MOLECULAR SURFACE ABSTRACTION

by

Gregory M. Cipriano

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2010

© Copyright by Gregory M. Cipriano 2010
All Rights Reserved

To Ila, for providing infinite inspiration and distraction.

ACKNOWLEDGMENTS

I would like to thank first of all the members of my committee for their advice and support throughout my time at the University of Wisconsin, especially my advisor, Michael Gleicher. Without you, this work would not have been possible.

I thank everyone in the Graphics Lab, past and present, for many lively discussions and for helping me work through a thousand thorny issues. I thank especially Rachel Heck for going above and beyond (and sacrificing a Saturday) in helping finish my first Vis video.

I thank my friends and family for never questioning the value of all this education, for encouraging me along the way, and for providing the many good times that I'll remember long after I've left Madison.

I thank my wonderful wife Ila for a thousand acts of generosity: for making me dinner when I didn't have time to feed myself, for rubbing my neck when I was stressed, and for making me laugh when I needed it most. Also, I thank her for never shying away from telling me when I was being unclear in my prose or when the colors were *all wrong*.

Finally, I thank CIBM and BACTER, for their financial support.

DISCARD THIS PAGE

TABLE OF CONTENTS

Page

LIST OF TABLES . . . vi

LIST OF FIGURES . . . vii

NOMENCLATURE . . . ix

ABSTRACT . . . x

1 Introduction . . . 1

1.1 Problem Overview . . . 4
1.2 Technical Contributions . . . 7
1.3 Technical Solutions Overview . . . 9
1.3.1 Molecular Surface Abstraction . . . 9
1.3.2 Multi-Scale Surface Curvature Estimation . . . 10
1.3.3 Ligand Binding Prediction . . . 11

2 Related Work . . . 13

2.1 Molecular Visualization . . . 13
2.2 Curvature Estimation . . . 14
2.3 Local Shape Descriptors . . . 16
2.4 Functional Surface Analysis . . . 18
2.4.1 Identifying Potential Binding Sites . . . 18
2.4.2 Alignment, Comparison and Classification . . . 22

3 Background . . . 25

3.1 The Geometric Surface . . . 25
3.1.1 The Van der Waals Surface . . . 25
3.1.2 Solvent-Excluded Surface . . . 26
3.2 Electrochemical Properties . . . 27
3.2.1 Electrostatic Potential . . . 28

Page

3.2.2 Hydropathy . . . 29
3.2.3 Hydrogen Donors/Acceptors . . . 29
3.3 The Lock and Key Metaphor . . . 30

4 Molecular Surface Abstraction . . . 31

4.1 Abstracted Surfaces . . . 32
4.1.1 Smoothing Surface Geometry . . . 34
4.1.2 Abstracting Surface Fields . . . 36
4.1.3 Removing Mid-Sized Features . . . 38
4.1.4 Decaling . . . 40
4.1.5 Results . . . 45
4.2 The Client Viewer . . . 48
4.2.1 High Quality Rendering of Abstracted Surfaces . . . 48
4.2.2 Multiple Surface Display and Comparison . . . 49
4.2.3 Results . . . 51
4.3 GRAPE: GRaphical Abstracted Protein Explorer . . . 52
4.3.1 Project Goals . . . 52
4.3.2 Server Side Processing . . . 53
4.3.3 Client Side Viewer . . . 56
4.3.4 Social Networking . . . 59
4.4 Discussion . . . 60

5 Multi-Scale Surface Shape Descriptors . . . 63

5.1 Local Shape Descriptors . . . 63
5.1.1 Contribution . . . 65
5.2 Multi-Scale Shape Descriptors . . . 66
5.2.1 Neighborhood Construction . . . 68
5.2.2 Height Field . . . 71
5.2.3 Fitting with Quadratics . . . 73
5.2.4 Moment-Based Surface Description . . . 74
5.3 Results . . . 76
5.3.1 Performance . . . 76
5.3.2 Evaluation . . . 77
5.4 Applications . . . 80
5.4.1 Multi-scale Lighting . . . 80
5.4.2 Segmentation . . . 82
5.4.3 Stylized Rendering . . . 83
5.4.4 Multi-scale/Anisotropic Curvature Matching . . . 84

Appendix

Page

6 Binding Prediction . . . 85

6.1 Introduction . . . 85
6.1.1 Contributions . . . 86
6.1.2 Method Overview . . . 87
6.2 The Functional Surface Descriptor . . . 88
6.2.1 Descriptor Scales . . . 88
6.2.2 Descriptor Features . . . 89
6.2.3 Normalization . . . 90
6.3 Per-Atom Training . . . 90
6.3.1 Building Training Examples . . . 91
6.3.2 Training an Atom Learner . . . 93
6.4 Per-Moiety Prediction . . . 97
6.4.1 Reducing Sample Count . . . 97
6.4.2 Predicting For an Atom . . . 100
6.4.3 Combining Atom Predictions . . . 100
6.5 Merging Moiety Predictions . . . 103
6.6 Evaluation . . . 103
6.6.1 Training . . . 104
6.6.2 Results . . . 106
6.6.3 Run-Time Performance . . . 113
6.7 Comparison to Existing Methods . . . 115
6.7.1 Thornton, Spherical Harmonics . . . 115
6.7.2 Kihara, Real Time Search . . . 116
6.8 Discussion . . . 117

7 Discussion . . . 119

7.1 Issues and Limitations . . . 120
7.1.1 Visual Abstraction . . . 120
7.1.2 Curvature . . . 122
7.1.3 Binding Prediction . . . 122
7.2 Future Work . . . 125

LIST OF REFERENCES . . . 127

APPENDIX Molecular Surface Feature Vector De nition . . . 138

DISCARD THIS PAGE

LIST OF TABLES

Table Page

6.1 Moiety matches used as training examples 105

6.2 Results of a calcium binding test 106

6.3 Test cases 109

Appendix
Table

A.1 A list of each feature contained within our surface descriptor. 140

DISCARD THIS PAGE

LIST OF FIGURES

Figure Page

1.1 An abstraction: ball-and-stick vs. ribbon diagrams 3

3.1 How electrostatics are sampled onto the surface 28

4.1 Adenylate Kinase (1ANK) drawn in Pymol, Qutemol, stylized and abstracted 33

4.2 Surface Field Simplification, before and after 36

4.3 Steps involved in geometric surface abstraction 38

4.4 Field of view for stickers vs. geometry 40

4.5 Local environment mapping diagram 41

4.6 Fixing issues with patch connectivity 42

4.7 Jagged vs. smooth (abstract) patch boundaries 43

4.8 Replacing ligands with ligand shadow stickers 44

4.9 Ribonuclease example 45

4.10 Abstraction vs. large probe size 46

4.11 Gallery of abstractions 47

4.12 Multiple surface display: 6 aligned RRMS 50

4.13 Demonstration: comparing ribonucleases 50

4.14 The GRAPE job queue 53

4.15 The four sticker types displayed in GRAPE 57

Figure Page

4.16 The GRAPE output window . . . 58
4.17 A GRAPE recommendation gadget . . . 59
5.1 Steps involved in generating a descriptor for a single neighborhood . . . 67
5.2 Shape description, represented at multiple scales . . . 68
5.3 Finding a disc: Dijkstra vs. 2-ring improvement . . . 69
5.4 Comparing curvature results: Dijkstra vs. 2-ring improvement . . . 69
5.5 Multi-scale descriptors: sensitivity to noise . . . 78
5.6 Multi-scale descriptors: sensitivity to tessellation . . . 79
5.7 Multi-scale lighting . . . 81
5.8 Multi-scale segmentation . . . 82
5.9 Multi-scale stylized rendering . . . 83
5.10 Multi-scale curvature matching . . . 84
6.1 Radii used in the functional surface descriptor . . . 89
6.2 Building a Corpus using Moiety Exemplars . . . 92
6.3 Atom training overview . . . 95
6.4 Computing the bounds for all pairs of atomic distances . . . 96
6.5 An illustration (in 2D) of how my method for grouping samples on the 3D surface works 99
6.6 Prediction Phase: combining atom surface functions to predict a ligand. . . . 101
6.7 Performance of calcium ion prediction, as the training corpus grows . . . 107
6.8 Confusion matrices for 4 test cases . . . 110
6.9 Results: ROC curves for each test case . . . 111
6.10 Results: ROC curves for each test case . . . 112

DISCARD THIS PAGE

NOMENCLATURE

APBS	Short for 'Adaptive Poisson-Boltzmann Solver", this software is used to evaluate the electrostatic properties of nanoscale biomolecular systems in the presence of water.
Functional Surface	The complex of geometrical as well as physio-chemical features on the bounding surface of a protein. Also called 'molecular surface'.
Moiety	A significant segment, portion or part of a molecule that may include a substructure of the functional group.
MSMS	The 'Michel Sanner Molecular Surface' package. This program takes as input a PDB file containing atom coordinates and radii and produces a triangulated solvent-excluded surface.
PDB	The Protein Data Bank, maintained by Research Collaboratory for Structural Bioinformatics (RCSB), is an open, online repository for MacroMolecular Structures, such as proteins.

MOLECULAR SURFACE ABSTRACTION

Gregory M. Cipriano

Under the supervision of Professor Michael Gleicher

At the University of Wisconsin-Madison

In every field of scientific inquiry, new tools and techniques are allowing scientists to acquire ever-increasing amounts of data. The field of molecular biology is an important example of this trend: not only are many molecules themselves large and complex, but the way in which they interact with their environment may be difficult to understand and describe. The region surrounding a protein, known as the surface of interaction or "functional surface", can provide valuable insight into its function. Unfortunately, due to the complexity of both their geometry and their surface fields, study of these surfaces can be slow and difficult, and important features may be hard to identify.

This dissertation describes tools and techniques that I have created to address these issues, all of which use abstraction as a method for reducing complexity. First, with the help of collaborators in biochemistry, I have created *novel abstract visualizations that more effectively convey the gestalt of the surface of large molecules*. Second, I have developed new multi-scale surface descriptors, to *aid in the discovery of potential binding partners and to better classify proteins of unknown function*. The main technical contributions include: a method for abstracting the functional surface of a protein, and an online system for building and displaying these abstractions; a method for describing the curvature and anisotropy of the surface of a triangulated mesh at multiple scales; a functional surface descriptor that combines this geometric information with multi-scale electrochemical features; and a method for using this descriptor to learn and predict ligand binding behavior on new protein surfaces.

Michael Gleicher

ABSTRACT

In every field of scientific inquiry, new tools and techniques are allowing scientists to acquire ever-increasing amounts of data. The field of molecular biology is an important example of this trend: not only are many molecules themselves large and complex, but the way in which they interact with their environment may be difficult to understand and describe. The region surrounding a protein, known as the surface of interaction or "functional surface", can provide valuable insight into its function. Unfortunately, due to the complexity of both their geometry and their surface fields, study of these surfaces can be slow and difficult, and important features may be hard to identify.

This dissertation describes tools and techniques that I have created to address these issues, all of which use abstraction as a method for reducing complexity. First, with the help of collaborators in biochemistry, I have created *novel abstract visualizations that more effectively convey the gestalt of the surface of large molecules*. Second, I have developed new multi-scale surface descriptors, to *aid in the discovery of potential binding partners and to better classify proteins of unknown function*. The main technical contributions include: a method for abstracting the functional surface of a protein, and an online system for building and displaying these abstractions; a method for describing the curvature and anisotropy of the surface of a triangulated mesh at multiple scales; a functional surface descriptor that combines this geometric information with multi-scale electrochemical features; and a method for using this descriptor to learn and predict ligand binding behavior on new protein surfaces.

Chapter 1

Introduction

In nearly every field of scientific inquiry, new tools and techniques are allowing scientists to acquire ever-increasing amounts of data. The resulting challenges are many: in storage and curation, in provenance, and in rectifying the unlimited scope of the data with the limited capacity for the human brain to understand it. It is this latter challenge that drives much of the field of scientific visualization, and the one that will be the focus of this dissertation.

The field of molecular biology is an important example of this trend: not only are many biological molecules themselves large and complex, but the way in which they interact with their environment may be difficult to understand and describe. An entire dissertation may be formed from the study of one such molecule, and careers made from the elucidation of the workings of a handful. Clearly, a need exists for better tools, to speed the time to discovery and to allow researchers to conduct larger and more involved comparative analyses.

Proteins are of central importance in most biological processes, and so are naturally the subject of extensive study within the larger field of molecular biology. Their apparent simplicity — they are always composed of a folded chain of amino acids, each chosen from a vocabulary of (usually) 22 options — gives rise to an extraordinary abundance of forms and functions. They may act as small cellular signals, shuttling between organelles and through membranes. They may be larger enzymes, playing an essential role in biochemistry by catalyzing reactions that would otherwise be energetically unfavorable. And they may combine to form even larger structures, such as in virus capsids or in the ribosome. Understanding protein function is, therefore, an essential component of understanding overall biological function.

Structural biologists and bioinformaticists typically rely on several hierarchical views of a given protein: its primary structure is the sequence of its amino acids; its secondary structure is how local segments of that sequence form higher-order units such as beta sheets and alpha helices; its tertiary structure is represented by the specific positions for each of its atoms; and finally, its quaternary structure, if one exists, is its arrangement into a multi-unit complex, either as a homo- or hetero-multimer.

Research has has been done at each level of this hierarchy to understand how a protein's structure relates to its function. Sequence comparison especially has been well studied as a tool for understanding protein function [1], and has found a great deal of use in techniques such as BLAST [2] and CLUSTALW [3]. And while a number of functional relationships between proteins have been found using these techniques, sequence comparison is neither necessary nor sufficient to completely describe surface binding. Indeed, many proteins share structural similarity without sharing sequence identity, which might be the result of convergent evolution. In contrast, two structurally homologous, evolutionarily-related proteins may have different functions.

Biologists, therefore, have increasingly begun using structural models to understand the binding interactions between small molecules and larger proteins, how molecular motion affects these bindings, and how all of these interactions fit into larger biological processes. But it was only with the advent of techniques for identifying tertiary structure through X-ray Crystallography and Nuclear Magnetic Resonance (NMR), that scientists could begin to directly observe the chemical mechanisms for interactions between ligands and their receptors at the molecular level.

Since then, the number of known 3D structures has grown dramatically, such that scientists now have a large repository of high-quality structural information stored the Protein DataBank (PDB) [4]. As a consequence, a great deal of recent attention has been focused on tertiary structure: both in visually depicting these often large and complex molecules, and in predicting function via the three-dimensional position of its atoms.

This dissertation is founded on the observation that in many proteins, the internal structure exists only as scaffolding to place various forces and chemical properties in proper spatial relationships with one another. Only the remaining atoms are involved in interactions with the outside

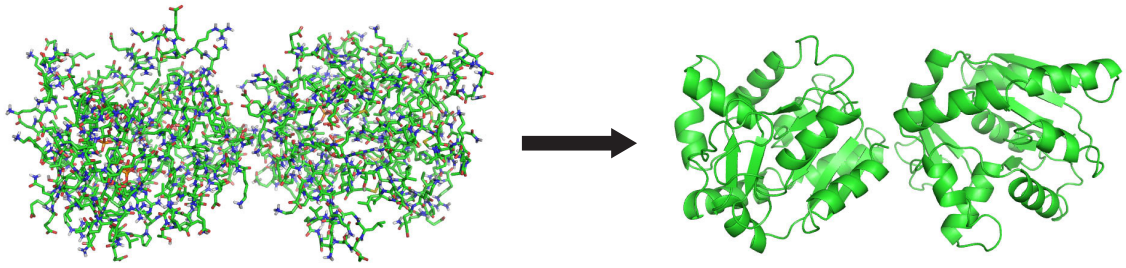


Figure 1.1 A ball-and-stick representation (left) of adenylate kinase contains too much information to be easily understandable. For larger scale views, biologists use abstracted representations such as ribbon diagrams (right). Such abstractions show major internal features of the molecule but do not convey the external surface.

world, and are therefore the most important for determining overall protein function. In this document, these are referred to as a "functional surface", which describes both the geometric boundary between the covalently-bonded protein interior and the outside environment, as well as electrochemical properties at that boundary, such as electrostatic potential and hydropathy.

Meanwhile, because these proteins can be large and complex, biologists often rely on visualizations that provide varying levels of abstraction: atoms are abstract representations of underlying quantum-mechanical forces, ribbon diagrams abstract atomic models, and so on (see Figure 1.1.) Importantly, each level of abstraction not only offers a simpler representation, but also may convey details not readily apparent in lower levels. For instance, while the ball-and-stick model does not depict electron orbitals, it provides a much more natural and intuitive representation of the bonds between atoms.

This representation, it should be noted, is *not* a replacement for the more nuanced orbital model, but rather an alternative and complementary view. Similarly, ribbon diagrams no longer accurately convey chemical structure, but they do allow for an easier understanding of the major components of protein composition, including beta sheets and alpha helices.

Functional surfaces can be just as complex as any of these representations, both in terms of their overall geometry, as well as in terms of the spatio-physico-chemical fields generated by underlying

amino acids. While tools exist for visualizing functional surfaces, and structural biologists use them extensively, none utilize abstract representations to deal with functional surface complexity.

This dissertation describes two novel abstract representations which are designed to address two distinct subproblems in functional surface understanding. The first is one of *visualization*: how can we depict the functional surface in such a way that emphasizes important, biologically relevant features, while deemphasizing those features that are less relevant? The second is one of *analysis*: how can we use the knowledge gained by looking at the binding patterns in known proteins to predict possible binding modes on an unclassified surface? ***It is my thesis that applying the approach of abstraction to the spatio-physico-chemical properties that form the functional surface of a protein allows us both to more readily visualize its significant features, such as pockets, clefts and overall shape, as well as to provide a concise set of descriptors to better understand and classify those surface features that interact with outside binding partners.***

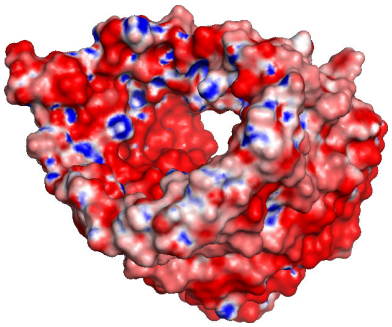
1.1 Problem Overview

An important aspect of protein analysis is the characterization of the non-covalent interactions between receptors and ligands, which are mainly driven by electrostatics, van der Waals forces and hydrogen bonding. This type of binding, which is involved in the majority of protein-ligand interactions, is characterized by the so-called "Lock and Key" principle, described in more detail in Section 3.3, which states that strong binding interactions require complementarity between these three features.

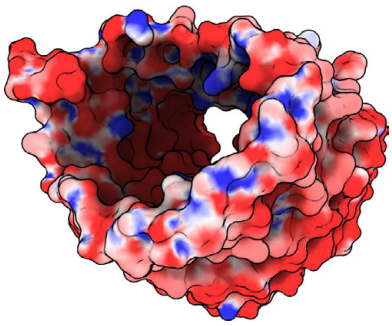
As these non-covalent interactions, by definition, take place on the surface of the molecule, a need has arisen for display and characterization of this surface along with its physicochemical properties, the combination of which is called the "functional surface". Existing methods for viewing the functional surface typically start by constructing a bounding surface mesh.

One typical method of building this mesh involves tessellating the points of contact between a spherical probe and the atoms of the protein (this will be described in more detail in Section 3.1). Physical or chemical properties, such as electrostatic potential, may be sampled onto the vertices

of the resulting mesh. The output of this process may be rendered into a picture such as this one, as shown in Pymol [5]:



Better use of lighting can emphasize the tubular nature of this porine molecule:



Fundamentally, however, visualizing even a small protein such as this one presents a challenge. In particular, the following visualization issues remain:

1. **Complex Surface Geometry** Many small bumps and bowls, a byproduct of faithfully reproducing the van der Waal shells of underlying atoms, present high frequency detail that make quick assessment of the surface difficult.

2. **Complex Surface Fields** Surface fields, such as electrostatic potential, may be sampled onto the surface. But the large number of sharp discontinuities in potential, such as when strongly positive and negative atoms are in close proximity, makes difficult the job of recognizing large regions of consistent potential.

3. **Multiple Properties** As mentioned above, other forces besides van der Waal and electrostatics are important for binding specificity: hydrogen bonding, zinc bridges and other interactions also play a role. It is difficult to depict all of these properties of the surface simultaneously using surface color alone. Most tools do not even try, at best requiring users to flip back and forth between different surface field representations.

In addition to visualization, the process of surface analysis also suffers from similar issues:

1. **High-Frequency Detail** Complex surface geometry and physicochemical forces require a large amount of storage to faithfully represent and a large amount of computation time to analyze.

2. **Multi-Scale Phenomena** The idea behind "induced fit", introduced by Koshland [6], is that often a receptor does not share strict complementarity with its ligand until it moves into place. This movement implies that strict fidelity to small-scale static molecular surface may not be desirable when predicting binding. Larger-scale features, however, such as overall pocket shape and charge, are less affected by this movement, and so may be better suited to this type of prediction.

3. **Localized Binding** Much of the surface of a protein is not involved in binding interactions. To achieve faster binding prediction, the surface must be triaged such that the majority of cycles are spent in areas that are more likely to bind.

The goal of this work is to provide both tools and techniques to address these issues. In particular, the aim of this dissertation is to show that by thinking of the surface at *multiple scales*, it is possible to address both sets of problems. To that end, I demonstrate two multi-scale techniques: first, a simplified (abstract) visual representation that captures the gestalt of a surface and allows for quick assessment and targeted hypothesis generation.

