $\mathcal{DM}$ is computed using a front propagation algorithm. The distances from a particular vertex i to all the vertices of the mesh is stored in a vector δ_i, which constitutes the i^{th} column of a reference array D (right). Here, the distance is expressed as a percentage of the maximal distance.

The implementation of the algorithm in the *vtkQuadricDecimation Class* of the VTK is used in this work.

5.1.2.3 Estimation Phase

Depending on the final application, the use can be different. Generally, distances are calculated either from a source vertex to a target vertex (itinerary), from a source vertex to equidistant vertices (geodesic circles), or from several source vertices to all vertices (Voronoï partition).

In the estimation phase, the Euclidean distances between end vertices (source and target) and close $\mathcal{DM}$ vertices are computed and added to mutual geodesic distances contained in the array D, constructed during the preparation phase. The expression of $\tilde{d}_s(t)$, the estimated geodesic distance between s and t is the following:

$$\tilde{d}_s(t) = \min_{s' \in \gamma(s), t' \in \gamma(t)} \left(\|\overrightarrow{ss'}\| + \delta_{s'}(t') + \|\overrightarrow{tt'}\| \right).$$

The set γ of $\mathcal{DM}$ vertices close to $\mathcal{OM}$ vertices have a size of N, which leads to $N \times N$ possibilities of finding the minimum.

For vertices in a close neighborhood around the starting vertex, a front propagation on $\mathcal{OM}$ is effectuated to avoid large over-estimations that would be generated by going back and forth to the closest vertex of $\mathcal{DM}$. The close neighborhood is defined as a geodesic disk with a radius equal to the longest edge of the decimated mesh. This estimation phase is illustrated in Figure 5.5.

An example of a geodesic distance map computed with the MDFP is shown in Figure 5.6.

5.1.3 From Parameterization to Geodesic Distance (PGD)

On a mesh, if two vertices are geodesically close, they have a similar geodesic distances map. In particular, if a vertex c lies on the geodesic between two other

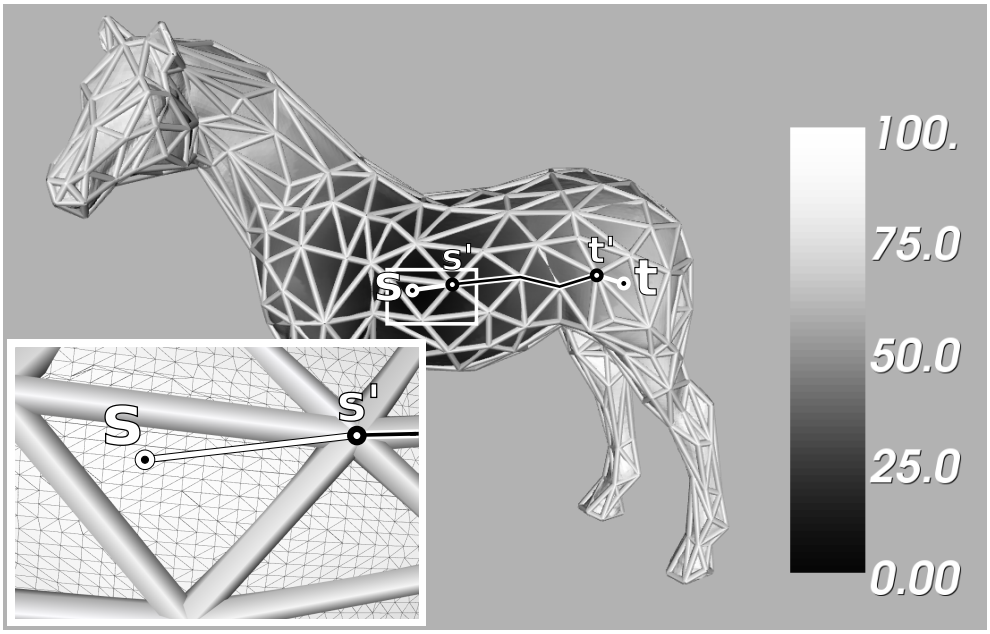

Figure 5.5 The purpose is to approximate the geodesic distance between a source vertex s and a target vertex t. The closest N vertices of $\mathcal{DM}$ from s (resp. t) are candidate to be the approximated source vertex s' (resp. target vertex t'). The approximated distance is the shortest one amongst the different possibilities, which are the $N \times N$ combinations of $\|\overrightarrow{ss'}\| + \delta_{s'}(t') + \|\overrightarrow{tt'}\|$.

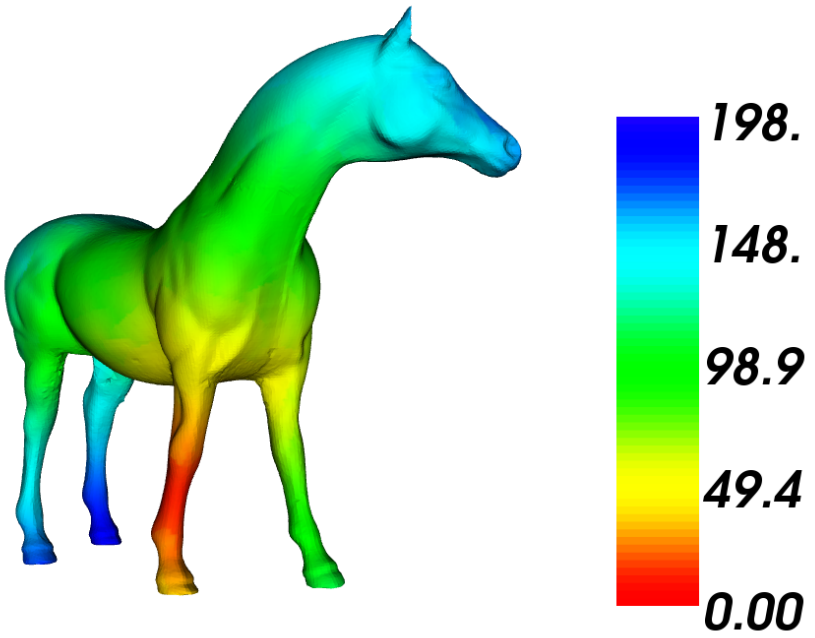

Figure 5.6 Example of a geodesic distance map using the MDFP.

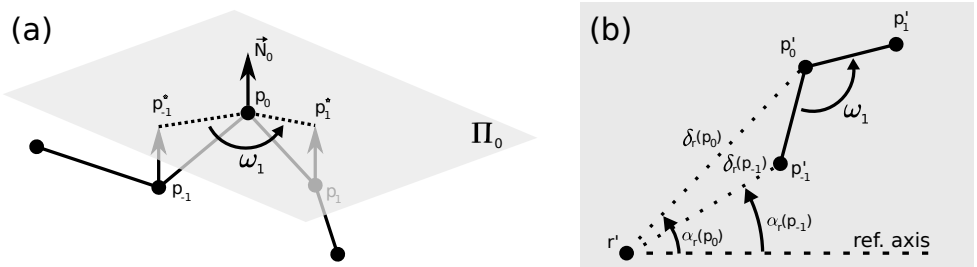

Figure 5.7 (a) Unfolding of edges in the tangent plane Π_0. p_0 is the current active vertex, p_{-1} is its parent in the unfolded tree and p_1 is a non-visited vertex. p_{-1} and p_1 are projected in the tangent plane along the normal vector $\vec{N}_0$. (b) Parameterization plane around r with the points p' of coordinate $(\delta_r(p), \alpha_r(p))$. Given the length of $[p_{-1}p_0]$, the length of $[p_0p_1]$, the projected angle ω_1 and the parameterizations of p_{-1} and p_0, respectively $(\delta_r(p_{-1}), \alpha_r(p_{-1}))$ and $(\delta_r(p_0), \alpha_r(p_0))$, it becomes possible to compute the parameterization of p_1.

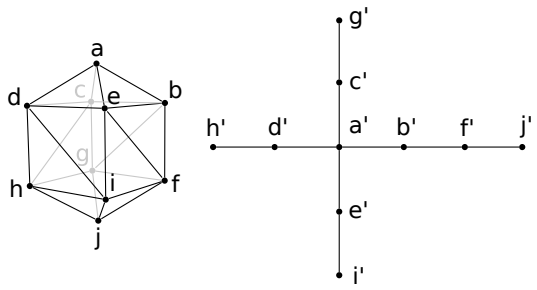

Figure 5.8 A simple polyhedron (left) and its unfolded version (right) around the vertex a.

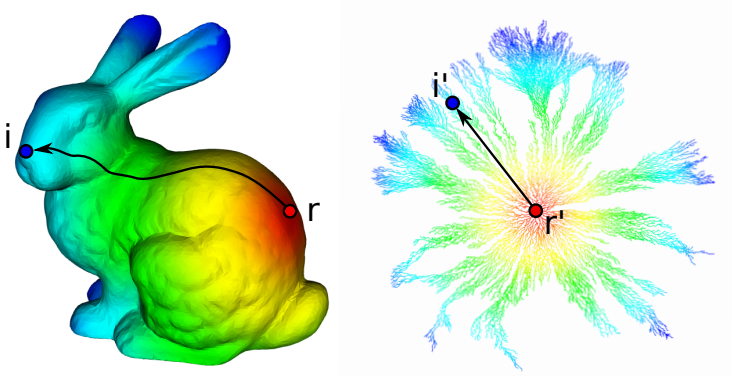

Figure 5.9 A distances map centered in the reference vertex r, and a geodesic between r and a target vertex i. The distances map is first computed on the unfolded version (right) and is mapped on the 3D shape (left). Distances on the unfolded version are computed as straight segment lengths in the plane. 3D model courtesy: Stanford Bunny, Greg Turk and Marc Levoy.

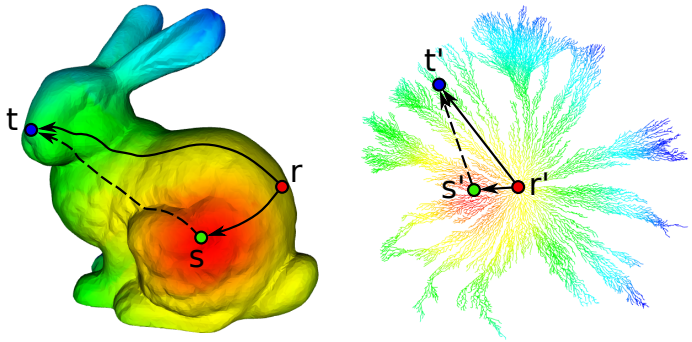

Figure 5.10 Estimation of the distances map centered in a source vertex s and a geodesic estimation between s and a target vertex t. The geodesic distance between s and t is estimated in the unfolding plane around r (right) based on the lengths of $\overrightarrow{r's'}$ and $\overrightarrow{r't'}$ and the angle between both vectors. The estimated distances map is mapped on the 3D shape (left).

vertices, a and b, the geodesic distance from c to b, $d_c(b)$, can directly be deducted from $d_a(b)$ and $d_a(c)$, both computed from the same vertex, a. This reasoning can be extended to vertices lying on different geodesics, bringing the problem into the plane instead of along a line and resolving a triangle. Then, it is possible to reuse information extracted during the computation of geodesics from a particular vertex for the approximation of geodesics from other vertices. In practice, it would be space-consuming to keep all the geodesics in memory. Thus, in PGD (from Parameterization to Geodesic Distance) method, a few reference vertices are selected and a geodesic distances map is computed for each of these vertices. This distances map is computed in a plane containing a parameterization of the shape after an unfolding. The PGD is divided into a preparation phase, during which a parameterization of the mesh is computed around some reference vertices, and a estimation phase, during which the parameterizations are used to estimate distances maps for concrete applications.

5.1.3.1 Preparation Phase

During the preparation phase, some reference vertices are selected and the mesh is parameterized around each of them. A parameterization is a one-to-one mapping from a surface to another one. Many ways to map vertices of a 3D mesh on other surfaces such as spheres or planes are found in the literature [97]. One of the major application of parameterizations is texture mapping [98]. Generally, geodesic distance are needed to produce the parameterization.

For the PGD, a new parameterization is used because it is needed to estimate the geodesic distances and not the opposite. The parameterization is designed to be computationally efficient and to guarantee that close vertices have similar parameterizations. This technique is a mix of shape unfolding [99] and of the front propagation. It maps vertices of the mesh to a plane parameterized with a distance δ to a reference point and an angle α to a arbitrary (but constant) reference axis. The mapping is organized as a tree where vertices are nodes and the branches of the tree represent the shortest edge paths computed by the front propagation. Let r be the selected reference vertex and $\delta_r(i)$, the estimated geodesic distance from r to i. As for the front propagation, at the first step,

$$\delta_r(i) = \begin{cases} \infty & \text{if } r \neq i \\ 0 & \text{if } r = i. \end{cases}$$

Then, the active vertex p_0 is chosen as the non-visited vertex with the smallest distance. At the first step $p_0 = r$. At each step, vertices connected to p_0 are unfolded by rotation of edges to bring them in the tangent plane of p_0 (see Figure 5.7).

For the direct neighbors of r, the angles α are computed in the plane tangent to r, relatively to a reference axis, and the parameters of distance δ are the lengths of the edges linking them to r. Let p_1 be one of these connected vertices and p_{-1} the

parent of p_0 in the unfolded tree. $p^\star$ is the projection of p in the local unfolding plane. The angle $\omega_1 = \widehat{p^\star_{-1} p_0 p^\star_1}$ allows to compute the parameterization of p_1 given the parameterizations of p_{-1} and p_0. At the end of the step, $\delta_r(p_1)$ and $\alpha_r(p_1)$, the polar parameters of p_1 are updated, p_0 is recorded to be the parent of p_1 and is labeled as visited. Then, the new p_0 is selected and so on until all vertices are visited. A simple polyhedron and an unfolded version are depicted in Figure 5.8.

The unfolding by successive projections on the tangent plane does not conserve neither the angles nor all the connectivity, but has some advantages for the PGD. Geodesics have to be parameterized as straight lines, therefore, the distance between two vertices can be computed using a simple operation. Conserving angles does not parameterize geodesics as approximated straight lines, what leads to crossing risks and obstruct the calculation of distances by simple operations.

An example of a distances map computed on a mesh based on its unfolding is depicted in Figure 5.9.

This parameterization is performed for all the references vertices and the parameters are kept in memory. During each parameterization the list of the vertices for which the current reference vertex is geodesically the closest is updated. A new reference vertex is selected as the vertex of the mesh being the geodesically furthest one from all other reference vertices.

5.1.3.2 Estimation Phase

During the estimation phase, parameterizations computed in the preparation phase are used to estimate geodesic distances from any vertex of a mesh. Let s and t be respectively the source and the target vertex in a practical application. Let r be the geodesically closest reference vertex from s. The parameters of all the vertices based on the unfolding around r are stored in the vectors δ_r (distance) and α_r (angle). The approximation of the geodesic distance between s and t, $\tilde{d}_s(t)$, is computed using the law of cosines:

$$\Theta^r_s(t) = \alpha_r(t) - \alpha_r(s),$$

$$\tilde{d}_s(t) = \sqrt{\delta_r^2(s) + \delta_r^2(t) - 2\delta_r^2(s)\delta_r^2(t)\cos\left(\Theta^r_s(t)\right)}.$$

An approximation example is shown in Figure 5.10. This application is based on the parameterization around r, depicted in the Figure 5.9.

5.2 Results and Discussion

Results concerning the estimation errors, the time complexity, and the computation time for the three methods are presented in this section.

In order to measure the estimation errors and the computation time, tests were performed on four different meshes: *head* (11703 vertices), *bunny* (14007 vertices), *horse* (48485 vertices) and *molecule* (435390 vertices).

5.2.1 Estimation Errors

In this section, possible errors due to the decimation or the use of the parameterization are exposed.

A mean absolute error score was computed during the tests. For the MDFP, the reference was the front propagation distance. For the PGD, the reference distance was the one considering the parameterization around the source point, and the estimation distance was the one considering the parameterization around another point.

The mean error ϵ for an "all sources to all targets" task is

$$\epsilon = \frac{1}{\sigma^2} \sum_{i=1}^{n} \sigma_i \left(\sum_{j=1}^{n} \frac{|d_i(j) - \tilde{d}_i(j)|}{d_{i,max}} \sigma_j \right), \tag{5.1}$$

where $d_i(j)$ is the reference geodesic distance between i and j, $\tilde{d}_i(j)$ is the estimated geodesic distance, $d_{i,max}$ is the maximal value of $\{d_i(j) | j \in [1, n]\}$, σ is the total area of the mesh and σ_j is a proportion factor depending on faces area around the vertex j.

5.2.1.1 MDFP

If the front propagation distance is considered to be the reference, there are two major sources of error. First, an over-estimation because the shortest theoretical path could cross $\mathcal{DM}$ edges, whereas the estimation will follow these edges. Second, an under-estimation because a succession of several segments approximating a curve in $\mathcal{OM}$ could be replaced by a single straight segment after the decimation. The decimation algorithm used for this algorithm removes vertices ordered by a local cost function until the desired threshold of decimation is attained [100]. It conserves the topology of $\mathcal{OM}$ so that there only are local and propagated errors, but no topological errors. Finally, in a close neighborhood, distances are computed by front propagation which locally gives an exact solution.

The error of the MDFP is not theoretically bounded because, with the used decimation technique [94], the angles of the triangles of the decimated mesh are not constrained. The distance may thus be largely overestimated if the optimal path would have to cross a long edge.

For each of the four meshes, an approximation of the error was calculated on 100 source points (instead of all points). The mean proportional global error (equation 5.1) as a function of the proportional distance ($\frac{d_i(j)}{d_{i,max}}$) is depicted in Figure 5.11 for three levels of decimation: keeping 100, 500 and 1000 points.

The mean error was calculated in reference with the front propagation algorithm. It worth noting that any geodesic distance algorithm can be used in this method. Thus the error method just takes into account the difference induced by the decimation.

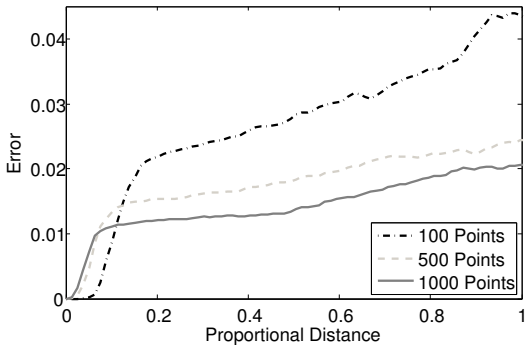

Figure 5.11 Absolute error graph of the estimation phase (MDFP) as a function of the proportional distance to the source for three levels of decimation: keeping 100, 500 and 1000 points.

5.2.1.2 PGD

For the geodesic distance based on the parameterization, some minor errors may appear. First, the parameterization makes the geodesics "follow" edges, whereas sometimes, going through a face would be shorter. This error becomes smaller if the size of the faces becomes smaller. Second, some crossings may appear in the unfolding, leading to discontinuities in the distances map. For relatively regular surfaces, however, these crossings are generally far from the source, where less precision is needed.

Bigger errors appear in the estimation phase. First, if the number of reference vertices is too low, two close vertices may be considered to be far from each other if they are on the border of the parameterization (overestimation). The error rate is unbounded because two close vertices may belong to different reference vertex zones, so an infinitely small distance may be estimated as a larger positive value. Second, if the surface is not smooth and if the geodesics from reference vertices follow valleys and crests, then, the geodesics from another source vertex, could cross the ripples without taking them into account (underestimation). The underestimation rate is unbounded because the number of ripples may be very high even close to the reference vertex.

For each of the four meshes, an approximation of the error was calculated on 100 source points (instead of all points) and for a unique parameterization. The mean proportional global error (equation 5.1) as a function of the proportional distance ($\frac{d_i(j)}{d_{i,max}}$) is depicted in Figure 5.12 for three categories of source points classified by their relative distance to the unique reference point:

- Close points, for which $\frac{d_r(s)}{d_{r,max}} < \frac{1}{20}$,
- Mid-close points, for which $\frac{d_r(s)}{d_{r,max}} < \frac{1}{4}$,
- All points, for which $\frac{d_r(s)}{d_{r,max}} \leq 1$,

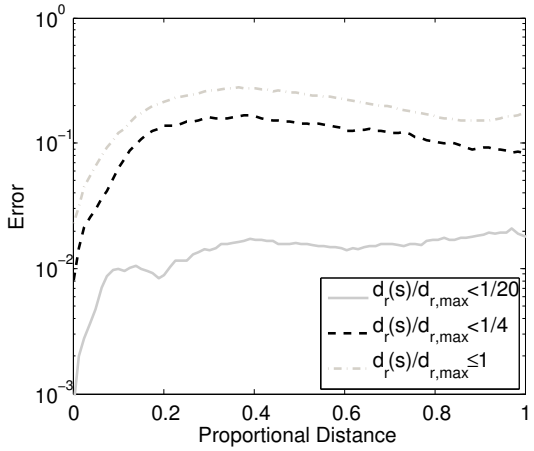

Figure 5.12 Absolute error graph of the estimation phase (PGD) as a function of the proportional distance to the source for three categories of source points: close points ($\frac{d_r(s)}{d_{r,max}} < \frac{1}{20}$), mid-close points ($\frac{d_r(s)}{d_{r,max}} < \frac{1}{4}$), all points ($\frac{d_r(s)}{d_{r,max}} \leq 1$).

where r is the reference point and s is the source point.

5.2.2 Complexity Analysis

5.2.2.1 Front Propagation

The time complexity of the front propagation algorithm is $O(n \log n)$.

5.2.2.2 MFPD

The time complexity of the MDFP, executed p times, from a single source vertex to all the vertices is $O(n \log n) + O(m^2 \log m) + O(p\frac{n}{m} \log \frac{n}{m})$, where n is the number of vertices in $\mathcal{OM}$, m is the number of vertices in $\mathcal{DM}$. The first term, $O(n \log n)$, comes from the quadric-based decimation algorithm [101]. The second term, $O(m^2 \log m)$, comes from the front propagation algorithm for the computation of the geodesics on $\mathcal{DM}$ [102]. The third term, $O(p\frac{n}{m} \log \frac{n}{m})$, comes from the front propagation in a close neighborhood around a source vertex containing about n/m vertices. Finding the shortest path from p to n vertices testing each time the NxN possibilities has a complexity of $O(pn) \in O(p\frac{n}{m} \log \frac{n}{m})$. Since distances between vertices of $\mathcal{DM}$ are stored in a array of size m^2, in practice, m is set to a constant, in order to avoid memory lacks. In this case the complexity of the MDFP is $O(pn \log n)$, the same than for the front propagation algorithm alone. In practice, however, coefficients are smaller for the MDFP because only local front propagations are needed. This is true only if the computed distances are long, if p is large or if n is large. Otherwise the third term, $O(p\frac{n}{m} \log \frac{n}{m})$, becomes comparable to both other terms, $O(n \log n)$ and $O(m^2 \log m)$, and the front propagation alone becomes less time consuming.

5.2.2.3 PGD

The time complexity for a mesh containing n vertices for a *"single source to single target"* estimation is $O(mn \log n) + O(1)$, where m is the number of reference vertices. The time complexity for the parameterization around a vertex is $O(n \log n)$ and this operation is repeated m times during the preparation phase, hence the fixed cost of the algorithm is $O(mn \log n)$. The variable cost depends directly on the number of sources and targets. It is the cost of the estimation phase which only contains simple operations. The *"all sources to all targets"* algorithm has a complexity of $O(mn \log n) + O(n^2)$ instead of $O(n^2 \log n)$ without estimation (in that case, $m = n$, and no estimation phase is needed).

5.2.3 Computation Time

To illustrate the time savings of the approximations, some comparisons with the front propagation algorithm are presented here. The computation times needed to perform an "all sources to all targets" task for the four meshes and the three methods are shown on Figure 5.13. The number of points in the decimated mesh for the MFPD was set to 500 and the number of reference points for the PGD was set to 100.

These parameters lead to mean absolute errors in the order of 10^{-2} (see Figures 5.11 and 5.12). For the PGD, in the particular case of a sphere, 49 uniformly distributed points should be sufficient to satisfy the condition that, for each source point s, there exists at least one reference point $\{r | \frac{d_r(s)}{d_{r,max}} < \frac{1}{20}\}$ [103]. Setting the number of reference points to 100 with an ordinary mesh should ensure to respect this condition.

The computation times for the preparation phases only are also shown in Figure 5.13. Since the computation times may be huge (up to three years for the molecule mesh using the front propagation), some of them were estimated using a linear extrapolation based and several "one source to all targets" tasks. By comparison, the Fast Marching algorithm [90] applied to the horse mesh would take approximately $1.67\,10^6 s.$; the exact method of Surazhsky et al. [91], $8.94\,10^6 s.$; and their approximate method, $1.18\,10^6 s$. These results are derived from those presented in [91].

5.3 Conclusion

In this chapter, three methods for geodesic distance computation are presented. The first one, the front propagation, is exact if we consider that the geodesic path cannot cross the faces and has to follow the edges. The second one, the MDFP is an approximation scheme based on a mesh decimation, and the third one, the PGD is another approximation scheme based on a parameterization.

The aim of the approximation methods is to make possible the estimation of the geodesic distances between a large number of points. This computation time reduction involves a degradation of the estimation quality. Slower methods, like those

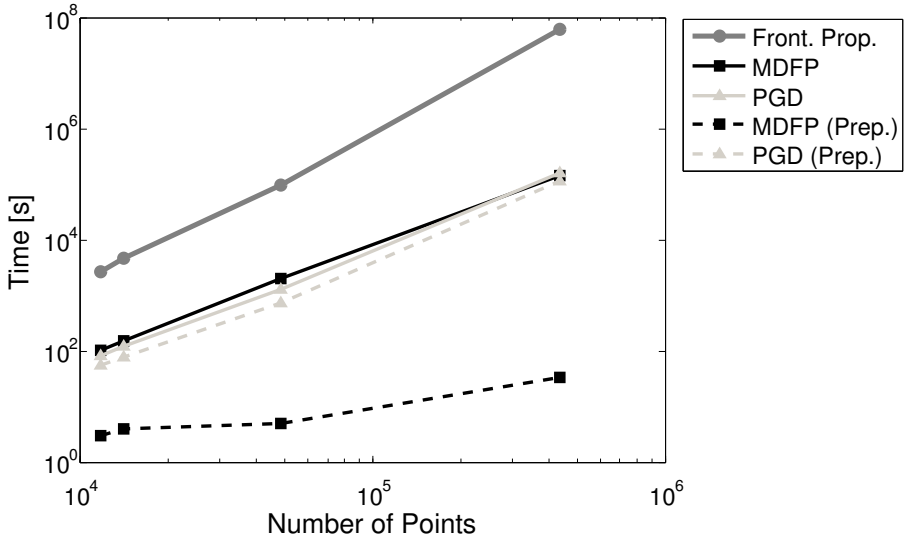

Figure 5.13 Computation time graph for geodesic distance estimation for an "all sources to all targets" task and with the three methods. The tests were performed on four meshes with a different number of points. Dashed lines represent the computation times of the preparation phases of the MFPD and the PGD. The axes are in a logarithmic scale.

proposed in [90] or [91], may be used to compute an exact or low-error geodesic distance estimation for which the geodesic paths may cross the mesh faces. The front propagation algorithm is a bit faster but is restricted to edges paths. It should consequently be used for meshes with relatively small faces. The MDFP is appropriate for relatively long distance computations because the error is not dramatically increasing in this condition. Without the local exact approximation, the MDFP would show important errors for close neighborhood distance evaluation. That is why it has been replaced by a front propagation in this case. A major interest of the MDFP is the short execution time of the preparation phase, making the method advantageous even for a few number of distances computations. It is not the case of the PGD, for which the preparation phase is longer. This method should be used when a large number of distances has to be computed. The estimation is very accurate for short distances but it is also suitable for longer distances.