Second, in Chapter 6, I describe a method for identifying potential binding partners on the surface of an unknown protein by using a learning algorithm trained on multi-scale surface features drawn from a corpus of proteins with known binding affinity. By predicting the location of potential ligand binding, this method can help to target small areas of a large protein for further study.

1.2 Technical Contributions

Molecular surface abstraction combines techniques for simplifying surface meshes and volumetric fields. Both techniques are designed to preserve the overall structure of a protein, while summarizing detail that might otherwise inhibit comprehension. In this dissertation, I will demonstrate the feasibility and utility of these techniques in both visualization as well as computational applications. In particular, I make the following specific technical contributions:

1. **A method for abstracting the functional surface of a protein (Chapter 4)** For the task of visualizing coarse-grained models, I provide a method that integrates *mesh and scalar- eld abstraction with stylized surface rendering* and decals. The end product is a system that implements these ideas, one which allows users to quickly assess the significant features of a protein, such as its pockets and clefts, without getting mired in detail. In addition, I will demonstrate how such abstracted views are useful for comparing molecules, showing two specific examples of aligned homologous proteins. One shows how the additional layering of information afforded by abstracted surfaces makes for easier comparison; the other, how patterns of electrostatic potential may be more easily compared after abstraction.

2. **An online system for building and displaying these abstractions (Chapter 4)** As my abstract surface visualizations are a departure from how scientists typically view the functional surface, an important first step in their acceptance is to allow the community of users to judge their utility for themselves. These users, however, are naturally reluctant to commit to the download and use of yet another bespoke application. To overcome this issue, I have created **GRAPE**, or "**GR**aphical **A**bstracted **P**rotein **E**xplorer" [7]. This website *lets users abstract any protein* of their choosing. Abstraction takes place asynchronously on the back-end and all resulting data is then sent back for display in a light-weight Java viewer. The combination of a simple client interface with extremely low computational requirements allows users a quick way to try abstractions for themselves.

3. **A method for describing the curvature and anisotropy of the surface of a triangulated mesh at multiple scales (Chapter 5)** I provide a simple multi-scale geometric surface descriptor that borrows intuitions from surface curvature. Rather than describing the instantaneous region around a point, however, this descriptor is designed to describe the gross curvature and anisotropy of a disc-shaped patch of surface surrounding a given point. The scale at which a point is characterized may be adjusted by simply changing the diameter of this patch. Because shape complementarity is an important component of binding specificity, and because this complementarity holds at multiple scales, my geometric descriptor was designed to be adaptable: it can accurately describe patches of surface from the size of an atom to the size of a pocket. I evaluate how well it compares to state-of-the-art local curvature estimation methods, as well as its resilience to noise and poor tessellation.

4. **A functional surface descriptor, which incorporates geometrical and electrochemical information at multiple scales (Chapter 6)** I demonstrate a functional surface descriptor, which combines geometry, electrostatic potential and hydropathy, each at multiple scales, along with other residue-specific surface features. I describe a simple scheme for sampling the surface with descriptor points. I also describe a procedure for aggregating nearby groups of samples that also cluster in feature space.

5. **A method for learning ligand binding behavior (Chapter 6)** I show how to use this functional surface descriptor to characterize binding behavior of a ligand, and how to use this characterization to predict the location and of that ligand's binding on a surface. As a test case, 4 ligands of varying sizes (ATP, testosterone, glucose and Heme) are first trained on, and then experiments run to gauge both the accuracy of my binding predictor, as well as the likelihood that predicted binding sites are confused (i.e., if a predictor is trained on ATP, does it find Heme binding sites?) I show that, except for glucose, all tests found known binding pockets, most with a high degree of specificity. I also evaluate its performance on the task of predicting Calcium ion binding, showing that its results compare favorably to those of FEATURE [8]. Finally, I show how incorporating multi-scale features into the descriptor can

improve the overall performance of the atomic-predictors, using this same calcium-binding set as a test case.

1.3 Technical Solutions Overview

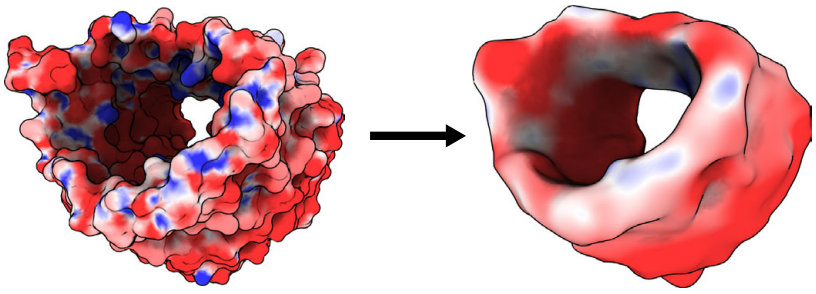
This section describes in more detail the individual technical problems I have addressed with my work and how I have solved these problems.

1.3.1 Molecular Surface Abstraction

I approach this problem by first noting that many of the features causing the problems mentioned are not essential for understanding the gestalt of a protein. Small peaks and bowls, for instance, may represent atoms whose positions are already uncertain, due to thermal vibration and small conformational movements. Further, their presence, much like the presence of shell orbitals in the quantum model of atomic bonds, may obscure higher-order phenomena on the surface.

Surface fields have a similar property: small amounts of positive and negative electrostatic potential may, for instance, obscure the fact that a pocket has an overall neutral potential. This visual clutter, again, makes it difficult to assess larger patches of uniform value, or to quickly gauge the overall character of a region on the surface.

To produce these abstract visualizations, these offending features must be removed: large bumps and pockets "sanded off", and electrostatic potential smoothed. The end product is a much simpler representation:

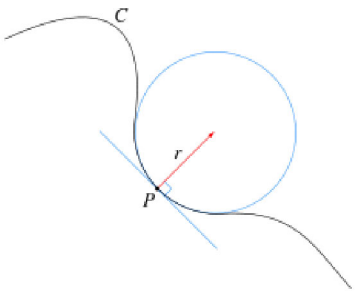


This work is described in detail in Chapter 4.

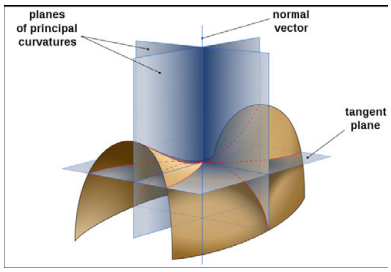
1.3.2 Multi-Scale Surface Curvature Estimation

An important aspect of functional surface analysis is the principle of surface shape complementarity: a binding partner for a protein will often have locally complementary shape to its region of binding. Much as a key fits only its matching lock, complementarity implies that binding is highly stereospecific. Therefore, by characterizing the shape of a known binding pocket, and then using this information to identify similar regions in other proteins, I may find new targets for a given partner. I note, however, that protein shape complementarity exists at *multiple scales*; from small atoms to large pockets, complementarity must hold.

Curvature, meanwhile, is only valid on a continuous surface and for an infinitesimal point. The reason is in the definition of curvature: to find the curvature for a curve C at a point P, first find the limit of the circle passing through three distinct points on C as these points approach P. This is the so-called osculating circle at point P, and its inverse is the curvature.



On a 3D manifold, a point P has a range of curvatures (one for each plane passing through P and its normal). The largest and smallest of these are called the *principle curvatures*, and along with the *principle directions*, are sufficient for completely describing the local surface at P.



Clearly, this definition of curvature does not extend to multiple scales, and for good reason: while the tensor of curvature at a point can be completely described by a quadratic form, and thus has three degrees of freedom, the number of degrees of freedom for an arbitrary patch of surface is much greater. Thus, a quadratic surface is not sufficient to fully describe most non-trivial surface patches.

Nevertheless, surface complementarity is most conveniently defined in terms of curvature: the curvature at complementary sides of a binding interaction are additive inverses of one another. And thus, we wish to retain both this property and the desirable intuitions that go along with surface curvature.

To tackle this problem, I have created a shape descriptor which is able to characterize the local neighborhood of a given point at multiple scales. Note that, unless the region being described is itself a quadratic, this characterization is only an approximation of its mean curvature and anisotropy. I show how such an approximation is sufficient for a number of tasks, some of which are described in Section 5.4. This work is described in detail in Chapter 5.

1.3.3 Ligand Binding Prediction

At this time, there are around 66961 unique entries in the Protein DataBank (PDB) [4], with around 37% having no known classification. These numbers grow every day as new structures are found. Add in computed protein structures, such as those obtained from threading a protein sequence through the known structure of a homologous protein, and there are many millions of surfaces, many of which have either incomplete or unknown binding affinity.

My goal with this work is to provide a set of predictors, each tuned for predicting the binding of one specific small ligand. These predictors can then be applied to the surface of a protein, resulting in an automated classification of the location and type of binding that may occur on its surface. This task, similar to one known as "blind docking", may yield new modes of functionality for classified proteins and new insights into the purpose of unclassified proteins.

A necessary component of this goal is a means to describe points on the functional surface of a given protein. To address this need, I have designed a feature vector which contains information

about the geometry and physio-chemical environment for a point on the surface, both at multiple scales. I describe this feature vector format in detail in Section 6.2. A complete description is formed after numerous samples are placed on the surface, and a functional surface descriptor built for each one.

As noted above, I assume complementarity holds at a number of scales. I further assume that a specific atom in a ligand interacts with every surface with which it binds in a fixed set of ways, often dictated by this need for complementarity. Polar atoms, for instance, will almost always be found in contact with polar residues on the surface of a protein. And atoms that can donate electrons will often be found bound to a region of the surface that can receive an electron (or vice versa) to form hydrogen bonds that stabilize an interaction.

These assumptions lead to a fundamental hypothesis that drives my research in this area: that the microenvironment surrounding a specific atom on a ligand will take on a fixed, small set of forms. Further, by studying that atom's microenvironment on a large corpus of exemplar proteins known to bind that ligand, we can enumerate those forms and use that information to predict its presence on the surface of a novel protein.

Of course, an atom-predictor is not particularly useful, as there may be numerous places on a protein surface that happen to match its microenvironment without actually being favorable for the complete ligand to bind. However, by combining predictions for all atoms in the ligand based on the known distances between pairs of atoms on the ligand itself, these false positives can be weeded out, ultimately producing a robust *ligand predictor*. This work is described in detail in Chapter 6.

Chapter 2

Related Work

2.1 Molecular Visualization

Because of the importance of molecular shape, structural biologists have depended on visual tools from the beginning. Visual tools predate computers and continue to be developed to this day (see [9, 10, 11, 12] for historically oriented surveys). One of the earliest of these tools, SURFNET [13], provided for the display of molecular surfaces, cavities and intermolecular interactions of the functional surface. More importantly, its pluggable interface spawned other cavity tools, such as [14], which adds a surface conservation metric to SURFNET to localize binding sites. Current state-of-the-art systems, such as Chimera [15], PyMol [5], and their competitors, provide large feature sets giving many options for the display of molecules.

Any visualization of a molecule necessarily involves some degree of abstraction. The field has developed a range of visual representations that provide different levels of abstraction; see [11] for a survey.

Surface simplification (see [16] for a survey) creates approximate models with fewer polygons. These methods are useful in improving efficiency while preserving the appearance. Simplification is an essential part of large molecule surface display [17, 18]. More recent work focuses on improving the quality of the initial solvent-excluded surface generation, while simultaneously reducing the number of triangles needed for faithful representation [19, 20].

In contrast, I seek to alter the appearance to be more abstract, which does not necessarily provide a performance benefit as number of triangles in the mesh is not significantly reduced.

Abstracted surfaces, however, are amenable to simplification. [17] applies smoothing to reduce the blocky appearance of coarse models of large molecules.

Display of other spatio-physico-chemical properties by color coding molecular surfaces became common as soon as surface representations were readily available. An early example was GRASP [21], which showed electrostatic potentials on surfaces. [22] unfolded the surfaces to better show their property distributions. My visualizations provide abstracted display of these properties as well as molecular shape.

Work on displaying molecular motion shows the uncertainty in molecular shape. [23] shows uncertainty and vibrational motion by blurring standard representations, and [24] clusters states to provide visual representations of ranges of conformations. [25] uses a combination of point-based rendering and random displacement to convey surface uncertainty in volumetric data. While my work does not deal directly with molecular motion, the idea that the absolute positions of atoms in a molecule are uncertain serves as a justification for the validity of visual abstraction.

The work of biochemist and artist David Goodsell showed the merit of using artistically stylized depictions of molecules. In [26] he describes a system for image processing molecular graphics that simulates a black-and-white line-art look, but which makes no abstraction of the shape, and in [27] he used similar depictions to conduct a visual survey of 136 homodimeric proteins.

In contrast to my abstractions, QuteMol [28], makes no attempt at all to simplify underlying protein models. Instead, it uses a combination of ambient occlusion and halos to enhance depth cues when displaying atomic models. The resulting renderings show the value of stylized shading in making molecular shape more comprehensible, but they provide no facility for display of other surface properties, such as charge, and still suffer from a profusion of detail.

2.2 Curvature Estimation

Many papers deal with the task of computing curvature for points on the surface: see [29] for an overview. Petitjean [30] surveys methods for estimating local surface quadratics, several of which inform my work. For instance, I borrow ideas for computing the so called ‘augmented Darboux frame’ from Sander and Zander [31], extending them into larger scales, and using them

as a basis for features used in my molecular surface descriptor. I also adapt the surface-covariance techniques from Berkmann, et al., [32] to robustly estimate the vectors of principle curvature. I draw inspiration for my point- and normal-weighting techniques from Page et al. [33], who use a technique they describe as "normal vector voting" to compute curvature in the presence of noise.

Curvature estimation may break down in certain cases, as shown in [34]. Rusinkiewicz [35] shows how to avoid similar issues in per-vertex curvature computation by estimating the second fundamental tensor per-face, and incorporating that into a description of the 1-ring curvature about a point.

As binding pockets may be highly anisotropic, a key requirement for surface curvature description for the task of binding prediction is the estimation of anisotropy. Sphere fitting methods, such as those by Coleman et al. [36], are able to characterize the surface at multiple scales, but are not able to produce anisotropy. Ellipsoid fitting, an extension of this process, can indicate anisotropy, but is computationally expensive, and is not able to model hyperbolic (saddle) surface regions [37]. Most discrete curvature fitting methods, such as *angle de cit* and *angle excess*, are very fast, but also do not provide principle directions, and so cannot give anisotropy information [29, 38].

Duncan and Olson [39], describe two techniques for approximating the tensor of curvature on a molecular surface at multiple scales, allowing them to find surface curvature and anisotropy and to identify regions such as saddles, ridges and troughs. One operates volumetrically, directly on the electron density function. The other is designed to operate on a triangulated surface, which they produce directly from the isocontours of electron density, represented as the sum of per-atom Gaussians and generated using Connollys AMS and TS programs. Derivatives are computed by convolving either the electron density or the surface normals with the derivatives of a Gaussian function. These surface normals are computed per-face, again, directly from the electron density function. Approximations of the principle directions and magnitudes of curvature are found on both surface representations at different scales by varying the size of the Gaussian (from 1Å to 5Å). The authors use both methods to compute and analyze the shape properties of a molecular surface, showing the value of multi-scale curvature. Though they do not discuss this, their method for computing the surface normal based on the weighted sum of normals in a patch could lead to

problems [40] (especially for larger scales). In regions with opposing normals, for instance, this could cause numerical instability. Fortunately, with enough samples (as is likely in their surfaces), conflicting normals have less effect on the final result, and spherical averaging is rarely necessary.

Olson, et al. [41] compute curvature volumetrically, directly on the atoms, by representing each atom as a gaussian. The size of the gaussian determines the 'blobbiness' of the resulting surface, with each value equivalent to representing that surface at a particular scale. Curvature can be computed directly from this representation, thus achieving a multi-scale curvature estimation. They show that ligand-interface complementarity holds across a range of scales, with maximal complementarity found at medium scales (equivalent to smoothing away atomic-level detail). Their method is simpler than ours because no solvent-excluded surface need be created and because gaussian sums are easy to compute. But in contrast to our method, their surface representation does not allow for finding surface correspondences between scales, as topology may change arbitrarily. This limits how easily their method can be integrated into a surface descriptor.

2.3 Local Shape Descriptors

In this section, I describe work done on geometric shape descriptors which relates to my work in Chapter 5. Most methods for molecular-surface description have focused on atomic models of the surface (operating on surface atoms or residues); my usage of the geometrical solvent-excluded surface distinguishes it from these previous methods. In turn, the methods described in this section, unless otherwise noted, were neither designed for nor used in biochemical applications.

Local shape descriptors are useful for a number of applications. Many have been developed, in numerous forms, for the task of shape matching, and share a subset of my goals. Körtgen, et al. [42] and Gatzke, et al. [43] both use the statistics over the neighborhood surrounding a point to perform robust matching. Unlike my method, the former does not directly utilize the surface. The latter, while more similar to mine, differs in that they build up their descriptor using differential curvature estimates, rather than estimating curvature on the patch as a whole. Spin-images [44] represent such a neighborhood as a 2D texture, which can be quickly built, and can be

used to perform rotationally-invariant matching. While fast, these are best suited for local feature comparison.

Gal [45] models the surface using local shape descriptors to construct partial matches, while [46] uses point signatures to detect rigid structures on a set of faces, which are then compared to one another. Goldman, et al. [47] describe a strictly local quadratic shape descriptor that they use for molecular similarity searching. All of these are local methods, and as such are not able to model larger-scale phenomena on the surface. See Section 5.4.4 for a discussion of why this is important.

Many graphics and visualization techniques use curvature, and therefore may benefit from an improved descriptor. Gumhold [48] describes an algorithm to optimize the placement of lights in a scene in order to emphasize high-curvature regions. Lee, et al. [49] have used curvature, along with globally discrepant lighting, to similarly emphasize the placement of specularities on the surface.

Toler-Franklin, et al. [50] describe how lighting derived from large-scale curvature can approximate local ambient occlusion, to better emphasize large concave features. Real-time rendering techniques also consider curvature: [51] show how extracting curvature information from image space can produce compelling lighting and shading in real-time, while [52] extracts curvature from volumetric data to enhance their visualizations.

Stylized rendering techniques benefit from the use of curvature as well. Principle curvatures can be used to depict the flow of curvature over the surface [53], and to then place textures along those flows, emphasizing important shape cues [54, 55, 56]. Line drawing techniques also utilize curvature [57, 58].

Curvature can directly inform another task, surface partitioning, by using high-water curvature as partition boundaries [59], or by placing seams in high-curvature regions to better hide them [60]. Mortara, et al. [61] uses the intersection of the surface and bubbles at multiple scales to estimate curvature for surface segmentation.

2.4 Functional Surface Analysis

It is hard to overstate the amount of varied research devoted to computational analysis of the functional surface. And while the idea of using a descriptor to characterize a region of the molecule is not novel, most techniques deal specifically with atomic arrangements, and not the geometric surface itself. Common among almost all these systems is the usage of electrostatic potential to understand function: see [62] for a survey of the reasons why electrostatics is important in protein-protein interactions.

2.4.1 Identifying Potential Binding Sites

Work to identify potential binding sites goes almost as far back as the work to visualize them. Though many overlap in classification, these can roughly be broken into the following categories: pocket detectors, which are designed to recognize pockets on the surface of a protein, but do not perform any surface characterization; binding-site predictors, which use chemiometric and/or geometric analysis of the functional surface to predict the location of ligand-binding; and alignment, matching and comparison techniques, which are designed to compare surfaces with one another, using information known about one surface to predict new function in another.

2.4.1.1 Pocket Detection

Many methods have been proposed to find pockets on the surface of a protein. LIGSITE [63] maps proteins to a 3D volumetric field, to identify potential ligand binding sites (pockets). A later extension, LIGSITEcsc [64], uses a notion of the Connolly surface and the degree of conservation (pulled from the ConSurf-HSSP database [65]) at particular surface points to improve performance. PocketPicker [66] also uses a 3D grid to detect pockets, casting rays to identify buried cells within the grid. The set of detected pockets is then refined by building a feature-vector, based on the shape and depth of clustered cells, and comparing that vector to known pockets. A later paper, [67] refines the method, and evaluates druggability on the PDBBind database [68], confirming that pocket size and depth are correlated with the presence of a ligand.

SURFNET [13] finds pockets by placing spheres into pockets; the spheres with maximal volume define the largest pockets. POCKET [69] and PHECOM [70] also use probe spheres. In a later paper, the same authors [71] use morphological operators on a grid representation to perform the same task at higher speed, and at multiple scales. PASS [72] uses probe spheres to fill cavities layer by layer, keeping probes with high atom counts after each iteration. [73] computes Voronoi diagrams of the exterior atoms to find pockets.

An, et al. [74] describe a system called PocketFinder, which can find and then characterize the shape of binding pockets using a smoothed approximation of the Lennard-Jones potential. They then use PocketFinder to construct a comprehensive database of pocket envelopes found in the PDB, called the Pocketome. Finally, they show how by clustering the Pocketome according to the shape and size of each pocket, they can predict potential drug targets for a unclassified pocket. My work shares their goal of characterizing the shape and electrochemical landscape of ligand binding. We differ in that ours uses supervised learning techniques, while theirs uses unsupervised. Further, my method, for better or worse, does not presume that binding occurs only in pockets.