The error of the MDFP is not bounded because the constraints of the decimation schemes are not made to face this kind of issues. The algorithm could be optimized by using a decimation algorithm optimizing the regularity of the triangles.

With the PGD, the error is not bounded. But in practice, with relatively smooth meshes, the approximation works well. The error can be theoretically bounded by using other techniques to rely application vertices to reference vertices. For instance, by combining several references or by considering an angle difference instead of a distance to choose them.

Travel Depth

6

The work of this chapter, made in collaboration with Patrice Rondão Alface has been presented and published in [104] and [105].

Depth is one of the most frequently used molecular descriptors in the literature. In this work, for example, it constitutes one of the properties analyzed in order to detect potential binding sites (Chapters 7 and 8). However, the definition of depth is generally qualitative. This is the reason why Coleman and Sharp (referred as CS in the following) introduced the notion of *travel depth* in 2006 [7].

The travel depth of a molecular surface point is defined as the length of the shortest path accessible for a solvent molecule between this point and the protein convex hull. The convex hull is the smallest convex 3D polyhedron that contains the entire set of the surface points. Travel depth is representative of the distance that a small molecule has to travel to approach a point on the surface of the molecule of interest. It is more global than a simple curvature computation [106,107] while keeping a good surface resolution. Moreover, it only requires surface information to be computed because the surface represents the frontier between accessible and non-accessible points.

Existing algorithms describing surface cavities [7, 108] are volume-based algorithms (VBA) that use the Dijkstra's shortest path algorithm on a regularly sampled bounding volume of the surface of interest. Since travel depth is defined on the molecular surface, these volume-based approaches are characterized by a large computational complexity due to the processing of too many unnecessary samples lying inside or outside the molecule. Subsequently, high resolution estimations for large macromolecules are limited by the large processing time and memory requirements of these methods. The use of a faster algorithm with reduced memory needs would make it possible to screen large databases and sort molecules on the basis of the cavities shapes.

In this chapter, we propose a surface-based algorithm in order to compute shortest paths between the points of the molecular surface and its convex hull (both represented as 3d triangular meshes). Unlike volume-based techniques, generally using Dijkstra's algorithm on tetrahedral or octree mesh models such as in [7], our tech-

nique only processes points defined on the surface. Figure 6.1 shows an intuitive comparison between the volume-based and the surface-based approaches. In the volume-based approach, each point on a three dimensional grid gets a depth value, whereas in the surface-based approach, only the mesh points and the convex hull are considered.

This surface-based algorithm (SBA) proposed here is somewhat similar in spirit with the A* search algorithm [109] well-known in the artificial intelligence domain. Indeed, we use an algorithm to restrict computations to surface and convex hull point samples instead of volume samples of the space between them (inter-volume). This algorithm is based on the intuitive idea that the shortest path between the molecular surface and its convex hull is composed of several shortest sub-paths which are either straight lines in the inter-volume or curved paths on the molecular surface (a.k.a. geodesic paths).

Straight paths in the volume between two surfaces can be efficiently estimated by an octree search with $O(\log s)$ complexity where s is the number of recursive subdivisions of the volume surrounding the two surfaces.

The main contribution of this work is an efficient combination of both techniques. The reference surface elements for which travel depth is already known are iteratively updated. Distances from reference elements to new points are computed as the shortest distance given by either a straight line (if this line does not cut the surface) or a geodesic path.

6.1 Method

In this section, a heuristic algorithm approximating the travel depth of a surface (or triangle mesh) is presented. The travel depth of a surface point is the length of the shortest path that a small probe should cover in order to reach this surface point from the convex hull of the surface.

The master idea of the algorithm lies in the classification of a surface between "visible" and "hidden" areas. "Visible" means that, for a given point of the surface, there exists a facet of the convex hull such that the segment between this point and its orthogonal projection on the facet does not intersect the surface other points. "Hidden" areas are simply defined as the surface areas that are not visible. In the case of hidden areas, the shortest path for a solvent molecule is constituted of sub-paths that either run along the surface together with sub-paths or are straight lines in the inter-volume between the surface and the convex hull. It will then follow the trajectory of a thread which would be tensed between a point of the surface and a point of the convex hull (see Figure 6.2).

The SBA is described in Section 6.1.1. An analysis of the strengths and limitations can be found in Section 6.1.2. The pseudo-code, given in Section 6.1.3, serves as a basis for the complexity analysis presented in Section 6.1.4.

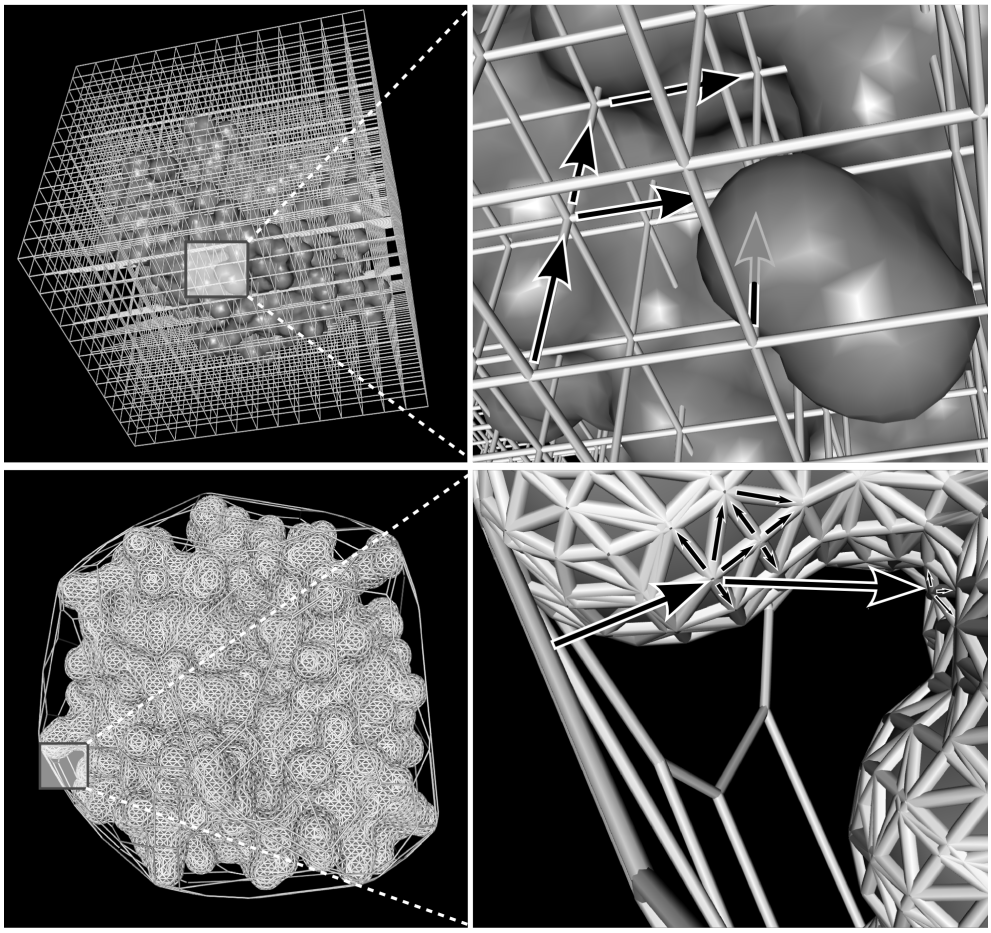

Figure 6.1 An intuitive view of a volume-based algorithm (VBA) for travel depth calculation on a protein surface (PDB code: 1ACB_A) and a zoom are represented, respectively on top left and right, . Each point on a 3D grid surrounding the protein constitutes a node of a large graph. A shortest path algorithm (e.g. Dijkstra) is performed on this graph. On this figure, grid size is set to 4 Å, but in practice computations are performed with a resolution of 1 Å. On the bottom left and right, an intuitive view of a surface-based algorithm (SBA) together with a zoom are represented, respectively. A shortest path algorithm is performed for points of the surface mesh representing the SES. This algorithm alternates local straight line and local geodesic shortest paths research. [105]

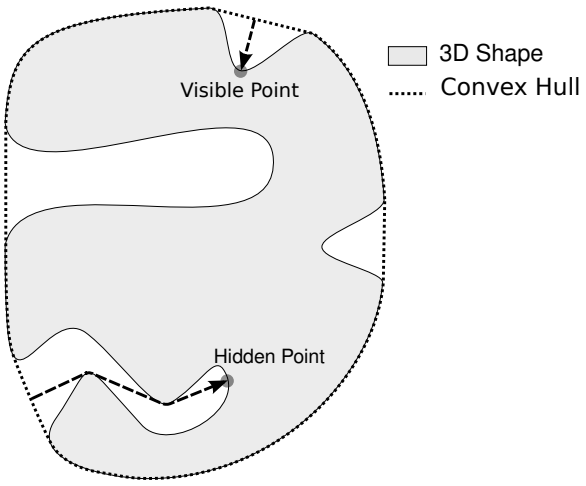

Figure 6.2 Two kinds of points are shown on a cutaway view of a 3D shape (gray zone outlined by a continuous black line). The visible point is the one for which a segment can be drawn all the way from the convex hull (black dotted line). The hidden point is the one that cannot be reached at shortest in a straight line from the convex hull. [105]

6.1.1 Algorithm

An initialization step is performed in order to generate a mesh representation of the convex hull (Step 0). The convex hull is the reference set for the first step. Then, the travel depth is computed for the visible points of the mesh. This is done in two steps: very close points for which no other computation must be done (Step 1A), and further points for which an intersection test is performed (Step 1B). Step 2 is iterated until the termination criterion is satisfied. In this step, the depth is calculated from points which were already visited, following a geodesic path (Step 2A) or a straight line (Step 2B) to a close neighborhood. At each iteration of Step 2, the newly calculated points act as a new set of reference points that replaces the convex hull. Step 3 is a geodesic propagation of the computed depths over the whole surface mesh. This algorithm is illustrated in Figure 6.3.

Step 0: Initialization During this step, the convex hull mesh of the input surface mesh is generated. It is done using a VTK algorithm, implemented in *vtkHull.h*, which iteratively cuts the surrounding cube with planes. As an initialization, all the points of the mesh receive an integer certitude label $c_i = -1$ and an infinite depth value. H, *the set of hidden points*, is created and is empty. R, *the reference set*, varies at each step and comprises high certitude points. At this step, R is filled with the points and the faces of the convex hull.

Step 1: Visible Points

Step 1A: Short Distances The Euclidean distance between each point of the mesh and the closest corresponding hull point is computed. If this distance is shorter than a parameter ε, this value is selected for the depth of the processed point. In the case

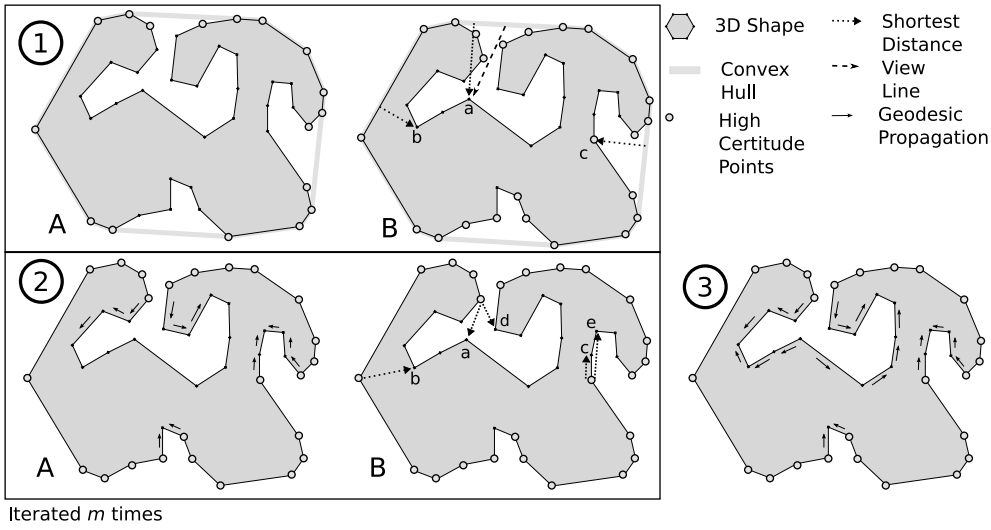

Figure 6.3 An example of the surface-base algorithm (SBA) for travel depth on a cutaway view of a 3D shape mesh (big gray zone delimited by the thin black line and small dots for the points). **Step 1A**: depth is definitely set (small gray circles) for points which are close enough from the convex hull (light gray thick line). These points have a high certitude. **Step 1B**: depth is set if the closest convex hull part is visible. These points have a high certitude. Dotted arrows depict the shortest segments to the hull for example points a, b and c. If this arrow does not go through the interior of the shape, the point is considered as visible, like c. Points a and b are hidden. It is possible to reach a following the view line depicted by the dashed arrow, but it would be longer than with several shorter segments. **Step 2A**: the depth is locally propagated following geodesics. **Step 2B**: the nearest point in the set of high certitude points is found for each point. If it is visible and if the new depth is smaller, the depth of the point is updated. Thus a is updated (it had an infinite depth); b is not updated (not visible); c is not updated (same depth as the one get at the step 2A); d is not updated (depth greater as the one get at the step 2A); e is updated (depth is smaller than the current one for this point). If the point get already a depth and is not updated, the certitude of this point is increased. New points are included in the set of high certitude points. Step 2 is iterated m times if there are still points to process. **Step 3**: propagation updates estimations if necessary. This final step is useful to cover points which are not covered by Step 2 and to correct discontinuities. [105]

of protein meshes, this operation is allowed with $\varepsilon = 1.5$ Å because the smallest atom usually present in proteins is the Oxygen (Hydrogen is considered to have an approximatively null size), which has a vdW radius of 1.52Å. Then, if a surface point is at a distance from the convex hull smaller than 1.5 Å, it cannot be hidden by another atom, otherwise, this other atom would push back the limit of the hull. Conversely, a point which is at a distance superior to 1.5 Å from the convex hull could be behind the atom equator. The certitude label c of these points is set to c_{max} (empirically set to 7), and their values of depth are optimal and will not be re-estimated in the following. In order to extend the method to any mesh, ε has to be set to 0, allocating a depth of zero to points belonging both to the surface and its convex hull.

Step 1B: Other Visible Points For the points which are not concerned by the step 1A, the number of intersections between the surface and the segment connecting the point to the closest hull point is computed. If there is no intersection, then the point is visible, and the depth is given by the length of this segment. The certitude label is then set to c_{max} for this point. As for other visible points, the depth values are optimal and will not be re-computed in following steps. If there exists an intersection, the point is considered to be hidden, and it is added to H, the list of hidden points. Depth of points contained in this list can be updated until the end of the algorithm if a shorter path reaching them is found. R, the reference set, is reset and is refilled with the high certitude points $\{i|c_i = c_{max}\}$.

Step 2: Local Iterated Propagations

Step 2A: Local Geodesic Propagation H is browsed until a depth is given to all the points belonging to close neighborhoods of the mesh points $\{i|c_i > -1\}$ by front propagation [84]. The close neighborhood of a point i contains all the the points which can be reached from the point i going through c_i (or less) other points. In other words, if a depth value of a point has a high certitude, its geodesic propagation is much wider. Each time a point is visited, it gets the smallest propagated depth from its neighbors in this way:

$$\delta_j = \min_{\{k \in N_j | c_k > 0\}} \left(\delta_k + \| \overrightarrow{jk} \| \right),$$

$$c_j = c_k - 1,$$

where δ_j is the current depth of the point j, N_j is the list of the points directly connected to the point j, and $\| \overrightarrow{jk} \|$ is the length of the edge between j and k. In other words, the propagated depth of a point j is the depth of a neighbor, k, added up to the distance between both points. The certitude of a point is inversely proportional to the distance to its reference point.

Step 2B: Local Straight Propagation For each point j, the closest point $k \in R$ is found. If the segment $[jk]$ does not intersect the surface and $\delta_j > \left(\delta_k + \| \overrightarrow{jk} \|\right)$, then

$$\delta_j = \left(\delta_k + \| \overrightarrow{jk} \|\right),$$

$$c_j = c_k - 1.$$

If the point is not updated, its certitude is incremented by 1. R is reset. It is refilled with points for which the certitude reaches c_{max} at this iteration.
Step 2 is repeated m times or less if R is empty. $m = 5$ is the empirical value that was chosen to minimize the execution time.

Step 3: Final Propagation A local geodesic front propagation as in Step 2B is performed, but in this case from all the points of the mesh, after certitude labels were reinitialized. This step has two purposes. First, some points of H could keep an infinite value during the previous steps because Step 2 is only repeated m times. The overestimation resulting from this process is negligible in comparison with the time gained. In practice, it does not concern many points because after $m = 5$ iterations of the Step 2, most points get already a depth value close to the theoretical value. Second, since c_{max} is arbitrarily set, it is possible that in some regions, points get a certitude value of c_{max} while some neighbors have a much smaller depth value. It may result in some discontinuities which would mostly disappear during this step.

An example of travel depth estimation algorithm execution is shown in Figure 6.4.

6.1.2 Strengths and Limitations

The main strength of this technique is that it is fast, accurate, and provides a smooth result. It is fast because it just needs to locate fixed points in the space which can be done easily with an octree. It is accurate and smooth because each point gets its own value instead of the same mean value for all the points lying in a cube as in VBA methods.

The main limitation is that, during Step 2, the depth can only be inherited from the closest point belonging to the set R. Problems occur if a pocket is deep and wide, so that the bottom of the pocket is closer from the non-accessible part of the convex hull than from visible parts. This kind of pockets will be accessible to the reference set R only after some iterations. Thus the path in order to reach the bottom would be to first follow the side of the pocket (Figure 6.5). In this case, the depth will be overestimated. The overestimation is proportional to the width of the cavity in this case, therefore this error is currently not bounded.

Nevertheless, this kind of situation is rare, and we have to keep in mind the motivation of this work: reduce the execution time necessary to find deep cavities, which could potentially be binding pockets. This objective is not affected by such limitations. Some small other errors may also occur, due to, for instance, the approximation of the convex hull.

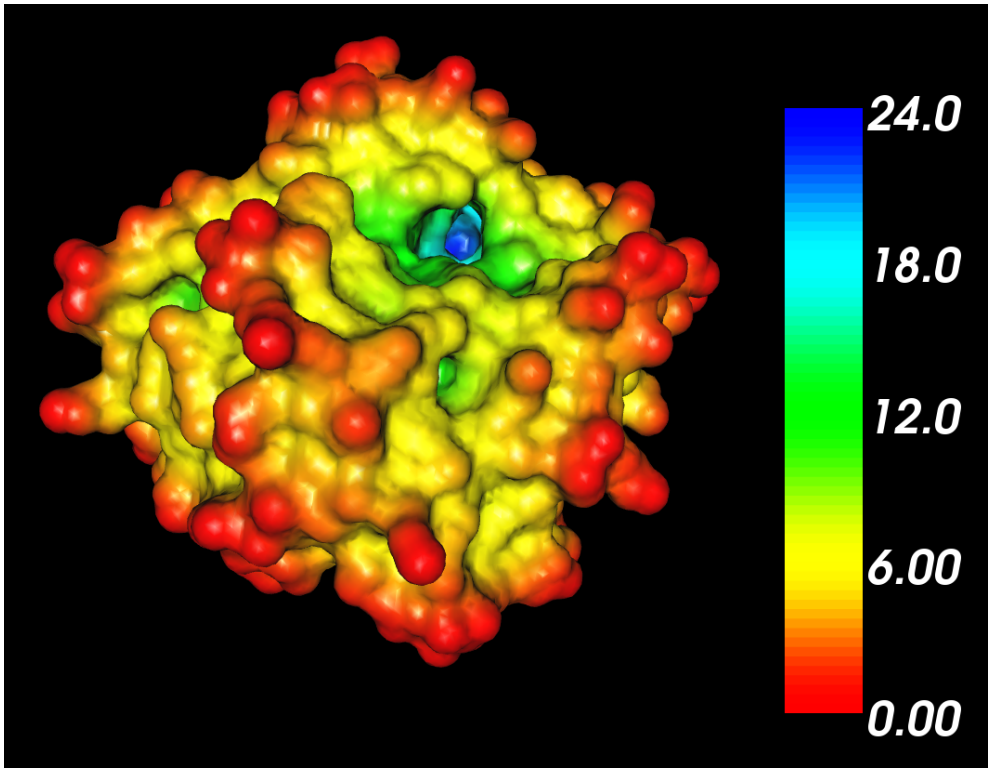

Figure 6.4 Example of a protein (PDB code: 1ACB_E) SES with a representation of the depth. Points colored in red are in contact with the convex hull an points colored in blue are the furthest from the hull (here 24Å far).

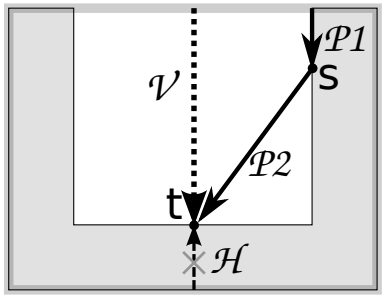

Figure 6.5 An illustration of the main limitation which leads to an overestimation occurring in deep and wide pockets. The depth of the target point t is theoretically the length of the path $\mathcal{V}$. However, the closest part of the reference set R at the first iteration is hidden and related to t by path $\mathcal{H}$. The depth of step point s is well estimated during the first iteration with path $\mathcal{P}1$. s is included in the reference set R during the second iteration, as the convex hull is removed from R. Thus at the second iteration, s is the closest reference point from t. The depth of t is estimated as the depth of s added to the length of path $\mathcal{P}2$. [105]

6.1.3 Pseudo-Code

Input: A 3D mesh (*mesh*) containing n points
and a mesh of its convex hull (*convex_hull_mesh*).
Output: A travel depth value for each point of the 3D mesh.

STEP 0
//Reference set initiated as the convex hull.
$R = convex\_hull\_mesh;$
//New reference set in construction at each iteration of step 2.
$R_{new} = \emptyset;$
//Set of hidden points.
$H = \emptyset;$
//Temporary set of hidden points for step 2A.
$H_t = \emptyset;$
//Certitude index of all points initiated to -1.
$\forall(i \in mesh)\ c(i) = -1;$
//Temporary certitude index for step 2A.
$c_t = \emptyset;$
//Depth of all points initiated to ∞
$\forall(i \in mesh)\ \delta_i = \infty;$

STEP 1
for$(i \in mesh)$
  $k = findClosest(i, R);$
  STEP 1A
  if$(\| \overrightarrow{ik} \| < \varepsilon)$ **then** $\delta_i = \| \overrightarrow{ik} \|;$
    $c(i) = c_{max};$
    $R_{new}.add(i);$
  STEP 1B
  else if$([ik] \cap mesh = \emptyset)$ **then**
    $\delta_i = \| \overrightarrow{ik} \|;$
    $c(i) = c_{max};$
    $R_{new}.add(i);$
  else $H.add(i);$
  endif
endfor
$R = R_{new};$
$R_{new} = \emptyset;$

STEP 2
$num\_It = 0;$

while$(R \neq \emptyset \quad \& \quad num\_It \leq m)$
  STEP 2A
  $num\_It++;$
  $H_t = H;$
  $c_t = c;$
  while$(H_t \neq \emptyset)$
    for$(j \in H_t)$
      $\delta_j = min_{(k \in N(j)) | c(k) > 0)}(\delta_k + \| \overrightarrow{jk} \|);$
        if$(k = j)$ **then** $c_t(j)++;$
        else $c(j) = c(k) - 1;$
        endif
    endfor
    if$(c_t(j) = c_{max})$ **then** $H_t.remove(j);$ **endif**
  endwhile

  STEP 2B
  for$(j \in H)$
    $k = findClosest(j, R);$
    if$([ik] \cap mesh = \emptyset)$ **then**
      $\delta_j = \| \overrightarrow{jk} \|;$
      $c(j) = c(k) - 1;$
    else $c(j)++;$
    endif
    if$(c(j) = c_{max})$ **then** $R_{new}.add(j);$ **endif**
  endfor
  $R = R_{new};$
  $R_{new} = \emptyset;$
endwhile

STEP 3
$\forall (j \in H) \quad c(j) = -1;$
while$(H \neq \emptyset)$
  for$(j \in H)$
    $\delta_j = min_{(k \in N(j)) | c(k) > 0)}(\delta_k + \| \overrightarrow{jk} \|);$
      if$(k_{min} = j)$ **then** $c(j)++;$
      else $c(j) = c(k_{min}) - 1;$
      endif
  endfor
  if$(c(j) = c_{max})$ **then** $H.remove(j);$ **endif**
endwhile

6.1.4 Time Complexity

The worst case time complexity of the SBA is

$$O(STEP0 + STEP1 + m(STEP2) + STEP3)$$
$$= O(p + p \log f_{CH} + m(c_{max} p_h + p_h \log (p - p_h)) + c_{max} p_h),$$

where p is the number of points in the mesh, f_{CH} is the number of faces in the convex hull, m is the number of iterations, p_h is the number of hidden points, and $(p - p_h)$ is the number of visible points. As m and c_{max} are small numbers that can be considered as constant, and p_h is proportional to p, the complexity can be written as:

$$O(p + p \log f_{CH} + p \log p).$$

As in practice, $p >> 1$ and $f_{CH} << p$, $O(p + p \log f_{CH} + p \log p) = O(p \log p)$. For infinitely large meshes, the proportion between the number of points (p), the number of edges (e) and the number faces (f) is constant [110] and is p:e:f = 1:3:2. These proportions are already very well approximated for genus-0 triangle meshes of any size like those used for representing SES in this work. In this case, for the big O time complexity notation, p, e and f are equivalent, therefore, the time complexity of the algorithm of the CS's algorithm can be written as:

$$O(p \log p + f_{CH} d^3 + (p + g) d^2 + (g^3 + p) \log g^3),$$

where g is the number of grid cubes in any dimension. The main difference between both complexities is the presence of d terms due to the volume-based approach that makes the computation time growing when the required precision is higher. In practice and with default parameters, g^2 and p have the same order of magnitude. Thus the g terms are not negligible.

6.2 Results

In Section 6.2.1, empirical results are presented. They show that the limitations presented in Section 6.1.2 concern only few points. They also show that mean time complexity is smaller than the worst case complexity (Section 6.1.4). In Section 6.2.2, results on accuracy (on chosen examples), smoothness and execution time are compared between the SBA and the CS's VBA.

6.2.1 Empirical Results

The error and the mean time complexity should be much more acceptable than in the worst case (Sections 6.1.2 and 6.1.4) regarding to experimental results. The routine has been executed on a small representative data set to note the proportion of points belonging to each category. The proportion of visible points, that is, points with a depth affected during the first step and constituting R at the first iteration was 0.86 ± 0.06. This observation means that the proportion of hidden points, the

dual set of the visible points, was 0.14 ± 0.06. In practice, c_{max} was set to 7 which empirically provided the best results. And, finally, the number of cells of the convex hull is neither depending on the number of atoms nor on the number of points of the mesh. It varies only on a parameter of the function which was kept constant, so that n_{CH} never exceeded 300, whereas n was varying from 2500 to 120000 in the data set used for the tests.

Consequently, the error on about 86% of the surface is really small (only the error due to the convex hull approximation) and the problems described in Section 6.1.2 only concern approximatively 14% of the surface.

6.2.2 Validation

To validate the results of the SBA, the algorithm presented here, some comparative tests with a volume-based method were made [7]. First, results of the SBA and the VBA of CS were compared on surfaces with a reference depth analytically computed. Then, it is illustrated on these surfaces that the VBA converges to the optimal solution as the 3D space division grid size tends to zero. A comparison on the value of the maximal depth was effectuated between SBA and the VBA with different grid sizes. Finally, a smoothness score comparison and an execution time comparison were performed.

The data used for the comparisons is made of proteins from the PDB, of a test protein (named test.pdb) available on the Sharp's website[1], and of 2 artificial virtually generated proteins. These artificial molecules have simple shapes in such a way that the theoretical travel depth could be analytically computed. They are rectangle parallelepiped with a hollow of some atoms along the surface. The first one is small (220 atoms) so that the hollow is deep in comparison with the molecule size. It gives prominence to the disadvantages of the algorithm presented here. The second molecule (see Figure 6.6) is quite bigger (3456 atoms), and the hollow is comparatively small. A least square error for both shapes and methods using different resolutions was computed. Meshes were generated from these molecules SES. They are referred in the following as *Art.1* and *Art.2*.

Least Square Error Comparison An error score was calculated on *Art.1* and *Art.2* between depths computed with both algorithms and the analytically computed depth. This error was weighted by the area. The comparison for the *Art.2* is shown in Figure 6.6. The computed depths are shown on the first line and the local error on the second one. The left-hand side column is for CS's VBA and the right-hand side column for the SBA. The error scores in $Å^4$ appear in Table 6.1. It appears that the SBA is accurate even when the shape was designed to highlight its approximation errors. In the VBA, the size of the 3D small cubes in the grid is constant. Thus if the molecule length in any dimension does not correspond to an integer number

[1] http://crystal.med.upenn.edu/

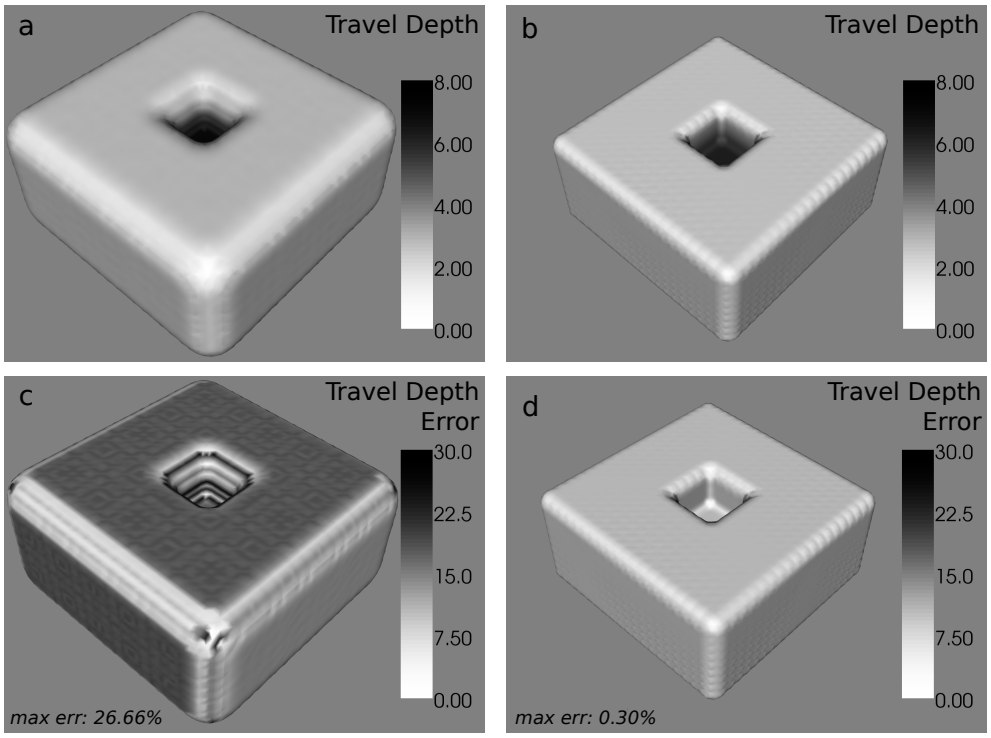

Figure 6.6 Visual comparison of two different methods. The molecule represented here is artificial, and virtually generated. It contains 3456 atoms. On the first row, the depths computed respectively with the VBA (**a**) and the SBA (**b**) are depicted by using a grayscale. On the second row, the error in comparison with an analytically computed depth is represented for the VBA (**c**) and the SBA (**d**). The volume-based method assigns the same depth for all the points of cubes, then we can see the errors due to depth gaps in Figure **c**. The largest error for a point with the SBA method is 0.3%, and then the surface seems completely white since the scale covers an interval from 0 to 30%. [105]

	Volume-based (VBA)	Surface-based (SBA)
Art.1	583.52±18.03	5.93±0.94
Art.2	2288.40±597.52	2.00±0.75

Table 6.1 Error scores for two methods and two meshes ($mean \pm std$ Å^4). [105]

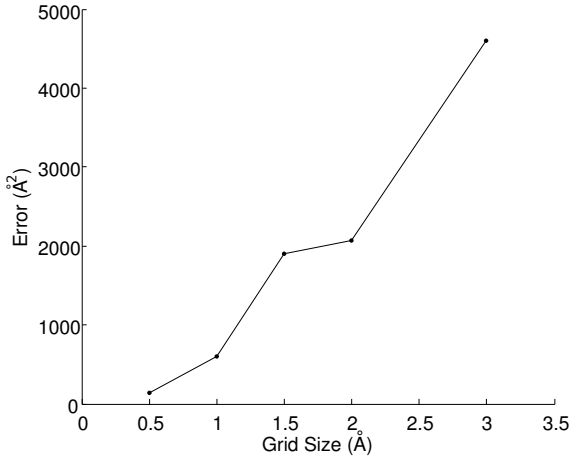

Figure 6.7 Error on the first artificial molecule mesh at different grid sizes using the CS's volume-based method. [105]

of cubes, positioning errors appears at the extremities. It could explain errors appearing on the external surface. The error in the pockets are also depending on the resolution. Points of the surface are considered to be at the same depth for 1Å. Thus the depth is exact only in the center of this area.

Maximal Value Comparison Another way to quantify the exactitude of the SBA is to compare the values of the maximum depth point for different grid sizes of the VBA and the value of the maximum depth point found with the SBA, based on the fact that the VBA tends to give the exact solution while the size of the division grid cubes tends to zero. Figure 6.7 shows the value of the error for the first artificial molecule mesh at different grid sizes. With the CS's volume-based method, the real maximum depth should be in the interval $[B_l, B_u] = [E_G - \frac{\sqrt{(3)}G}{2}, E_G + \frac{\sqrt{(3)}G}{2}]$, where B_l and B_u are respectively the lower and the upper bound of the acceptable interval, and E_G is the estimated maximum depth with a grid size of G, because $\frac{\sqrt{(3)}G}{2}$ is the biggest possible distance from the center of a cube (where the depth is estimated) to any point lying in this cube. Small variations due to errors announced in the CS's paper are possible, and are expected to decrease proportionally with the grid size. Figure 6.8 depicts the value of the maximum depth for 3 proteins at different grid sizes and

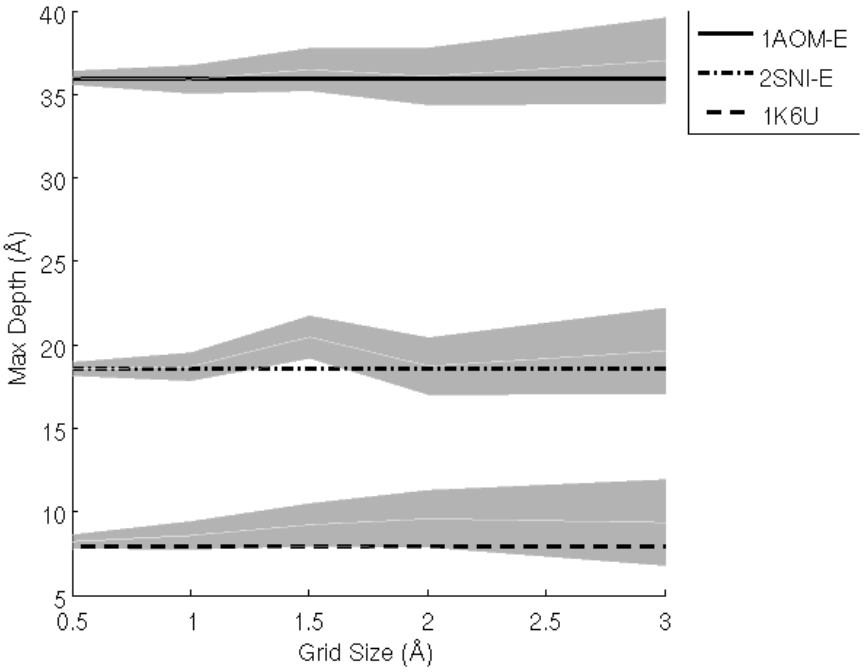

Figure 6.8 Maximal depth of three different proteins with different methods. White thin lines depict the evolution of the maximal depth at different grid sizes with the VBA. It is surrounded by a confidence interval countering approximation due to large grid sizes. Thick lines represent the maximal value obtained with the SBA. It is constant because their resolution is independent. The three proteins shown in example are the 1AOM-E (top, continuous line), the 2SNI-E (middle, dash-dotted line) and the 1K6U (bottom, dashed line). [105]

an interval (gray zone) where the real maximum depth is expected to be located. The thick lines represent the value of the depth obtained with the SBA.

Table 6.2 presents mean results for tests effectuated on 11 meshes. First and second columns are the differences in Å between maximum depth value obtained with the SBA, and, respectively the lower bound (B_l) and the upper bound (B_u) at different grid sizes. Third and fourth columns present the differences between the maximum depth value for the CS's VBA with a grid size of 1 (default size) and B_l and B_u at different grid sizes. Each line corresponds to a grid size used to compute the acceptable bound. E_s is the mean of the maximal depth obtained with the SBA and E_1 is the mean of the maximal depth obtained with the CS's method with defaults parameters (grid size of 1). The mean value are all positive. It means that the maximal depth is, in mean, in the acceptable interval. Both volume- and surface-based methods give equivalent results.

	$E_s - B_l$	$B_u - E_s$	$E_1 - B_l$	$B_u - E_1$
$G = 0.5$	0.67	0.19	0.23	0.64
$G = 1$	0.90	0.83	0.87	0.87
$G = 1.5$	0.85	1.75	1.78	0.81
$G = 2$	1.92	1.54	1.58	1.89
$G = 3$	2.81	2.39	2.42	2.78

Table 6.2 Mean differences between maximal depth values and acceptable bounds (Å). [105]

Smoothness Comparison In addition, a test of smoothness was performed. The smoothness score of an edge between the points i and j is

$$\max\left(|\delta_i - \delta_j| - \| \overrightarrow{ij} \|, 0\right),$$

This means that the importance of the depth gap is taken into account in the score, if it is greater than the distance separating two points. The total smoothness score is the sum of the scores of all the edges. The exact solution should have a null smoothness score because, if the difference of travel depth $|\delta_i - \delta_j|$ between a point i and a point j is greater than the Euclidean distance $\| \overrightarrow{ij} \|$, then the depth of j, δ_j, could be replaced by $\delta_i \| \overrightarrow{ij} \|$. For the SBA, this score was 0.05 ± 0.15 Å. For the VBA, the smoothness score was 812.55 ± 734.41Å.

Execution Time Comparison The last comparison concerned the execution time which is the main advantage. It is shown in Figure 6.9 that the execution time remains reasonable even for meshes with a good resolution (16s for almost 100,000 points). For example, computing the depth of a molecule containing about 1800 atoms with a resolution of 64 voxels/27Å^3 (64 voxels for a cube surrounding a single atom) would take about 6s with the SBA, which is insignificant in comparison with the 1300s that it would take with the VBA. It must be mentioned, however, that the VBA was implemented in python, and that it could have been more efficient using C or C++.

Table 6.3 shows the execution times in seconds for all the proteins used in the experiments for meshes created from 3D images with a resolution of $\frac{2}{3} \times \frac{2}{3} \times \frac{2}{3}$ voxel/Å^3, which is generally sufficient to work on the protein keeping low execution times.

6.3 Discussion

In this section, we proposed an algorithm to estimate the travel depth as accurately as needed in the shortest possible time. This was achieved using a surface-based approach instead of a volume-based approach. It is a solution to maximize

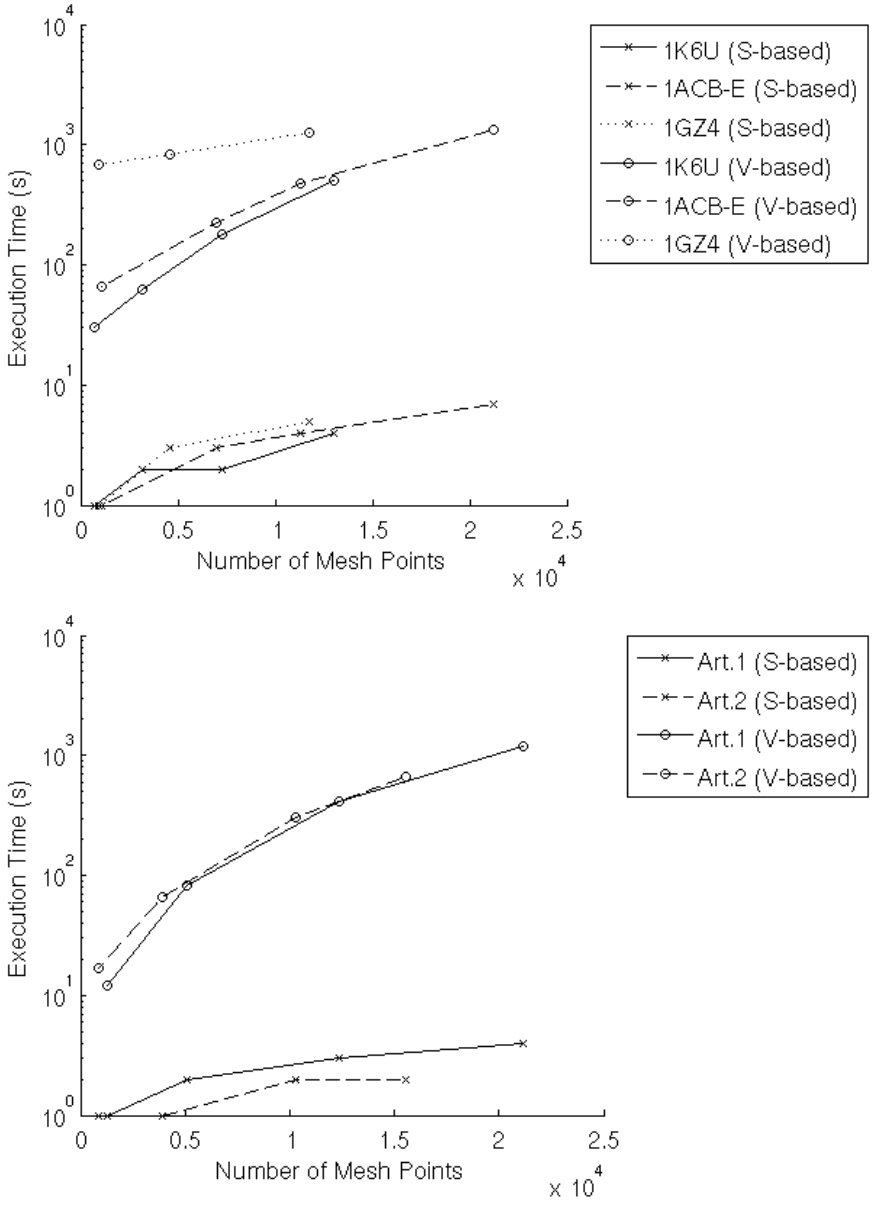

Figure 6.9 Execution times for 5 different surfaces at 4 different mesh resolutions (number of points). The test was performed on the same surfaces with the SBA (crosses) and with the VBA (circles). The upper graph depicts the results for 3 surfaces of proteins from the PDB with both methods and the lower graph illustrates the results for the 2 surfaces of the virtual molecules *Art.1* and *Art.2* with both methods. The time scale (Y-axis) is logarithmic. [105]

Molecule	Type	# Atoms	# Mesh Points	Time
Art.1	Artificial	220	1172	0.132
1EDN	Vasoconstrictor	171	1692	0.216
1PPT	Hormone	302	2588	0.352
1K6U	Enzyme Inhibitor	673	3310	0.372
Art.2	Artificial	3456	3412	0.200
test	Unknown	1009	5452	0.588
1W5W:A	Enzyme	1516	7212	0.728
1A0F:A	Enzyme	1612	7562	0.780
2SNI:E	Enzyme	1940	7732	0.716
1ACB:E	Enzyme	11769	8150	0.848
1A4U:A	Enzyme	1963	8760	0.944
1A4U:B	Enzyme	1963	9300	1.024
1ACB	Enzyme/Inhibitor Complex	2291	9426	0.988
1AOM:A	Enzyme	3434	11856	1.304
1A4Y:A	Enzyme	3410	12326	1.368
1SOX:A	Enzyme	3577	13802	1.752
1SOX:B	Enzyme	3613	14062	1.704
117E	Enzyme	4943	14918	1.624
1PRT	Toxin	14504	40340	6.260
1GZ4	Enzyme	18289	46738	7.304

Table 6.3 Execution times in common situations with the SBA (*s*). [105]

the quality/execution time rate, permitting to perform deep spot detections with execution times on the order of the second, that is, two orders of magnitude faster than with previous methods. The results accuracy was equivalent to the one obtained with default parameters of the tested volume-based approach [7].

A fast algorithm to compute travel can be used in databases screening, researching molecule with particular cavities shape [111]. For example, in a database containing 1000 compounds, detecting cavities on proteins would take around one hour with the surface-based approach and more than one day with the volume-based approach. It is also suitable for visualization because depth can be computed and displayed in a few seconds even for large meshes.

The precision of the algorithm could be improved by resolving the overestimation problem explained in Section 6.1.2. The succession of points forming the path to reach a particular point of the surface from the convex hull could be recorded and reanalyzed in order to find shortcuts. This kind of improvement would obviously increase the computation time. This algorithm could also be partly parallelized: in each step, lists of points are browsed and the points are treated independently. The

propagation steps, for instance, can be done separately for each frontier point and be followed by a single gathering step in which all the possible updates are compared.

Protein Properties

7

The analysis of the surface properties is essential in the context of protein docking. In this work, several surface properties are studied in order to detect potential binding sites (Chapter 8).

Properties of different types are affected to surface points of the proteins. They can be classified into three groups: geometric properties (Section 7.1), physicochemical properties (Section 7.2), and conservation score (Section 7.3).

Some of them are directly related to the surface points and other ones are related to the atoms or the residues of the protein. In order to make the calculations more uniform, these properties can be considered as surface points properties. To do so, each point of the surface is related to the closest atom center and inherits its properties (Figure 7.1).

7.1 Geometric Properties

7.1.1 Travel Depth

The travel depth is a property which is inversely proportional to the accessibility of a surface point for solvent molecule. Chapter 6 is dedicated to this property.

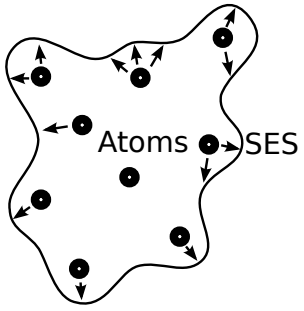

Figure 7.1 The properties related to the atoms are mapped on the surface points. Each surface point inherits the properties of the closest atom.

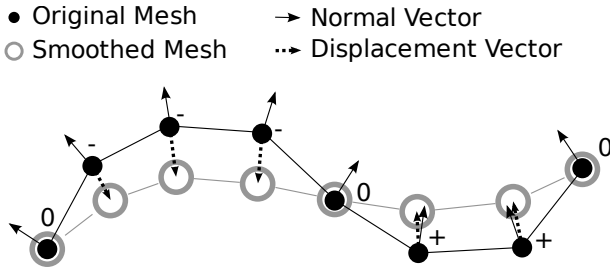

Figure 7.2 Scheme of the mean curvature estimation computation. The original mesh (black disks and links) is smoothed by a Laplacian operator to give a smoothed mesh (gray circles and links). The displacement vectors between both meshes are depicted with thick dotted arrows. The dot products between these vectors and the normal vectors of the original mesh (thin arrows) are calculated. The signs of the results are shown next to the original mesh vertices.

7.1.2 Curvature

The curvature of a surface in a 3D environment can be defined in different ways, as opposed to the curvature of a curve in a 2D environment, which has only one standard definition: the inverse of the radius of the circle that approximates at best the curve at this point. There is, for example, the normal curvature, κ_n, defined as the curvature of the intersection curve between the surface and a normal plane. A normal plane is a plane containing the surface normal vector. The principal curvatures, denoted κ_1 and κ_2, are the minimal and the maximal normal curvatures. The mean curvature H and the Gaussian curvature K are defined as

$$H = \tfrac{1}{2}(\kappa_1 + \kappa_2),$$
$$K = \kappa_1 \kappa_2.$$

Intuitively, the mean curvature is positive for caps or ridges, negative for cups or valleys and zero for planes or saddle points. The Gaussian curvature is positive for caps or cups, negative for saddle points and zero for ridges, valleys or planes.

The algorithms used in this work to compute those two curvatures are based on a Laplacian smoothing of the mesh, which consists in the replacement of each vertex by the mean position of its neighbors. This operation can be iterated (typically, 20 times) and the displacement can be controlled by a relaxation factor to avoid instabilities. The more the filter is iterated, the more the surface is filtered, so that the curvature reflects the global shape instead of local curvatures that can be influenced by noise. Since the surfaces of molecules have similar local shapes, the parameters can be tuned such that the curvature is constant on small spherical parts corresponding to isolated atoms.

The mean curvature is estimated as the dot products between the smoothing displacement vectors and the normal vectors of the original mesh (Figure 7.2). The Gaussian curvature is estimated as the ratio between the surfaces of the polygons of

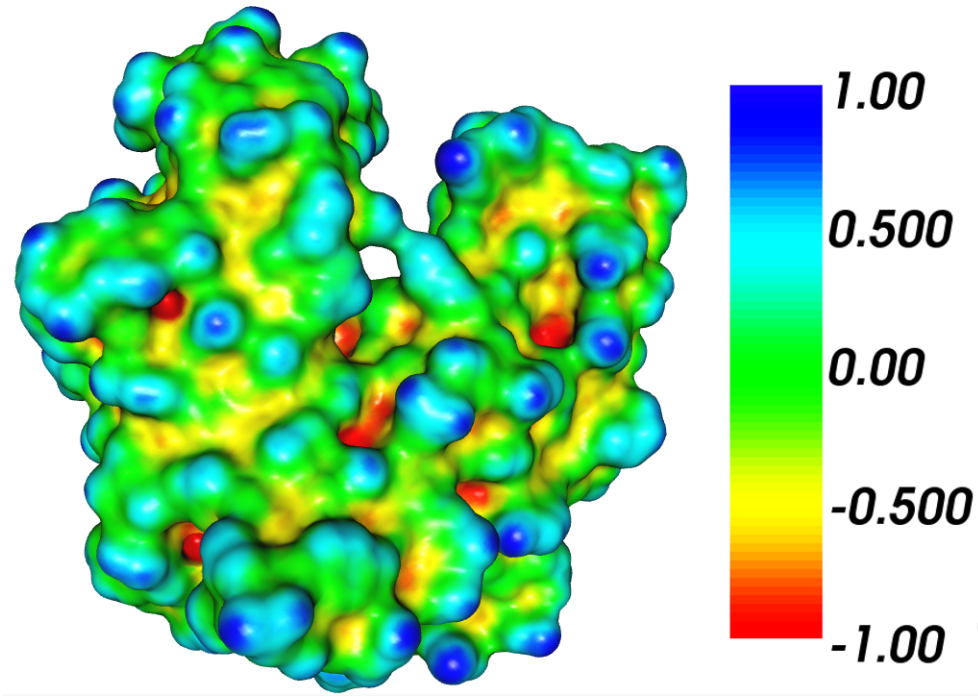

Figure 7.3 Mean curvature on a protein surface (PDB code: 1A0F_A), normalized between -1 and 1.

the smoothed mesh and the polygons (with the same indices) of the original mesh. Examples of mean and Gaussian curvatures on a protein SES are shown in Figures 7.3 and 7.4.

7.1.3 Electron Cloud Density

The work of this section, made in collaboration with Rose Josiane Keumo Tcheuko and Jérôme Ambroise has been presented in [112].

Knowing the mobility of the atoms of a proteins is of main interest in the study of molecular interactions. As mentioned in Section 2.6, proteins are not static and this mobility is almost always necessary for a docking to occur (Section 3.6).

The large amount of parameters and the high vibration frequency of the atoms make generally the explicit simulations very slow. Then, it is useful to find methods to speed up the computations. Some existing method use the normal vibration modes of the molecule. They are generally based on the eigenvector decomposition of the adjacency matrix (considering the covalent bonds) [61, 113, 114].

In this section, we propose a method to predict the atoms mobility using Fourier filtering on an electron density map (or potential field).