For each of these methods, the goal is strictly geometric pocket-finding (and not characterization). Consequently, in contrast to my descriptor, physio-chemical features, such as electrostatic potential and hydrophobicity, are never considered. Also, no attempt is made to use this information to predict *which* ligand might fit in a specific pocket.

2.4.1.2 Binding Site Prediction

In comparison to simple pocket detection, in silico binding site prediction is considerably more complicated. A complete solution must address both physical issues (such as steric clashes, flexibility and conformational change) as well as electrochemical issues (such as hydrophobicity and electrostatics). A ligand may potentially bind at any position on a surface and at any relative orientation. Considering ligand flexibility, the space of possible binding configurations is enormous. Nevertheless, despite its difficulties, inferring protein binding affinity is an extremely important step in rational drug design; see [75] for a discussion.

In one of the first examples of binding site prediction, DOCK [76] performs a brute-force evaluation of potential ligand positions and orientations relative to a protein surface, attempting to minimize steric overlap. Later, Norel, et al. [77] utilize shape complementary as a means to match ligands with receptors more efficiently.

FEATURE [78], creates a descriptor of protein micro-environments using physical and chemical properties at multiple levels of detail (the atomic, chemical group, residue, and secondary structural levels), aggregating statistics in thin shells, each extending radially outward. Using these, they employ a Bayesian supervised approach to identify potential ligand binding sites. Later, they expand earlier work on calcium binding identification by using a modification of their previous approach to the task of identifying calcium binding sites in proteins, with a high degree of success [8]. In contrast to my proposed methods, they make no attempt to characterize the shape, and assume radial symmetry when collecting data in each shell.

Binding classification does require some care. On protein-protein interfaces and in DNA binding, complementarity is often found, as shown by Shahbaz, et al. [79]. However, a 2002 paper by Ma, et al. [80] show that, while ligand recognition does generally derive from shape complementarity, the shape of the binding pocket is influenced heavily by the shape of the ligand bound to that pocket. Consequently, many pockets many not bind with just one, but a multitude to potential partners. Kahraman, et al. [81] show that geometrical complementarity in general is not sufficient to drive molecular recognition. And Mobley, et al. [82] demonstrate the instability of the energy landscape of small ligand binding interactions to changing ligand shape and orientation. My method attempts to overcome these limitations by incorporating information at multiple scales; at larger scales, the surface exhibits less shape instability, and complementarity is more likely to hold.

SITEHound [83] uses Molecular Interaction Fields (MIFs) [84] to predict ligand binding sites. The type of binding site produced is dependent on the type of probe used to construct a particular MIF. This idea inspires some of my work, in that I tune my binding prediction to the particular environment for a given atom. In a followup paper [85] they describe a web server that implements their ideas.

TCBRP [86] maps binding pockets to a query protein by a combination of sequence similarity and structural information. Homologous proteins are first identified by sequence similarity. Existing pockets in those proteins are then classified, and mapped back to the query protein based on sequence correspondence; these are predicted to be pockets. Structural information is used to identify binding residues.

Nassif, et al. [87] train a SVM classifier using FEATURE descriptors to predict glucose binding. They show how by applying supervised learning techniques can produce accurate predictions of glucose binding. An important aspect of their work is the analysis of how much each specific electrochemical feature impacts binding prediction at each scale. Interestingly, their method does not use surface shape as a feature.

Hoffman, et al. [88] propose a similarity measure based on convolving the cloud of atoms between two pockets. This score is then used to attempt to classify new pockets by their similarity to pockets known to bind to specific ligands. My methods use a similar approach, using a set of exemplars to model the binding of a specific ligand, though I take a more atomistic (vs pocket-based) approach. A similar method by Chikhi, et al. [89] uses compact pseudo-Zernike and Zernike descriptors as a basis for scoring pocket similarity. Because Zernike descriptors are very compact, requiring no pose normalization, they provide for very fast search.

While each of these methods address some facet of this large task, none completely solve the problem, and there is significant room for improvement. My proposed method contains several key improvements. In contrast to Norel [77], my method allows for ligand flexibility in matching. Unlike Chikhi [89] and Hoffman [88], I make no assumptions that binding occurs on pockets, or about their shape and size, and am therefore more likely to find binding outside of pockets. Like TCBRP, my method utilizes the surface, but because it does not assume structural homology, it is not limited to testing homologous proteins.

My method does have limitations: unlike DOCK [76], it does not return back a specific binding arrangement, but only a prediction of the *location* of binding on the surface. Compared to methods such as [89], which make speed a priority, my method for binding prediction takes several orders

of magnitude longer to run. While the method of Nassif [87] and mine share many similarities, theirs significantly outperform mine. I compare these results in Section 6.6.2.3.

2.4.2 Alignment, Comparison and Classi cation

An early paper by Lawrence, et al. [90] uses a statistical descriptor to evaluate shape complementarity at protein/protein interfaces. This descriptor, however, only contained a rather simple, local description of shape, and no notion of other chemical properties.

Several papers use spatial transforms to map surfaces into a more tractable format. PEST [91], or (Property-Encoded Surface Translator) utilized a fragment-based wavelet coefficient descriptor, while Morris [92] uses a technique for the comparison of protein binding pockets using the coefficients of a real spherical harmonics expansion to describe the shape of a protein's binding pocket. Shape similarity is computed as the RMS distance in coefficient space. Unlike my descriptors, however, spherical harmonics descriptors are unable to capture the shape of neighborhoods that are not homotopically equivalent to a sphere. This precludes them from accurately describing flat regions, and pockets of high (local) genus.

COMPASS [93], predicts biological activities of molecules based on the activities and three-dimensional structures of other molecules. Their technique represents only the "surface" of a molecule, defined as those atoms nearest to exterior of a molecule. Similarly CASTp [94] attempts to infer functional relationships between proteins via their pockets: their method first triangulates the surface atoms using alpha shapes, grouping triangles into larger collections; a pocket, then, is defined a collection of empty triangles. Surface amino acid patterns for these cavities form the 'description' of the pocket, and are used in a bag-of-words model to compare against other pockets.

Other work, such as RECON [95], uses pre-built descriptors for different atomic charge-density configurations to allow for quick reconstruction of molecular charge densities and charge density-based electronic properties of molecules. PEST [91] expands on this work to create hybrid, alignment-free shape-property descriptors. This work is ultimately integrated into a support vector machine to support virtual high-throughput screening. COLIBRI [96] utilizes RECON descriptors to identify complementary ligands/binding sites, formed by using Delauney tessellation to isolate

the protein atoms that make contacts with bound ligands. RECON descriptors are compact, but unlike mine, are not inherently multi-scale and do not specifically consider the functional surface. Given my assumption that protein interaction primarily takes place on the functional surface, this latter property means that RECON, like other volumetric descriptors, end up characterizing the less-meaningful subsurface atoms within a protein along with more-meaningful surface atoms.

FADE [97] used Fourier transforms to quickly compute an estimate of the atomic density at a collection of arbitrary points about a molecule. Shape descriptors can be used to analyze a single molecule or to evaluate shape complementarity. PADRE, also described in [97], uses a similar method to compute topographical information, which can be used to identify crevices, grooves and protrusions. Their formulation is inherently volumetric, and so is valid everywhere, not just at the surface. Later KFC [98] combined FADE with knowledge-based biochemical contact analysis to more accurately predict mutagenesis hotspots for protein/protein interaction on the surface.

SURFCOMP [99] builds an association graph, made up of the correspondences between convex and concave critical points on two surfaces to be compared. Harmonic shape image matching is used to detect locally similar regions from within this graph, augmented with properties such as the electrostatic potential or hydrophobicity, to compare the two surfaces. This technique shows the promise of whole-surface comparison. It is tuned toward whole-surface comparison, is exceedingly compact, but unlike mine, is likely not descriptive enough to handle ligand binding prediction, as the surface graphs constructed are very sparse.

Several papers describe the construction of large-scale databases for the purpose of shape-comparison. The CATH (Class, Architecture, Topology, and Homologous superfamily) database [100] contains hierarchically classified structural elements (domains) of the proteins stored in the PDB. The CATH system uses automatic methods for the classification of domains, as well as manual classification by experts when automatic methods fail to give reliable results. The pvSOAR [101] website allows users to input a surface pattern as a query against the CASTp pocket database.

Bock, et al. [102], use spin-image profiles of points on the surface of a molecule to perform geometric shape matching to find other, similar points on other proteins. Their approach makes no use of the physicochemical properties on the surface, unlike my proposed approach, but their

results show that geometric comparison alone can still help identify regions of interest, giving structural biologists targets for deeper inspection.

S-BLEST [103] allows for annotation of previously-uncharacterized proteins by encoding the local environment around individual amino acids in a protein into a type of descriptor. Novel proteins are found by performing a K-nearest-neighbor search against a known protein database to find similar amino-acid environments, and from them proteins of potentially similar function.

Many algorithms attempt a global surface alignment across proteins. Baum, et al. [104] focus directly on the molecular surface for alignment. Their algorithm aligns point sets on two surfaces, each computed using an iterative refinement of Voronoi diagrams. They show that relatively sparse sets of points can still achieve good alignment results, which gives us confidence in my own surface sampling techniques. Yi Fang, et al. [105] adapt the Local Diameter (LD) descriptor, first described in [106], for comparison of flexible proteins. They show that by considering flexibility, they achieve better recognition performance than by rigid alignment. My method is designed to allow for flexibility as well, both on the part of the ligand and on the part of the binding pocket.

Chapter 3

Background

The work in this dissertation builds on a large body of work in molecular biology and computer science. In this section, I define concepts that I use as the basis for my research.

3.1 The Geometric Surface

It should be noted that the molecules and atoms that we work with are very small objects, and as such, fall into the realm of quantum mechanical interactions, where positions and boundaries are not precisely known. Thus the bounding surface, which delineates the interior from the exterior of such a molecule, is not as easily defined as, say, the surface of a rubber ball. Nevertheless, the molecular surface can be defined as the *boundary outside of which the molecule only shows weak non-covalent interactions with another molecule*. This definition will be refined in the next sections.

3.1.1 The Van der Waals Surface

According to the definition derived from van der Waals' study of the deviation of real gases from their ideal behavior, we can model each class of atoms in a protein as a hard sphere with a specific radius. This is the "van der Waals radius", which demarcates the exclusive region for an atom, where no other atom may reside. At short distances, the repulsion between two atoms increases rapidly due to an overlap between their electron clouds, which increasingly violate the Pauli exclusion principle.

In molecular mechanics interactions, the van der Waals energy is usually described in terms of the Lennard-Jones potential:

$$V(r) = 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right] \quad [3.1]$$

where **r** is the interatomic distance, ϵ is the depth of the potential well, σ is the (finite) distance at which the inter-particle potential is zero. Typically, ϵ and σ are fitted to reproduce experimental data.

A surface can be built directly from the union of the van der Waals spheres surrounding each atom, such that the interior of the surface consists of mainly covalent interactions between the atoms of the protein, and the exterior of non-covalent interactions. Such a surface, however, does not provide a lot of extra information: it mainly serves to give the atom points a volume.

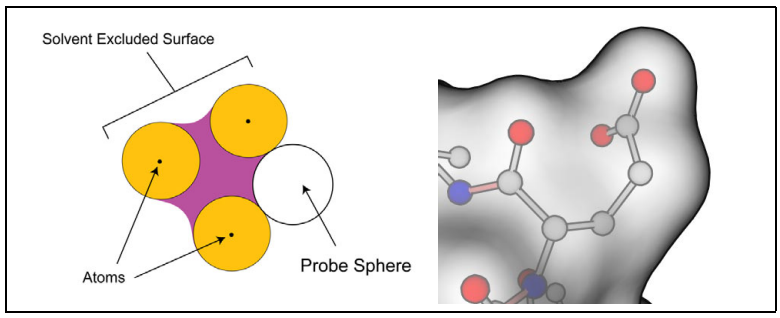
3.1.2 Solvent-Excluded Surface

The van der Waals surface also does not clearly define the interior and exterior in terms of accessibility: cracks and pockets may exist on the van der Waals surface where no covalent interactions are taking place, but where nevertheless an atom or molecule may not fit due to steric clashes. This is a crucial component of the activity of a biomolecule, as its properties are influenced by effects that involve — directly or indirectly — the presence of a solvent, typically water.

The "Solvent-Excluded Surface" models these effects directly. In contrast to the van der Waals surface, which simply divides the space containing covalent interactions from non-covalent interactions, the solvent-excluded surface divides those regions that are exposed to the solvent from those regions that are hidden. The bounding surface then represents the points of contact between protein and solvent.

This surface is described as follows: first, the desired solvent molecule is represented by a probe, a sphere whose radius is chosen to approximate the size of the solvent. Usually this is chosen to be 1.5 Ångströms, the size of a water molecule. Unless otherwise noted, this is the case

for all figures in this document. Next the probe is rolled over the van der Waals surface of the molecule.



Lee and Richards [107] defined the "Solvent-Accessible Surface" by the trace of the center of the sphere at each point of contact. This surface is built by simply extending the van der Waals radii of all atoms by the radius of the probe sphere. Unfortunately, this surface is not smooth (i.e., C^1 continuous), and further, does not precisely represent the interface between the protein and solvent.

The solvent-excluded surface was later developed to be a better representation of the true molecular surface, using the points of contact themselves, rather than the center of the probe sphere. A major advantage of this representation is that it is (for the most part) smooth, while still conforming to the shape of the van der Waals surface. For this reason, all surfaces in this document are (or begin life as) solvent-excluded surfaces, generated using Connolly's MSMS program [108].

3.2 Electrochemical Properties

In addition to their shape, protein functional surfaces exhibit a number of electrochemical properties that play a large role in determining the function of the protein. In this section I describe those that appear in my molecular surface descriptor. Though not a complete list of functional surface properties, these are commonly used as features in binding-prediction tasks because they exhibit a high degree of complementarity between ligand and binding surface [109], and are thus

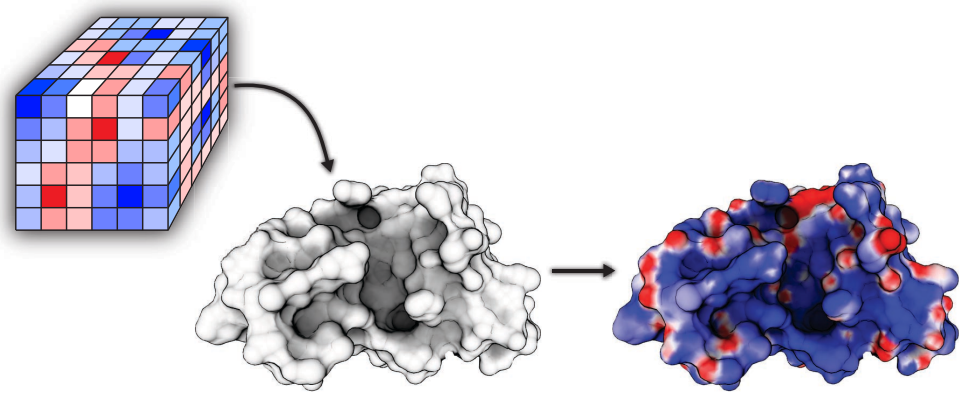


Figure 3.1 How electrostatics are sampled onto the surface. At left, the result of an Adaptive Poisson-Boltzman potential computation, which takes the form of charges sampled into a regular grid. The protein surface is embedded into this grid, and for each vertex, the charge is computed by looking up the cube that this point occupies, and performing trilinear interpolation of the potentials at the corners this cube.

necessary for fully characterizing binding interactions. For this reason, these are the electrochemical properties I use in my descriptor formulation (Section 6.2.2).

3.2.1 Electrostatic Potential

Atoms in isolation have a charge which is proportional to the difference between the number of electrons forming the outer 'cloud' surrounding its nucleus, and the number of protons in the nucleus. In a molecule, the density of each atom's electron cloud is changed due to electron donating and withdrawing between molecular bonds. Thus each atom in a protein can be considered to have a partial charge, which may be different than its charge in isolation.

The "electrostatic potential" of a protein describes the potential energy of a unit charge at a point in space, in a field generated by each atom's point charge. This potential has critical implications to formation of non-covalent bonding: the binding specificity that a ligand has for a region of the functional surface is proportional to the free energy of that configuration. Free energy is lowest when the point charges for each atom in the ligand within the electrostatic field are complementary.

The electrostatic computations within my system have two phases. First, the partial point charges must be assigned to each atom within the protein. If the protein file doesn't already include hydrogens (which many don't), the positions of these missing hydrogens is calculated. Force fields can then be computed in a number of different ways (AMBER94, CHARMM27, PARSE, etc.). Because of its more-accurate charge models [110], AMBER94 is used to compute all electrostatics in this document. In my system, these steps are handled by PDB2PQR [111, 112].

The effects of the solvent on the electrostatics of a protein in an ionic solution (such as water) can be significant, and should be accounted for. The PoissonBoltzmann equation is a differential equation that describes these electrostatic interactions. Because it is not possible to solve this equation analytically, typically a grid is constructed spanning the dimensions of the protein, and the final potential is computed on that grid. I use APBS [113] (or Adaptive Poisson-Boltzmann Solver) for this step.

Once electrostatics have been computed, they are sampled onto the surface. See Figure 3.1 for a diagram of this process.

3.2.2 Hydropathy

The hydrophobic effect plays an important role in the formation and interaction of the surface of a protein. Hydrophilic amino acids tend to appear on the surface of the folded protein, as they can form transient hydrogen bonds with water which stabilize the structure. Conversely, hydrophobic amino acids tend to be internal to the protein. I use the score formulated by Kyte and Doolittle [114] which maps hydropathy to a scale of -4.5 (least hydrophobic) to 4.5 (most hydrophobic).

My system maps hydropathy to the geometric surface by first deriving this score for individual residues in the protein structure, and then, for each vertex, using the score of the nearest residue to that vertex.

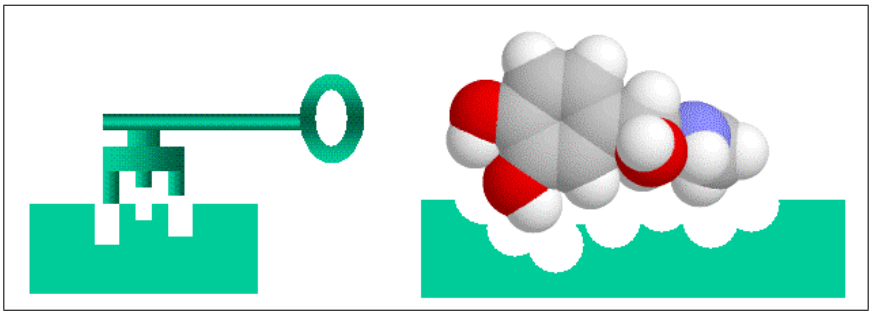
3.2.3 Hydrogen Donors/Acceptors

Proteins are further stabilized by internal hydrogen bonds, which are attractive interactions between a side-chain hydrogen atom and an electronegative atom, such as nitrogen or oxygen.

These hydrogen bonds can form between amino acids that are far apart in the protein sequence, and frequently result in meaningful structures: alpha helices, for instance, are formations which arise when regular hydrogen bonds occur between the residues in positions i and $i+4$ in the protein sequence.

Hydrogen bonds can also form between atoms of the protein and outside binding partners. When available, these "external" hydrogen bonds further help stabilize binding interactions.

3.3 The Lock and Key Metaphor



Recognized by Emil Fisher at the end of the 19th century, the "Lock and Key" metaphor describes the conditions necessary for binding to occur between small organic compounds (ligands) and larger proteins. Much like a lock only recognizes the specific shape of its corresponding key (or, perhaps, a range of keys), binding sites on the surface of a protein are highly specialized to favor interaction with specific partners. Further, both sides of a binding interaction must have complementary features: where geometric peaks appear on one side of the interaction, valleys must appear on the other; positive charges must match negative; and, whenever they are used, hydrogen bond donors must match acceptors.

This complementarity implies an important property that I will take full advantage of in Chapter 6: because a site that matches a certain ligand must be complementary to that ligand, then we expect all sites matching that ligand to share similar shape and charge configurations. Therefore, for example, a positively charged atom in some ligand will prefer (all else being equal) binding to a negatively charged surface, and we can limit our search to negative patches.

Chapter 4

Molecular Surface Abstraction

As described in Section 1.1, one goal of structural biology is to understand the chemical and physical properties of macro-molecules (especially proteins) and how this enables the chemical reactions behind life's processes. In order to study these large and complex molecules, biochemists rely on visualizations that provide various levels of abstraction. The more abstract visualizations portray a molecule's internal structure. However, protein interactions involve the "functional surface" presented: to a large degree, the internal structure simply exists as scaffolding to place various forces and chemical properties in proper spatial relationships with one another.

While visualizations of these functional surfaces exist, they portray all of the detail and complexity of large molecules. The complexity of these visualizations is problematic as they do not afford rapid assessment, and details may obscure larger scale phenomena. To date, the degree of abstraction provided for internal structure has not been shown for external properties.

In this chapter, I describe a method that provides for abstracted views of the boundary of a molecule and the physical and chemical properties at this boundary. I call the result an *abstracted molecular surface.* My goal in creating this visualization is to provide simplified visual representations of molecules such that scientists can rapidly assess the most significant features of their surfaces, even when drawn at a small size.

This chapter will be broken into three sections. First, in Section 4.1, I describe the method itself, and show how such abstracted views are useful for studying molecules while unencumbered by small details. Next, in Section 4.2, I describe a client-side application for exploring molecular abstractions, including a method for aligning and displaying multiple copies of the same protein. Finally, in Section 4.3, I describe a web server that lets any researcher to easily create and view an

abstract representation of any protein, effectively lowering the barrier of entry for anyone wanting to try abstractions for themselves.

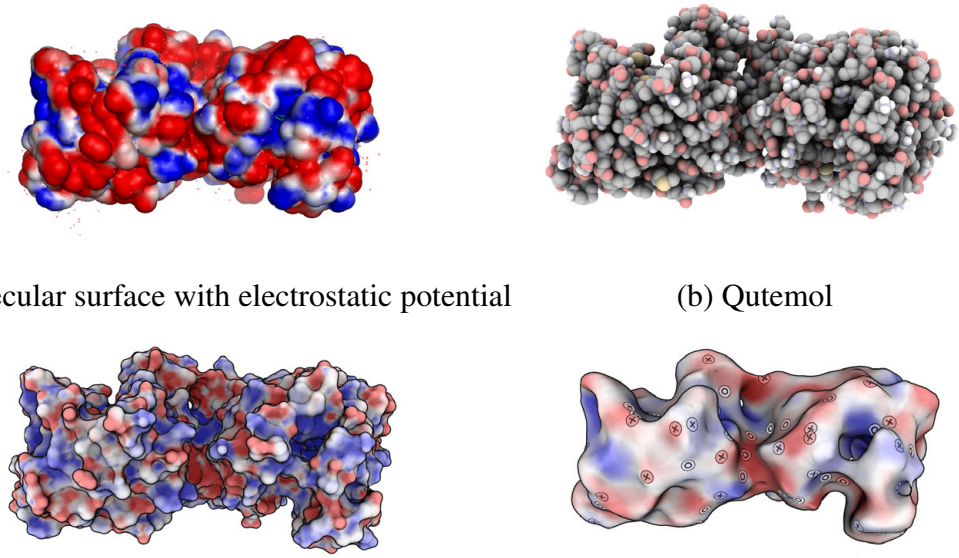
All of these sections share in common the same abstraction mechanism, which processes the detailed information about the molecule to provide a visually simplified representation. The shape of the molecular surface is simplified, removing small details to better convey the basic shape. Significant shape features, such as clefts and pockets, become more prominent when the visual clutter of smaller features is removed. A comparison with other display methods is shown in Figure 4.1. Additionally, abstractions are more amenable to stylized rendering that accentuates shape, and their lower curvature allows for the use of surface markings to display other information. Finally, their lower amount of surface detail should lead to better readability at lower resolutions, allowing gallery displays.

Molecular surface abstraction also applies abstraction to properties other than shape. Scalar fields along the surface, such as electrostatic potential, are simplified for clarity, and other properties are displayed as symbols on the surface. These presentations allow significant features to be seen quickly and clearly.

I am motivated by an increased need for tools that enable quick and comparative visual analysis. Advances in structural biology, such as high-throughput crystallography and NMR spectroscopy, together with better prediction and simulation, have led to a marked increase in the number of proteins for which the three-dimensional atomic structure is known. Repositories for structural information, such as the PDB [4], have in turn grown dramatically. This wealth of information creates the need to look at large collections of molecules, requiring quick judgement.

4.1 Abstracted Surfaces

In this section, I describe the general abstraction algorithm. This work was published in the 2007 IEEE Transactions on Visualization [115], which introduced the concept of molecular surface abstractions, and described methods for creating them.



(a) Molecular surface with electrostatic potential

(b) Qutemol

(c) Stylized display

(d) Abstracted

Figure 4.1 Depictions of the surface and electrostatic potential distribution of Adenylate Kinase (1ANK). The standard approach (a), drawn with Pymol [5], shows the molecular surface pseudo-colored from red to blue, representing negative to positive potential, respectively. Qutemol [28] (b), applies stylized lighting to a space filling representation. (c) applies stylized shading directly to the molecular surface. (d) shows an abstracted surface depicted with stylized rendering. This molecule has binding partners that fit into channels formed in each of its lobes. From this view, one such channel should be visible in the center of the right lobe. This is made more readily visible by abstraction.

The abstraction process removes small details from molecular surfaces and their associated properties. These details are unlikely to be biologically significant, but nevertheless detract from a viewer's ability to interpret larger patterns.

Section 4.1.3 describes the process for abstracting surface geometry. In this section, I adapt standard methods for shape smoothing by explicitly removing regions where the methods are likely to fail. Section 4.1.4 describes how surface decals can be used to display removed features, as well as other information. Section 4.1.2 describes the methods I use for more effectively portraying scalar fields on the surfaces.

To create an abstracted representation of a molecule, my approach takes as input a triangle mesh of the molecular surface as well as information about the properties of the molecule. My implementation uses external tools to create these inputs from PDB files. For the figures in this document, MSMS [116] was used to create surface meshes and APBS [113] was used to compute electrostatic potential. The triangle mesh must be sampled finely enough to appear smooth at the scale of interest, but need not be uniform. Properties are associated with mesh vertices; scalar fields are sampled at these points before abstraction.

4.1.1 Smoothing Surface Geometry

The primary step of abstraction is to remove small details in shape by smoothing the mesh. The choice of the scale of "small" is chosen to be smaller than a residue, but larger than an atom. My implementation uses Taubin's filter [117, 118] as it is efficient and sufficiently volume-preserving. Local operations are performed on vertices lying in disc surrounding a given point, with importance falling off with the inverse of distance, as given by Fujiwara [119].

Taubin smoothing requires several parameters be set, each of which affects the degree of smoothness in the final mesh. First, a disc radius R must be chosen: larger radii take into account more points when averaging, increasing the effect of smoothing. λ is chosen as the blend weight between the starting surface points and the points produced by the umbrella operator. With $\lambda = 0$, no smoothing is applied. $\lambda = 1$ indicates that the weighted average over the disc is returned

back for each vertex. Values between 0 and 1 result in a weighted blend between these two positions for each vertex. μ is always a function of λ, and reflects the amount of 'unsmoothing' needed to 'expand' the mesh back, to achieve volume-preserving smoothing.

I have empirically determined that the filter parameters based on the size of the features I wish to remove: $R = 4$Å, reflecting the approximate size of 'mid-sized' features, such as most amino acid sidechains. Choice of λ directly affects the frequency of the band-pass filter. $\lambda = .8$ (with $\mu = -.87$) was chosen to reflect a degree of smoothing that works well for visualization and for later parameterization of the surface. Finally, given these parameters I found that 10 iterations are needed to ensure that all offending features are removed, such as protruding side-chain moieties, 'fingers' from loose terminal chains and pockets smaller than the size an atom.

These parameters are a function of the scale at which I am interested in studying (i.e. larger than atomic scale interactions), not of particular molecules. All examples in this document use the same parameter values. While parameter tuning might produce better results, empirically, the current version produces abstraction results that I have found to be both useful and visually pleasing.

The smoothing process creates a potential problem: "Mid-sized" features that are larger than what is removed reliably by the filter are often distorted by it. These features, such as peaks formed by protruding side chains and small divots, may be biologically significant. While a more sophisticated filtering mechanism might better preserve them, even undistorted, these features are undesirable for abstraction as they are still difficult to portray, and, because of their higher curvature, more difficult to parameterize. This latter property makes them a less-than ideal canvas on which to apply stickers.

To handle mid-sized features, I apply a different strategy shown in Figure 4.3. My approach identifies these features and removes them from the mesh, leaving a smoother surface. However, it remembers that a feature was removed and depicts that feature using a surface marking. This approach has the advantages that it avoids artifacts from filtering and provides more control over how features are displayed. Small surface markings are better for abstracted representations than

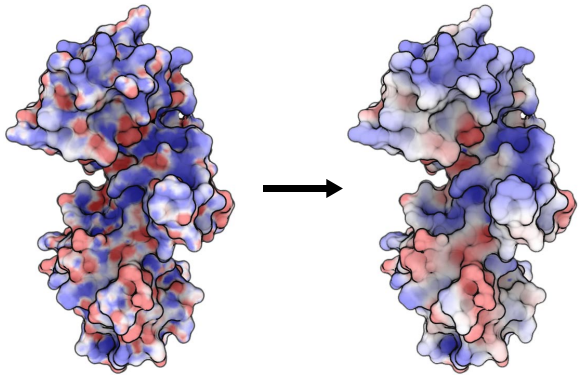


Figure 4.2 On the left, a molecular surface with electrostatic potential sampled from an underlying electrostatic field. On the right, after surface field abstraction. Note the preservation of large areas of charge, and the absence of high-frequency coloring.

small bumps and divots because they do not detract from the overall shape and are visible from a wide range of viewpoints (see Figure 4.4).

4.1.2 Abstracting Surface Fields

Many important properties beyond the atomic forces that form the molecular surfaces, such as electrostatic potential and hydrophobicity are typically represented as scalar fields on the molecular surface and displayed using pseudo-coloring. Such properties suffer from the same profusion of small details as the shape itself, with similar issues in comprehension and display at small size.

Therefore, I abstract the scalar fields on the surface of the molecule. The scalar properties are attached to vertices before any simplification. This is particularly important because the true geometry determines the value (i.e. the position used to sample a volumetric scalar field such as electrostatic potential). Simplification should account for the real geometry in finding features (such as regions) in the scalar field as property values attached to vertices can be moved by the shape simplification. While this will distort the shape of the field, significant features of the field, such as regions of large magnitude, will remain with approximately the same shape and value.

The field abstraction process aims to remove small, less relevant details but to also preserve the coherent regions where the field has a definite value. Therefore, my system applies a boundary-preserving low-pass filter to the surface scalar field. Specifically, I adapt the bilateral filter[120] to the irregular lattice of the triangle mesh.

As in the image case, the bilateral filter computes a new value at a vertex by taking a weighted average of the other vertices in its neighborhood, where these weights are determined by using both the spatial and value differences. I achieve larger kernel sizes by iterated application of smaller ones.

The general formula for a bilateral filter at iteration i for vertex v with value $val_i(v)$, where $d(v, w)$ represents Euclidean distance from vertex v to w, and $N(v)$ represents the set of v's neighbors, is:

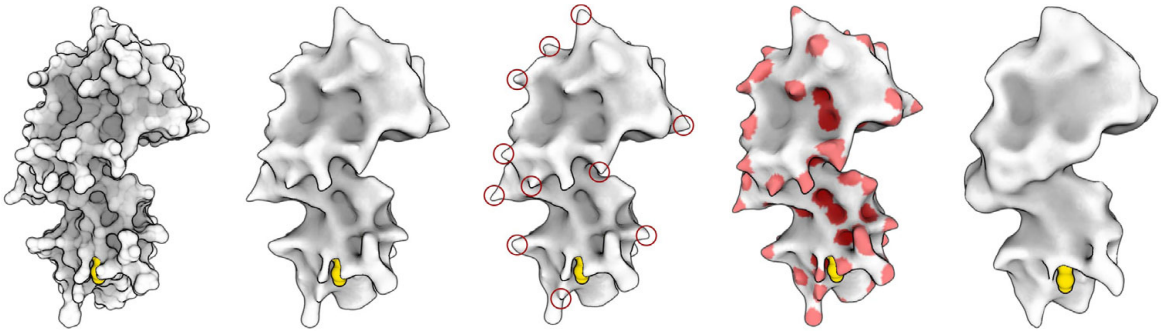
$$val_{i+1}(v) = \frac{1}{k_i(v)} \cdot \sum_{w \in N(v)} val_i(w) \cdot c_i(w, v) \qquad [4.1]$$

$$c_i(w, v) = e^{-\frac{d(v,w)^2}{2}} \cdot e^{\frac{(val_i(v) - val_i(w))^2}{2 \cdot \sigma_i^2}}, \quad k_i(v) = \sum_{w \in N(v)} c_i(w, v) \qquad [4.2]$$

Thus, my system applies a Gaussian filter for both distance and value-similarity weights. For the latter, though, I have found that by using a larger kernel (σ_i) for the first few iterations and then progressively reducing it at later iterations, I can prevent areas of uniform value from completely diffusing into areas of different value.

My method to adapt kernel size is to use a kernel proportional to the standard deviation over the values at each vertex. Since the variance of the values themselves will be converging as a result of smoothing, this results in a progressively smaller sigma, which in turn gives higher weight to the value-similarity kernel.

The algorithm iterates until asymptotic convergence, which is reached when the average over all vertices v of $val_{i+1}(v) - val_i(v)$ is less than ϵ. This value determines the overall field smoothness, and can be adjusted to attain a desired level of abstraction. In all figures in this document, $\epsilon = .005$, chosen empirically to provide a good balance between fidelity to the original field and smoothness.



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

Figure 4.3 Steps involved in geometric surface abstraction. The original molecular surface (a) is first smoothed (b). "Mid-sized" peaks and bowls are identified (c), and filtered to produce patches of 2.5Å in diameter (d). Those patches are removed from the original surface by repeated edge contraction, then smoothing is applied to produce the final abstracted geometry (e).

4.1.3 Removing Mid-Sized Features

The next step in the abstraction process is for mid-sized features to be identified and removed. At present, my methods identify bumps and bowls that are large enough to be potentially interesting, but small enough to be problematic for smoothing. Similar approaches could be applied for other shape features, such as ridges and valleys.

The process for removing bumps and bowls consists of several steps, illustrated in Figure 4.3. These features are identified by finding points that have high curvature after smoothing, which is indicative of a filtering artifact. An initial round of smoothing is applied specifically for feature detection. Vertices whose curvature are outliers are chosen as features.

The system first computes the 10th percentile of the absolute value of principle curvatures ($\kappa 1$ and $\kappa 2$) for all vertices over the mesh. Vertices are chosen to be outliers if their Gaussian curvature $K = \kappa 1 \kappa 2$) is greater than P times the square of this value. Empirically, I have found $P = 30$ to provide a good balance between mesh smoothness and overly aggressive feature removal. The exact value of this parameter does not matter since precise identification is unimportant; an excess or missed point is likely to be grouped with another point in the next stage.

For each of these seed vertices, my system constructs a group containing other vertices within 2.5 Ångströms along the surface. This distance was empirically found to correspond to the approximate size of most high-curvature features. If groups overlap, then it is likely that they are larger aggregates of individual features on the original mesh, so the process repeatedly merges groups until no overlaps are found.

This step results in a collection of patches on the surface representing mid-sized features. These regions are then removed from the original, unsmoothed mesh. To "sand them off," my system removes the majority of the vertices in the region and simultaneously "deflates" those that remain. To accomplish both, it first sorts the vertices according to how far away they are from their nearest seed vertex. It then takes the closest 80% and removes them, one by one, by edge contracting each with its closest neighbor in the graph, provided this contraction doesn't cause topological problems. This ensures that smaller triangles will be removed first, and also that vertices will be removed top-down, in the case of peaks, or bottom-up for divots.

The edge contraction process produces a mesh with its mid-sized features "sanded off" but the removal process often negatively impacts mesh connectivity, leaving many high-order vertices. My system performs edge flips to improve the mesh by identifying high order vertices and for each one flipping its outgoing edge that is connected the highest order neighbor. It also finds extremely low-order vertices and contracts the outgoing edge connecting to their lowest order neighbor.

It's important to note that when a mid-sized feature is removed, scalar field information contained in that feature is lost. This information is potentially important: for example, protrusions with significant electrostatic potential may be biologically significant. To preserve this important field information, the removal process associates a field value with the decal representing a removed feature, determined by averaging the values of that feature's vertices.

After these methods remove mid-sized features, the resulting surface is smoothed, as before using a lambda/mu filter. This entire process may be repeated, if desired, to achieve an even more aggressive visual abstraction. All figures in this document, however, use a single pass of geometric abstraction.

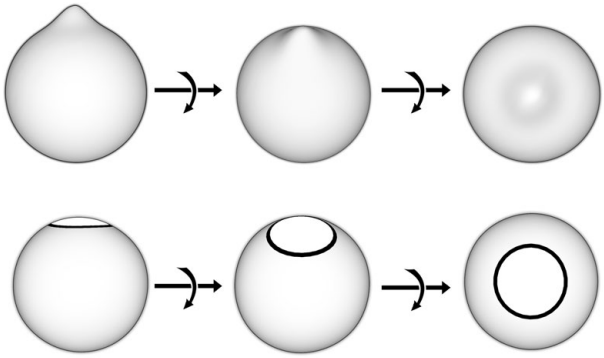


Figure 4.4 A sphere with a bump facing upward, rotated forward $45°$, and at right $90°$. Top: traditional geometric display. Bottom: an abstracted view makes the location more apparent.

4.1.4 Decaling

Decals are surface markings that are independent of the underlying triangulation. This is an important property: otherwise, a coarse or uneven triangulation might lead to jagged, irregular shaped markings.

Texture mapping provides for surface markings independent of triangulation, but requires a parameterization of the surface to provide texture coordinates. Unfortunately, molecular surfaces are difficult to parameterize globally, owing both to their size and arbitrary topology, so I instead use the local approach of [121] to place textures on regions of the surface. Their approach, called Exponential Maps, creates a local parameterization of a region of the surface. This approach works well for my needs because abstracted surfaces are relatively smooth, and because the markings I wish to apply are local.

4.1.4.1 Decal Parameterization

[121] use a discrete exponential map to create a local parameterization of a surface in the neighborhood of a point. Exponential maps take a point on the surface and map the surface surrounding that point to its tangent plane (see Figure 4.5), in a manner that yields mappings that preserve distances well. That plane serves as a local parameterization of the surface, and can be used to apply a texture with minimal distortion.

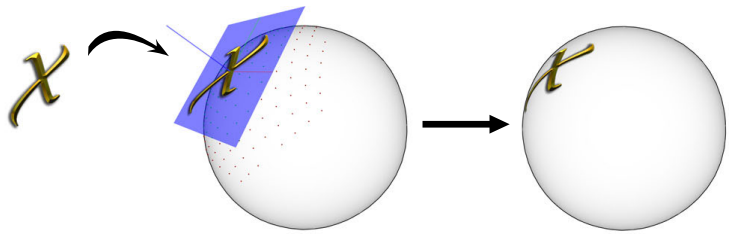


Figure 4.5 The abstract surface is textured using local parameterizations generated using an exponential map. First, a plane is constructed tangent to the desired position of the texture. Next, points surrounding that point (here in dark red) are mapped to that plane. Finally, the texture is placed on the surface according to that map.

Exponential maps were presented to support interactive decal placement. To apply it within the molecular abstraction process, the process of choosing the seed point must be automated. These issues are challenging because poor choices may lead to parameterizations that distort the textures as they get further from the seed point.

To solve these problems my system attempts to locate an ideal starting vertex within the patch. Two competing goals intersect here: this vertex should lie as close as possible to the center of the region it represents, and also the normals on the surface should deviate as little as possible from its normal. This latter property is much more important to the overall quality of the parameterization, so my system first removes from consideration any vertices where it doesn't hold (i.e. $N_{vertex} \cdot N_{plane} < .05$).

If all vertices are removed, then the patch cannot form a good parameterization, and so that patch is not shown. Otherwise, the starting vertex is picked that has minimal distance to its most remote neighbor, which most often is a vertex lying roughly in the center of the patch.

4.1.4.2 Choosing Decal Placement

I consider two types of markings: fixed sized glyphs centered at a point, and arbitrary shaped regions. The former are used in my system to display symbols, such as circles and checks, to denote various features on the surface. To create a glyph decal, the point position is used as the seed for creating the parameterization.

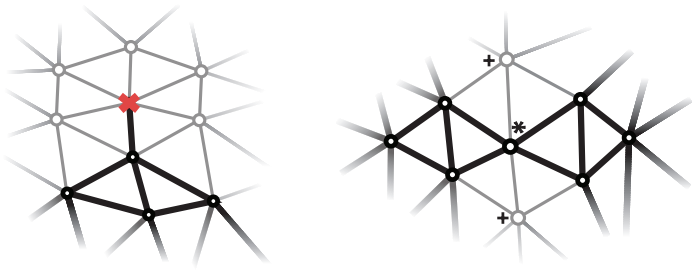


Figure 4.6 At left, a vertex in the patch has only one other neighbor that is also in the patch. These are removed. At right, a vertex, denoted by a '*', joins two otherwise locally disconnected sets. Its neighbors, denoted by a '+', will be added to the patch.

Regions are represented as a subset of the mesh vertices. To create a decal corresponding to a region on the surface, my approach selects the best vertex (using the criteria in Section 4.1.4.1) and builds a parameterization around it. This parameterization determines where each of the vertices in the region lie in the texture plane, providing a 2D mesh that can be drawn on that plane. This patch is drawn to a texture such that the region outside of it is made transparent with alpha blending.

The shape of a feature may have been distorted by both the filtering operations to create the smooth mesh and the mapping process, which may lead to patches with small holes, disconnected or poorly-connected vertices, and a jagged boundary. Removing these artifacts leads to abstracted markings that not only prevent problems in display, but also fit better into an abstracted representation and dispel any illusion that the fine details of the patch boundary are significant.

My system abstracts patches in a number of steps. First, it applies standard binary image processing operations adapted to the non-uniform lattice of the 2D mesh. Then it uses morphological operations [122] to remove outlying points and fill in small niches and holes. Dilation and erosion operators are defined based on the neighbors of a vertex. One step of dilation expands the patch out to include all immediate neighbors of the outermost vertices, while one step of erosion contracts the patch to remove all outermost vertices.