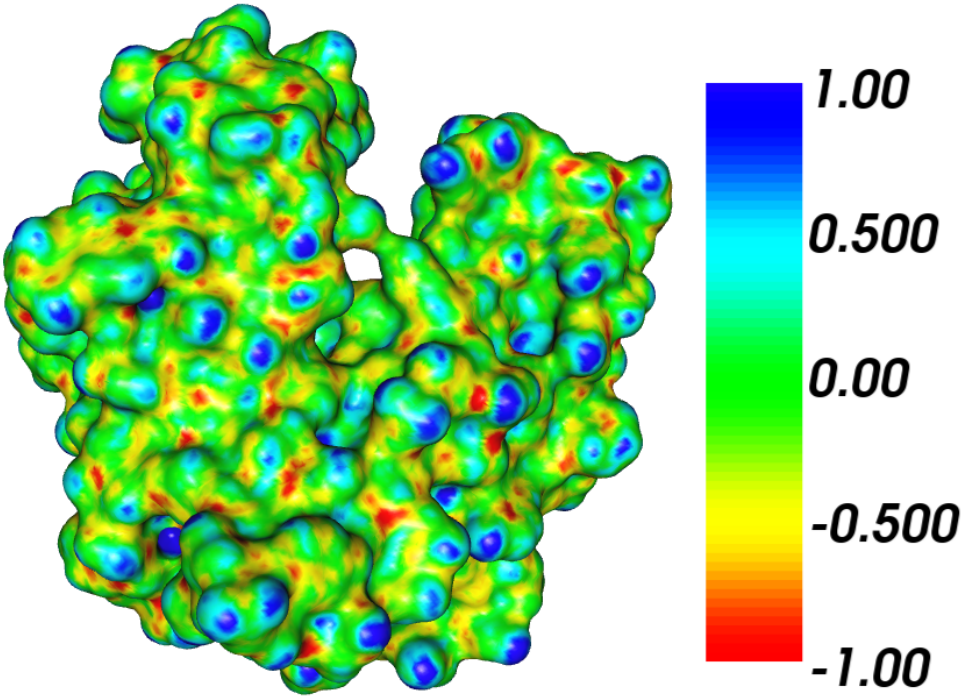

Figure 7.4 Gaussian curvature on a protein surface (PDB code: 1A0F_A), normalized between -1 and 1.

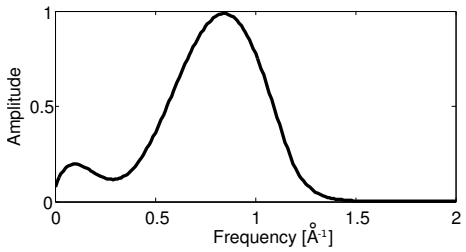

Figure 7.5 Filter designed empirically for the evaluation of the atom stability. This filter is applied to the 3D Fourier transform of the electron density map described in Section 4.3.1.1.

7.1.3.1 Method

First of all, a training data set containing ten kinase proteins was build. A dynamic simulation using NAMD [60, 115] at 310K with a time step of 2fs for 250ps (125000 steps) was performed on this data set. For each protein, ten conformations as different as possible were selected. This selection was done using a greedy recursive algorithm, adding at each step the conformation with the largest RMSD (root mean square deviation, see Section 3.2) in relation to the conformations which were already selected. A stability score ζ was affected to each atom of each protein. This score for the atom i is $\zeta_i = 1 - RMSF_i$ where $RMSF_i$ is the root mean square fluctuation of the atom among the ten conformations. In order to approximate the ζ's with a stability score, a frequency filter is designed as follow: two potential fields are generated by centering 3D Gaussian functions on the atoms of one of the proteins. The first field is parameterized such that, for a certain threshold, the isosurfaces are the vdW spheres. The second field is parameterized such that the isosurface spheres radius, at the same threshold, are varying from 0 to 1 proportionally to ζ. The 3D Fourier transforms are computed for both potential fields. The histogram of the ratio between both transforms makes appear that some part of the frequency spectrum are more expressed for the stability field than for the classical vdW field. The filter is designed such that these parts of the spectrum are enhanced (see Figure 7.5). The frequency space has three dimensions but since this filter has to be rotation invariant, a point in the frequency space has influence on the three coordinates it corresponds to. Once the filter is designed, it can be applied on the potential field transforms of the ten proteins. The inverse transform of the filtered frequency field is the approximation of the stability scores field (see Figure 7.6).

7.1.3.2 Results

The mean correlation (among the ten proteins) between the predicted stability scores ζ_{pred}'s and the stability score ζ's computed using dynamic simulation was 0.71 (with a standard deviation of 0.05). An example of both scores mapped on the protein structure is shown in Figure 7.7 (PDB code 1GZ3, correlation: 0.70). Simulations were also performed with the ElNémo server [5] returning a similar score

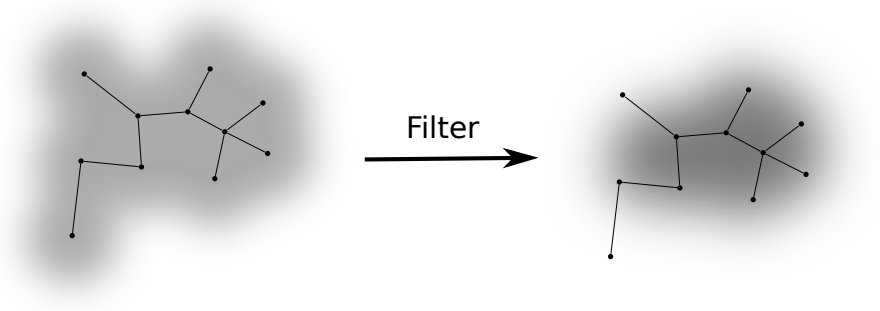

Figure 7.6 Two dimensional illustration of the effect of the filter on the electron density map. The points and the lines represent the atoms and the covalent bonds.

called the B-Factor. The correlation between the B-Factor and the stability score was 0.51 (with a standard deviation of 0.07).

7.1.3.3 Discussion

We presented here a method to predict the stability of the atoms of a protein with a low computation time (a few seconds instead of several hours for the dynamic simulations). The predictions are well correlated with the stability of the atoms during dynamic simulations. Our method gives better results than the ElNémo server, which seems to be better at predicting large conformation changes, instead of small side chain movements. The prediction of such stability scores can be used in dynamic simulations to make move only the non stable atoms. This result could be improved by designing a filter based on a spherical Fourier transform and by designing the filter using more training proteins.

7.2 Physicochemical Properties

The chemical properties considered are twenty composition properties and the hydrogen bond directionality (only for the epitope application).

The composition properties are the proportion of each surface amino acid composing a surface patch. For example, if a fifth of the points in a patch corresponds to a leucine and the rest to a proline, the composition of the patch is Leu: 0.2, Pro: 0.8, other amino acids: 0.

In the epitope study, 544 other physicochemical properties are derived from the composition properties and from the 544 propensity scales found in the AAindex database [116]. For a particular scale, the property of a given patch is computed as a weighted mean of the values assigned to each residues in this scale. The weights are the proportion of the amino acid composing this patch.

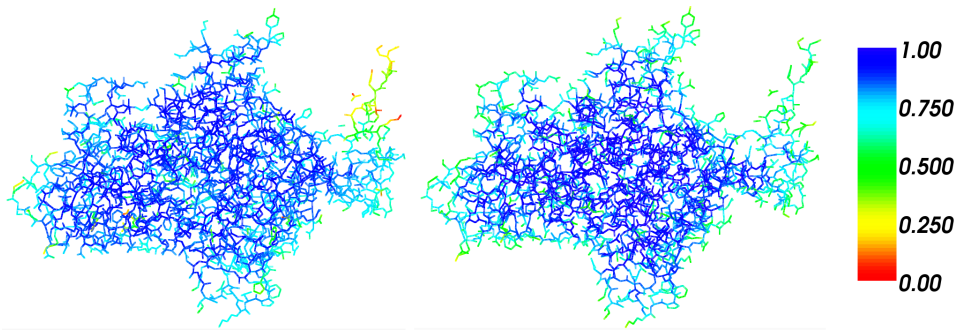

Figure 7.7 Comparison between the atoms stabilities (right) and the predictions (left) (PDB Code: 3EBZ). Blue vertices correspond to very stable atoms and red vertices to very mobile atoms.

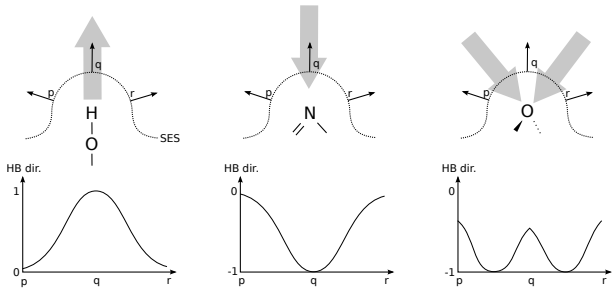

Figure 7.8 Three examples of the hydrogen bond directionality profile along a surface. O-H (left) is donor and the hydrogen bond orientation vector is aligned with the O-H axis. Thus the profile is maximum when the normal vector is aligned with this vector (in q) and decreases as the normal vectors change their orientation (in p and r). N with one free electron pair (center) is an acceptor and the profile of the descriptor is the opposite to the one of O-H. O with two free electron pairs is a double acceptor, so the profile has an inflection point halfway between both acceptor orbitals (in q).

7.2.1 Hydrogen Bonds Directionality

Hydrogen bonds play an important role in protein-protein interactions [117,118]. Their simple presence, but also their orientation are considered in various application such as energy score functions of docking algorithms [119][1].

In order to conserve a scalar surface descriptor vision, the hydrogen bond directionality is described in this work as the scalar product between a unitary hydrogen bond acceptor or donor vector of the closest atom and the normal vector at this point of the surface (Figure 7.8). For each point p of the surface, the closest non-hydrogen atom a is localized. The donor and acceptor hydrogen bond vectors (see Section 2.1.2) are treated separately. Let $\vec{b}^{+}_{a,i}$ be the i^{th} unitary hydrogen bond donor vector

[1] Thanks to Philippe Manivet for the interesting discussions about the importance of hydrogen bonds.

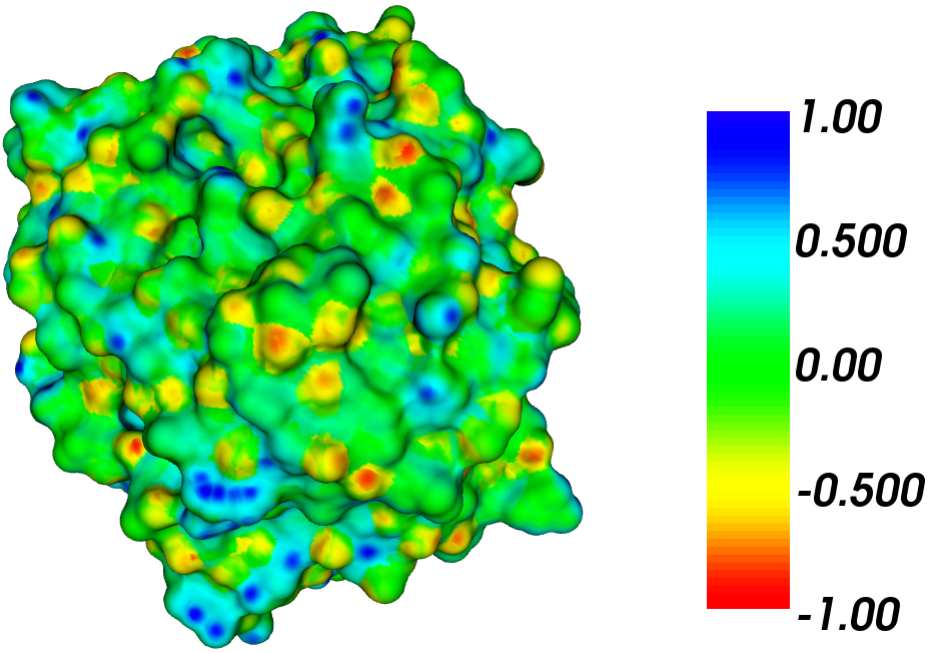

Figure 7.9 Hydrogen bonds directionality on a protein surface (PDB code: 1A14_N). Negative values (red) represent acceptors and positive values (blue) represent donors.

attached to a, if any, and $\vec{b}^{-}_{a,j}$ be the j^{th} unitary hydrogen bond acceptor vector attached to a. All the $\vec{b}$ are directed from the donor to the acceptor. The hydrogen bond directionality score attached to the considered point is

$$hbd = \max_{i \in H_a} \frac{\vec{n}_p . \vec{b}^{+}_{a,i}}{\|\overrightarrow{ap}\|} + \max_{j \in O_a} \frac{\vec{n}_p . \vec{b}^{-}_{a,j}}{\|\overrightarrow{ap}\|}$$

where $\vec{n}_p$ is the normal vector of the SES at p, H_a is the set of donor vectors and O_a is the set of acceptor vectors (depending on the acceptor orbitals) attached to a. An example of such a directionality mapped field is shown in Figure 7.9.

7.3 Conservation Score

The conservation score of an amino acid residue is related to the evolution and the mutation rate [120]. This score is high if the amino acid is well conserved. To quantify the conservation score, the part of the sequence containing this particular residue is compared by pairwise alignment to proteins of the same family (see Section 3.1). A high alignment score is the expression of a highly conserved sequence part, which is potentially important for the structure and/or the function of the protein for which a mutation would have been lethal.

Binding Site Detection

8

The work of this chapter, made in collaboration with Jérôme Ambroise has been published in [121].

In binding site detection methods, the objective is to predict the location of potential binding sites of a protein with any other molecule. The applications of such methods are various: they can be used, for example, to accelerate a docking process by selecting the limited search areas, or to create drugs mimicking the binding site and causing the same effects.

The aim of this chapter is the description of a patch-based protein-protein binding site localization method (the PBSL for Patch Binding Site Localization) including statistical tools. Since the patches are defined using geodesic distances, they are suitable to take possible conformation changes into account and thus the PBSL is adapted to prediction on unbound structures. The PBSL is illustrated by two application examples. The first one concerns a heterogeneous data set and the binding site to be predicted can be an enzyme catalytic site (see Section 2.6.1) or the interface between two subunits of a dimer (see Section 2.3.4). In that study, several statistical tools are compared in order to find which one provides the best predictions. The second example concerns the B-Cell epitopes, i.e. the part of the antigens which is recognizable by an antibody (see Section 2.6.2).

This chapter is organized as follows. The state of the art for both applications, the various binding sites and the epitopes, are introduced in the background section (Section 8.1). Then, in Section 8.2, the pipeline of the method, which is common for both applications, is described in details. The results and the comparisons with existing methods, separated for both applications, are exposed in Section 8.3. Finally, the results and the perspectives are discussed in the conclusion (Section 8.4).

8.1 Background

8.1.1 Various Binding Site

The use of machine learning and statistical tools to combine the properties of proteins in order to localize a binding site have already been well explored. Petrova and

Wu [122] compared 26 machine learning classifiers in order to identify the catalytic residues constituting the active site of an enzyme. Gutteridge et al. [123] trained a neural network classifier to score the residues of a protein by their likelihood to be catalytic. The location of the active site was determined by searching for clusters of high-ranking residues.

Shulman-Peleg et al. proposed a method, called SiteEngine [124,125], which compares the properties at different regions of the proteins surface, such as conformational properties, in order to find proteins with similar functions in databases. The same kind of approach has been adopted by Shatsky et al. [126] to recognize binding patterns common to a set of protein structures. Jones and Thornton [127] presented a method for predicting protein-protein interaction sites. The prediction was based on the calculation of relative combined scores for patches constructed on the surface of protein structures. Zhou and Shan [128] published a method based on a neural network for predicting individual residues in protein-protein interfaces. Since then, many methods predicting individual residues or patches that overlap with binding sites have been proposed [122, 129–134].

Bradford et al. [130] proposed a protein binding site predictor based on a Support Vector Machine (SVM) discriminating the patches constructed by segmentation of the surface. After the segmentation and the computation of some relevant properties for each patch of each protein, the SVM was trained and became usable to distinguish binding site from non-binding site surface parts. Bradford and Westhead [135] used a Bayesian network in combination with a surface patch analysis to design a protein-protein binding site predictor. Petsalaki et al. [136] presented a different approach to predict peptide binding sites on protein surface. This approach is based on the construction of spatial position specific scoring matrices for each of the twenty standard amino acids. Several groups performed the comparison of different tools used for the prediction of protein-protein binding sites [137, 138].

8.1.2 Epitopes

B-cell epitopes are the portions of antigens that are recognized by a specific antibody or B-Cell receptor. Detecting the immunogenic region of the antigen is useful in numerous immunodetection and immunotherapeutics applications. It allows, for example, the design of short peptides with similar properties and activating immunologic response without carrying the harmful character of the antigen. This kind of synthetic short peptides mimicking epitopes, can be used in vaccine design [139] and for immunodiagnosis reagents [140, 141].

As the experimental approaches used for detecting immunogenic regions are resource-intensive, computational methods for the prediction of epitopes were developed during the last 25 years. Early methods for detecting epitopes on antigens were based on the sequence and on the residues properties. These methods were based on amino acid propensity scales such as the hydrophobicity [142]. A score

drawn from the propensity scale was assigned to each amino-acid of the antigen sequence and high scoring segments were inferred as the candidate epitopes. However, Blythe and Flower [143] showed that approaches using single-scale amino acid property conduct to predictions, on average, only marginally better than random.

More recently, several properties were combined using machine learning approaches such as SVM (COBEpro [144]) or neural networks (ABCPred [145]). These methods were developed for the prediction of continuous epitopes, composed of residues that are directly connected in the sequence [146]. Crystallographic studies, however, have shown that more than 90% of the epitopes in protein antigens are not connected in the sequence [147]. Thus efficient methods for predicting discontinuous epitopes are needed.

For discontinuous epitopes prediction, spatial conformation of the protein has to be taken into account. When the 3D structure of the antigen is available, 3D properties of amino acid can be extracted and used to improve the prediction of the immunogenic regions. The considered conformational properties are, for instance, the backbone flexibility [148], the protrusion [149], the accessible surface area [150] or the accessibility (CEP [151]). Such as for single amino acid properties, 3D descriptors can be combined with other 3D descriptors or with amino acid properties. After studying the histogram of contact numbers (the number of C_α atoms within a certain distance threshold) and an amino-acid propensity scale, Andersen et al. combined them using a linear model in the Discotope method [152]. Sweredoski and Baldi used the same approach adding accessibility information in the PEPITO (later renamed BEpro) method [153]. Epitopia [154] is a method combining several properties using a Naive Bayes Classifier. For this method, the relevant properties were selected using statistical inference tools [155].

8.1.3 Contributions

The first contribution of this work is the description of a patch-based protein-protein binding site localization method (the PBSL) including either regression or classification tools. In the literature, patch-based binding site localization methods generally use supervised classifier for finding the relationship between the patches properties and their overlap with the true binding site. Machine learning or statistical techniques are used to discriminate patches totally superposed with the real binding site from patches lacking joined surface with it. In these cases, the response variable is categorical because it only takes two distinct values. Therefore, supervised binary classifiers are used to predict the response variable from the predictor variables. However, when predicting the location of the binding site of a protein, many patches constructed are partially superposed with the real binding site. Using regression tools instead of classification tools allows the inclusion of partially superposed patches during the training process. It should improve the predictive performance of the model. The predictive performance of several regression and

classification tools are compared using leave-one-out cross-validation on the first application (various binding sites).

The second contribution is to include original properties in the model, such as the travel depth and the electron density (see Sections 7.1.3 and 7.2.1).

The third contribution is the selection of appropriate properties as input of the statistical model, using of a greedy backward elimination method for variable selection [156]. The principle of such a selection is to eliminate the variables that are the less significant.

The fourth contribution is the construction of a final patch combining patches with a good score. In the literature, for patch-based methods, the predicted binding site in chosen among the original patches according to the ranking of their probability to be the real binding site. The PBSL takes into account the output of the statistical model on the whole surface to construct a new patch, which forms the predicted binding site. The size of this predicted binding site can be adjusted according to the application.

The fifth contribution is the use of the PBSL to the particular case of antigen epitopes. Although it lies on the same principles than binding site detection, epitope detection constitutes a separate field of research with different methods. Reusing the same ideas in epitopes as for various binding site gives better results than previous methods.

8.2 Method

The aim of the PBSL is to predict the location of potential binding sites in a protein according to its 3D structure. The development of the PBSL is composed of two successive parts: the training and the application. A scheme of the method pipeline is depicted in Figure 8.1. Input data of the training part are the PDB files of several protein complexes. External surfaces of the proteins are constructed by using the 3D structures encoded in the PDB files (see Section 4.3) . Some characteristics are extracted or mapped on these surfaces (see Chapter 7). Then, the surfaces are segmented into patches. The mean values of the properties, their standard deviations, and an index quantifying the overlap with the real binding site are computed for each patch. Finally, a statistical model is used to find a relationship between the patches properties and their overlap indices.

Input data of the application part are one or more PDB files. Patches are constructed in the same way as in the training part except that the overlap index with the real binding site cannot be computed because the location of the real binding site is unknown. The statistical model previously trained is then used to allocate a score to each patch. Finally, a predicted binding site is located by using the score mapped on the protein surface.

The application part can be tested on a protein of known binding site location. In this case, the location is considered as unknown during the prediction of the binding

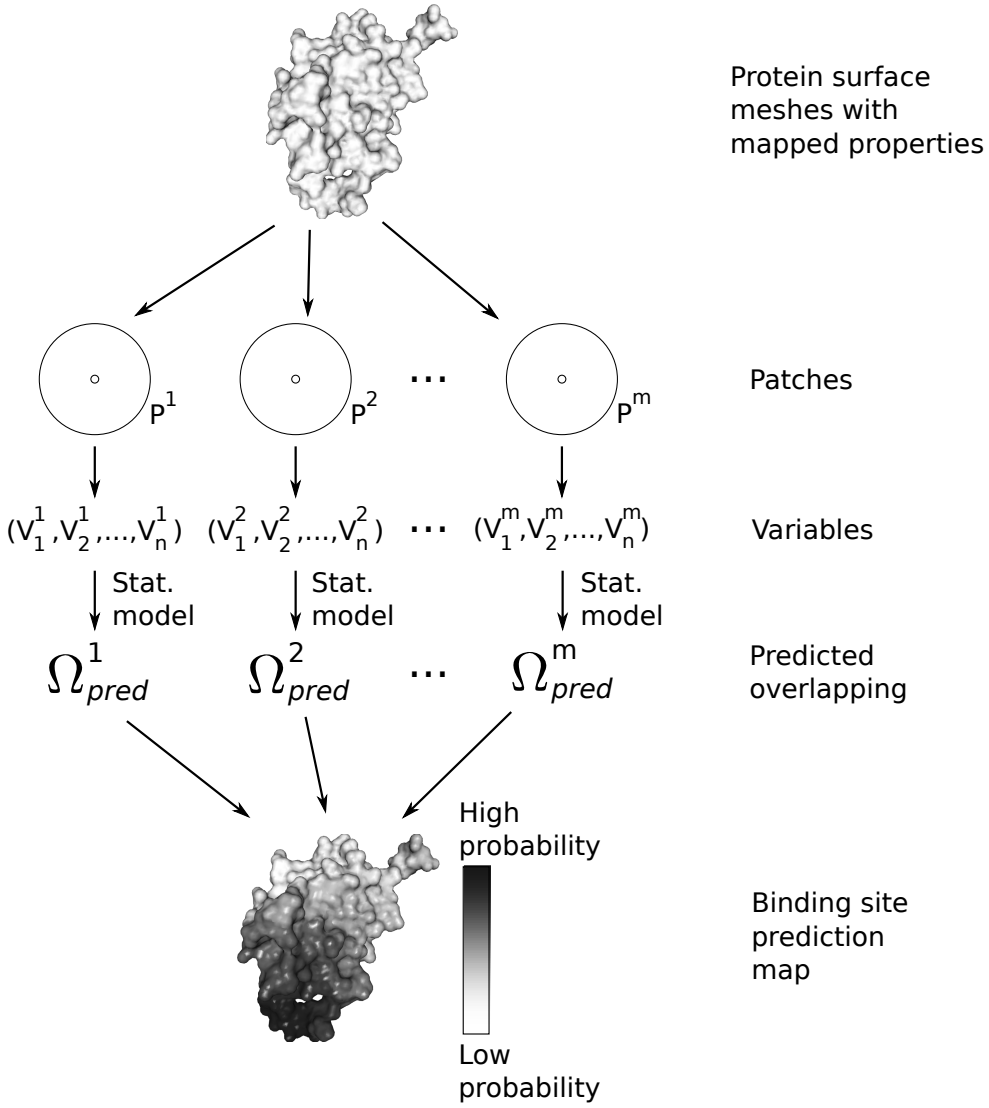

Figure 8.1 An overview of the PBSL. The first step is the surface construction and the extraction of the protein surface properties (PDB code:1AFV_A). Then, the surface is divided into patches. A set of variables is affected to each patch. These variables are the means and the standard deviations of the surface properties as well as the proportions of some amino acid residues inside the patch. They constitute the prediction variables for a statistical model of which the response variable is the predicted overlap Ω_{pred} between the patch and the binding site. The statistical model is previously trained on protein for which the binding site location and the the real Ω's are known. The final steps are the mapping of these predictions on the protein surface and the prediction of the binding site location by thresholding this mapping.

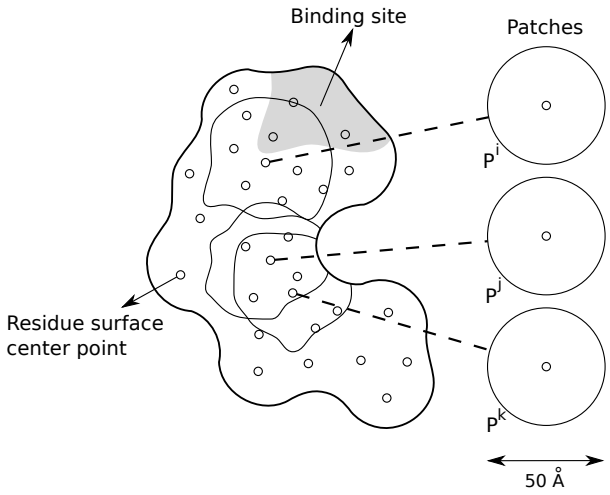

Figure 8.2 Construction of the patches on the protein surface. There is one patch centered on each surface amino acid residue. The patches are delimited by geodesic circles with a diameter of 50 Å. P^i, P^j and P^k are three examples of such patches. An important overlap exist between patches, in practice, a surface point belongs on average to fifteen patches. The aim of the PBSL is to predict the overlapping index Ω between the patches and the real epitope. Here, $\Omega^i \simeq 0.2$ and $\Omega^j = \Omega^k = 0$.

site location. Comparing the predicted location of the binding site with the real one, is used to validate the PBSL.

8.2.1 Properties

Properties of different types are affected to surface points of the proteins. These properties can be classified into three groups: geometric properties, conservation score and physicochemical properties. All these properties are described in details in Chapter 7.

8.2.2 Binding Site Definition

The way the binding site is defined is not universal. Several definitions exist and are generally based on known complexes. In this work, it is defined as the set of residues in which any atom of the epitope residue is separated from any antibody atom by a distance < 4Å with a minimal SES area of 1Å^2 [157].

8.2.3 Patches

Once the mesh is created, patches are constructed. The construction is very simple (Figure 8.2). Patches are delimited by geodesic circles centered on each surface amino acid. A geodesic circle is a set of points at a constant geodesic distance from a center point. The diameter of these circles is constant and is set empirically to 50 Å. It is a compromise between robustness to conformational changes (small patches) and

consideration of the surrounding areas properties (large patches). The patches are wider than the amino acid SES so that an important overlap exists between patches. In this case, each surface point belongs on average to fifteen patches. The mean and the standard deviation of the properties is computed for each patch. They constitute the variables used to predict an overlapping index with the binding site.