Rather than defining larger structuring elements, these immediate connectivity operators are applied repeatedly: 4 iterations of the close operator (dilation followed by erosion) provide a good balance of problem removal and shape preservation.

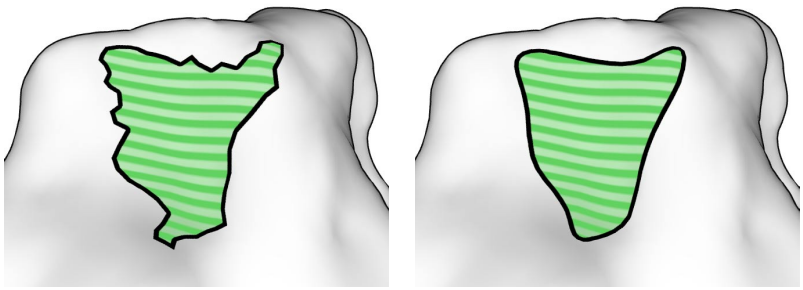


Figure 4.7 At left, a patch before boundary smoothing. Nodes on the boundary are placed directly on vertices on the mesh, leaving a jagged exterior. At right, after smoothing.

Morphological operators may leave thin threads and bridges, as shown in Figure 4.6. These are removed by eliminating vertices with only one connected neighbor, and by expanding the patch around bridge vertices, which are defined as any vertex in the patch that has at least two neighbors not in the patch, and which do not themselves share a neighbor that is not in the patch.

After these cleaning steps, my system then finds all closed loops that lie on the border of the patch. This boundary is then smoothed (in the 2D map) by applying a low-pass filter to the 2D positions in the chains. This boundary is drawn with a stroke around its edge and the enclosed region filled, either with a flat color or with a texture defined over the plane. See Figure 4.7 for an example.

4.1.4.3 Using Decals

Decals help to present additional information about the molecule in several ways. Because decals are semi-transparent, they overlay nicely on one another. However, displaying too much information may lead to clutter, so my system can optionally disable certain types of decals, if desired. Decal positions can be determined from a number of tools, or can be provided manually for annotation. New methods for identifying features to mark could be easily added within this framework.

For specific positional features, such as the location of hydrogen bond acceptors, my system chooses a single position on the surface and places a symbolic decal like the X in Figure 4.5. A

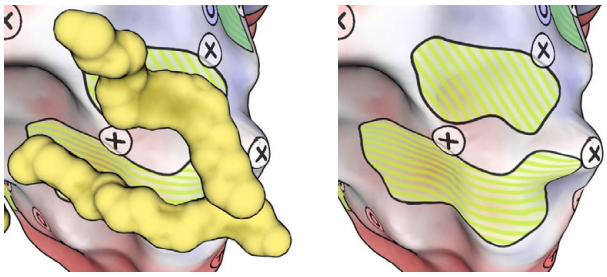


Figure 4.8 At left, surface features are obscured by binding ligands. At right, projecting each ligand's location onto the surface allows simultaneous viewing of both ligand location and underlying surface properties.

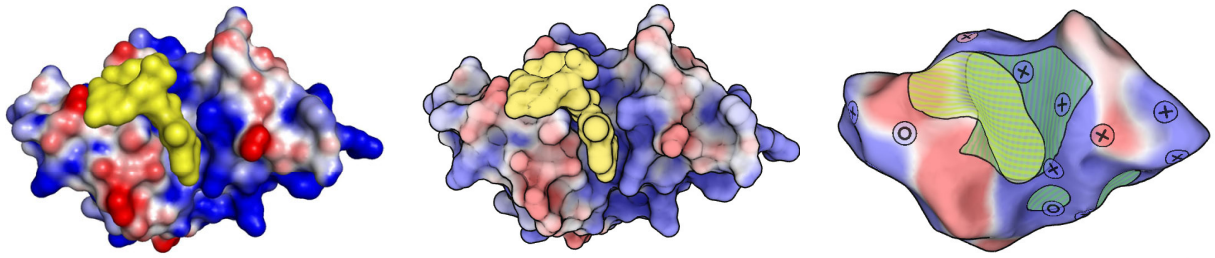
surface point near an internal feature, such as an atom center, is chosen somewhat arbitrarily as small differences in positions are not important in the abstracted representation.

I also use decals to indicate the mid-sized features removed in Section 4.1.3. While the set of vertices in the feature that remain after the removal process could be used to denote a region, my experience is that after contraction and smoothing, this patch bears little relationship in shape that of the removed feature. Therefore, my system instead uses a circular symbol of fixed size (1.5 Ångströmsin radius), as a circle doesn't imply anything (for better or worse) about the original shape. Glyphs within the circles are used to differentiate peaks from bowls.

4.1.4.4 Binding Pockets

Decals can also be used to indicate larger regions corresponding to other information that is known about the molecule. Biologists use a myriad of tools to attempt to locate biologically significant areas on a molecule's surface. When binding partners are known, regions of the surface near ligands can be marked. This representation makes visible the portion of the surface involved in the interactions (Figure 4.8). The output of region detectors, such as pocket finders, can also be displayed this way. My system presently includes an an implementation of Ligsite [63] to identify potential pockets. The output of these detectors is noisy, so before constructing decals, small or low-confidence regions are removed to avoid clutter.

For either type of sticker, excessively large regions may arise. These are dealt with in the following way: during each attempt to parameterize a region, my system simply evaluates the



(a) Pymol (b) Stylized (b) Abstracted

Figure 4.9 An example of Bullfrog Ribonuclease (1M07) before and after the abstraction process. The green striped areas represent parts of the surface that were identified as putative ligand binding sites. The yellow, ligand shadows, or areas of the surface nearest to known ligand locations.

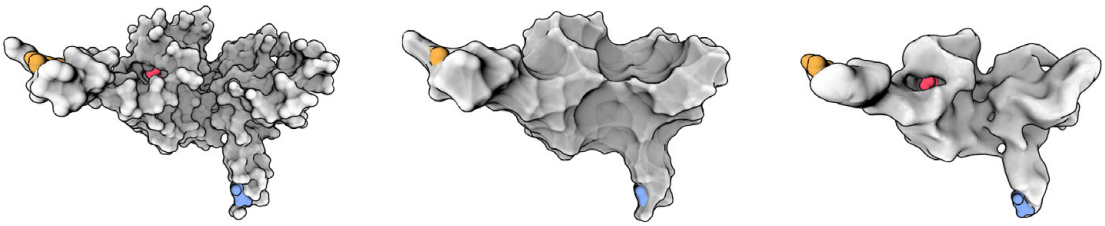
distortion during parameterization as it proceeds outward from the seed vertex. Errors are compounded as parameterization moves further from the seed. So if a vertex fails a conformality test, then both that vertex and all vertices which fall further along the same path from the seed are left out of the chart. This method is repeated, then, adding charts to the patch until none are left.

The system displays the output of different region detectors using different patterns for each, allowing multiple features to be shown simultaneously and compared (Figure 4.9).

4.1.5 Results

I emphasize that abstraction is not the same as changing the probe size. Larger probe sizes will fill in crevices that may be important pockets, while leaving bumpy details that detract from comprehension as shown in Figure 4.10. Indeed, abstraction can be applied to surfaces generated with any probe size. All examples in this document were generated using a probe size of 1.5 Ångströms, (except Figure 4.10b), equivalent to radius of the sphere surrounding a water molecule, the most likely solvent.

Figures throughout this document show the results of abstraction applied to various molecules. Figure 4.9 shows how important aspects of the molecule are made clear by abstraction.



(a) Original, with 1.5 Å probe (b) With 4 Å probe (c) Abstracted

Figure 4.10 A demonstration of how using a larger probe size, while resulting in a slightly smoother mesh, will destroy fine details. Bright yellow, blue and red surfaces denote ligands, included to emphasize the important pockets. In particular, channels containing important ligands are completely removed in (b), along with other structural detail. Surface abstraction (c) preserves these features.

Major shape features, such as pockets and clefts, become very clear on abstracted surfaces because there are fewer small details to distract the viewer, and the smoothness allows silhouette shading, contouring and ambient occlusion lighting to emphasize the shape.

To date, I have evaluated quality only informally during development, when I chose smoothing parameters. Absent input from skilled users, constructing metrics for quality over and above merely screening for visual anomalies would be difficult. Abstraction quality, I argue, is inherently tied to the usefulness of the abstraction. Therefore, any future assessment of the quality of surface abstractions would require concerted interaction with potential users.

The asymptotic complexity of the abstraction process scales linearly with the number of vertices of the input mesh, which scales (at worst) linearly with the number of atoms in the molecule. This was confirmed empirically during performance evaluation.

4.1.5.1 Performance

To assess performance of surface abstraction, I selected 60 proteins of various sizes from the Astex test set [123]. When determining the timings for the abstraction process, I consider smoothing, decal construction, and surface field relaxation; but not the preprocesses to determine the initial mesh, electrostatic potentials, or binding pockets. I also exclude the time required to load

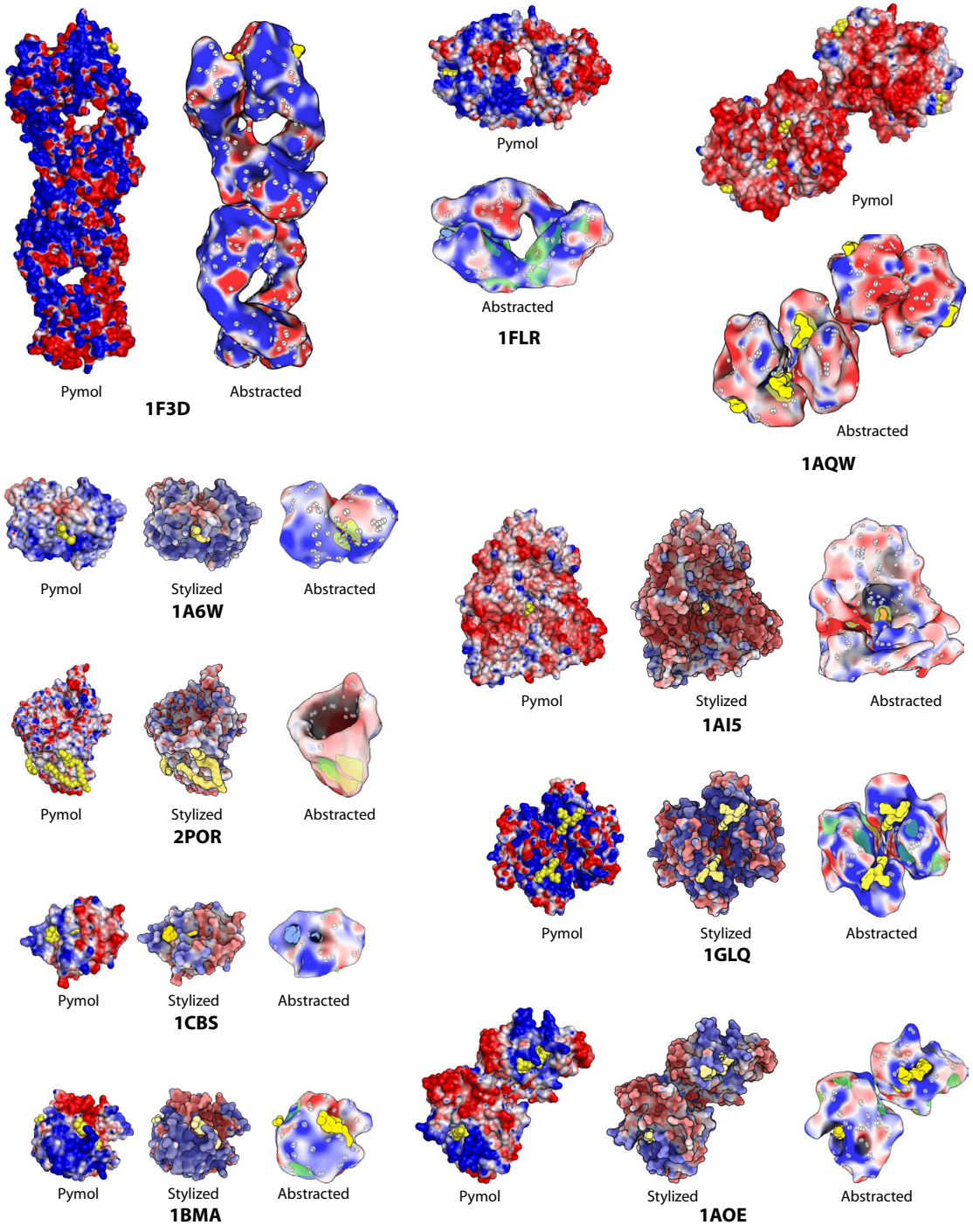


Figure 4.11 A gallery of example proteins of various sizes, shown before and after abstraction. Traditional images, rendered with Pymol [5], show molecular surfaces for the proteins, and spheres for the ligands. Stylized images use the rendering techniques described in Section 4.2.1 on non-abstracted surfaces.

data from disk and compute ambient occlusion lighting. Timings were performed on a PC with an Athlon 4400 CPU, 2GB of RAM, and NVidia 7900GT graphics.

On this test set, the time needed to perform abstraction ranged between 7 and 113 seconds. These correspond to the smallest molecule in the set (1CBS with 137 residues and 1092 atoms) and the largest (1CX2, 2200 residues, 21764 atoms). The expected linear performance scaling was observed.

4.2 The Client Viewer

As a platform to test ideas, I first developed a client viewer, written in a combination of C++ and OpenGL. This bespoke viewer not only implements abstraction, but a number of other rendering technologies that I will describe in this section.

Note that the GRAPE web-based viewer, which I describe in Section 4.3, *does not* yet implement these features, with the exception of precomputed ambient occlusion.

4.2.1 High Quality Rendering of Abstracted Surfaces

The resulting abstracted surfaces still provides a shape display problem. The visual presentations must not impede the use of the surface markings indicating small shape features. Also, I prefer a visual style consistent with the abstraction, rather than the "realistic" shiny plastic more commonly used to display molecular surfaces. Therefore, the primary display is stylized within the client viewer. Qutemol [28] showed the utility of stylized shading for molecular depiction. I apply several of their concepts to molecular surfaces. My system also applies stylized rendering to non-abstracted surface models (seen in many figures throughout this document).

To enhance shape portrayal, I apply per-pixel silhouette shading [124]. This shading technique sets pixels on faces that orient directly toward the viewer at maximal brightness, with decreasing brightness as the face normal orients away. Concretely, $brightness = 1 - p^{\frac{cos^{-1}(n_z)}{\pi/2}}$, where n_z is the z component of the surface normal and p is a tunable constant that I have set to .3 in all figures in this document. The underlying surface color is multiplied by this brightness value to produce a final color.

Ambient occlusion (AO) lighting [124] is applied as it accentuates global shape. As pointed out by [28], the regions made darker by ambient occlusion because of lower lighting accessibility are related to the regions with lower chemical accessibility. Interior points of clefts and pockets are made darker.

My implementation of AO uses the graphics hardware to sample light directions. While this computation is not realtime (typically taking from less than a second to 10 seconds for large molecules), it is generally fast enough.

As a final step, the client viewer strokes along contours of the mesh, which can be defined as those edges that border both a front-facing and a back-facing face. This not only enhances shape perception, but gives a stylized look that gives a constant reminder of the degree of abstraction in the representation. The smoothness of abstracted surfaces makes more sophisticated contouring methods unnecessary. Experiments with Suggestive Contours [58, 125], show that they add few contours beyond the simple ones on abstracted surfaces, and that these additional contours were typically small enough to be difficult to notice.

As in traditional molecular surface display, scalar fields are indicated on the surface by pseudo-coloring. For the examples shown in this document, electrostatic potential is displayed with the red to blue scale that is commonly used. Colors for other decals are chosen such that they can be seen when overlayed on these colors.

4.2.2 Multiple Surface Display and Comparison

As an initial prototype, I have adapted my client viewer to allow me to align and display multiple protein surfaces (see Figure 4.12). Comparing multiple surfaces in a gallery presents a number of additional challenges that are not present in single-surface display. First and foremost, the size and resolution of each molecule's depiction is limited. Second, there is less opportunity for interactively rotating each molecule to find views that show shape features.

I address the resolution problem by leveraging the reduction in apparent detail complexity afforded by abstract representations. Detailed solvent-excluded surfaces may be finely tesselated, with many triangles. Though abstraction does not appreciably reduce the number of triangles,

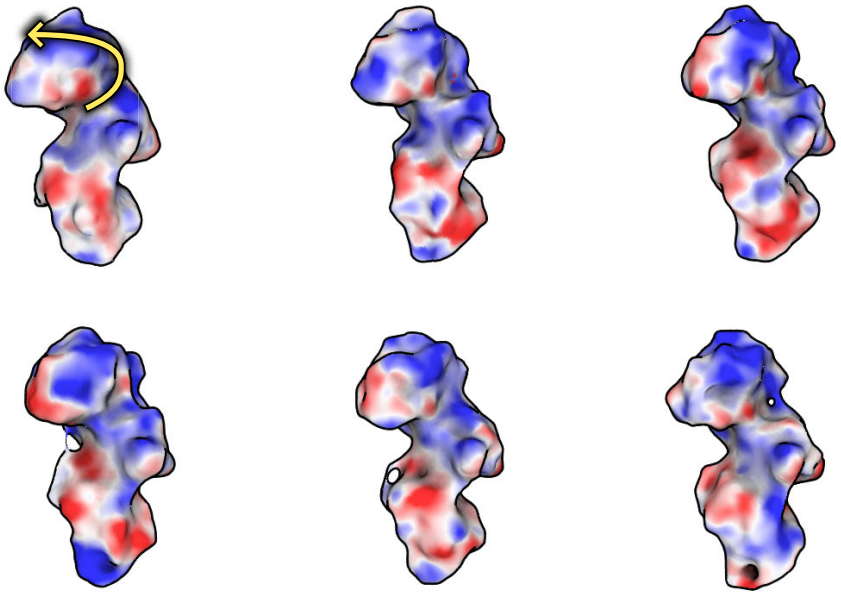


Figure 4.12 6 aligned RRMS: 2GHP, and 5 threaded surfaces of homologous proteins: castelli, glibrata, gossypii, kluyveri and lactis. Note that regions on the surface with positive electrostatic potential associated with known RNA binding, which winds up through the crevice and along the top (demonstrated by the yellow arrow), are mostly conserved, while other areas are less so.

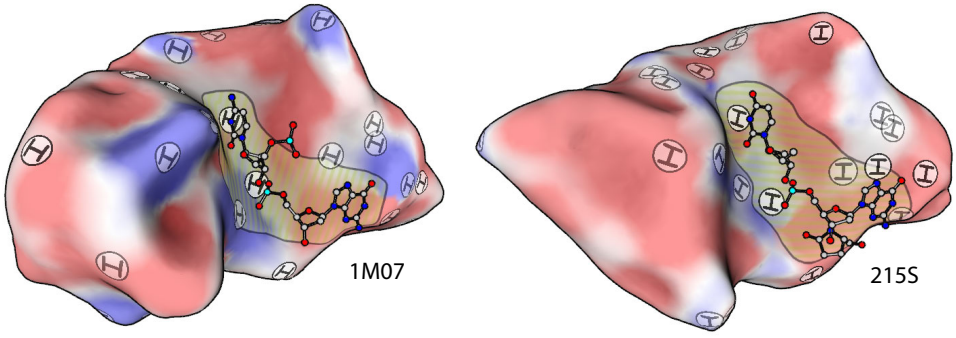


Figure 4.13 Abstraction aids comparison by concisely showing multiple properties of the surface in the same image. Shown here are ribonuclease proteins from two frog species whose enzymatic activity varies by a factor of 100. In both pictures, regions in yellow represent areas of the surface in proximity to bound RNA. Electrostatic potential and sites of potential hydrogen bonding are also depicted. As these depictions show, two hydrogen bonds are formed with the guanine nucleobase in 2I5S, improving its catalytic efficiency. These bonds are not present in 1M07.

abstract surface have, by design, significantly lower surface curvature. As a result, the abstracted surfaces can be represented with fewer triangles while still retaining their overall appearance.

To perform this triangle reduction, I have integrated Garland's quadratic-error mesh decimation algorithm [126], at several levels of detail, into my system. The end result: it can simultaneously render dozens of proteins at interactive rates. It should be noted that decimation may also be applied when browsing multiple detailed surfaces. In this case, the smaller number of available pixels available counteract otherwise severe surface deformations caused by decimation.

I address the interactive rotation issue for one important case: when all displayed proteins are homologous (or nearly so). In this scenario, all proteins may be aligned to use the same camera view, such that when a user rotates one, the others also rotate. I perform this alignment by first identifying sets of paired points in each protein. To find these sets of paired points, I use the Needleman-Wunsch method [127] to find a sequence alignment between the sequence of amino acids in two proteins. This lets me align homologous proteins with a reasonable tolerance against insertions and deletions.

The next step is to find a transformation that aligns the proteins to one another. For this, I use Horn's method [128], which takes two paired sets of points, S_1 and S_2, and constructs a rigid transformation that, when applied to S_1, produces a new set of points, S_1' that has the lowest-possible RMS deviation from S_2. For each of the aligned amino acids (ignoring gaps) found, I use their respective alpha carbons as a pair in Horn's algorithm.

4.2.3 Results

I have found that this method works quite well when aligning on a common small ligand across a group of proteins (provided the ligand does not exhibit a large conformational change), and for aligning proteins that share a high degree of sequence identity. I have not yet evaluated the degree of identity required to produce a reasonable alignment, though Horn's method does seem to produce a reasonably correct answer, even with a large number of misalignments.

Though the results shown in Figures 4.12 and 4.13 are preliminary, they have been met with enthusiasm by our collaborators, providing for an interesting use case for abstraction and an opportunity for further research.

4.3 GRAPE: GRaphical Abstracted Protein Explorer

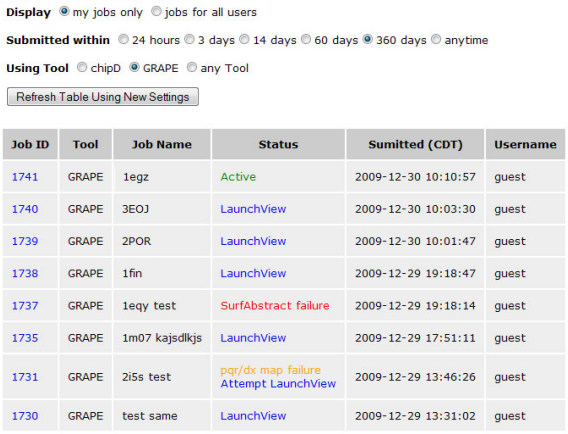
GRAPE represents the first completely web-based system for constructing and displaying abstracted molecular surfaces, using PDB data as input. Its functionality can be broken into two categories:

- A back-end system, which takes PDB files as input, coordinates the job of abstraction among a server pool, and produces an output file encapsulating the abstracted surface and its varied fields.
- A front end viewer, which uses a combination of Java and OpenGL to embed a live, explorable representation of the abstracted surface.

4.3.1 Project Goals

GRAPE is motivated by the dual goals of making surface abstractions easy to generate, and to give as many users as possible the ability to use them on their own molecules. I have organized GRAPE so that all computations, including those for high-quality lighting and mesh generation, are performed on the server; the viewer is a thin client that merely reads in the results of those computations and renders them. This reduces the computational burden on the user's computer, and ensures its availability to users with low-end hardware.

In recent years, many online protein viewers have come to be widely used, such as Jmol [129]. These viewers are ideal for presenting a low barrier to entry for exploration: no software need be installed, and results are available from any computer, regardless of platform. GRAPE is designed to be similarly accessible: the client software itself is quite small, and requires only modest graphics hardware for texture rendering. For this version, because GRAPE does require graphics hardware, layering its required functionality on top of existing molecular viewers was not feasible.



Job ID	Tool	Job Name	Status	Sumitted (CDT)	Username
1741	GRAPE	1egz	Active	2009-12-30 10:10:57	guest
1740	GRAPE	3EOJ	LaunchView	2009-12-30 10:03:30	guest
1739	GRAPE	2POR	LaunchView	2009-12-30 10:01:47	guest
1738	GRAPE	1fin	LaunchView	2009-12-29 19:18:47	guest
1737	GRAPE	1eqy test	SurfAbstract failure	2009-12-29 19:18:14	guest
1735	GRAPE	1m07 kajsdlkjs	LaunchView	2009-12-29 17:51:11	guest
1731	GRAPE	2i5s test	pqr/dx map failure Attempt LaunchView	2009-12-29 13:46:26	guest
1730	GRAPE	test same	LaunchView	2009-12-29 13:31:02	guest

Figure 4.14 The GRAPE job queue. The job queue is a list of pending and completed jobs, along with information on what user submitted each job and the time of submission.

The process of creating and using an abstraction in GRAPE has three steps: first, obtain the data about a molecule; then, abstract it into a useful form; finally, load this data into a viewer. The first two steps are performed on the server, and result in a compressed data file.

4.3.2 Server Side Processing

4.3.2.1 Input Data

GRAPE takes as input a PDB file, which may be either uploaded from a local copy, or fetched directly from the Protein Data Bank [4]. Optionally, users may also upload a PQR file to supply a custom protonation state, overriding the automatic protonation computations done within GRAPE.

Abstracting a protein can take a long time, depending on its size and complexity, so the GRAPE server creates a separate job for each submitted protein. Jobs runs asynchronously on the server; after a submission, users are redirected to a job queue to monitor the status of their job.

4.3.2.2 The Job Queue

Along with the PDB file itself, each job has the following metadata associated with it:

- JobID: The unique identifier for this job
- Tool: GRAPE

- Job Name: The name a user assigns to this job (not the name of the protein)
- Status: Active refers to a job that is currently running. LaunchView appears when a job has successfully finished. Clicking this will take you to the viewer. pqr/dx map failure indicates that either pdb2pqr or APBS failed to produce charge and SurfAbstract failure indicates that abstraction failed, for an unknown reason. For either of these failure modes, clicking the Job ID, and then clicking on the 'log' link on the following page, may indicate the source of the failure.

This data can be seen by browsing to the "job queue" page, as shown in Figure 4.14.

4.3.2.3 Authentication

To allow users to better manage their queue, GRAPE provides an optional authentication mechanism. Users that wish to authenticate may create an account by first providing a username and password. They then gain the ability to filter the job queue to show only their jobs, to receive an email when their jobs finish, and to mark specific jobs as 'private'.

By default all jobs in GRAPE are marked as public, which means that all GRAPE users will be able to view the results of a job. For users with sensitive data, such as prepublication proteins, the optional ability to mark a job as 'private' ensures that only that user will see the results.

The authentication and queue management infrastructure used in GRAPE is derived from work done for the KFC server [98].

4.3.2.4 Processing a Job

All major processing takes place on the server back end, where jobs are farmed out to a cluster of computers, in first-come first-served order. Each computer takes on the task of abstracting a single protein. This can be divided into two phases: the *data collection phase*, in which the shape and electro-chemical properties of the original "solvent-excluded" surface is computed, and the *abstraction phase*.

All of the server-side code described below is based on the algorithms described above in Section 4.1.

The data collection phase breaks down as follows:

1. Compute the solvent excluded surface using MSMS [116]

2. Compute electrostatic potential using PDB2PQR [111, 112] (or a user-supplied PQR file) and APBS [113].

3. Compute hydrophobicity by finding, for each point, the hydropathic index of the closest amino acid in the protein.

4. Predict potential hydrogen bond donor/receptor locations by locating atoms in the side-chains of each amino acid that are both potential donors/receivers, and also near the solvent excluded surface.

5. Find "interface regions", areas of the surface which are within a radius (1.6 Ångströms) of the van der Waals surface of another part of the molecular assembly (i.e. a different chain in the PDB file).

6. Compute ambient occlusion lighting [124], which darkens interior regions of the molecule. High-quality global lighting has been shown to be useful as a shape cue, as recently demonstrated in QuteMol [28], but cannot be done in real-time without high end hardware. The GRAPE server pre-computes ambient occlusion lighting, reducing hardware requirements for its users.

After collecting surface data, the second phase of a server job is to abstract this data, transforming the detailed results of the first phase into a (visually) simpler form. This process is identical to that described in Section 4.1, which again can be broken down into a series of steps:

1. Smooth fields, such as electrostatic potential, along the surface. This removes small features, while preserving larger regions of consistent value.

2. Perform a series of mesh restructuring operations on the mesh geometry to first smooth and then 'sand off' small bumps and pockets. The resulting mesh is a smooth approximation of the original solvent accessible surface.

3. Apply stickers directly to the surface, as seen in Figure 4.15.

4.3.2.5 Surface Data Format

The final results of the abstraction process are compressed into a single ZIP file that stores the information required for the client to draw both traditional and abstracted views of the protein. This ZIP container contains a number of smaller files, in three major categories:

- **Surface geometry:** Each chain in the PDB file is given its own set of surface geometry, one for the solvent-excluded surface and (for chains that are not hetero-compounds) one for the abstracted surface. Every surface is contained in a binary .ply file.

- **Stickers:** The picture for each sticker on the surface is stored in a compressed .png file. Note that some stickers, such as those for hydrogen bonds, are only stored once, and then reused over the surface.

- **Surface Fields:** One XML file connects surface geometry with field data, identifies the location of each sticker on the surface, and matches chains in the PDB with each set of .PLY files.

The ZIP file, which ranges in size from 200KB to 12MB, depending on the size of the protein, is stored on the server in a job-specific location.

4.3.3 Client Side Viewer

After the GRAPE server has completely finished a job, its status changes to a link titled 'LaunchView'. Clicking this link brings up the results of all abstraction computations. We have built a GRAPE viewer, shown in Figure 4.16 which can be run directly within the output web page. Standard viewing controls are provided, to let users navigate the molecular surface.

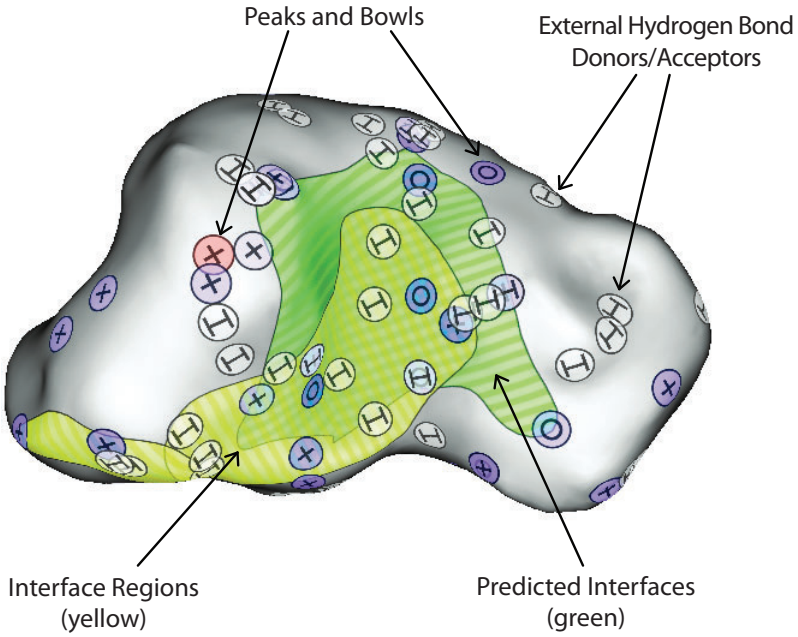


Figure 4.15 **Stickers:** Four types of stickers are applied to abstracted surfaces, each of which may be independently turned on or off. *Hydrogen bond stickers* are placed on the surface in areas that are close to one or more atoms that could form an external hydrogen bond. *Detected Pocket stickers* indicate regions of the surface that resemble binding pockets, according to a variant of the LIGSITE pocket detector. *Interface stickers* indicate regions of the surface in close proximity to another chain. And *peak/bowl stickers* display as an 'X' or 'O', respectively, points where significant peaks or bumps in the original solvent excluded surface were removed.

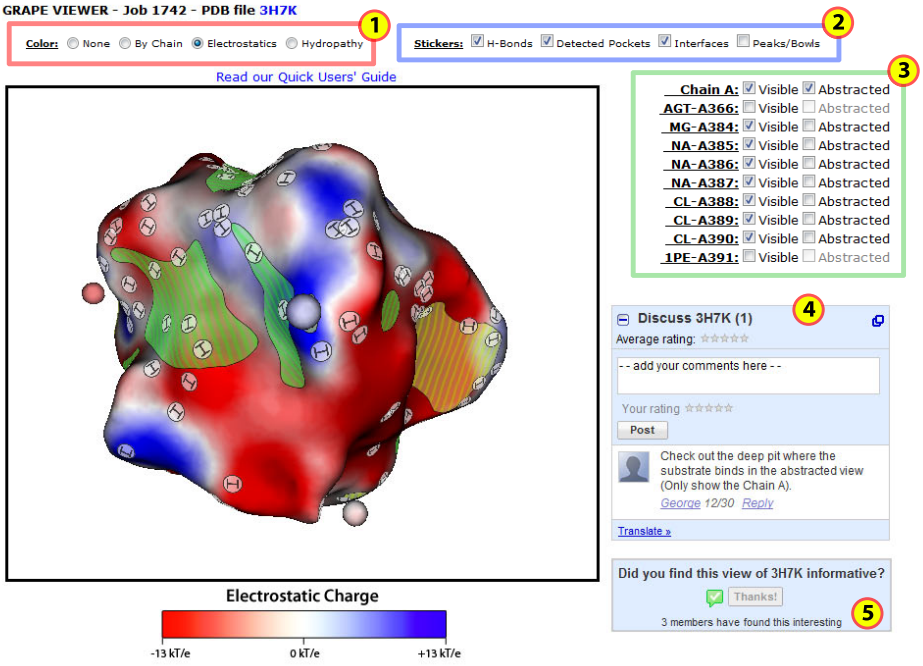


Figure 4.16 The GRAPE output window. Shown is the abstracted surface of a protein, each chain of which can be independently customized in the following ways: **1.** Coloring: Chains may be white, uniquely colored, colored by electrostatic potential, or colored by hydropathic index; **2.** The different sticker types may be hidden (see Figure 4.15); **3.** Display: Chains may be hidden, displayed as the solvent excluded surface, or displayed in abstracted form. Also on this page are: **4.** the discussion gadget, where users may leave comments, and **5.** the recommendation gadget, where users may recommend their favorite proteins to one another.

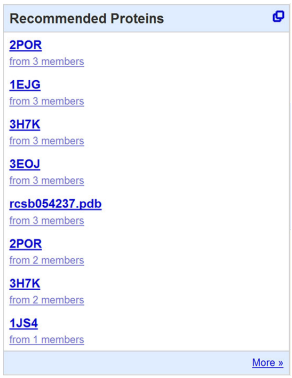


Figure 4.17 A GRAPE recommendation gadget. The proteins most recommended by GRAPE users are shown at the top of this gadget, which appears in several places on the site. Each link immediately directs a user to the viewing page for that protein.

The viewer itself is written in Java, and uses the Java OpenGL (http://opengl.j3d.org/), or JOGL, binding library to render the surface. On page load, a small JAR file is downloaded for the GRAPE applet, followed by native JOGL libraries, if necessary. Finally, the ZIP file described in the **Surface Data Format** section above is downloaded from the server, and loaded into memory.

A link is also provided on this page to download abstraction data as a raw ZIP file. Currently, the GRAPE viewer is the only tool that can completely use this data, though we envision plugins for existing protein viewers that would allow us to merge abstract surfaces into existing methods of display.

4.3.4 Social Networking

GRAPE uses Google Friend Connect throughout to foster discussion about protein surfaces, to link researchers together, to allow new users to quickly discover the surfaces that others have found interesting and to provide a mechanism for others to give us feedback about the tool.

Friend Connect gadgets expand the usefulness of the site in two ways: first, GRAPE uses the "recommendation" gadget to give users the ability to recommend proteins to one another, as in Figure 4.17. So experienced users can discover interesting new models, and new users can quickly

see the benefits of abstraction on existing proteins, before they try their own. In addition to the these recommended proteins, I have added several curated examples in a separate gadget, to ensure that new users always have a set of high-quality examples with which to begin their exploration of GRAPE.

Second, the viewer page for each job has an additional "ratings and reviews" gadget, where users can discuss aspects of a single protein. An example can be seen in Figure 4.16.

4.4 Discussion

The symbolic display of smaller features has a number of advantages. It makes shape features more readily apparent from a wider range of viewing directions (Figure 4.4). Symbols are visible in static displays, while shape features often are only obvious in regular displays when the object is moving. At small sizes, sampling issues may make small geometric features difficult to display. With surface textures, texture sampling hardware can perform sampling using mip-mapping, and decals can be omitted at very small sizes.

Displaying symbols as decals on the surface provides a mechanism for indicating a variety of properties about the underlying molecule. Such decaling would be difficult on non-abstracted surfaces: their non-smoothness would make parameterization difficult, and the small features would obscure the symbols with clutter and occlusion. Because the symbolic display is imprecise, they fit in better with the abstracted surfaces as abstracted surfaces imply a lack of positional precision.

Molecular abstraction has been applied to study features at a specific scale. Exploring other scales would require retuning the methods, and possibly designing a new set of feature detectors. Studying molecules at scales much different than the atomic level interactions we consider, such as macro-molecular assemblies, would provide more challenges, including performance.

In creating the GRAPE server, I endeavored to make abstractions more accessible for researchers to use. In doing so, however, I made several tradeoffs. As described in Section 4.3.1, my primary goal was to give researchers the ability to quickly use abstractions, to judge their utility for themselves. So rather than attempting to fit into the many different workflows used by researchers, I chose instead to build a simple, easy-to-use server that provides quick results. This decision does

ultimately limit how useful GRAPE can be: because it does not integrate into researchers' workflows, they cannot use the tools to which they have grown accustomed. In the future, I would like to fix this limitation by providing a PyMOL plugin that can understand and display the server-created data format.

I also chose to make the client hardware requirements as low as possible: the GRAPE viewer itself is very thin, and all abstraction is done on the server. While I could have implemented abstraction directly on the web-client, which would have potentially provided more functionality and lowered the time between submitting a protein and viewing its abstraction, this would shift the burden of creating abstractions to the client's computer, which would in turn limit the number of users who could use them.

Nevertheless, the GRAPE viewer is currently missing several important features: there is no way to identify which regions of the surface are proximal to specific amino acids in the sequence, ribbon and stick-and-ball visualizations are not available, and certain parameters (such as the color maps for electrostatic potential and the degree of abstraction) are fixed. These features will be added in future revisions.

The most important step for this work, however, is to assess how effective these representations are for scientists. Though GRAPE does afford more opportunities to see how scientists use abstracted views, to date, my testing of abstracted molecular surfaces with biochemist collaborators has been limited, so my observations are anecdotal. In all cases, however, their initial reactions were extremely positive. They immediately appreciated the simplified views. On molecules familiar to them the views matched what they "expected" them to look like.

In several cases they would make comments like "I never noticed that before, I wonder ..." which is particularly encouraging as it implies a new way of looking at things might lead to new hypotheses. Figure 4.13 represents such a case: collaborators who were shown an abstracted surface showing charge overlaid with hydrogen-bond and binding interface stickers were instantly able to notice interesting patterns. Though in this example the results were known in advance, in a real-world case, these patterns could quickly lead to a hypothesis about the correlation between high RNA-binding affinity in onconase and the number and arrangement of hydrogen bonds.

The methods I have chosen to produce surface abstractions do have several limitations. First, because there is currently no quick way to adjust the level of abstraction, important phenomena on the surface may be hidden or removed. For instance, biologically-relevant features with small shapes could be smoothed away. Similarly, important fine-grained electrostatic features could also be blended. In either of these scenarios, a researcher potentially could be pointed in the wrong direction. That said, abstractions were designed not to replace conventional surface representations, but rather as a complement to them. Therefore, while the current version doesn't provide a *complete* picture of the surface of a protein, it does provide a new way of looking at these surfaces, one which is useful for generating additional insight.

As discussed in Section 4.1.4.4, I take steps to produce visually pleasing stickers over large regions, such as pockets and binding interfaces. Unfortunately, regions where the mesh folds over on itself, or where surface irregularities prevent good parameterization, may result in stickers that have defects, such as warping and tearing. Fortunately, these are usually small and infrequent. Using better surface fairing and parameterization methods would likely further reduce the impact of this problem.

Chapter 5

Multi-Scale Surface Shape Descriptors

5.1 Local Shape Descriptors

Local shape descriptors distill the shape of a region of a surface into a short vector of numbers, each corresponding to a property of the region. These descriptors have broad application when working with shapes: for example, they are used in visualizing and analyzing scientific data, in shape matching, and in stylized rendering. While various local shape descriptors and methods for computing them exist, their inability to summarize the shape of larger regions limits their utility.

In this chapter, I present a local surface shape descriptor that is applicable at different scales to summarize the shape of differently sized neighborhoods on a triangular mesh. This allows it to be applied to smaller regions to capture small-scale detail, or to larger neighborhoods to summarize their overall shape. Regions of the surface may have one shape at a small scale, but a different shape at a larger scale (e.g. a small bump within a large bowl). This approach, called *multi-scale surface descriptors*, was published in the 2009 IEEE Transactions on Visualization [130]. Described in this chapter, it will further serve as the basis for a full molecular descriptor in Chapter 6.

The shape of any finite region may contain arbitrary amounts of detail, therefore a shape descriptor can only provide a summary. For an infinitesimal region, the amount of detail is limited, so the shape can be completely described by its *curvature.* Curvature provides a compact local descriptor: three or four numbers are sufficient to characterize the shape for an infinitesimally small region. For finite-sized regions, however, the mathematics of curvature do not apply.

I provide a descriptor that captures the most significant features of the shape of a local surface region. The descriptor considers a local neighborhood around a central point with a roughly circular area specified by radial distance. It measures the degree and type of non-planarity of the region, for example encoding whether something is a steep bump or a shallow bowl. It also captures the degree and direction of anisotropy, identifying troughs and ridges. A key insight of my approach is that while these quantities are not sufficient to capture all details of the shape of a finite region, they do capture the most significant aspects of shape. I introduce robust and practical methods for computing these larger-scale surface descriptors, and show their usefulness in a number of applications.

One of my key motivating applications is the description of molecular surfaces. As described in Section 3.3, stereospecificity implies that shape is a useful predictor of binding affinity. Further, this phenomena holds at multiple scales. Therefore, by characterizing the shape of a known binding interface, and then using this information to identify similar regions in other proteins, new targets may be found for a given ligand.