8.2.4 Overlapping Index Definition

An overlapping index between the binding site and each patch is defined as follow:

$$\Omega = \frac{S(P \cap B)}{S(P)},$$

where S is the area on the SES, P is the analyzed patch and B is the binding site. To give an intuitive vision of this index, Ω is equal to one when the patch is completely included in the binding site. In the training part, the true binding site is also considered as an additional patch.

8.2.5 Variables Selection

One objective of this study is to find relevant variables for the identification of binding sites in order to design more efficient and less heavy statistical models. Such variables can be identified with various tools. Here, the filters and the greedy backward elimination methods are presented [156].

8.2.5.1 Filters (Relevance Scores)

In filter techniques, a relevance score is computed for each variable and the low-scoring variables are removed. The disadvantage of this method is that it does not take the interactions between variables into account. In this work, it was not used in practice but relevance scores were computed to confirm the results of the other variables selection methods.

Two scores computed for each variable separately are presented in this section. The first score highlights variables that are highly different between the patches perfectly overlapping the epitopes ($\Omega = 1$) and the patches sharing no common surface with the epitope ($\Omega = 0$). A t-test is performed on these two groups and for each property separately. The p-value associated to the t-test is very small if a given property is highly different on the epitope than on the rest of the surface. The $-\log(\text{p-value})$ of the t-test is chosen as first score. Relevant properties are associated with high value for this score.

The second relevance score takes into account all the patches, even those partially overlapping the epitope. For a given variable, a Pearson correlation coefficient between the overlapping index Ω and this variable is computed. Relevant variables are those associated with a high absolute value of the correlation coefficient.

8.2.5.2 Greedy Backward Elimination for Regression

First, a complete regression model is constructed with all the variables. Each variable is associated to a p-value reflecting its influence on the response variable. The variable with the higher p-value (low relevance) is removed from the model and a new regression model is constructed with this subset of variables. The process stops when all the variables have a p-value > 0.05.

8.2.5.3 Greedy Backward Elimination (Wrapper)

The idea is to start with a model containing all the variables and to evaluate its predictive performance with a leave-one-out cross-validation. Every variable is eliminated and a new model is constructed with the left subset of variables. The variable whose the elimination improved the best the predictive performance of the model is removed definitely. The backward elimination stops when no more irrelevant variables are found.

8.2.6 Statistical tools

The objective of the statistical model is to assign a score to patches of a protein of unknown binding site location.

During the training part, the input data of the statistical model are the patches associated to their Ω's and the properties extracted in the previous steps. The objective of the training is to find a relationship between the extracted patches properties (predictor variables), and the patches Ω (response variable).

The relationship is used afterwards in the application part to allocate a score to the patches resulting from the segmentation of a target protein.

In this section, different statistical tools are presented. They can be separated into two categories: the classifiers, which are more commonly used to treat this kind of issues, and the regression tools. The output of a classifier is the probability of each patch to represent the real binding site. The output of a regression model is a predicted Ω, for each patch. For the sake of simplicity, the statistical model output score, whether it be a probability or a predicted Ω, is written Ω_{pred}. It has to be highly correlated with Ω to ensure the success of the PBSL.

8.2.6.1 Multiple Linear Regression (Lin. Reg.)

Multiple Linear Regression [158] is a statistical technique that fits the relationship between the response variable and the predictor variables, usually, by minimizing the sum of the squared deviations. The best multiple linear model can be found by using a stepwise selection of the predictor variables.

8.2.6.2 Partial Least Square Regression (PLSR)

Partial Least Square Regression [159] is an extension of the multiple linear regression. Using multiple linear regression with a high number of predictor factors leads to over-fitting: the model fits the sampled data closely but fails to correctly predict

new data responses. In a partial least square regression, a few latent factors are extracted and replace the original predictor factors for the response fitting. Partial least square is useful to find a good predictive model without necessarily understanding the existing relationship between variables.

8.2.6.3 Principal Component Regression (PCR)

Principal Component Regression [160] uses principal component analysis to fit the relationship between the predictor variables and the response variable. The first step is to compute the principal components of the predictor variables. The second step is the regression on a subset of the principal components.

8.2.6.4 Multilayer Perceptron (MLP)

Multilayer Perceptron [161] is a machine learning technique using multiple layers of neurons. A neuron is a simple processing element, connected to the neurons of the previous and of the following layers. The connections between the neurons are characterized by weights which are adjusted during the training step. Multilayer Perceptrons can be used for regression as well as for supervised classification task. In this work, it was used as a regression tool.

8.2.6.5 Random Forest for Regression (RFR) and Classification (RFC)

Random Forest [162] is a machine learning tool used for classification and regression tasks. In both cases, a random forest is a set of trees created by bootstrapping samples of the training data set. Random forests for regression use a set of regression trees. In this case, the prediction is made by averaging the predictions of the different trees. Random forests for classification use a set of classification trees. In this case, the prediction is made by a majority vote on the predictions of the different trees.

Regression trees and classification trees are both decision trees. A decision tree is a technique used to predict a response variable using a set of predictor variables. Regression trees are used for continuous response variables, whereas classification trees are used for discrete response variables. In both cases, it uses a series of *"if then else"* conditions based on values of the predictors to predict the response. During the training phase, the decision tree is built through an iterative process of data splitting. This iterative process continues until each node reaches a fixed minimum size. For example, if there are three predictor variables, the data can be separated from the root according to the value of the first variable. At each new node, the data can then be separated according to the value of other variables, and then, at children nodes, they can be separated according to the first variable again, etc. During the prediction phase, a new data starts from the root and follows the branches corresponding to the value of its predictor variables. The leaves determine the value of the response variable.

8.2.6.6 Naive Bayes Classifier (NBC)

Naive Bayes Classifier [163] is a machine learning tool using the Bayes theorem to compute the probability of a case to belong to a category. The NBC is based on a conditional independence assumption. Given the value of the response variable, the predictor variables have to be independent. Despite the fact that this assumption is rarely true in reality, naive Bayes classifier often performs better than expected.

8.2.6.7 Support Vector Machine (SVM)

Support Vector Machine [164] is a machine learning tool used for classification and regression. When used for binary classification, the objective of the SVM is to map the training data into a property space by the aid of a kernel function, and to constructs an N-dimensional hyperplane that optimally separates the case according to the category of their response variable.

8.2.7 Predicted Binding Site Construction

After statistical analyses, the score Ω_{pred} assigned to each patch is used to construct a score map on the protein SES. Each point of the surface is associated to the geodesically closest center of patch and get the Ω_{pred} affected to this patch. Next, the points of the mesh representing the SES are added successively to the predicted binding site in descending order of score. The growth of the predicted binding site stops when either a predetermined size or a score threshold is reached.

For the epitopes study, the score is redistributed by residue in order to be able to compare the results with those of other methods. In this case, the score affected to each residue is the weighted mean of the scores of the corresponding surface points. The residues are predicted to be a part of the epitope in descending order of score.

8.3 Results

8.3.1 Various Binding Sites

The first application of the PBSL is the detection of binding sites of various proteins. This first study is mainly focused on the choice of an appropriate statistical model for a patch-based binding site prediction method.

Results comparing, on one hand, the different statistical tools, and, on the other, the PBSL to other existing methods are presented in this section. The different statistical tools were compared by calculating the correlation coefficient between predictions and real Ω's. Next, they were compared after the mapping step by analyzing the Ω_{pred} distribution inside and outside the genuine binding site. Finally, overlapping indices between the predicted and the real binding site were computed. These indices were used to compare the results obtained with different statistical models, as well as to compare results of the PBSL with results of three existing methods usable through Web Servers.The first method, named SHARP^2 [131], is a patch-based binding site localization method using a combined scoring function. The second

method, named PINUP [132], is based on an empirical scoring function including a side-chain energy term, a solvent accessibility term and a conservation term. The third method named cons-PPISP [129] is based on a neural network taking PSI-blast sequence profile and solvent accessibility as input. The predictive performance are evaluated using leave-one-out cross-validation.

8.3.1.1 Data Sets

Two data sets are used in this work and both are composed of known protein complexes. The first one, used to train the models and to compare the statistical tools, is the one used by Bradford et al [130] (See Appendix A). This data set is composed of proteins in their bounded conformation and had already been filtered at 20% sequence identity using a PSI-BLAST homology search [17]. The second one is used to compare the PBSL with existing methods. It is composed of 35 proteins in the enzyme/inhibitor category of Docking Benchmark 2.0 [165], after filtering at 35% identity using a clustalW multiple alignment search [18] (See Appendix B). The genuine binding sites were identified using the bounded complexes but the tests were performed on the unbounded structures. As the proteins of the training data set originate from PDB files containing proteins complexes, the locations of their binding sites are already known.

8.3.1.2 Variables Selection

Relevance Scores Both relevance scores, the $-\log(\text{p-value})$ of the t-test and the absolute value of Pearson correlation coefficient were computed for the three data sets. Results are displayed in Figure 8.3. The outstanding variables are the mean travel depth and the mean conservation score.

Backward Variables Selection A backward variables selection was applied on the data with the Lin. Reg. model. The aim of this selection is not to optimize the performance of the final model but to keep the same set of variables for all the statistical models in order to compare them. The selected variables were the mean of the travel depth (deptm), the mean of the mean curvature (curvm) , the mean of the conservation score (consm), the proportion of aspartic acid (propD), glutamic acid (propE), lysine (propK) and phenylalanine (propF). The proportion of all the amino acids, however, were included in the model. No standard deviation variable was selected.

8.3.1.3 Statistical Models Comparison

The performance of the PBSL strongly depends on the ability of the statistical model to correctly classify a patch or to predict the corresponding Ω. For both classification and regression tools, the output is a number between 0 and 1 called the Ω_{pred} of the patch.

Each classification and regression tool was evaluated through leave-one-out cross-validation, which is an iterated process. At each step, the data set, compris-

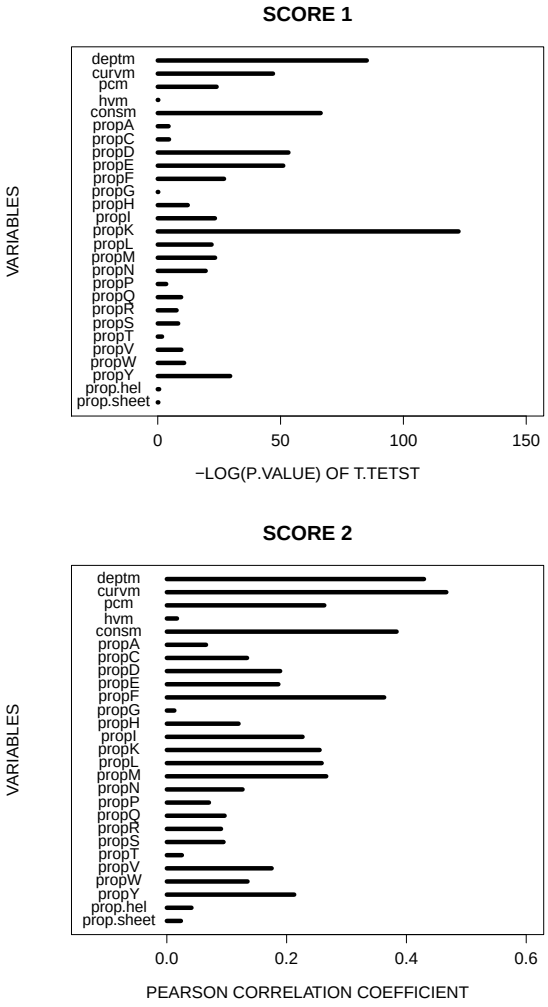

Figure 8.3 Scores of the variables for the binding site prediction.

ing the patches and their properties, was divided into two sub-sets: a test data set, which contained the patches of one protein, and a training data set, which contained the patches of all the other proteins. The statistical model was trained on the training data set, and used to affect a Ω_{pred} to the patches of the test data set. The Ω_{pred} were finally compared to the Ω's through the computation of the Pearson Correlation Coefficient. The protein in the test data set was different at each iteration until prediction and correlation coefficient were obtained for the patches of each protein in the complete data set.

After the leave-one-out cross-validation, each statistical tool was associated to 180 correlation coefficients, each of them corresponding to one protein. Distributions of these correlation coefficients for regressions and classifications are compared in the Box Plot shown in Figure 8.4. Using regression to fit the relationship between the properties and the Ω frequently led to better Ω_{pred} correlations than using classification.

As the correlation coefficients vary considerably from one protein to another for the same statistical tool, the box plots are stretched and the overlap is important. Relative box plot is more suitable to visualize which statistical tool generally produces the best results. For this purpose, relative correlation coefficients were computed for each protein separately, relatively to the mean of the scores for all the statistical tools. A positive relative coefficient means that the statistical tool better works than the average. The distribution of these relative coefficients is depicted in a box plot in Figure 8.5. The box plot of the relative correlation coefficients shows that, for one particular protein, regressions generally provided better prediction than the average, whereas classification generally provided worse prediction than the average.

8.3.1.4 Binding Site Localization

After statistical prediction, a predicted binding site was constructed in two steps, namely the score map construction and the thresholding on the score map. The score map construction was validated by comparing the score distribution inside and outside the genuine binding site, using a histogram. The thresholding on the score map was validated by analyzing the Positive Predicted Value (PPV or precision) and the sensitivity (or recall) of the resulting predicted binding site. These indices are defined as follow:

$$Sens = \frac{S(P \cap B)}{S(B)},$$

$$PPV = \frac{S(P \cap B)}{S(P)},$$

where S is the area of the SES, P is the analyzed patch and B is the true binding site. The validation results for the different statistical tools are compared in this section.

Score Map Histograms For each statistical tool, the score distribution of the points within the real binding site was compared with the score distribution of the points

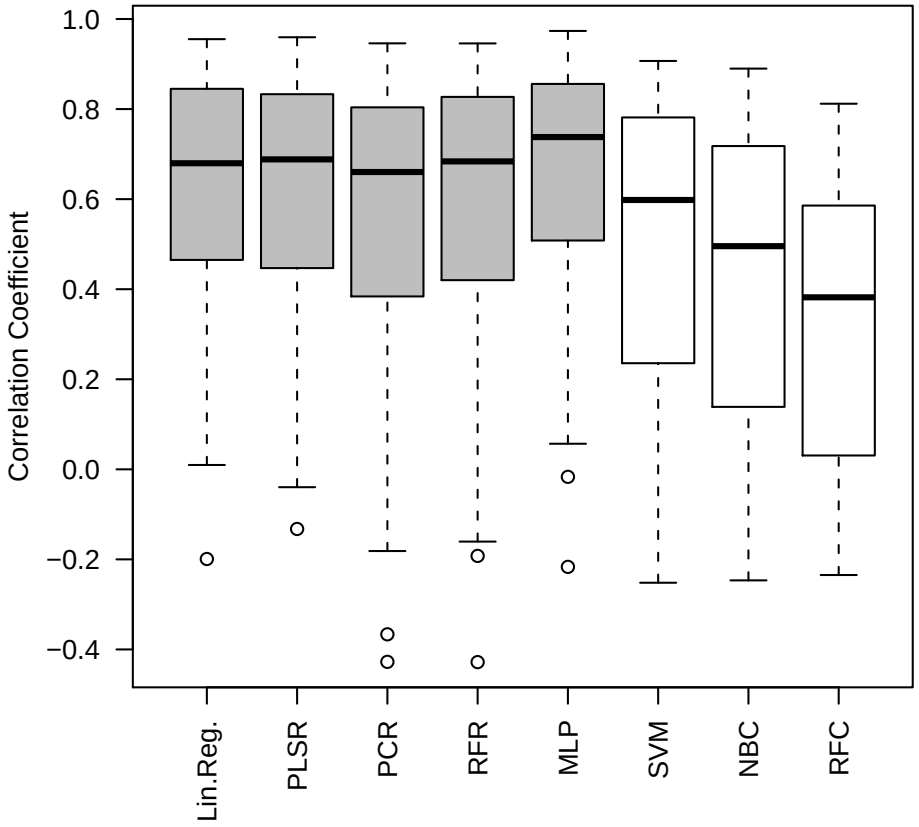

Figure 8.4 Box plot of the correlation coefficients for each statistical tool. Each correlation coefficient is computed between the scores of the patches of one protein and their real Ω. Each statistical tool is characterized by 180 correlation coefficients, corresponding to the 180 proteins used for the validation of the PBSL. Coefficients for regressions are shown in gray and coefficients for supervised classification are shown in white.

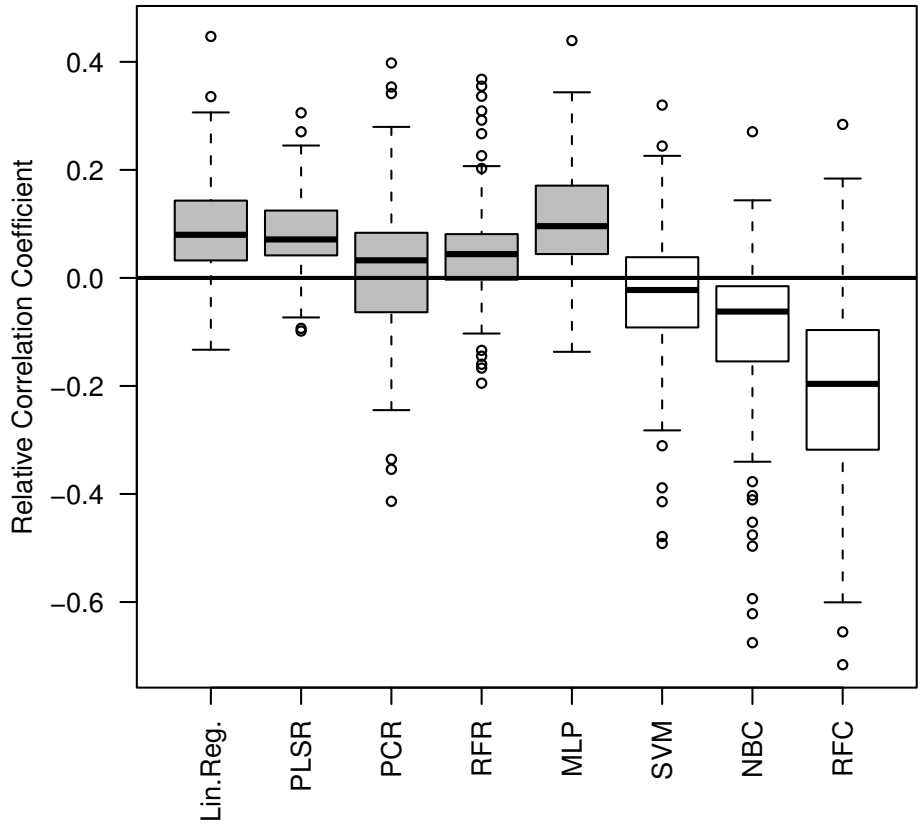

Figure 8.5 Box plot of the relative correlation coefficients for each statistical tool. The relative correlation coefficients are positive if the statistical tool gives better prediction than average. Coefficients for regressions are shown in gray and coefficients for supervised classification are shown in white.

within the rest of the protein surface. The score was weighted by the area of the targeted zones. Ten bins histograms are presented in Figure 8.6. A histogram is given when the exact Ω's are taken as the Ω_{pred}, another one represents the expected values for a random method, and each other one corresponds to a specific statistical tool. Red hatched bars represent the proportion of score for surface points belonging to the binding site and green plain bars for points belonging to the rest of the surface. An ideal histogram would show 100% of the observation in the first bin for the non-site parts and 100% of the observations in the last bin for the site parts. In the histograms corresponding to classifications (RFC, SVM, NBC), many binding site parts have a small score, what leads to many false negative results. On the other hand, in the histograms corresponding to regressions (Lin . Reg., RFR, MLP, PLSR, PCR), as for the "Exact" histogram, the repartition is more distinct between site and non-site zones, what makes the prediction more accurate.

Binding Site Prediction The location of potential binding site is predicted by applying a threshold on the score map. Then, the results vary with the chosen threshold and can be adapted to limit the false positive or false negative rates. This method was tested using different statistical tools. A Precision-Recall graph comparing results for all these tools appears in Figure 8.7. Curves corresponding to regression tools generally are above the one corresponding to classification tools, what is consistent with the previous observations. In this experiment, the size of the predicted binding site was arbitrarily set to 1/10 of the whole surface of the protein, in other words, points with the highest score were added to the predicted binding site until it attained 1/10 of the total surface.

Results were compared to the expected indices obtained via random selection of the predicted binding site. As in this application, the area of the binding site and the area of the predicted binding site are unchanged for one protein, the expected values of both indices are estimated as follows:

$$E(Sens) = E\left(\frac{S(P \cap B)}{S(B)}\right) = \frac{E\left(S(P \cap B)\right)}{S(B)} = \frac{S(B)\frac{S(P)}{S_T}}{S(B)} = \frac{S(P)}{S_T},$$

$$E(PPV) = E\left(\frac{S(P \cap B)}{S(P)}\right) = \frac{E\left(S(P \cap B)\right)}{S(P)} = \frac{S(P)\frac{S(B)}{S_T}}{S(P)} = \frac{S(B)}{S_T},$$

where S_T is the whole surface area. $S(B)$ and $S(P)$ are considered to be constant. Thus, in these formulas, the only variable is the overlap between P and B. The expected proportion of P inside B is the same as anywhere else on the surface. Then, $E\left(S(P \cap B)\right)$ is this proportion weighted by $S(B)$. The reasoning is the same for the proportion of B inside P. To quantify the success rate of the PBSL with each statistical tool, the percentage of proteins of the data set with a higher index than expected (via random selection) was calculated (Table 8.1). As for other tests, the results are better for regressions than for classifications. In 84% of the cases, the MLP gave a

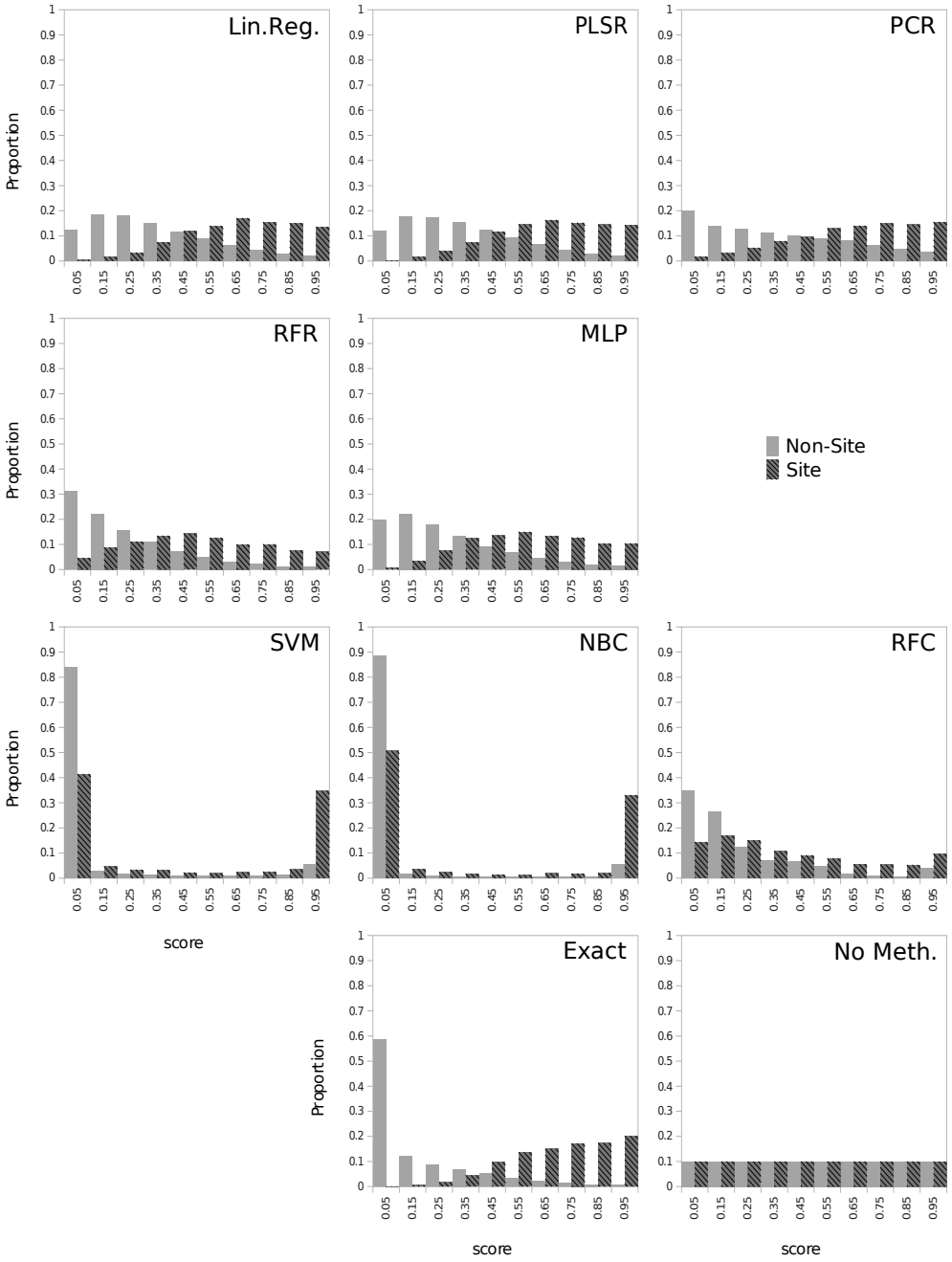

Figure 8.6 Histograms of the score repartition inside the real binding site (darker gray hatched bars) and on the rest of the molecular surface (lighter gray bars). The ideal histogram would be a single light gray bar on the far left side and a single dark gray hatched bar on the far right side. The "Exact" histogram (lower center) is obtained by considering the real patches Ω as the Ω_{pred}. The resulting error comes from the patches approximation only. The "No Meth." histogram (lower right) represents the expected histogram obtained if no binding site prediction method is used. Other histograms are for the different statistical tools.

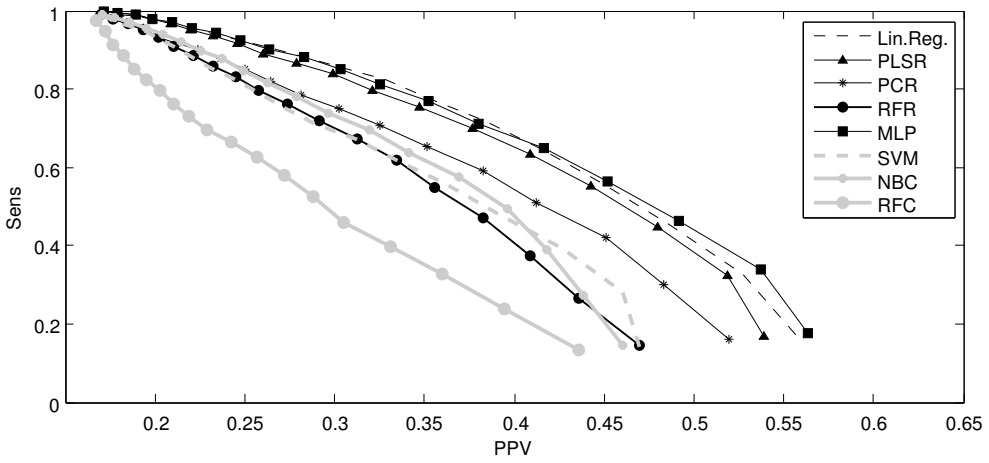

Figure 8.7 Precision-recall curves comparing the results obtained with different statistical models. The y-axis represents the mean sensitivity (or the precision) over the 180 proteins and the x-axis represents the mean PPV (or Recall). Curves corresponding to regression tools are depicted with thin black lines and curves corresponding to classification tools are depicted with thick gray lines.

better result than the expected value, whereas the best classification tool (NBC) gave a result of 78%.

The tool giving the best results was the MLP with a mean sensitivity and PPV of 0.34 and 0.54, respectively. A molecule with a predicted binding site representing approximately these mean values is depicted in Figure 8.8.

Comparison with Other Methods Finally, the PBSL was compared to other methods for which applications are available on the web: Cons-PPISP [129], PINUP [132] and Sharp^2 [131]. These three methods return a score for each residue. These scores were mapped on the protein surfaces and the binding site locations were predicted as was done for the scores resulting from the different statistical models. The tests were performed on the second data set (See Supplementary Materials 2). A Precision-Recall graph comparing results for all these methods appears in Figure 8.9. Performances of the PBSL using MLP are higher than those of Sharp^2 method and comparable to those of the PINUP and Cons-PPISP methods. The percentage of proteins of the data set with a higher index than expected (via random selection) was also calculated (Table 8.2). For both the PBSL using MLP and the PINUP method, the result was better than expected in 71% of the cases. These results are a bit worse than for the other data set, probably because the training data set was made of proteins from bounded structures.

Statistical Tool	Success Rate
Exact	100
Lin. Reg.	81.67
PLSR	82.78
PCR	76.67
RFR	77.78
MLP	**83.89**
SVM	75.00
NBC	77.78
RFC	69.44

Table 8.1 Success rate for different statistical tools: percentage of proteins in the whole data set resulting on a PPV higher than the expected value for each tool.

Method	Success Rate
MLP	**71.43**
Cons-PPISP	69.70
PINUP	**71.43**
Sharp^2	62.86

Table 8.2 Success rate for different methods: percentage of proteins in the whole data set resulting on a PPV higher than the expected value for each method.

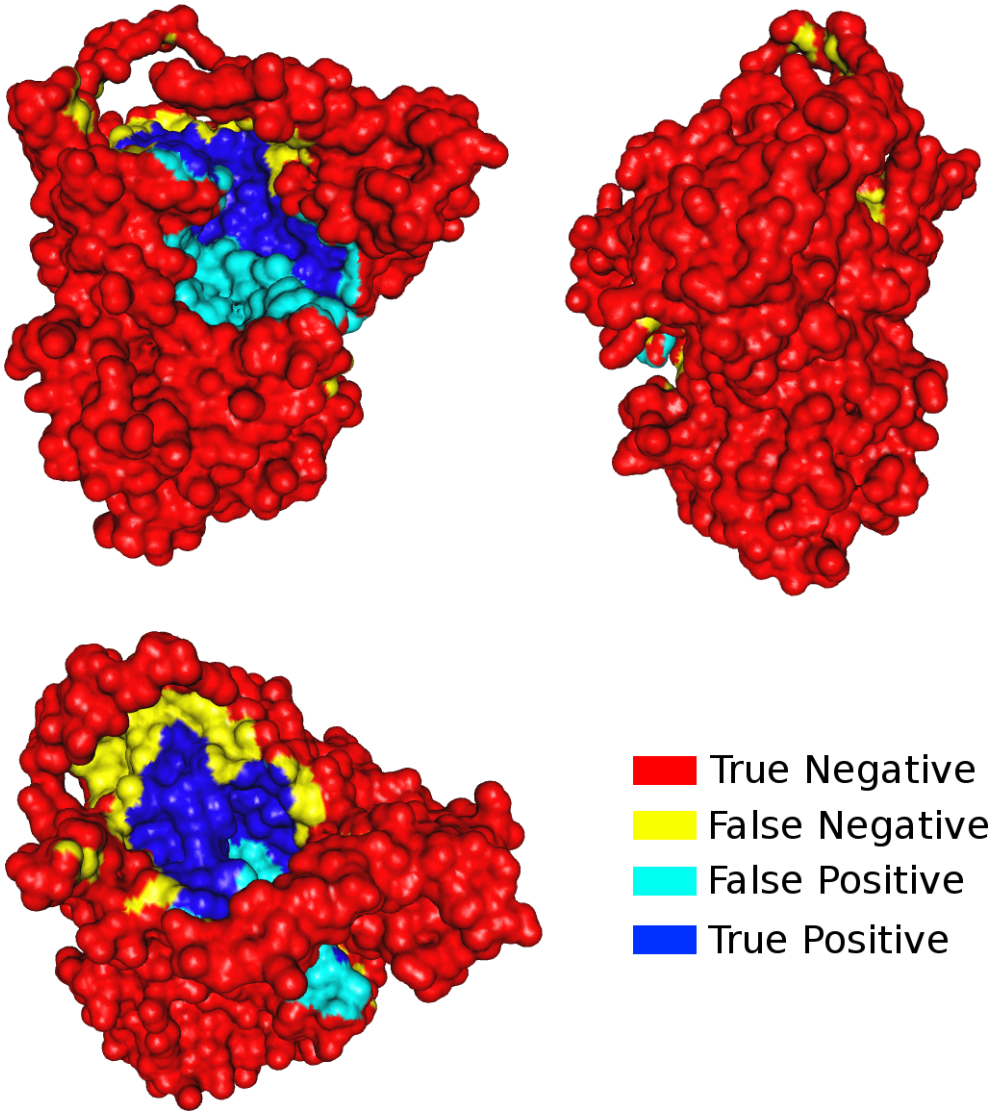

Figure 8.8 Three views of protein SES with non labeled zones (red), zones belonging to the real binding site and not to the predicted one (yellow), zones belonging to the predicted binding site and not to the real one (light blue), and well predicted binding site zones (medium blue). In summary, the predicted binding site is the union of the light blue and the medium blue zone, and the real binding site is the union of the yellow and the medium blue zones. Here, the predicted binding site has a sensitivity=0.44 and a PPV=0.54, which represent approximately the mean results using the MLP.

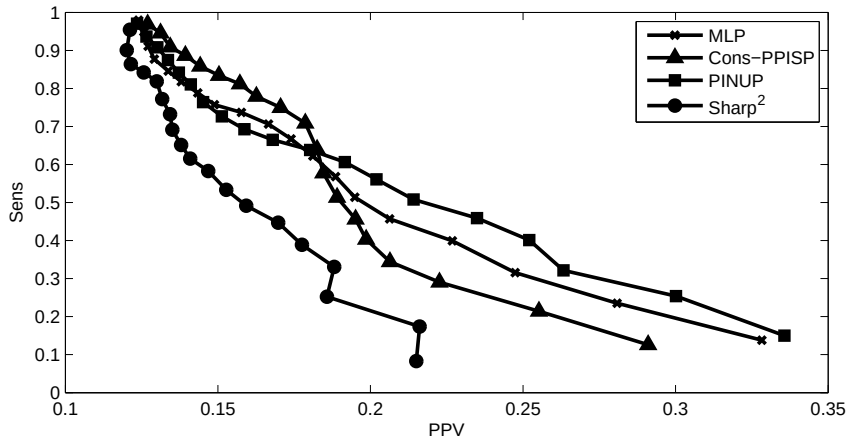

Figure 8.9 Precision-recall curves comparing the results obtained with different models. The y-axis represents the mean sensitivity (or the precision) over the 180 proteins and the x-axis represents the mean PPV (or Recall). The MLP curve (line with crosses) is obtained using the PBSL with a Multilayer Perceptron.

8.3.2 Epitopes

The second application of the PBSL is the detection of epitopes on antigens surfaces. This second study is mainly focused on the variables selection and on the prediction results obtained on this protein family of great interest. The identification of relevant variables, the backward variable selection and the comparison of the PBSL with web server tools were performed on three data sets. The only statistical tool used in this study is the MLP because it was associated to the best results in the first study (Section 8.3.1). The data sets and the results associated to these three steps are presented in this section.

8.3.2.1 Data Sets

The first two data sets contain bound antibody-antigen complexes whereas the third one contains unbound structures.

The first data set is **Discotope**. It was created by Andersen et al. [152] in 2006 and contains 75 antigens grouped into 25 non-homologous families.

The second one is a benchmark data set of 59 epitopes published by **Pono-marenko** et al. [166].

The third data set is the one of **Liang** et al. [167]. This data set is different from the first two because it contains unbound antigens, which is closer to the reality of the application. Indeed, when a new antigen is discovered and has to be studied, there is no guarantee that it is in its bound conformation and that it will not be deformed when interacting with an antibody. The data set is divided into a training part (19 antigens) and a testing part (16 antigens). However, the data set used in this study

is slightly different from the original one. First, the 3BQU(bound)/2F5A(unbound) was removed from the set because it only contains antibody chains. Second, in the 2J6E(bound)/2DTQ(unbound), only the chain A was considered because A and B are identical with very similar epitopes.

8.3.2.2 Variables Selection

Variables Relevance Score Calculation Both relevance scores, the $-\log(\text{p-value})$ of the t-test and the absolute value of Pearson correlation coefficient were computed for the three data sets. Results are displayed for the Discotope data set in Figure 8.10. For both score, the mean of a property was generally more discriminant than the standard deviation except for the hydrogen bond directionality (hbdm and hbdsd). The most discriminant variable was the depth (deptm) followed by the curvature (curvm) and the density (densm), the three geometric properties. Among the composition variables, the proportions of glycine (propG), glutamate (propE), phenylalanine (propF), and valine (propV) were associated with high relevance scores for both indices.

The relevance scores associated to the properties computed on the antigens of the Ponomarenko data set are displayed in Figure 8.11. In contrast with the Discotope data set, the density was less relevant, whereas the conservation score (consm) was more relevant. Among the composition variable, the proportions of cysteine (propC), glycine (propG), and valine (propV) were among the most discriminant.

The relevance scores for the Liang data set are displayed in Figure 8.12. The geometric properties were less discriminant than for the first two data sets. The conservation (consm) was particularly relevant according to the second score. The proportion of glycine (propG) and valine (propV) were again the most relevant composition variables followed by the proportion of leucine (propL) and glutamine (propQ).

Both relevance scores were also computed for the 544 physicochemical properties derived from the propensity scales of the AAindex database [116]. Among these properties, none was associated to a relevance score higher than the score associated to high ranked composition variables. Thus no physicochemical properties derived from a propensity scale were included because they were redundant with the composition variables and they were not more discriminant.

Backward Variable Selection The greedy backward selection of the variable with the MLP1 was applied on each data set. For the Discotope data set, the final variable set contained two geometric variables (deptm and curvsd), one conservation variable (consm) and four chemical variables (hbdsd, propA, propC, propG and propV). See the legend of Figure 8.10 for the meaning of the abbreviations. Results for the two other data sets are quite similar. The final set obtained for the Ponomarenko contained eight variables (deptm, curvm, curvsd, hydm, consm, propC, propG and

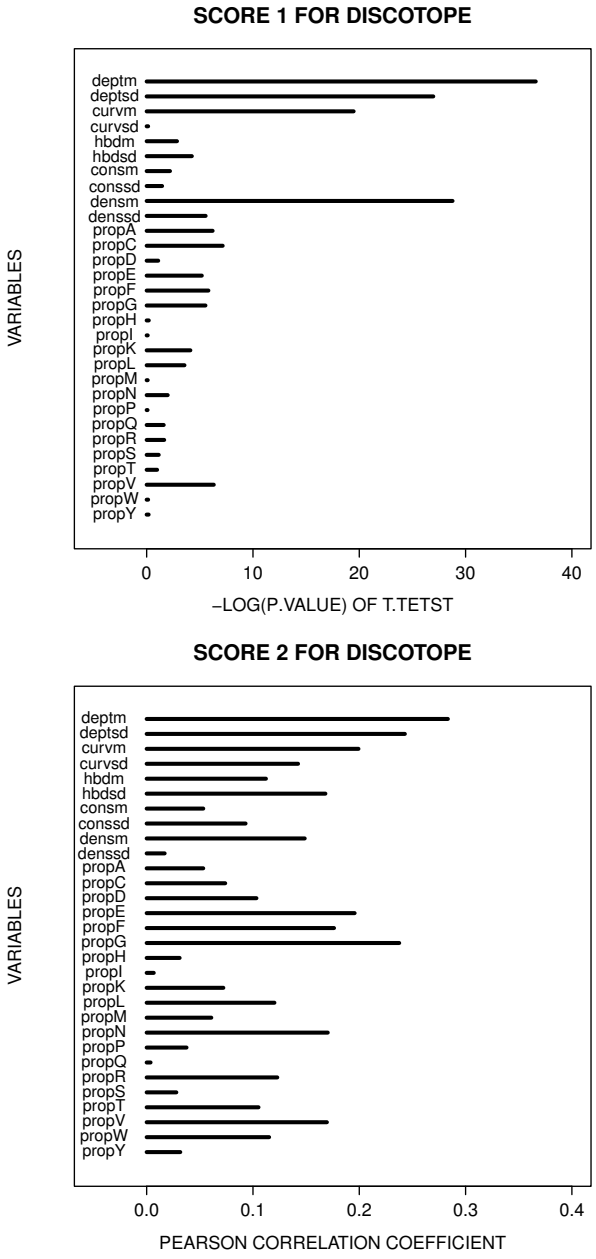

Figure 8.10 Relevance scores of the variables for the Discotope data set. The first relevance score results from a t-test performed for each variable on patches perfectly overlapping the epitope and those sharing no common surface with it. The $-\log(\text{p-value})$ of the t-test is the first score. The second relevance score is the absolute value of the correlation between a variable and the overlapping index. The suffixes "m" and "sd" of the names on the left stand respectively for *mean* and *standard deviation*. "dept" stands for *travel depth*, "curv" for *mean curvature*, "hbd" for *hydrogen bonds directionality*, "cons" for *conservation score* and "dens" for *density*. "porpX" stands for the *proportion of the amino acid X* where X is the one-letter code of the amino acid

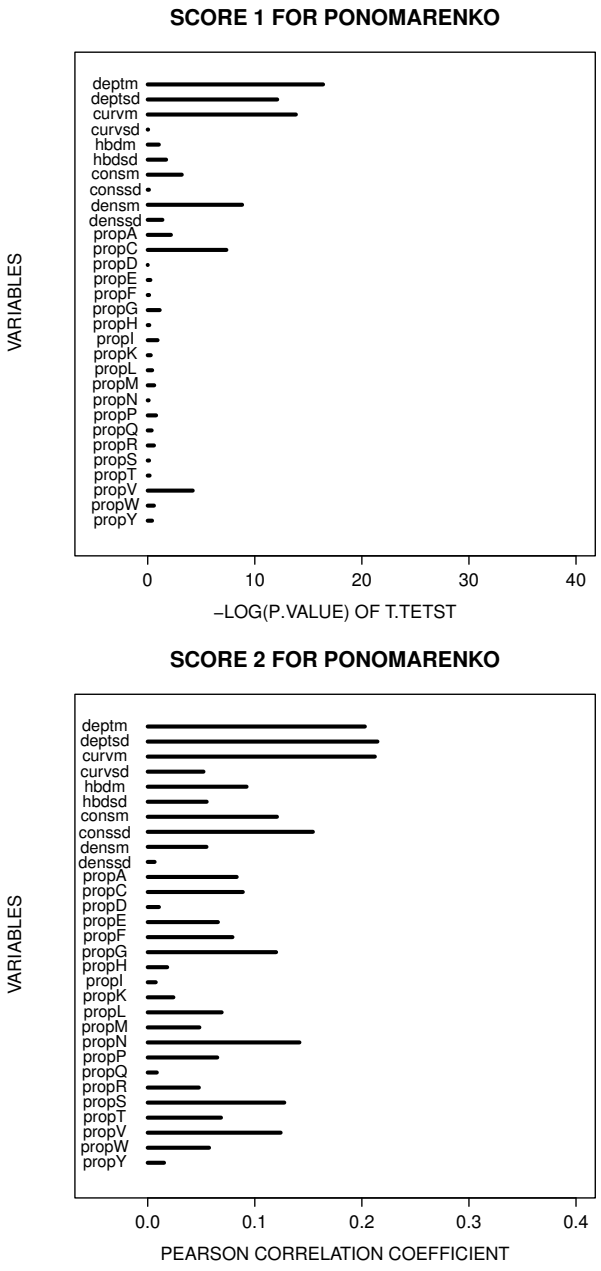

Figure 8.11 Relevance scores of the variables for the Ponomarenko data set.

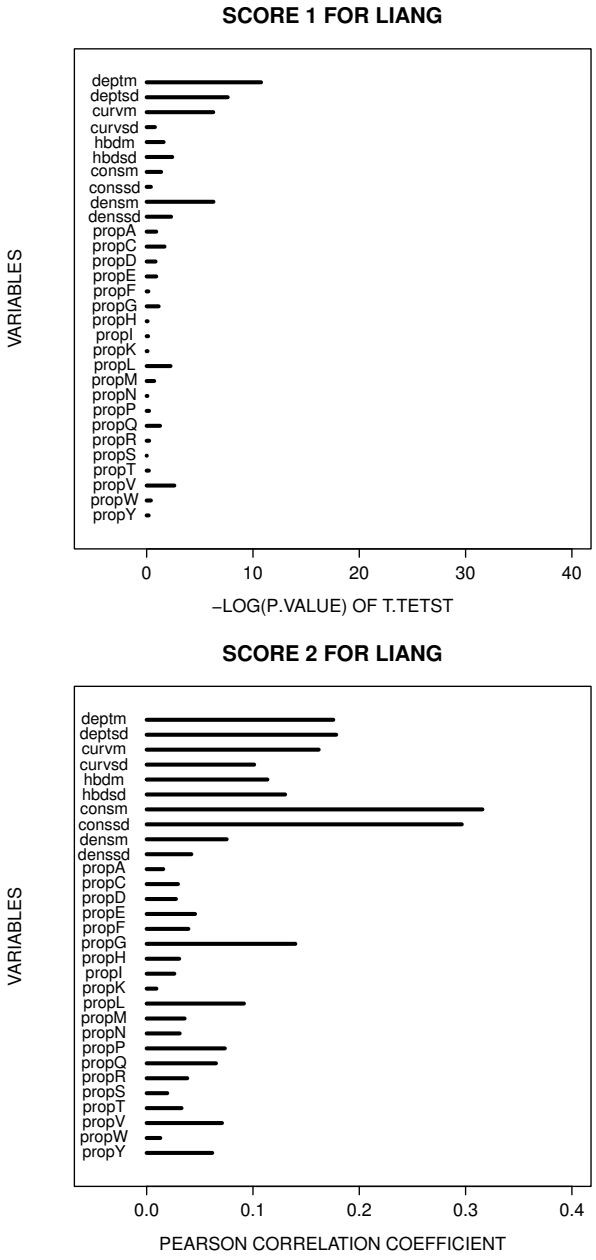

Figure 8.12 Relevance scores of the variables for the Liang data set.

propV) and those of Liang seven variable (deptm, curvsd, hbdm, consm, propA, propG and propV).

After the variables selection step, a leave one out cross-validation was performed and the predicted overlapping index Ω_{pred}'s were used to predict the location of the epitopes.

8.3.2.3 Comparison with Web-Server Tools

Comparisons of performance were done on the three data sets. The comparisons are based on the success indices reported in the literature for different methods. In these methods, a score is affected to each residue, reflecting its probability to be a part of the epitope and residues are added to the predicted epitope in decreasing order of score. Here, the index compared between methods is the area under curve (AUC) of the *receiver operating characteristics* (ROC). The ROC is the plot of the *true positive rate* (TPR) vs. the *false positive rate* (FPR), defined as:

$$TPR = \frac{TP}{P},$$

$$FPR = \frac{FP}{N},$$

where TP is the number of *true positive* residues, that is, the number of residues predicted to be a part of the epitope which are actually in the epitope, P (*positive*) is the number of residues in the epitope, FP is the number of *false positive* residues, and N (*negative*) is the number of residues out of the epitope. The AUC takes a value between 0 and 1 with an expected value of 0.5 for random ranking.

Discotope Data Set The results of the PBSL were compared with the results of Andersen et al. [152] and with the results of BePro [153] and COBEPro [144]. Detailed comparisons with BePro grouped by family appear in Table 8.3.

The mean AUC of the families was 0.75 for BePro, 0.60 for COBEpro, 0.72 for Discotope, and 0.75 for the PBSL (see Table 8.6).

Ponomarenko Data Set Ponomarenko et al. [166] compared a series of open access tools on their benchmark data set. These tools are general binding site predictors, epitopes predictors, docking methods, or simply a property calculator. The PBSL was compared with the specific epitopes predictors, which produced the best results excepting the docking methods, for which the other molecule is known. Results appear in Table 8.4. The mean AUC was 0.66 for Discotope, 0.53 for CEP, 0.54 for Consurf, and 0.69 for the PBSL (see Table 8.6). However, some antigens of this data set also appear in the Discotope training data set. When removing these antigens from the list, the mean AUC was 0.63 for Discotope and 0.75 for the PBSL.

Family representative	PBSL	BEPro
1a14:N	0.65	**0.9**
1ar1:B	**0.94**	0.61
1cz8:W	**0.97**	0.92
1egj:A	0.85	**0.88**
1eo8:A	**0.54**	0.31
1ezv:E	0.81	**0.83**
1fj1:F	0.4	**0.76**
1fsk:A	0.87	**0.88**
1g7l:C	**0.75**	0.7
1g9m:G	**0.53**	0.48
1h0d:C	**0.95**	0.63
1iqd:C	**0.89**	0.82
1jrh:I	0.52	**0.66**
1k4c:C	**0.85**	0.72
1lk3:A	**0.82**	0.8
1mhp:A	0.78	**0.86**
1nfd:B	0.7	**0.96**
1oaz:A	0.34	**0.63**
1ors:C	**0.98**	0.63
1ots:A	**0.94**	0.88
1tqc:A	0.55	**0.61**
1xiw:A	**0.91**	0.9
2hmi:B	0.44	**0.95**
2jel:P	**0.94**	0.71
2vdn:A	0.8	**0.86**
Mean	**0.75**	**0.75**

Table 8.3 Mean AUC of the ROC for BEPro and the PBSL on the Discotope data set. A row represents a family of antigen and only the first member of this family (in alphabetical order) is indicated here.

PDB ID	PBSL	CEP	Discotope	Consurf
1a14:N	0.73	0.47	**0.87**	0.76
1afv:A	0.6	**0.63**	0.53	0.41
1ar1:B	**0.72**	0.49	0.57	0.64
1bgx:T	0.73	0.52	**0.74**	0.5
1bvk:C	0.39	0.46	**0.66**	0.61
1e6j:P	**0.76**	0.55	0.23	0.39
1egj:A	0.57	0.61	**0.88**	-
1eo8:A	0.61	0.48	0.27	**0.65**
1ezv:E	0.83	0.47	**0.85**	0.64
1fe8:B	**0.95**	0.6	0.81	0.67
1fj1:F	**0.81**	0.54	0.68	0.5
1fns:A	0.56	0.44	**0.92**	0.57
1fsk:A	**0.95**	0.5	0.82	0.33
1g9m:G	**0.55**	0.53	0.44	0.48
1h0d:C	**0.86**	0.55	0.5	0.51
1i9r:A	**0.87**	0.48	0.71	0.74
1iqd:C	**0.87**	0.45	0.78	0.81
1jhl:A	0.58	0.39	**0.79**	0.73
1jps:T	0.23	0.54	0.62	**0.72**
1jrh:I	0.51	0.53	**0.62**	-
1ken:A	0.67	0.51	**0.76**	0.62
1lk3:A	0.71	0.52	**0.81**	0.76
1mhp:B	0.74	0.52	**0.81**	0.42
1n8z:C	0.52	0.57	**0.59**	0.46
1nca:N	**0.87**	0.47	0.84	0.69
1ndg:C	**0.79**	0.57	0.74	0.65
1nfd:D	0.49	0.55	**0.88**	0.71
1nl0:G	0.63	**0.78**	0.61	-
1nsn:S	0.43	0.41	0.58	**0.78**
1oaz:A	**0.93**	0.64	0.61	0.25
1orq:C	0.5	0.6	0.48	**0.61**
1ors:C	**0.82**	0.66	0.5	-
1osp:O	0.28	0.49	**0.76**	0.38
1p2c:C	**0.8**	0.65	0.74	0.57
1pkq:E	**0.92**	0.56	0.71	0.7
1qfu:A	0.23	0.5	0.38	**0.64**
1r0a:B	0.93	0.45	**0.94**	0.54
1r3j:C	0.78	0.52	0.72	**0.81**
1rjl:C	0.54	0.53	**0.73**	0.48
1s78:B	**0.9**	0.5	0.55	0.57
1sy6:A	0.71	0.58	**0.82**	-
1tqb:A	**0.9**	0.41	0.59	0.44
1txv:A	0.25	0.52	0.87	**0.89**
1tzi:V	**0.74**	0.6	0.52	0.55
1v7m:V	0.41	**0.56**	0.47	-
1wej:F	0.66	0.39	0.47	**0.83**
1xiw:A	0.83	0.45	0.87	**0.9**
1xiw:F	**0.9**	0.48	0.59	0.74
1yjd:C	**0.93**	0.52	0.58	0.54
1ynt:F	**0.66**	0.51	0.49	-
1yy9:A	**0.89**	0.48	0.68	0.75
1z3g:A	**0.88**	0.53	0.66	0.59
1za3:R	**0.83**	0.65	0.69	0.77
1ztx:E	**0.74**	0.54	0.47	0.63
2adf:A	**0.81**	0.57	0.53	0.62
2aep:A	0.61	0.52	**0.7**	0.51
2b4c:G	0.44	**0.48**	0.43	0.44
2jel:P	0.42	0.44	0.66	**0.7**
2vit:C	**0.91**	0.54	0.58	0.63
Mean	**0.69**	**0.53**	**0.66**	**0.61**

Table 8.4 Mean AUC of the ROC for different methods on the Ponomarenko data set.

PDB ID	PBSL	Liang
1NEZ:G	0.56	**0.6**
1DOK:A	**0.6**	0.28
1YWH:A	**0.75**	0.62
2GHV:E	**0.75**	0.73
1D4V:A	0.69	**0.72**
2DTQ:AB	**0.94**	0.61
1YG9:A	0.72	**0.87**
2VUA:A	0.71	**0.81**
1KF2:A	0.35	**0.55**
1Z40:A	**0.8**	0.65
1KEX:A	**0.84**	0.74
1OK8:A	**0.67**	0.57
1GX9:A	**0.8**	0.41
2EVW:X	0.3	**0.51**
1EAX:A	0.36	**0.58**
3D87:A	**0.89**	0.59
Mean	**0.67**	**0.61**

Table 8.5 Mean AUC of the ROC for the Liang method and the PBSL on the Liang data set.