This application, which I discuss in Section 5.4.4, highlights many of the requirements for practical, effective local shape descriptors and the methods to compute them: they must operate over large enough neighborhoods to be chemically significant (see Section 6.2.1 for a discussion of scale); they must be efficient, as descriptors must be computed for all points on each molecule in a database; because MSMS produces highly irregular meshes with many sliver triangles, they must be robust against these noisy and poorly tesselated meshes: any resulting errors in curvature estimation could lower predictive performance; and they must correspond to domain scientists' intuition about shape and neighborhood. I provide the first shape descriptors that I feel are able to meet these needs.

In Chapter 6, I expand these shape descriptors to incorporate electrochemical information. This allows them to serve as the basis for a *molecular* surface descriptor, with which I describe the microenvironment surrounding specific atoms in a ligand at multiple scales. Once described, they may then be used to predict potential sites of binding on an unclassified protein.

5.1.1 Contribution

The contribution in this chapter is an approach to local shape descriptors that provides a method for characterizing neighborhoods at multiple scales on the surface of a mesh. My approach is the first to address all of the following goals for such descriptors:

- *It scales to describe larger neighborhoods.* Curvature captures only infinitesimal regions and prior approaches to curvature computation focus on minimizing the size of regions to better approximate the differential case. In Section 5.2 I present methods that summarize a non-trivial region statistically.

- *It corresponds to intuitions about curvature.* Like curvature, my descriptor captures the degree and type of non-planarity of a region, and the degree and direction of anisotropy.

- *It allows for control of scale in a simple way.* Neighborhoods are specified by their center and a radius, so regions are controlled by physically relevant quantities of the surface. In contrast, methods such as mesh pre-filtering require a less direct specification of neighborhood size in terms of frequency (which can be difficult to explain to domain scientists), and filtering causes points to move, precluding localized assessment.

- *It compensates for issues in tessellation.* The methods of 5.2.1 account for discretization, allowing my descriptor to be robust to poorly tesselated surfaces.

- *It affords ef cient computation.* Throughout Section 5.2 I present methods for computing the descriptors efficiently. In particular, my approach avoids expensive parameterizations and my approximations can provide good performance without resorting to expensive mesh operations such as exact geodesic computations.

- *It works in applications.* In Section 5.4 I describe how my descriptor is effective in several applications. By providing a larger-scale summarization of surface regions, it allows for simpler algorithms to make effective use of descriptors across a number of applications. In Chapter 6 I describe a greatly expanded *molecular* surface descriptor which is used for ligand binding prediction.

My approach is closely related to prior work on using planar parameterizations and polynomial fitting for curvature estimation, which will be discussed in Section 2.2. However, unlike these methods that focus on the challenge of being robust with as small a region of surface as possible, my approach is designed to work with larger surface regions. This requires us to provide effective methods for finding the correct regions (Section 5.2.1) as well as to provide techniques for statistically summarizing the shape over these regions (Section 5.2.3, Section 5.2.3.1).

5.2 Multi-Scale Shape Descriptors

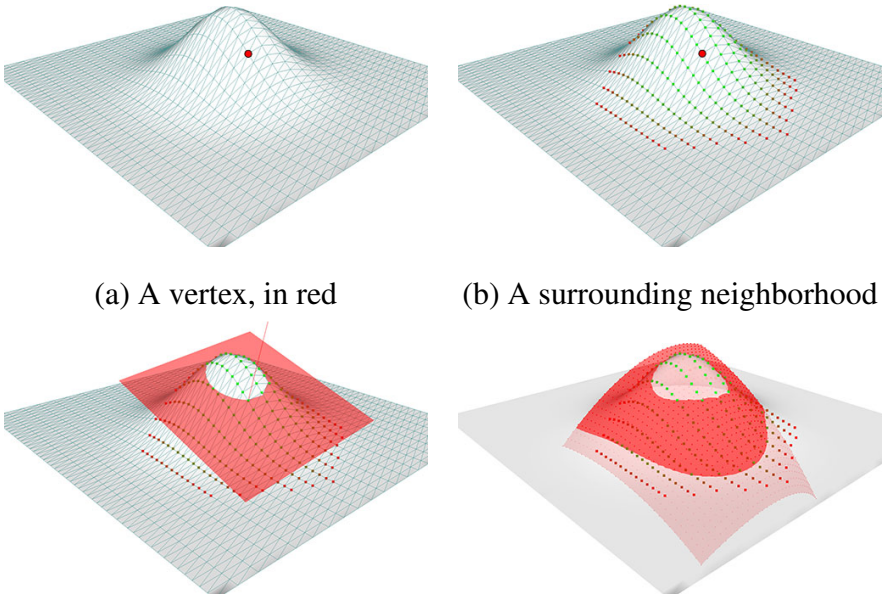
Given a triangle mesh and a neighborhood defined by a center point i (a vertex in the mesh) and size d, my approach computes a local descriptor of this region as a statistical characterization of its shape. See Figure 5.1 for a depiction of this process. The radius, d, gives control over the scale of the neighborhood and has units of length with the same scale as the surface itself. The choice of radius depends on the application domain. For example, when considering molecular surfaces I choose d to be a size of biological relevance.

My approach has the following steps:

1. The set of points, $N(i)$, in the neighborhood around i are determined and a weight computed for each to compensate for issues in tessellation. In Section 5.2.1 I describe how the points can be determined efficiently and how the weights are chosen.

2. $N(i)$ is represented as a height field on the surface tangent plane at i. This representation, motivated by methods in curvature estimation, simplifies the computation of the statistics in the next step. In Section 5.2.2 I describe how the tangent plane is determined and explain the height field representation.

3. The height field is characterized statistically to yield the descriptor. I provide two methods for this summarization. In Section 5.2.3 I describe an approach using robust least-squares fitting that directly applies the intuitions of curvature. In Section 5.2.4 I provide a method that is based on simpler intuitions of shape. In Section 5.3.2.1 I compare both approaches.



(a) A vertex, in red

(b) A surrounding neighborhood

(c) With height-field plane

(d) Best-fit quadratic

Figure 5.1 A demonstration of the steps involved in generating a descriptor for a single neighborhood. My method starts with a vertex on the surface (a). It then generates a set of patches surrounding that vertex. One is shown in (b), with color given according to each vertex's weight (greener vertices have high weight, and redder have low weight). A plane is constructed using a weighted average of the normals of that patch (c), onto which all points are projected. Finally, a quadratic surface is found to approximate that height field (d).

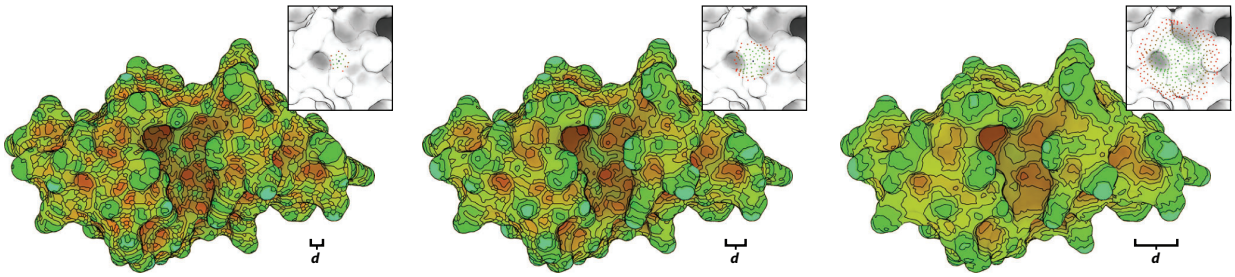


Figure 5.2 Shape description, represented at multiple scales, with peak-like regions in green and bowl-like regions in red. From left to right, the radius of the region, indicated by d, is increased. Inset, a single patch is shown for each scale, representing the neighborhood used for a single vertex.

The result of these steps is: one number that describes the type and degree of non-planarity of the neighborhood, one number assessing the degree of anisotropy, and a vector giving the direction of anisotropy, assuming that there is sufficient anisotropy to determine the direction.

5.2.1 Neighborhood Construction

To characterize a given vertex, my method first finds the neighborhood of all vertices that are within a distance d from that vertex along the surface of the mesh, called a *d-disc*. Descriptors at multiple scales are constructed, then, by choosing a set of d values for a given surface. Small values characterize local regions, while larger values take more of the surface into account. In the latter case, small surface variations are deemphasized. Figure 5.2 shows the effect of varying patch sizes over the mesh.

To create uniformly-shaped patches on the surface, the geodesic distance should be used between points, as geodesics follow the shortest path on the surface. Computing geodesic distance on a mesh, however, is difficult because this path does not necessarily follow edges on the mesh. Even with efficient algorithms, such as [131], exact geodesic computation is prohibitively time-consuming and, as I describe below, unnecessary.

A common approximation to geodesic distance is simply to follow the edges between vertices. The distance between any two vertices, then, is the shortest patch along the edges connecting those vertices. This so-called Dijkstra path is an approximation to the true geodesic distance.

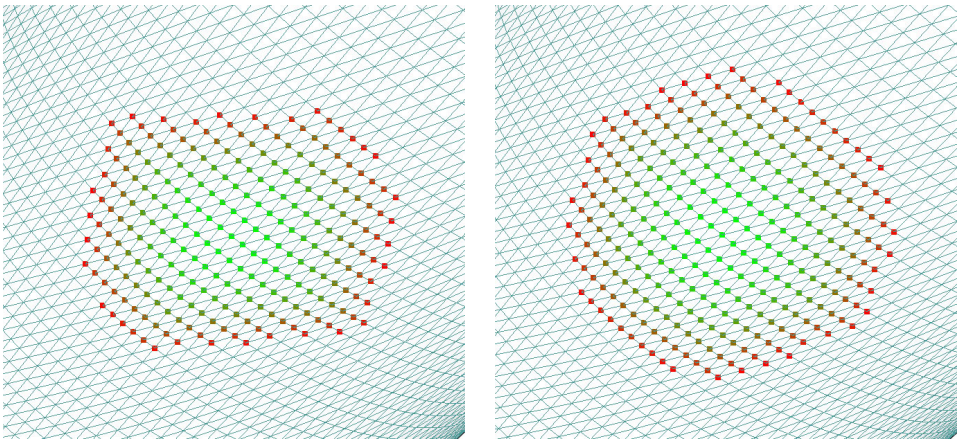


Figure 5.3 On the left is the result of performing a search along edges for those points that are less than a particular distance from a single vertex. Though, intuitively, this should form a circular patch because of the mesh's tessellation, the patch has a non-circular shape. By adding virtual edges to 2-neighbors, a better result can be achieved, at minimal incremental cost.

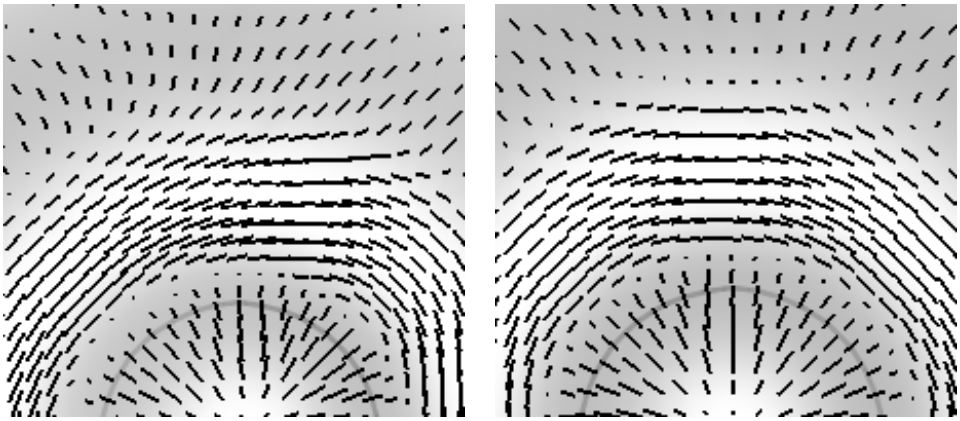


Figure 5.4 Here, directions of least curvature are found for a radially symmetric cosine wave. On the left, I show the result from computing distance along edges of the mesh. On the right, edges are added between 2-neighbors to better avoid tessellation issues. Note that the resulting directions are not symmetric on the left, but are on the right.

Though fast to compute, Dijkstra distance can be an arbitrarily poor approximation to geodesic distance. This problem is significant in practice, as the use of Dijkstra distance in determining the neighborhood gives irregularly shaped regions, as shown in Figure 5.3, and is unacceptable for descriptor computation. The problem is not pathological, but rather, occurs frequently on real meshes.

I introduce a modification to Dijkstra distance that improves its distance estimates. My insight is that a large discrepancy in the distance computed between two vertices results from a large difference between the geodesic path and the edgewise path. Further, the majority of these cases arise in an edgewise path of length 2 (e.g., two hops). To close the gap, my method simply adds virtual edges between each point and all of its 2-neighbors, and performs a Dijkstra computation on the original mesh with these added neighbors. In theory, these "corner cutting paths" may still have lengths that are arbitrarily poor approximations of the geodesic distances, but I have not encountered such pathological cases in practice. Figures 5.3 and 5.4 show the results of this simple modification. I compare this method with true geodesics in Section 5.3.2.2.

Values for d can be chosen in one of two ways. For surfaces with an inherent scale, such as proteins, d may be set according to the size of known features to describe, for instance, atom- or residue-sized neighborhoods on the surface; for molecular examples in this chapter, I set d to 4 and 8 Ångströms, respectively. For surfaces that have no inherent scale, the user can simply choose their own, or use a heuristic, such as a multiple of the median edge length, to automatically pick a reasonable size.

5.2.1.1 Weighting

Before using the constructed neighborhood, $N(i)$, my algorithm first adjusts the weights of its individual vertices:

- Points are weighted according to the area of the surface nearest to them. This reduces the effect of any variability of the size of triangles on a surface.

- Points closer to the edge of the patch are weighted to contribute less to the overall shape description than those closer to the center. This compensates for the fact that, for any given angular wedge, outer regions will contain more surface area than inner regions.

Given these two factors, the equation for the weight of a vertex $p \quad N(i)$ is:

$$W_i(p) = \frac{A(p)}{D(i,p) + \epsilon} \tag{5.1}$$

where $A(p)$ refers to the area of the faces surrounding p and $D(i,p)$ refers to the shortest distance from p to i. Note that dividing by $D(i,p)$ ensures that concentric rings in the neighborhood have equal weight. Normalizing these weights produces the final values:

$$\widehat{W_i(p)} = \frac{W_i(p)}{\sum_{v \in N(i)} W_i(v)} \tag{5.2}$$

5.2.2 Height Field

As described in [29], a common method for estimating curvature on the surface of a mesh is to construct a height-field function, parameterized by two variables, u and v:

$$F(u,v) = (u, v, f(u,v)) \tag{5.3}$$

There are several reasons one might want to do this. First, projection to a height field acts as a change of coordinate system. In this case, the height field approximates a tangent plane, and is therefore a more natural basis for computing differential curvature, as well as multi-scale curvature. Second, by projecting a set of points onto a height-field, a 3D problem is reduced to a 2D problem, reducing its degrees of freedom, and therefore, its complexity.

The coordinates for each point on the patch are found by projecting that point onto a plane. The choice of plane will affect how good the overall parameterization will be. As I will describe below in Section 5.2.2.1, one measure of 'goodness' is to minimize the difference between the normals on the surface, and the normal of the plane of projection.

So my method finds the plane normal $N_i(p)$ as the weighted average of all surrounding vertex normals:

$$N_h(p) = \sum_{v \in N(i)} \widehat{W_i(v)} \cdot normal(v) \qquad [5.4]$$

This vector is normalized to produce the final height-field normal, $\widehat{N_h(p)}$. It should be noted that while [40] give a more accurate method for averaging normals than linear approximation, I have not seen a case in practice that would benefit from their method.

As a final step, I ensure that the central vertex in a patch is placed at the origin (so $u_0 = v_0 = 0$). This is accomplished by subtracting the projected u, v for that vertex from the u and v of all other vertices.

5.2.2.1 Avoiding Issues with Height Fields

Several problems can arise from the use of height functions: first, projection does not preserve relative distances in the (u, v) parameter space. Second is the issue of foldover: $F(u, v)$ may have multiple, conflicting values at a given u and v. This is generally the result of including triangles that face away from the plane of projection.

I have experimented with various parameterization techniques, including exponential maps [121], to directly address both of these issues. Though this is a viable alternative, my technique does not require a parameterization of the surface. Also, because I am using an orthonormal projection, relative 3D distance is preserved. Thus I am still able to achieve a reliable fit and projection isn't a large issue. Local parameterizations, on the other hand, besides being slower to compute than height-field projections, may break down with distance; since I fit relatively large patches, this presents a problem.

The second problem, foldover, is more of a concern, but it can be mitigated by simply throwing away those vertices whose normals point away from the plane of projection. While this doesn't completely remove foldover, remaining conflicting vertices have very low weight, so they do not

tend to affect the result, as described in Section 5.3.2.5. Usage of robust statistics to lower the weight of outliers, as described below in Section 5.2.3.1, further improves matters.

5.2.3 Fitting with Quadratics

After representing the surface region as a height field, my approach must compute a descriptor that summarizes its shape statistically. Because the region may consist of a large number of points, the height field may have an arbitrarily complex shape. To summarize this shape, I first approximate it by a simpler shape that is easier to characterize. I fit a quadratic function to the height field. I choose quadratic functions because they are the simplest form that can sufficiently express the shape variability I need to encode.

Note that if the region were infinitesimal, its shape would exactly fit the quadratic form, and the coefficients of this quadratic would yield the curvature at the center point of the region. This is the basis of several curvature estimation approaches [29]. However, rather than trying to find a minimal set of points to constrain the quadratic, I instead use a large collection of points over the neighborhood and find the best-fit quadratic.

The form of equation to fit is:

$$z_i = f(u_i, v_i) = Au_i^2 + Bu_iv_i + Cv_i^2 + Du_i + Ev_i \qquad [5.5]$$

Each point of the height field (u_i, v_i, z_i) provides a linear constraint on the 5 degrees of freedom of the quadratic. Note that I do not include a constant term, which forces the function through the central vertex in a patch.

The simplest fitting method is to solve the set of linear constraints according to a weighted least-squares metric, with the weights $\widehat{W_i(v)}$ from Section 5.2.1.1. Because there is a small number of variables, my implementation solves these linear least-squares problems by forming the normal equations (multiplying the matrix by its transpose), and solving the resulting linear system using the Cholesky factorization. The use of linear least squares for polynomial fitting is discussed in [132].

Once parameters have been found for the quadratic, the principle directions and magnitude of curvature of the quadratic patch can be estimated by finding the eigenvectors and eigenvalues, respectively, of the second fundamental form:

$$II = \begin{bmatrix} A & \frac{B}{2} \\ \frac{B}{2} & C \end{bmatrix}. \quad [5.6]$$

For my descriptor, I use these principle curvatures of the quadratic. Specifically, I include the eigenvector corresponding to the smallest eigenvalue (i.e., with shallowest curvature), along with the degree of anisotropy, which is derived from the ratio of the largest absolute eigenvalue to the smallest.

5.2.3.1 Robust Statistics

When there are many points, even the best fit quadratic surface may be a poor approximation. In particular, the least squares metric used for fitting is sensitive to outliers: a small number of badly fit points can skew the results. In local descriptor computations, outliers can come from surface foldovers, sharp discontinuities, as well as noise on the surface itself.

I find that by applying robust statistical techniques [133] to my initial fit, I can lessen the impact of outliers. Using M-estimation, with the original quadratic fit as a prior, regions that have folded under are quickly identified as outliers. Their weights are then lowered, reducing their contribution, and a new surface is fit. This can be iteratively applied to reweight points, driving the least-squares solution toward a better fit with those points that remain. In practice, I find that a single iteration of this re-weighting is required to provide sufficient robustness.

5.2.4 Moment-Based Surface Description

A different description of the shape of the local region is based on a simpler intuition that less directly corresponds with curvature. In Section 5.3.2.3, I contrast the two descriptors and show that their results are similar.

Consider the height field patch from the region as a rigid object, with the points having mass proportional to their weights. If the center of mass of the object is above the plane, the center point is likely to be at the bottom of a bowlike shape (or the top of a peak if the mass center is below). The distance between the center and the plane gives an assessment of how peak/bowl-like the shape is. Similarly, I can consider the statistical trend of the mass distribution. If the moment of inertia is strongly directional, then the region is anisotropic.

These simple intuitions lead to a very efficient statistical characterization of the height field. The centroid of the neighborhood is found quickly by averaging, and its height value gives the assessment of non-planarity. By treating the height field as an image (with the height mapping to intensity), image statistic methods can be used to determine the distribution. Image moment computations [122] determine the degree and direction of anisotropy.

5.2.4.1 2D Moment Computation

Image moments can be used to statistically deconstruct an image to find, for instance, its area, center of mass, and orientation. It is this last property, described using the *second central moment*, that I use to interpret anisotropy in a height field.

The second central moment is used to find the direction of highest intensity in an image, which can be used to uncover its orientation. For my method, I consider the projected distance (or height) of a point on the plane as 'intensity'. Intuitively, ridges in the height field will have the highest such intensity. Ridges, by definition, have lowest curvature along their major axis. By finding this direction, I then find an approximation to the direction of least curvature of the neighborhood.