Liang Data Set Comparisons of the results of the PBSL with the results of Liang [167] appear in Table 8.5. The mean AUC was 0.61 for Liang, 0.59 for Discotope, 0.60 for BePro, and 0.67 for the PBSL (see Table 8.6). For this data set, only the surface residues were considered.

8.4 Conclusion

In this chapter, a patch-based binding site prediction method based on either classification or regression tools was presented. Patch-based methods presented in the

Data Set	PBSL	Disco.	BEP.	COBEP.	CEP	ConS.	Liang
Disco.	**0.75**	0.72	**0.75**	0.60	-	-	-
Pono.	**0.69 (0.75)**	0.66 (0.63)	-	-	0.53	0.54	-
Liang	**0.67**	0.59	0.60	-	-	-	0.61

Table 8.6 Summary of the mean AUC of the ROC for different data sets: Discotope, Ponomarenko and Liang, and different methods: PBSL, Discotope, BEPro, COBEPro, CEP, ConSurf and Liang. The value in parentheses for the Ponomarenko data set is the mean AUC for antigens that are not in the Discotope training set.

literature generally use classification tools. In this case, a binary classifier is trained to discriminate two types of patches:

- Patches totally superposed with the real binding site.
- Patches with no common surface with the real binding site.

Using regression instead of classification allows the consideration of patches partially overlapping the genuine binding site during the training step. The variable estimated the regression is the overlap between the real binding site and the patches constructed on the protein surface.

In the first study (Section 8.3.1), using leave-one-out cross-validation, we showed that regression tools have better predictive performance than classification ones. As the patches constructed during the application step partially overlap the genuine binding site, the predictions for these patches were generally more correlated with their overlapping index Ω when regression was used. Among regression tools, the Multilayer Perceptron was the most efficient. In 84% of the cases, with the data set 1 (see Appendix A), the method using an MLP for regression, allowed a better prediction than the expected value via random selection. This method used with MLP was also compared with three methods usable through a web server. This method performed better than Sharp^2, which is also a patch-based method, and had equivalent performance than both other methods.

In summary, regression tools appeared to be more efficient than classification tools when included in a new patch-based method, the latter is comparable with existing binding site prediction methods. When possible, using regression instead of classification for other predictors will probably improve the results, not only when patches are used, but every time the output is a continuous variable.

It is interesting to notice that the outstanding variables are the geometric ones (depth and curvature), the conservation score and four amino acid composition variables: the proportion of aspartic acid, glutamic acid, lysine and phenylalanine. The first three are charged amino acids while the phenylalanine is hydrophobic.

In the second study (Section 8.3.2), the PBSL was applied to the particular case of epitopes and also produced good results. The only statistical tool used for this study is the MLP. The method produced equivalent or better predictions than other recent efficient methods regarding the AUC of the ROC. On the first data set (Discotope), only BEPro had equivalent performances. This method, however, uses parameters found on the original Discotope data set. This lack in cross-validation introduces a positive bias on the results. On the second data set (Ponomarenko), the PBSL outperformed all other methods. It still obtained better results when removing data contained in the training data set of Discotope. The high performance of the PBSL compared to previous ones on the third data set (Liang) shows that it is suitable for the prediction of epitope locations on unbound structures, a situation closer to reality.

Regarding the variables selection, the geometric properties were associated with the highest relevance scores especially for the first two data sets. Indeed, epitope are generally located on a prong, i.e. a site with a small depth and a high curvature. For the third data set, relevance scores of the geometric properties were lower; it is probably due to the nature of this data set. Indeed, the structures of the antigens are in their unbound conformation. Therefore, it seems logical that geometric properties are not yet so discriminant. The conservation score was associated with good relevance scores, especially for the last two data sets. It has already been shown by Ponomarenko et al. [166] that the conservation score alone can be used to predict epitopes locations. The glycine and the valine proportions were relevant in the three data sets. They are both small amino acids with a hydrophobic tendency, which is coherent with previous studies [142]. No common points were identified between the other relevant amino acid proportions.

The variable sets selected with the greedy backward selection on the MLP1 were almost the same for the three data sets. All the variables selected with this wrapper technique presented high relevance scores. The absence of the density variable in the three data sets can be explained by its high correlation with the other two geometric properties. The proportion of glycine, valine, alanine and cysteine were also selected. These amino acids are all small and with a hydrophobic tendency.

No actual time comparisons were made because execution times do generally not appear in the reference papers. It is not possible either to deduce those computation times when testing the web-based tools because the whole process comprises the data transfers and a potential queue. The PBSL takes about one minute from the surface generation to the final binding site localization. It is therefore realizable to analyze hundreds of proteins a day. This method could also be parallelized at different levels. First, once the model is trained, each molecule can be treated separately. Second, during the overlapping prediction step, the patches can be treated independently.

Conclusion

9

In this thesis, 3D image processing was used for the analysis of proteins properties. Proteins are organic macromolecules intervening in all the levels of life organization, from the transportation of dietary minerals to the structuration of cells. Understanding how they work by studying their structures and their interactions (protein docking) is essential for applications such as drug design (conception of medication) or diagnosis.

In this work, proteins were mainly modeled as polygonal meshes, considering only the SES. The computation of geometrical features on meshes is generally efficient although some algorithms may be quite time consuming, such as for the geodesic distance. Given the huge number of structures contained in the data bases, there is a need of rapidity in the analysis in order to screen these data bases and to find molecule structures with the required properties for a particular application. As an example, if the side effects of an artificial inhibitor has to be verified on all the non-target regulatory enzymes (more than 1000 structures in the PDB), reducing the compatibility search from 1000 seconds to 10 seconds reduces the total screening time from 12 days to 3 hours. This reduction of computation time would permit to test more different artificial inhibitors.

The objective of this thesis was to design fast algorithms while keeping a sufficient precision or even improving this precision.

9.1 Contributions

9.1.1 Travel Depth

The travel depth is defined as the shortest distance from the convex hull without penetrating the molecule surface. The contribution is the design of a fast and accurate travel depth estimation algorithm (Chapter 6). The calculation times are reduced by considering external surfaces instead of volumes. The algorithm in a hybrid euristic between Euclidean and geodesic distance front propagation, similarly to the Dijkstra algorithm.

9.1.2 Geodesic Distances

Geodesic distances are distances along the surface. The contributions are the design of two algorithms for the fast approximation of the geodesic distance (Chapter 5). The first one is based on the computation of the geodesic distances on a simplified mesh and the second one is based on planar parameterizations around reference points.

9.1.3 Binding Sites Detection

Binding sites detection is the identification of parts of the protein that are potential binding sites. The contribution is the development a method to detect binding sites using surface properties (Chapter 8). This method is based on a classifier or a regression model that takes protein properties as input and returns the location of probable binding sites. Comparisons between different statistical models and tests were performed on data sets containing various binding sites or more specifically, antigens epitopes.

9.1.4 Solvent Excluded Surface

The SES is the frontier between accessible and unaccessible zones for a solvent molecule. The contribution is the conception of a method to compute the solvent excluded surface (Chapter 4). The mesh representing the SES is obtained by filtering an electron density map with a frequency filter after a Fourier transform.

9.1.5 Electron Density

Electron density is a new descriptor of the electronic density taking atom movements into account. The contribution is the design of this descriptor and a method for its computation (Section 7.1.3). This method uses the same kind of filter as the SES calculation.

9.1.6 A Study about Spectral Properties

In that study, a Delaunay triangulation is applied to the atom's positions and the eigenvector of the connectivity matrix of this triangulation is analyzed. It can be useful in the context of binding sites detection or to compute similarity scores between structures.

9.2 Perspectives

Each part of the methods developed in this thesis could be improved separately. This section focuses on the possible future works considering the global application: the use of protein structural properties in order to detect sites of interest and to proceed to protein docking.

The next natural step of this work would be the integration of the method of binding sites detection in a global docking process. The identification of some zones

with a high binding potential could be combined with a complementarity research algorithm and an algorithm of data bases screening in order to find rapidly complementary molecules to a given one. It could also be integrated in drug design applications for the identification of the site where the drug will have to act.

Drugs are generally steric blockers, that is, small molecules physically obstructing the binding site preventing any other interaction. The specificity of such molecules is quite low, so that they also block binding sites of other proteins leading to negative side effects. Therefore, it would be interesting to study the mechanisms of allosteric inhibitors or activators, which are more specific to a particular protein. An allosteric inhibitor or activator is a molecule which is able to modify the 3D conformation of the target protein in order to make unaccessible or accessible a binding site located elsewhere on the surface.

Another interesting perspective is the integration of the binding sites detection in an application that takes dynamics into account. For example, the dynamic agitation of the protein can be simulated and a similar binding sites identification scheme may be used on some sampled conformations. Dynamic models may also be integrated in the binding sites detection scheme to detect zones that are able to open and make a potential binding site accessible.

The binding site detection method could be combined with a structure prediction algorithm in order to proceed to sites detection given the sequence only. A similar perspective is the localization of binding sites on mutated proteins structures. The comparison between binding sites localized on mutated proteins and wild type (i.e. not mutated) proteins would be useful for the understanding of genetic disease symptoms.

List of Publications

Journal Papers

- Joachim Giard, Patrice Rondao Alface, Jean-Luc Gala And Benoît Macq. Fast Surface-Based Travel Depth Estimation Algorithm for Macromolecule Surface Shape Description, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, IEEE computer Society Digital Library, **2009**. [105]
- Joachim Giard, Jérôme Ambroise, Jean-Luc Gala And Benoît Macq. Regression applied to protein binding site prediction and comparison with classification, *BMC Bioinformatics*, 10:276, **2009**. [121]
- Joachim Giard And Benoît Macq. Molecular Surface Mesh Generation by Filtering Electron Density Map, *Hindawi International Journal of Biomedical Imaging*, vol. 2010, Article ID 923780, 9 pages, **2010**. [63]

Proceedings

- Joachim Giard, Patrice Rondao Alface And Benoît Macq. Fast and accurate travel depth estimation for protein active site prediction, *Proceedings of SPIE Image Processing*, San Jose, CA, USA, Vol. 6812, **2008**. [104]
- Joachim Giard And Benoît Macq. Fast Geodesic Distance Approximation using Mesh Decimation and Front Propagation, *Proceedings of SPIE Image Processing*, San Jose, CA, USA, Vol. 7245A, **2009**. [81]
- Joachim Giard And Benoît Macq. Molecules 3D Delaunay Triangulation: a Spectral Study (Poster), *Proceedings of SPIE Medical Imaging*, Orlando, FL, USA, Vol. 7262, **2009**. [168]

Conferences

- Joachim Giard And Benoît Macq. From Mesh Parameterization to Geodesic Distance Estimation, *Euro CG*, Brussels, Belgium, **2009**. [82]
- Joachim Giard, Jérôme Ambroise, Rose Josiane Keumo Tcheuko, Benoît Macq. Predict Protein Atoms Mobility Using Fourier Transform (Poster), *4th IEEE EMBS Benelux Chapter*, University of Twente, NL, **2009**. [112]

Data Set 1 for the Various Binding Site Detection

A

Protein Chains Involved in Non-Obligate Interactions

Enzyme-inhibitors

1a4y_A ribonuclease inhibitor
1a4y_B angiogenin
1ava_A alpha amylase
1avw_A trypsin
1avw_B soybean trypsin inhibitor
1ay7_A ribonuclease Sa
1ay7_B Barstar
1bvn_T tendamistat
1clv_I alpha amylase inhibitor
1cse_I eglin C
1dpj_A proteinase A
1dpj_B proteinase inhibitor Ia3
1dtd_A carboxypeptidase A2
1dtd_B metallocarboxypeptidase inhibitor
1eai_C elastase/chymostrypsin inhibitor I
1f34_B major pepsin inhibitor
1fss_A acetylcholinesterase
1fss_B fasciculin II
1gla_F Glucose-Specific Factor III
1kxq_H antibody Vhh fragment
1mct_I squash seed inhibitor
1smp_A serratia metalloproteinase
1smp_I erwinia chysathemi inhibitor
1tab_I bowman-birk inhibitor
1tgs_I pancreatic secretory trypsin inhibitor
1udi_E uracil-DNA glycosylase
1udi_I UDG inhibitor

1viw_B alpha amylase inhibitor
2ptc_I bovine pancreatic trypsin inhibitor
2sic_E subtilisin BPN
2sic_I SSI
4cpa_I potato carboxypeptidase A inhibitor
4sgb_E serine proteinase B
4sgb_I potato chymotrypsin inhibitor I
7cei_A Im7
7cei_B DNAase

Others

1agr_E Rgs 4
1atn_D Deoxyribonuclease I
1b6c_A Fk506-Binding Protein
1b6c_B Tgf-B Superfamily Receptor Type I
1bkd_S Son Of Sevenless-1
1buh_B Cell cycle regulatory protein Ckshs1
1d2z_B Death Domain Of Tube
1dow_A alpha-catenin
1dow_B beta-catenin A
1eay_A chey
1eay_C chea
1euv_A Ulp1 Protease
1euv_B Ubitquitin-Like Protein Smt3
1f3v_B traf2 2.0A
1f5q_A Cyclin Dependent Kinase 2
1f5q_B Cyclin
1he1_A Exoenzyme S
1hx1_B Bag-Family Molecular Chaperone Regulator-1
1i2m_B Regulator Of Chromosome Condensation
1i8l_C Cytotoxic T-Lymphocyte Protein 4
1kac_B Adenovirus Receptor
1pdk_A Chaperone Protein Papd
1pdk_B Protein Papk
1qav_A alpha-1 Syntrophin
1tx4_A P50-Rhogap
1tx4_B Rhoa
1xdt_R Heparin-Binding Epidermal Growth Factor
1xdt_T Diphtheria Toxin
3ygs_C Apoptotic Protease Activating Factor

3ygs_P Procaspase 9

Protein Chains Involved in Obligate Interactions

Hetero-Dimers

1ahj_A nitrite hydratase
1ahj_B nitrite hydratase
1aht_H alpha thrombin
1aht_L alpha thrombin
1b34_A Small Nuclear Ribonucleoprotein Sm 2
1b34_B Small Nuclear Ribonucleoprotein Sm 2
1bun_A beta-2-Bungarotoxin
1dce_A Rab Geranylgeranyltransferase
1dce_B Rab Geranylgeranyltransferase
1dj7_A ferredoxin thioredoxin
1efv_A electron transfer flavoprotein
1efv_B electron transfer flavoprotein
1g4y_B calcium-activated potassium channel Rsk
1g4y_R calmodulin
1gux_A Retinoblastoma Protein
1gux_B Retinoblastoma Protein
1h2a_L Ferricytochrome-C3 Oxidoreductase
1h2a_S Ferricytochrome-C3 Oxidoreductase
1luc_B Bacterial Luciferase
1pnk_A Penicillin Amidohydrolase
1pnk_B Penicillin Amidohydrolase
1req_A Methylmalonyl-Coa Mutase
1req_B Methylmalonyl-Coa Mutase
1tco_A serine/threonine phosphatase 2B
1tco_B serine/threonine phosphatase 2B
2aai_A ricin
2aai_B ricin

Homo-Dimers

1a0f_B Glutathione S-Transferase
1a4i_A Methylenetetrahydrofolate Dehydrogenase
1a4u_A Alcohol Dehydrogenase
1afr_F delta-9 Stearoyl-Acyl Carrier Protein
1afw_A 3-Ketoacetyl-Coa Thiolase
1aj8_A Citrate Synthase

1ajs_A Aspartate Aminotransferase
1aom_A Nitrite Reductase
1aq6_A L-2-Haloacid Dehalogenase
1at3_A Herpes Simplex Virus Type II Protease
1az3_B Ecorv Endonuclease
1b3a_B Rantes
1b5e_A Deoxycytidylate Hydroxymethylase
1b7b_A Carbamate Kinase
1b8a_A Aspartyl-tRNA Synthetase
1b8j_B Alkaline Phosphatase
1b9m_B Molybdate-Dependent Transcriptional Regulator
1bbh_A cytochrome C
1bft_A Nuclear Factor Nf-kappa-B P65
1bjn_B Phosphoserine Aminotransferase
1bo1_A Phosphatidylinositol Phosphate Kinase
1brm_A Aspartate-Semialdehyde Dehydrogenase
1bw0_B Tyrosine Aminotransferase
1byf_A Polyandrocarpa lectin
1byk_B Trehalose Operon Repressor
1c7n_A Cystalysin
1cli_B Phosphoribosyl-Aminoimidazole Synthetase
1cmb_B Met Apo-Repressor (Metj)
1cnz_A 3-Isopropylmalate Dehydrogenase
1coz_A Glycerol-3-Phosphate Cytidylyltransferase
1cp2_A Nitrogenase Iron Protein
1dor_A Dihydroorotate Dehydrogenase A
1e0b_A Swi6 Protein
1ete_A Flt3 Ligand
1f5m_A Gaf
1f6y_A Iron Sulfur Protein Methyltransferase
1f8r_A L-Amino Acid Oxidase
1gpe_B Glucose Oxidase
1hgx_B Hypoxanthine-Guanine-Xanthine Phosphoribosyltransferase
1hjr_C RuvC resolvase
1hss_A 0.19 alpha-Amylase Inhibitor
1hul_A Interleukin-5
1isa_B Iron(II) Superoxide Dismutase
1jkm_A Brefeldin A Esterase
1kpe_A Protein Kinase C Interacting Protein
1mka_A beta-Hydroxydecanoyl Thiol Ester Dehydrogenase
1msp_A Major Sperm Protein

1nse_A Nitric Oxide Synthase
1nsy_A Nad Synthetase
1one_A Enolase
1pp2_L Phospholipase A2
1pvu_A Pvuii Restriction Endonuclease
1qae_B Extracellular Endonuclease
1qax_A 3-Hydroxy-3-Methylglutaryl-Coenzyme
1qbi_A Soluble Quinoprotein Glucose Dehydrogenase
1qfe_A 3-Dehydroquinate Dehydratase
1qfh_B Actin Binding Protein 120
1qi9_B Vanadium Bromoperoxidase
1qor_B Quinone Oxidoreductase
1qqj_A Fumarylacetoacetate Hydrolase
1qu7_B Methyl-Accepting Chemotaxis Protein
1scf_B Stem Cell Factor
1smt_A Transcriptional Repressor Smtb
1sox_A Sulfite Oxidase
1spu_A Copper Amine Oxidase
1trk_B Transketolase
1vfr_B Nad(P)H: Fmn Oxidoreductase
1vhi_A Epstein Barr Virus Nuclear Antigen-1
1vlt_A Aspartate Receptor
1vok_A Tata-Box-Binding Protein
1vsg_B Variant Surface Glycoprotein
1wgj_A Inorganic Pyrophosphatase
1xik_A Protein R2 Of Ribonucleotide Reductase
1xso_A Cu, Zn Superoxide Dismutase
1ypi_A Triose Phosphate Isomerase (TIM)
1yve_J Acetohydroxy Acid Isomeroreductase
2ae2_A Tropinone Reductase-II
2arc_A Arabinose Operon Regulatory Protein
2gsa_B Glutamate Semialdehyde Aminotransferase
2hdh_A L-3-Hydroxyacyl Coa Dehydrogenase
2hhm_B Inositol Monophosphatase
2nac_B Formate Dehydrogenase
2pfl_B Pyruvate Formate-Lyase
2utg_B Uteroglobin
3tmk_B Thymidylate Kinase
4mdh_A Cytoplasmic Malate Dehydrogenase
5hvp_B HIV-1 Protease

Data Set 2 for the Various Binding Site Detection

B

Complex	Enzyme	Inhibitor
1AVX_A:B	-	Soybean trypsin inhibitor
1AY7_A:B	Barnase	Barstar
1BVN_P:T	-	Tendamistat
1CGI_E:I	-	PSTI
1D6R_A:I	-	BowmanBirk inhibitor
1DFJ_E:I	Ribonuclease A	Rnase inhibitor ACE
1E6E_A:B	Adrenoxin reductase FAD	Adrenoxin
1EAW_A:B	Matriptase	BPTI
1EWY_A:C	Ferredoxin reductase FAD	Ferredoxin
1EZU_C:AB	-	Ecotin
1F34_A:B	Porcine pepsin	Ascaris inhibitor
1HIA_AB:I	Kallikrein	Hirustatin
1MAH_A:F	Acetylcholinesterase	Fasciculin
1TMQ_A:B	alphaamylase 5HP	RAGI inhibitor
1UDI_E:I	UracylDNA glycosylase	Glycosylase inhibitor
2MTA_HL:A	Methylamine dehydrogenase	Amicyanin
2PCC_A:B	Cyt C peroxidase HEM	Cytochrome C HEM
2SIC_E:I	-	Streptomyces subtilisin inhibitor
2SNI_E:I	Subtilisin SOC	Chymotrypsin inhibitor
7CEI_A:B	Colicin E7 nuclease	Im7 immunity protein

Bibliography

[1] C. Yan, D. Dobbs, and V. Honavar, "A two-stage classifier for identifiation of protein-protein interface residues," *BIOINFORMATICS*, vol. 20, no. Suppl. 1, pp. i371–i378, 2004.

[2] A. A. Bogan and K. S. Thorn, "Anatomy of hot spots in protein interfaces," *J. Mol. Biol.*, vol. 280, pp. 1–9, 1998.

[3] M. Prevost, S. J. Wodak, B. Tidor, and M. Karplus, "Contribution of the hydrophobic effect to protein stability: Analysis based on simulations of the ile-96- ala mutation in barnase," *PNAS*, vol. 88, no. 23, pp. 10880–10884, 1991.

[4] L. Young, R. L. Jernigan, and D. G. Covell, "A role for surface hydrophobicity in protein-protein recognition," *Protein Sci.*, vol. 3, pp. 717–729, 1994.

[5] G. Cheng, B. Qian, R. Samudrala, and D. Baker, "Improvement in protein functional site prediction by distinguishing structural and functional constraints on protein family evolution using computational design," *Nucleic Acids Research*, vol. 33, no. 18, pp. 5861–5867, 2005.

[6] M. Ota, K. Kinoshita, and K. Nishikawa, "Prediction of catalytic residues in enzymes based on known tertiary structure, stability profie, and sequence conservation," *J. Mol. Biol.*, vol. 327, pp. 1053–1064, 2003.

[7] R. G. Coleman and K. A. Sharp, "Travel depth, a new shape descriptor for macromolecules: Application to ligand binding," *J. Mol. Biol.*, vol. 362, no. 3, pp. 441–458, 2006.

[8] D. Nelson, A. Lehninger, and M. Cox, *Lehninger Principles of Biochemistry Lecture Notebook*. WH Freeman, 2004.

[9] M. Betts and R. Russell, "Amino-Acid Properties and Consequences of Substitutions," *Bioinformatics for Geneticists: A Bioinformatics Primer for the Analysis of Genetic Data*, p. 311, 2007.

[10] I. Kuntz, "Structure-based strategies for drug design and discovery," *Science*, vol. 257, no. 5073, pp. 1078–1082, 1992.

[11] M. Congreve, C. Murray, and T. Blundell, "Keynote review: Structural biology and drug discovery," *Drug discovery today*, vol. 10, no. 13, pp. 895–907, 2005.

[12] F. Chauveau, C. Pestourie, and B. Tavitian, "Aptamers: selection and scope of applications," *Pathologie-biologie*, vol. 54, no. 4, p. 251, 2006.

[13] T. Hermann and D. Patel, "Adaptive recognition by nucleic acid aptamers," *Science*, vol. 287, no. 5454, p. 820, 2000.

[14] S. Henikoff and J. Henikoff, "Amino acid substitution matrices from protein blocks," *Proceedings of the National Academy of Sciences*, vol. 89, no. 22, p. 10915, 1992.

[15] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman, "Basic local alignment search tool," *Journal of molecular biology*, vol. 215, no. 3, pp. 403–410, 1990.

[16] W. Pearson, "Rapid and sensitive sequence comparison with FASTP and FASTA," *Methods in Enzymology*, vol. 183, pp. 63–98, 1990.

[17] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman, "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs," *Nucleic Acids Research*, vol. 25, no. 17, pp. 3389–3402, 1997.

[18] J. Thompson, D. Higgins, and T. Gibson, "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *Nucleic acids research*, vol. 22, no. 22, pp. 4673–4680, 1994.

[19] R. Edgar, "MUSCLE: multiple sequence alignment with high accuracy and high throughput," *Nucleic acids research*, vol. 32, no. 5, p. 1792, 2004.

[20] C. Notredame, D. Higgins, and J. Heringa, "T-COFFEE: A novel method for fast and accurate multiple sequence alignment," *Journal of molecular biology*, vol. 302, no. 1, pp. 205–217, 2000.

[21] I. Shindyalov, "Protein structure alignment by incremental combinatorial extension (CE) of the optimal path," *Protein Engineering Design and Selection*, vol. 11, no. 9, pp. 739–747, 1998.

[22] L. Holm and C. Sander, "Protein Structure Comparison by Alignment of Distance Matrices," *JOURNAL OF MOLECULAR BIOLOGY*, vol. 233, pp. 123–123, 1993.

[23] I. Wohlers, L. Petzold, F. Domingues, and G. Klau, "PAUL: protein structural alignment using integer linear programming and Lagrangian relaxation," *BMC Bioinformatics*, vol. 10, no. Suppl 13, p. P2, 2009.

[24] D. Baker and A. Sali, "Protein structure prediction and structural genomics," *Science*, vol. 294, no. 5540, pp. 93–96, 2001.

[25] R. Bonneau, J. Tsai, I. Ruczinski, D. Chivian, C. Rohl, C. Strauss, and D. Baker, "Rosetta in CASP4: progress in ab initio protein structure prediction," *Proteins: Structure, Function, and Genetics*, vol. 45, 2001.

[26] K. Ginalski, "Comparative modeling for protein structure prediction," *Current opinion in structural biology*, vol. 16, no. 2, pp. 172–177, 2006.

[27] H. Berman, J. Westbrook, G. G. Z. Feng, T. Bhat, H. Weissig, I. Shinkdyalov, and P. E.Bourne, "The protein data bank," *Nucleic Acid Res.*, vol. 28, pp. 235–242, 2000.

[28] F. J. Burkowski, *Structural Bioinformatics: An Algorithmic Approach*. Chapman & Hall/CRC, 2008.

[29] M. L. Connolly, *Molecular surfaces: A review*. Network Science, 1996.

[30] C. Camacho and S. Vajda, "Protein–protein association kinetics and protein docking," *Current opinion in structural biology*, vol. 12, no. 1, pp. 36–40, 2002.

[31] N. Brooijmans and I. Kuntz, "Molecular recognition and docking algorithms," *Annu. Rev. Biophys. Biomol. Struct.*, vol. 32, pp. 335–373, 2003.

[32] S. Bespamyatnikh, V. Choi, H. Edelsbrunner, and J. Rudolph, "Accurate protein docking by shape complementarity alone," *Manuscript, Duke Univ., Durham, NC*, 2004.

[33] E. Brigham and C. Yuen, "The fast Fourier transform," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 8, no. 2, pp. 146–146, 1978.

[34] E. Katchalski-Katzir, I. Shariv, M. Eisenstein, A. Friesem, C. Aflalo, and I. Vakser, "Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques," *Proceedings of the National Academy of Sciences*, vol. 89, no. 6, pp. 2195–2199, 1992.

[35] H. Gabb, R. Jackson, and M. Sternberg, "Modelling protein docking using shape complementarity, electrostatics and biochemical information," *Journal of Molecular Biology*, vol. 272, no. 1, pp. 106–120, 1997.

[36] M. J. E. Sternberg and G. Moont, "Modelling protein-protein and protein-dna docking," *Bioinformatics - From Genomes to Drugs*, pp. 361–404, 2004.

[37] I. Vakser, "Protein docking for low-resolution structures," *Protein Engineering Design and Selection*, vol. 8, no. 4, pp. 371–378, 1995.

[38] R. Chen, L. Li, and Z. Weng, "ZDOCK: an initial-stage protein-docking algorithm," *PROTEINS: Structure, Function, and Genetics*, vol. 52, no. 1, 2003.

[39] B. B. Goldman and W. T. Wipke, "Qsd: quadratic shape descriptors. 2. molecular docking using quadratic shape descriptors (qsdock)," *Proteins*, vol. 38, pp. 79–94, 2000.

[40] H. Lenhof, *Bioinformatics: From Nucleic Acids and Proteins to Cell Metabolism*, ch. An algorithm for the protein docking problem, pp. 125–139. Wiley, 1995.

[41] M. L. Connolly, "Analytical molecular surface calculation," *J. Appl. Crystallogr.*, vol. 6, pp. 548–558, 1983.

[42] D. Fischer, S. Lin, H. Wolfson, and R. Nussinov, "A geometry-based suite of moleculardocking processes," *Journal of molecular biology*, vol. 248, no. 2, pp. 459–477, 1995.

[43] Y. Lamdan and H. Wolfson, "Geometric hashing: A general and efficient model-based recognition scheme," in *Second International Conference on Computer Vision*, pp. 238–249, 1988.

[44] M. Mitchell, *An introduction to genetic algorithms*. The MIT press, 1998.

[45] N. Metropolis and S. Ulam, "The monte carlo method," *Journal of the American Statistical Association*, pp. 335–341, 1949.

[46] J. Bryngelson, J. Onuchic, N. Socci, and P. Wolynes, "Funnels, pathways and the energy landscape of protein folding: a synthesis," *Proteins: Struct. Funct. Genet.*, vol. 21, pp. 167–195, 1995.

[47] S. Yang, S. Cho, Y. Levy, M. Cheung, H. Levine, P. Wolynes, and J. Onuchic, "Domain swapping is a consequence of minimal frustration," *Proc. Natl. Acad. Sci. USA*, vol. 101, no. 38, pp. 13786– 13791, 2004.

[48] M. Teodoro, G. Phillips Jr, and L. Kavraki, "Molecular docking: a problem with thousands of degrees of freedom," *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 1, pp. 960–965, 2001.

[49] B. Sandak, R. Nussinov, and H. J. Wolfson, "A method for biomolecular structural recognition and docking allowing conformational flexibility," *J. Comput. Biol*, vol. 5, pp. 631–654, 1998.

[50] N. Andrusier, E. Mashiach, R. Nussinov, and H. Wolfson, "Principles of flexible protein-protein docking," *Proteins*, vol. 73, no. 2, pp. 271–289, 2008.

[51] G. Bartlett, C. Porter, N. Borkakoti, and J. Thornton, "Analysis of catalytic residues in enzyme active sites," *Journal of molecular biology*, vol. 324, no. 1, pp. 105–121, 2002.

[52] R. Kosloff, "Time-dependent quantum-mechanical methods for molecular dynamics," *The Journal of Physical Chemistry*, vol. 92, no. 8, pp. 2087–2100, 1988.

[53] S. Marrink, H. Risselada, S. Yefimov, D. Tieleman, and A. de Vries, "The MARTINI force field: coarse grained model for biomolecular simulations," *Journal of Physical Chemistry B-Condensed Phase*, vol. 111, no. 27, pp. 7812–7824, 2007.

[54] V. Hornak, R. Abel, A. Okur, B. Strockbine, A. Roitberg, and C. Simmerling, "Comparison of multiple Amber force fields and development of improved protein backbone parameters," *Proteins: Struct Funct Bioinf*, vol. 65, pp. 712–725, 2006.

[55] W. Smith and T. Forester, "DL_POLY_2. 0: A general-purpose parallel molecular dynamics simulation package," *Journal of molecular graphics*, vol. 14, no. 3, pp. 136–141, 1996.

[56] S. Mayo, B. Olafson, and W. Goddard III, "DREIDING: a generic force field for molecular simulations," *The Journal of Physical Chemistry*, vol. 94, no. 26, pp. 8897–8909, 1990.

[57] F. Momany and R. Rone, "Validation of the general purpose QUANTA® 3. 2/CHARMm® force field," *Journal of Computational Chemistry*, vol. 13, no. 7, pp. 888–900, 1992.

[58] W. Cornell, P. Cieplak, C. Bayly, I. Gould, K. Merz, D. Ferguson, D. Spellmeyer, T. Fox, J. Caldwell, and P. Kollman, "A second generation force field for the simulation of proteins, nucleic acids, and organic molecules," *Journal of the American Chemical Society*, vol. 117, no. 19, pp. 5179–5197, 1995.

[59] A. Jaramillo and S. Wodak, "Computational Protein Design Is a Challenge for Implicit Solvation Models," *Biophysical Journal*, vol. 88, no. 1, pp. 156–171, 2005.

[60] J. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. Skeel, L. Kale, and K. Schulten, "Scalable molecular dynamics with NAMD," *Journal of Computational Chemistry*, vol. 26, no. 16, p. 1781, 2005.

[61] M. Levitt, C. Sander, and P. Stern, "Protein normal-mode dynamics: trypsin inhibitor, crambin, ribonuclease and lysozyme," *J. Mol. Biol*, vol. 181, no. 3, pp. 423–447, 1985.

[62] F. Tama and Y. Sanejouand, "Conformational change of proteins arising from normal mode calculations," *Protein Engineering Design and Selection*, vol. 14, no. 1, p. 1, 2001.

[63] J. Giard and B. Macq, "Molecular Surface Mesh Generation by Filtering Electron Density Map," *International Journal of Biomedical Imaging*, vol. 2010, p. 923780, 2010.

[64] N. Max, "Progress in scientific visualization," *The Visual Computer*, vol. 21, no. 12, pp. 979–984, 2005.

[65] C. Bajaj, V. Pascucci, A. Shamir, R. Holt, and A. Netravali, "Dynamic maintenance and visualization of molecular surfaces," *Discrete Applied Mathematics*, vol. 127, no. 1, pp. 23–51, 2003.

[66] P. Laug and H. Borouchaki, "Molecular Surface Modeling and Meshing," *Engineering with Computers*, vol. 18, no. 3, pp. 199–210, 2002.

[67] M. Sanner and A. Olson, "Reduced Surface: An Efficient Way to Compute Molecular Surfaces ," *Biopolymers*, vol. 38, pp. 305–320, 1996.

[68] H. Edelsbrunner and E. Mücke, "Three-dimensional alpha shapes," in *Proceedings of the 1992 workshop on Volume visualization*, pp. 75–82, ACM New York, NY, USA, 1992.

[69] D. Kim, J. Seo, D. Kim, J. Ryu, and C. Cho, "Three-dimensional beta shapes," *Computer-Aided Design*, vol. 38, no. 11, pp. 1179–1191, 2006.

[70] J. Ryu, R. Park, and D.-S. Kim, "Molecular surfaces on proteins via beta shapes," *Computer-Aided Design*, vol. 39, no. 12, pp. 1042–1057, 2007.

[71] H.-L. Cheng and X. Shi, "Quality mesh generation for molecular skin surfaces using restricted union of balls," *Comput. Geom.*, vol. 42, no. 3, pp. 196–206, 2009.

[72] Y. Zhang, G. Xu, and C. L. Bajaj, "Quality meshing of implicit solvation models of biomolecular structures," *Computer Aided Geometric Design*, vol. 23, no. 6, pp. 510–530, 2006.

[73] T. Can, C. Chen, and Y. Wang, "Efficient molecular surface generation using level-set methods," *Journal of Molecular Graphics and Modelling*, vol. 25, no. 4, pp. 442–454, 2006.

[74] W. Lorensen and H. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pp. 163–169, ACM New York, NY, USA, 1987.

[75] I. Bloch and H. Maitre, "Fuzzy mathematical morphologies: A comparative study," *Pattern Recognition*, vol. 28, no. 9, pp. 1341–1387, 1995.

[76] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle, "Piecewise smooth surface reconstruction," in *Proceedings of SIGGRAPH*, vol. 94, pp. 295–302, Citeseer, 1994.

[77] D. Zorin, P. Schröder, and W. Sweldens, "Interpolating subdivision for meshes with arbitrary topology," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 189–192, ACM New York, NY, USA, 1996.

[78] E. Pettersen, T. Goddard, C. Huang, G. Couch, D. Greenblatt, E. Meng, and T. Ferrin, "UCSF Chimera-a visualization system for exploratory research and analysis," *Journal of computational chemistry*, vol. 25, no. 13, pp. 1605–1612, 2004.

[79] N. Guex and M. Peitsch, "SWISS-MODEL and the Swiss-Pdb Viewer: an environment for comparative protein modeling," *Electrophoresis*, vol. 18, no. 15, 1997.

[80] W. DeLano, "The PyMOL molecular graphics system. 2002," *USA: DeLano Scientific, San Carlos*, 2002.

[81] J. Giard and B. Macq, "Fast geodesic distance approximation using mesh decimation and front propagation," in *Proceedings of SPIE*, vol. 7245, p. 72450B, 2009.

[82] J. Giard and B. Macq, "From mesh parameterization to geodesic distance estimation," in *EuroCG*, 2009.

[83] O. Sifri, A. Sheffer, and C. Gotsman, "Geodesic-based surface remeshing," in *Proc. 12th International Meshing Roundtable*, pp. 189–199, 2003.

[84] G. Peyré and L. Cohen, "Geodesic Remeshing Using Front Propagation," *International Journal of Computer Vision*, vol. 69, no. 1, pp. 145–156, 2006.

[85] S. Katz, G. Leifman, and A. Tal, "Mesh segmentation using feature point and core extraction," *The Visual Computer*, vol. 21, no. 8, pp. 649–658, 2005.

[86] H. Lee, L. Kim, M. Meyer, and M. Desbrun, "Meshes on Fire," in *Computer Animation and Simulation 2001: Proceedings of the Eurographics Workshop in Manchester, UK, September 2-3, 2001*, Springer, 2001.

[87] S. Liu, X. Jin, C. C. L. Wang, and J. X. Chen, "Water wave animation on mesh surfaces," *Computing in Science and Engg.*, vol. 8, no. 5, pp. 81–87, 2006.

[88] J. Mitchell, D. Mount, and C. Papadimitriou, "The Discrete Geodesic Problem," *SIAM Journal on Computing*, vol. 16, p. 647, 1987.

[89] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[90] R. Kimmel and J. Sethian, "Computing geodesic paths on manifolds," *Proceedings of the National Academy of Sciences*, vol. 95, no. 15, pp. 8431–8435, 1998.

[91] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. Gortler, and H. Hoppe, "Fast exact and approximate geodesics on meshes," *Proceedings of ACM SIGGRAPH 2005*, vol. 24, no. 3, pp. 553–560, 2005.

[92] M. Meyer, M. Desbrun, P. Schroder, and A. Barr, "Discrete differential-geometry operators for triangulated 2-manifolds," *Visualization and Mathematics*, vol. 3, pp. 35–57, 2003.

[93] W. Schroeder, J. Zarge, W. Lorensen, *et al.*, "Decimation of triangle meshes," *COMPUTER GRAPHICS-NEW YORK-ASSOCIATION FOR COMPUTING MACHINERY-*, vol. 26, pp. 65–65, 1992.

[94] M. Garland and P. Heckbert, "Surface simplification using quadric error metrics," pp. 209–216, ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 1997.

[95] S. Kim, C. Kim, and D. Levin, "Surface simplification using a discrete curvature norm," *Computers & Graphics*, vol. 26, no. 5, pp. 657–663, 2002.

[96] B. Chazelle, "Triangulating a simple polygon in linear time," *Discrete and Computational Geometry*, vol. 6, no. 1, pp. 485–524, 1991.

[97] A. Sheffer, E. Praun, K. Rose, and I. NetLibrary, "Mesh Parameterization Methods and Their Applications," *Foundations and Trends® in Computer Graphics and Vision*, vol. 2, no. 2, pp. 105–171, 2006.

[98] G. Zigelman, R. Kimmel, and N. Kiryati, "Texture mapping using surface flattening via multidimensional scaling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 2, pp. 198–207, 2002.

[99] E. Demaine and J. O'Rourke, "A survey of folding and unfolding in computational geometry," *Combinatorial and Computational Geometry*, 2005.

[100] H. Hoppe, "New quadric metric for simplifiying meshes with appearance attributes," pp. 59–66, IEEE Computer Society Press Los Alamitos, CA, USA, 1999.

[101] M. Garland, *Quadric-Based Polygonal Surface Simplification*. PhD thesis, Georgia Institute of Technology, 1999.

[102] G. Peyré and L. Cohen, "Heuristically Driven Front Propagation for Geodesic Paths Extraction," *LECTURE NOTES IN COMPUTER SCIENCE*, vol. 3752, p. 173, 2005.

[103] M. Tegmark, "An icosahedron-based method for pixelizing the celestial sphere," *Astrophysical Journal*, vol. 470, pp. L81–L84, 1996.

[104] J. Giard, P. Alface, and B. Macq, "Fast and accurate travel depth estimation for protein active site prediction," *Proceedings of SPIE*, vol. 6812, p. 68120Q, 2008.

[105] J. Giard, P. Alface, J. Gala, and B. Macq, "Fast Surface-Based Travel Depth Estimation Algorithm for Macromolecule Surface Shape Description," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 99, no. 1, 2009.

[106] J. Sethian, "Curvature and the evolution of fronts," *Communications in Mathematical Physics*, vol. 101, no. 4, pp. 487–499, 1985.

[107] D. Page, Y. Sun, A. Koschan, J. Paik, and M. Abidi, "Normal Vector Voting: Crease Detection and Curvature Estimation on Large, Noisy Meshes," *GRAPHICAL MODELS*, vol. 64, pp. 199–229, 2002.

[108] M. Stahl, C. Taroni, and G. Schneider, "Mapping of protein surface cavities and prediction of enzyme class by a self-organizing neural network," *Protein Engineering*, vol. 13, no. 2, pp. 83–88, 2000.

[109] R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality af A*," *Journal of the ACM (JACM)*, vol. 32, no. 3, pp. 505–536, 1985.

[110] D. Harker, "Two-dimensionally infinite polyhedra with vertices related by symmetry operations," *Proceedings of the National Academy of Sciences*, vol. 88, no. 2, pp. 585–587, 1991.

[111] H. Eckert and J. Bajorath, "Molecular similarity analysis in virtual screening: foundations, limitations and novel approaches," *Drug discovery today*, vol. 12, no. 5-6, pp. 225–233, 2007.

[112] J. Giard, J. Ambroise, R. J. K. Tcheuko, and B. Macq, "Predict protein atoms mobility using fourier transform," in *4th IEEE EMBS Benelux Chapter*, 2009.

[113] P. Doruker, R. Jernigan, I. Navizet, and R. Hernandez, "Important fluctuation dynamics of large protein structures are preserved upon coarse-grained renormalization," *International Journal of Quantum Chemistry*, vol. 90, no. 2, pp. 822–837, 2002.

[114] K. Suhre and Y. Sanejouand, "ElNemo: a normal mode web server for protein movement analysis and the generation of templates for molecular replacement," *Nucleic acids research*, vol. 32, no. Web Server Issue, p. W610, 2004.

[115] W. Humphrey, A. Dalke, and K. Schulten, "Vmd-visual molecular dynamics," *J. Mol.Graphics*, vol. 15, pp. 33–38, 1996.

[116] S. Kawashima and M. Kanehisa, "AAindex: amino acid index database," *Nucleic Acids Research*, vol. 28, no. 1, p. 374, 2000.

[117] D. Boobbyer, P. Goodford, P. McWhinnie, and R. Wade, "New hydrogen-bond potentials for use in determining energetically favorable binding sites on molecules of known structure," *Journal of Medicinal Chemistry*, vol. 32, no. 5, pp. 1083–1094, 1989.

[118] I. Halperin, B. Ma, H. Wolfson, and R. Nussinov, "Principles of docking: an overview of search algorithms and a guide to scoring functions," *PROTEINS-NEW YORK-*, vol. 47, no. 4, pp. 409–443, 2002.

[119] R. Wang, Y. Lu, and S. Wang, "Comparative evaluation of 11 scoring functions for molecular docking," *Journal of Medicinal Chemistry*, vol. 46, no. 12, pp. 2287–2303, 2003.

[120] M. Landau, I. Mayrose, Y. Rosenberg, F. Glaser, E. Martz, T. Pupko, and N. Ben-Tal, "Consurf 2005: the projection of evolutionary conservation scores of residues on protein structures," *Nucleic Acids Research*, vol. 33, no. Web-Server-Issue, pp. 299–302, 2005.

[121] J. Giard, J. Ambroise, J. Gala, and B. Macq, "Regression applied to protein binding site prediction and comparison with classification," *BMC bioinformatics*, vol. 10, no. 1, p. 276, 2009.

[122] N. Petrova and C. Wu, "Prediction of catalytic residues using Support Vector Machine with selected protein sequence and structural properties," *BMC Bioinformatics*, vol. 7, no. 312, pp. 1471–2105, 2006.

[123] A. Gutteridge, G. Bartlett, and J. Thornton, "Using A Neural Network and Spatial Clustering to Predict the Location of Active Sites in Enzymes," *Journal of Molecular Biology*, vol. 330, no. 4, pp. 719–734, 2003.

[124] A. Shulman-Peleg, R. Nussinov, and H. Wolfson, "Recognition of functional sites in protein structures," *Journal of molecular biology*, vol. 339, no. 3, pp. 607–633, 2004.

[125] A. Shulman-Peleg, R. Nussinov, and H. Wolfson, "SiteEngines: recognition and comparison of binding sites and protein-protein interfaces," *Nucleic acids research*, vol. 33, no. Web Server Issue, p. W337, 2005.

[126] M. Shatsky, A. Shulman-Peleg, R. Nussinov, and H. Wolfson, "The multiple common point set problem and its application to molecule binding pattern detection," *Journal of Computational Biology*, vol. 13, no. 2, pp. 407–428, 2006.

[127] S. Jones and J. Thornton, "Prediction of protein-protein interaction sites using patch analysis," *Journal of molecular biology*, vol. 272, no. 1, pp. 133–143, 1997.

[128] H. Zhou and Y. Shan, "Prediction of protein interaction sites from sequence profile and residue neighbor list," *Proteins: Structure, Function, and Genetics*, vol. 44, no. 3, pp. 336–43, 2001.

[129] H. Chen and H. X. Zhou, "Prediction of interface residues in protein-protein complexes by a consensus neural network method: test against nmr data.," *Proteins.*, vol. 61, pp. 21–35, October 2005.

[130] J. R. Bradford and D. R. Westhead, "Improved prediction of protein-protein binding sites using a support vector machines approach," *BIOINFORMATICS*, vol. 21, no. 8, pp. 1487–1494, 2005.

[131] Y. Murakami and S. Jones, "SHARP2: protein-protein interaction predictions using patch analysis," *Bioinformatics*, vol. 22, no. 14, pp. 1794–1795, 2006.

[132] S. Liang, C. Zhang, S. Liu, and Y. Zhou, "Protein binding site prediction using an empirical scoring function," *Nucleic acids research*, vol. 34, no. 13, pp. 3698–3707, 2006.

[133] S. Qin and H. Zhou, "meta-PPISP: a meta web server for protein-protein interaction site prediction," *Bioinformatics*, vol. 23, no. 24, pp. 3386–3387, 2007.

[134] N. Li, Z. Sun, and F. Jiang, "Prediction of protein-protein binding site by using core interface residue and support vector machine," *BMC bioinformatics*, vol. 9, no. 1, p. 553, 2008.

[135] J. R. Bradford, C. J. Needham, A. J. Bulpitt, and D. R. Westhead, "Insights into protein-protein interfaces using a bayesian network prediction method," *Journal of Molecular Biology*, vol. 362, no. 2, pp. 365–386, 2006.

[136] E. Petsalaki, A. Stark, E. García-Urdiales, and R. Russell, "Accurate Prediction of Peptide Binding Sites on Protein Surfaces," *PLoS Computational Biology*, vol. 5, no. 3, 2009.

[137] H. Zhou and S. Qin, "Interaction-site prediction for protein complexes: a critical assessment," *Bioinformatics*, vol. 23, no. 17, pp. 2203–2209, 2007.

[138] N. Tuncbag, G. Kar, O. Keskin, A. Gürsoy, and R. Nussinov, "A survey of available tools and web servers for analysis of protein-protein interactions and interfaces," *Briefings in Bioinformatics*, vol. 10, no. 3, pp. 217–232, 2009.

[139] P. Taylor and D. Flower, *Immunoinformatics and Computational Vaccinology: A Brief Introduction*, p. 23. Springer Verlag, 2007.

[140] R. Meloen, W. Puijk, J. Langeveld, J. Langedijk, and P. Timmerman, "Design of synthetic peptides for diagnostics.," *Current protein & peptide science*, vol. 4, no. 4, p. 253, 2003.

[141] M. Gomara and I. Haro, "Synthetic peptides for the immunodiagnosis of human diseases," *Current medicinal chemistry*, vol. 14, no. 5, pp. 531–546, 2007.

[142] T. Hopp and K. Woods, "Prediction of protein antigenic determinants from amino acid sequences," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 78, no. 6, pp. 3824–3828, 1981.

[143] M. Blythe and D. Flower, "Benchmarking B cell epitope prediction: underperformance of existing methods," *Protein Science: A Publication of the Protein Society*, vol. 14, no. 1, p. 246, 2005.

[144] M. Sweredoski and P. Baldi, "COBEpro: a novel system for predicting continuous B-cell epitopes," *Protein Engineering Design and Selection*, vol. 22, no. 3, p. 113, 2009.

[145] S. Saha and G. Raghava, "Prediction of continuous B-cell epitopes in an antigen using recurrent neural network," *PROTEINS: Structure, Function, and Bioinformatics*, vol. 65, pp. 42–9, 2006.

[146] M. Van Regenmortel, "Mapping epitope structure and activity: from one-dimensional prediction to four-dimensional description of antigenic specificity," *Methods*, vol. 9, no. 3, pp. 465–472, 1996.

[147] D. Barlow, M. Edwards, and J. Thornton, "Continuous and discontinuous protein antigenic determinants," *Nature*, vol. 322, no. 6081, pp. 747–748, 1986.

[148] E. Westhof, D. Altschuh, D. Moras, A. Bloomer, A. Mondragon, A. Klug, and M. Van Regenmortel, "Correlation between segmental mobility and the location of antigenic determinants in proteins.," *Nature*, vol. 311, no. 5982, p. 123, 1984.

[149] J. Thornton, M. Edwards, W. Taylor, and D. Barlow, "Location of'continuous' antigenic determinants in the protruding regions of proteins.," *The EMBO Journal*, vol. 5, no. 2, p. 409, 1986.

[150] J. Novotny, M. Handschumacher, E. Haber, R. Bruccoleri, W. Carlson, D. Fanning, J. Smith, and G. Rose, "Antigenic determinants in proteins coincide with surface regions accessible to large probes (antibody domains)," *Proc. Natl. Acad. Sci. USA*, vol. 83, no. 1, pp. 226–230, 1986.

[151] U. Kulkarni-Kale, S. Bhosle, and A. Kolaskar, "CEP: a conformational epitope prediction server," *Nucleic acids research*, vol. 33, no. Web Server Issue, p. W168, 2005.

[152] P. Andersen, M. Nielsen, and O. Lund, "Prediction of residues in discontinuous B-cell epitopes using protein 3D structures," *Protein Science: A Publication of the Protein Society*, vol. 15, no. 11, p. 2558, 2006.

[153] M. Sweredoski and P. Baldi, "PEPITO: improved discontinuous B-cell epitope prediction using multiple distance thresholds and half sphere exposure," *Bioinformatics*, vol. 24, no. 12, p. 1459, 2008.

[154] N. Rubinstein, I. Mayrose, E. Martz, and T. Pupko, "Epitopia: a web-server for predicting B-cell epitopes," *BMC bioinformatics*, vol. 10, no. 1, p. 287, 2009.

[155] N. Rubinstein, I. Mayrose, D. Halperin, D. Yekutieli, J. Gershoni, and T. Pupko, "Computational characterization of B-cell epitopes," *Molecular Immunology*, vol. 45, no. 12, pp. 3477–3489, 2008.

[156] Y. Saeys, I. Inza, and P. Larranaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, p. 2507, 2007.

[157] S. Jones and J. Thornton, "Principles of protein-protein interactions.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 93, no. 1, pp. 13–20, 1996.

[158] E. Vittinghoff, D. Glidden, S. Shiboski, and C. McCulloch, *Regression Methods In Biostatistics: Linear, Logistic, Survival, and Repeated Measures Models*. Springer New York, 2005.

[159] R. Tobias, "An introduction to partial least squares regression," in *Proceedings of the Twentieth Annual SAS Users Group International Conference, Cary, NC: SAS Institute Inc*, pp. 1250–1257, 1995.

[160] Y. Xie and J. Kalivas, "Evaluation of principal component selection methods to form a global prediction model by principal component regression," *Analytica Chimica Acta*, vol. 348, no. 1-3, pp. 19–27, 1997.

[161] G. Zhang, B. Eddy Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *International Journal of Forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[162] C. Chen, A. Liaw, and L. Breiman, "Using random forest to learn imbalanced data," tech. rep., University of California at Berkeley: Statistics Department, 2004.

[163] H. Zhang and J. Su, "Naive bayesian classifiers for ranking," in *European Conference on Machine Learning*, pp. 501–512, 2004.

[164] S. Gunn, "Support Vector Machines for Classification and Regression," tech. rep., Faculty of Engineering, Science and Mathematics School of Electronics and Computer Science, 1998.

[165] J. Mintseris, K. Wiehe, B. Pierce, R. Anderson, R. Chen, J. Janin, and Z. Weng, "Protein-protein docking benchmark 2.0: an update," *PROTEINS-NEW YORK-*, vol. 60, no. 2, pp. 214–6, 2005.

[166] J. Ponomarenko and P. Bourne, "Antibody-protein interactions: benchmark datasets and prediction tools evaluation," *BMC Struct. Biol*, vol. 7, p. 64, 2007.

[167] S. Liang, D. Zheng, C. Zhang, and M. Zacharias, "Prediction of antigenic epitopes on protein surfaces by consensus scoring," *BMC bioinformatics*, vol. 10, no. 1, p. 302, 2009.

[168] J. Giard and B. Macq, "Molecules 3D Delaunay triangulation: a spectral study," in *Proceedings of SPIE*, vol. 7262, p. 90, 2009.