These major and minor axes of intensity can be found by first constructing the central moments [122] up to order two:

$$M_{pq} = \sum_i u_i^p v_i^q F(u_i, v_i) \qquad [5.7]$$

$$\begin{aligned} \mu'_{20} &= M_{20}/M_{00} - \bar{u}^2 \\ \mu'_{02} &= M_{02}/M_{00} - \bar{v}^2 \\ \mu'_{11} &= M_{11}/M_{00} - \bar{u}\bar{v} \end{aligned} \qquad [5.8]$$

Axes of intensity are then found by taking solving for the eigenvectors and eigenvalues of the covariance matrix:

$$cov[F(u,v)] = \begin{bmatrix} \acute{\mu_{20}} & \acute{\mu_{11}} \\ \acute{\mu_{11}} & \acute{\mu_{02}} \end{bmatrix} \quad [5.9]$$

One minor wrinkle: as described, this technique works only for ridges. To properly characterize neighborhoods that represent valleys, I simply invert the heights, replacing $F(u,v)$ with $-F(u,v)$ in equation (5.7).

5.3 Results

In this section, I discuss the performance and correctness of multi-scale geometry descriptors. For all results below, the symbol '$\pm$' represents a range of one standard deviation from a given value.

5.3.1 Performance

To assess the performance of my method, I built a corpus of 30 meshes, ranging in size from 2,600 to 170,000 vertices. Descriptors were computed over the surface of each mesh. On my test set, the time needed to construct these descriptors using quadratics ranged from 3 seconds to 4 minutes per mesh on an Intel Core 2 Duo, E8500 with 3GB of RAM. Runtime was generally a function of the largest neighborhood size, as a larger neighborhood has more vertices to consider, as well as the size of the mesh itself, as a set of descriptors is built for each vertex. In testing, construction of vertex neighborhoods takes about the same amount of time as fitting the quadratics over those neighborhoods. I found moment computation to run three times faster than quadratic fitting. Combining these timings, moments can be calculated, on average, 50% faster than quadratics.

My system provides the ability to cache the results of computing descriptors onto disk for future use, so in cases when meshes must be visited many times, such as in the matching scenario I describe in Section 5.4.4, I do not have to repeatedly reconstruct them.

5.3.2 Evaluation

Tradeoffs were made in the construction of multi-scale descriptor to achieve high performance. In the following sections I assess the impact of these tradeoffs, both in terms of accuracy, as well as in terms of sensitivity to both noise and to differences in surface tessellation.

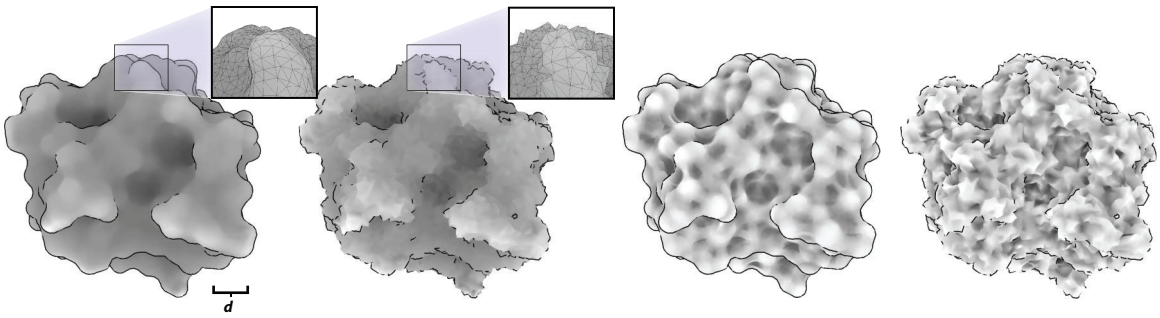
5.3.2.1 Descriptor Equivalence

I describe a set of moment-based descriptors which further reduce computational requirements. These may not necessarily agree with quadratic descriptors. In these cases, I consider the quadratic descriptors to be the the 'correct' result. To assess the degree to which moments get the wrong answer, I ran both algorithms against the same corpus of meshes. On each mesh, I found the average angular difference over each mesh between the principle direction produced by moments and the direction produced by quadratics. For all meshes, this value averaged $16^\circ \pm 12^\circ$. The curvature values produced agreed with each other with an R^2 value ranging from .81 to .95. This suggests that moments, by and large, produce similar results to quadratics, a result reinforced by visual inspection.

I also compared multi-scale curvature to Rusinkiewicz's local curvature estimation [35]. To do so, I constructed descriptors with small neighborhoods (equivalent, approximately, to 2-rings). I found that the principle directions found by quadratic descriptors deviated, on average, $14^\circ \pm 11^\circ$. My curvature values agreed with the mean of theirs with an average R^2 value of .83. It is difficult, however, to compare our methods directly as their method aims to deal with instantaneous neighborhoods directly, while ours cannot.

5.3.2.2 Geodesics vs. 2-Ring Approximation

I described in Section 5.2.1 a simple method for improving Dijkstra searches by adding virtual edges between 2-ring neighbors. This method, while more tolerant of issues with tessellation, is still an approximation to true geodesic distance. To quantify the improvements my method produces, and to determine how close my approximation comes, I compare geodesic neighborhoods



(a) Large scale (b) With surface noise (c) Local curvature (d) With surface noise

Figure 5.5 A depiction of the results from a test of my shape descriptors' sensitivity to surface noise. (a) Shows my results on the surface of a protein, with lighter colors denoting areas of higher positive curvature. (b) Shows the same surface after noise is introduced. This test is repeated in (c) and (d) using local curvature estimation. Note the similarities between the results in (a) and (b), indicating a resilience to noisy surfaces, unlike (c) and (d).

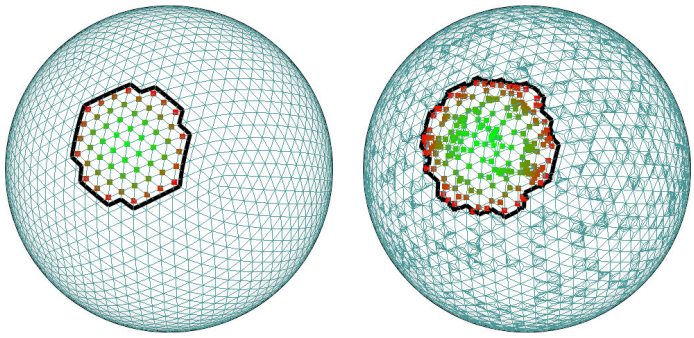
of varying size, generated using the accurate method in [131], on the surface of my corpus to both my method, as well as to standard Dijkstra distance.

I find that for all protein surfaces in my corpus, for a patch radius of 8 Ångströms, my 2-ring approximation finds between $97-100\%$ of the vertices in a geodesic patch of the same radius, with an average distance for the missing vertices of .1 Ångströms from the patch boundary. Standard Dijkstra finds $84-90\%$, with an average distance within .5 Ångströms of the boundary.

I also tested against a torus model, containing a "grain" in its tessellation. Because, unlike proteins, this model does not have an intrinsic scale, I assign one according to its median edge length, 'e_m' $= .8$. For patches of size $d = 10 * e_m$ $.6$, my approximation finds $92-95\%$ of the vertices, with an average distance of .02. Standard Dijkstra finds only $76-80\%$, with average distance of .05.

5.3.2.3 Noise Sensitivity

To gauge the ability of my surface descriptor to tolerate noise, I used the same models as above, and then perturbed each vertex a random distance, up to .5 Ångströms, along its normal. Figure 5.5 shows one example protein, before and after this process. For all other models, vertices were



(a) A sphere (b) Randomly tesselated

Figure 5.6 This figure illustrates one of my test cases for tessellation sensitivity. (a) Shows a sphere, with 5,120 faces. (b) Shows the same sphere after 5,000 rounds of subdivision, with replacement. Also shown is a sample neighborhood on each sphere. Note the uneven distribution of sample points arising from subdivision.

perturbed up to $.75 * e_m$. Curvatures were assessed for each vertex on the original model, then compared against the same vertex on the perturbed model.

For small patch sizes of around 2 Ångströms (or $2 * e_m$), I found that this difference between respective vertices, on average, accounted for $19\% \pm 26\%$ error in the reported curvature, versus the actual curvature. For patches 4 times larger, this dropped to $2\% \pm 6\%$. In contrast, local curvature had $25\% \pm 36\%$ error. This is in accord with my visual observations that descriptors formed from larger neighborhoods seem more resilient to noise.

5.3.2.4 Tessellation Sensitivity

To test my method's sensitivity to varying tessellation, I used models with uniform tessellation and known curvature. Tessellation "noise" was introduced by repeatedly selecting a triangle from the mesh, inserting a new vertex in the center of this triangle, and projecting this new vertex onto the surface of the model (for this reason, models with known analytic forms were used). This process creates meshes with high variability in the tessellation, and many examples of bad tessellations, such as high valence vertices and sliver triangles.

Two models were used in testing, a unit sphere and a unit torus; the sphere model began with 2,562 vertices, to which 5,000 more are added during subdivision. Figure 5.6 shows the sphere

before and after subdivision. The torus begins with 4,800 vertices, and again 5,000 are added. One minor wrinkle in my testing: because the tesselated meshes had a much smaller e_m relative to the non-tesselated meshes, I set the tesselated model's scale equal the original.

Tests were conducted in a similar manner as in Section 5.3.2.3: each vertex in the original mesh was compared against the same vertex in the tesselated mesh. In the sphere, average error ranged from $.02\% \pm .01\%$ for small patches to $.07\% \pm .1\%$ in large patches. In the torus, error ranged from $.2\% \pm .1\%$ to $1\% \pm .3\%$, respectively.

5.3.2.5 Foldover

As discussed in Section 5.2.2.1, my algorithm discards vertices that face away from the plane to avoid having foldover bias my results. In my protein corpus, I have found that, on average, $23\% \pm 12\%$ of the vertices of the largest patches (8 Ångströms) are discarded. Though this sounds wasteful, on average the closest discarded vertex is 2.75 $\pm$ .73 Ångströms from the center of the patch, which means that, on average, only the largest descriptors have any foldover, and those have enough samples to be meaningful. Other meshes in my test set performed similarly, or better.

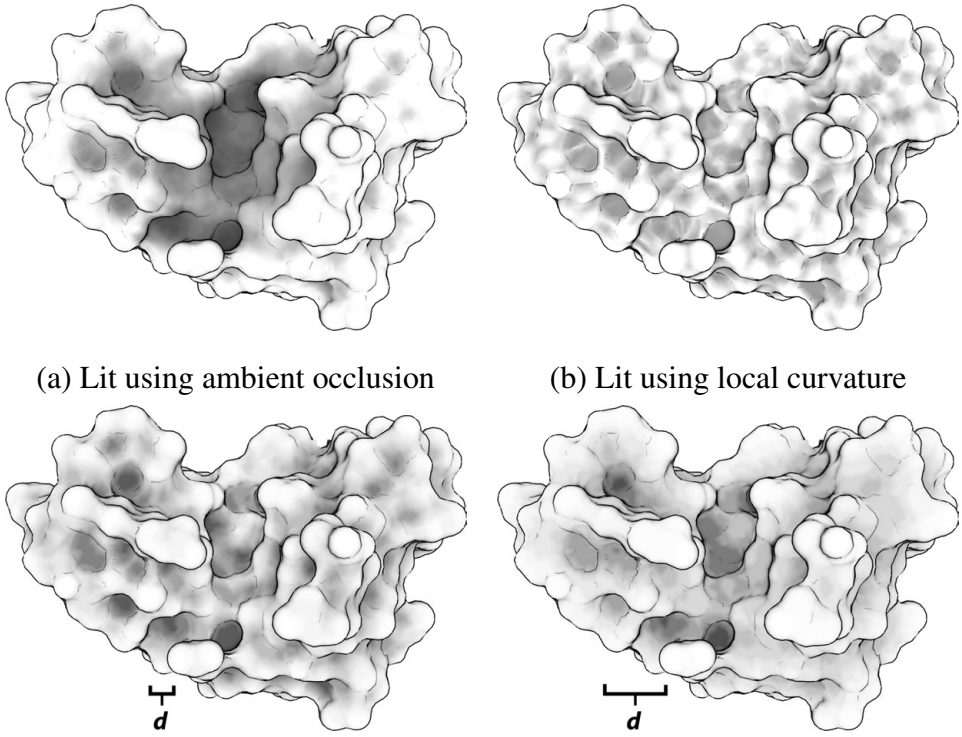
5.4 Applications

I have found a number of applications that benefit from multi-scale, anisotropic descriptors. In the following sections, I describe several of these applications. This list is not intended to be comprehensive, but rather a sampling of the possible uses for my method.

5.4.1 Multi-scale Lighting

Rusinkiewicz, et al., [50] describe a method for multi-scale lighting which uses curvature information at large scales. They show that by darkening regions of the mesh that are concave at these scales, they produce results similar to those of local ambient occlusion.

My system does not use curvature directly, but rather uses a descriptor built from large neighborhoods to identify concave regions. These are similarly darkened. Figure 5.7 demonstrates my results, and compares it against simply lighting using local curvature. Note that, especially with



(a) Lit using ambient occlusion (b) Lit using local curvature

(c) Lit using small descriptors (d) Lit using large descriptors

Figure 5.7 Depicted here are four lighting schemes applied to a ribonuclease molecule (PDB ID 1MO7). (a) Is lit using ambient occlusion, which darkens interior points. This is a global effect, which helps to emphasize the large active cleft. (b) Uses local curvature, darkening concave regions. Note that very small features are emphasized, but the cleft is hard to make out. (c) Uses my descriptor, with an atom-sized (4 Ångström) neighborhood radius. (d) With a residue-sized (8 Ångström) radius.

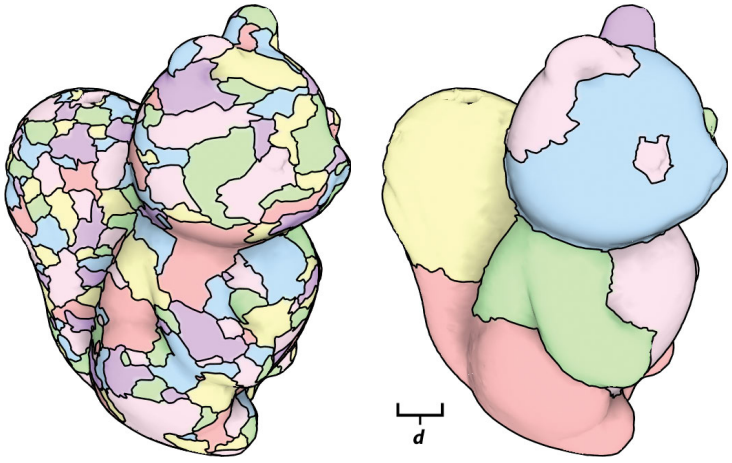


Figure 5.8 A demonstration of the usage of large-scale descriptors within the watershed algorithm [59]. On the left, the result of segmenting the surface using local curvature values. On the right, shape is estimated using my descriptor, and the same algorithm run. Note the improvement in patch boundaries, and the lack of small, isolated segments. The same overflow threshold of .9 is used in both cases.

larger descriptors, lighting with a multi-scale descriptor produces similar results to ambient occlusion. Because, however, my algorithm only captures local phenomena, it is unable to correctly shade flat areas that are deep within a groove. Nevertheless, I achieve interesting lighting results, which can, for instance, darken features of a particular size for better emphasis.

5.4.2 Segmentation

I now look at the task of surface segmentation, which operates on the curvature of a surface to produce regions of similar curvature. An early surface segmentation technique, called the watershed method [59], segmented the surface according to curvature. Segments are demarcated by ridges of high curvature, such that all vertices within a segment have lesser or equal curvature than the boundary vertices. Segments can flow into neighbors if their height is lower than a threshold parameter. This parameter, defined by the difference between the curvature at the highest-curvature vertex in the segment and the lowest, has a large effect on the number of segments produced.

The watershed method, while fast and simple, is sensitive to noise in curvature. Tuning for the ‘minimum height’ of a segment can help by merging some small patches caused by noise.

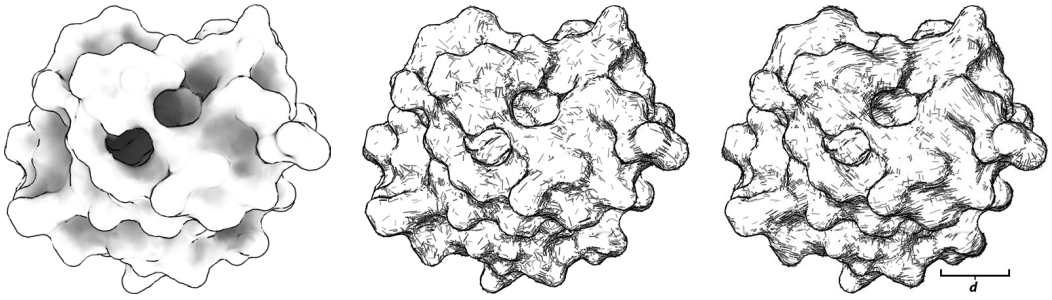


Figure 5.9 A demonstration of hatching using multi-scale descriptors, with lines drawn along directions of principle curvature. On the left, a Hydrolase molecule (PDB ID 6RNT). In the center, local curvature is used to place lines, using the method described in [35]. On the right, curvature is estimated using a much larger patch (with a radius of 8 Ångströms). Note that the stroke lines are more uniformly oriented, with fewer 'stray' lines.

Unfortunately, this may also cause larger segments to merge, too, making precise segmentation difficult. In Figure 5.8, I show that by simply changing the segmentation to use larger patches, which in turn exhibit smoother variations in curvature, I see a significantly improved segmentation, without a need for excessive tuning.

5.4.3 Stylized Rendering

Next, I demonstrate the effects of large-scale curvature on hatching. Hatching simulates the brush strokes an artist might make to convey the shape of a surface. Girshick, et al. [53] note that these strokes very often are made along the lines of principle curvature on a surface, and that their length and distribution is directly related to the degree to which a surface is curved.

Flat regions, containing little detail, need only a few strokes, or possibly none at all, to convey their shape. Highly curved regions, meanwhile, need far more to properly convey.

I show that depending on application, it may not be desirable to emphasize all small features, especially bumps and wiggles. To do so may require adding numerous additional strokes, cluttering up the image, reducing understandability and increasing rendering times. On the surface of a protein, for instance, atom-scale features abound. Representing all of these using hatching strokes makes for a less comprehensible image (see Figure 5.9).

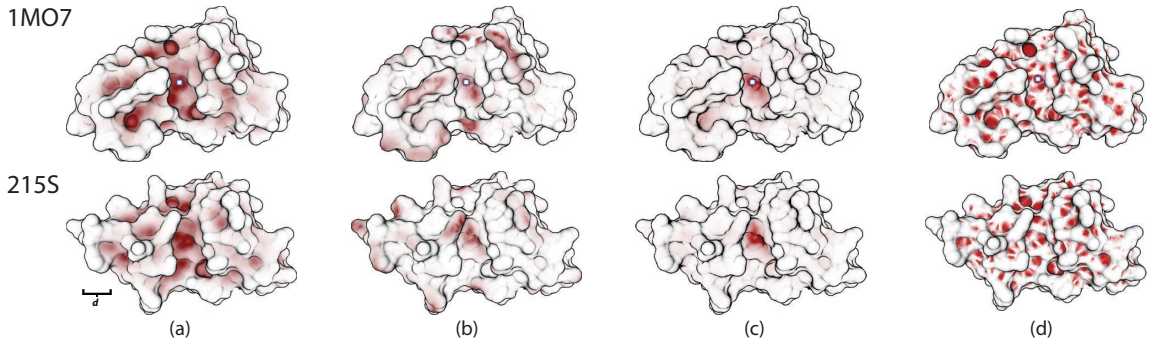


Figure 5.10 Matching the surface geometry for two ribonuclease proteins (PDB IDs 1MO7 and 215S). Both have similar functional sites, occupying the center groove. (a) Shows large-scale curvature matching for the picked site (blue and white circle in the center of each top image). In (b), degree-of-anisotropy is used instead. In (c), both metrics are combined, yielding a more accurate match in the bottom image to the picked point in the top image. Finally, (d) shows, for comparison, the results from matching using Rusinkiewicz's local curvature estimation technique [50].

5.4.4 Multi-scale/Anisotropic Curvature Matching

Finally, I look at the task of surface matching: given a point on the surface, find similar points, either on the same model, or on other models. I am interested in this task as a component of matching functional sites between proteins. Figure 5.10 demonstrates a preliminary result, produced by simply matching the curvature and degree of anisotropy for a source point against those of all other points.

Each vertex is assigned a feature vector with 8 values, formed from the curvature and anisotropy of four differently-sized patches centered at that vertex. The match between two vertices is then given by the normalized dot product of the feature vector for each vertex.

This process forms the basis for my binding prediction algorithm, described in the next chapter.

Chapter 6

Binding Prediction

6.1 Introduction

In the last chapter, I presented a surface descriptor that was able to characterize the shape of a surface at multiple scales. The formulation of this descriptor arose naturally from my investigations into the binding of small ligands to the surface of a protein. I found that a classical measure of curvature, being local and instantaneous, did not robustly describe binding complementarity. Indeed, the curvature in a pocket complements that of the bound ligand not only at atomic scale, but also at the scale of residues and even pockets.

In this chapter, I first describe work done to incorporate physio-chemical features into my multi-scale *shape* descriptor to turn it into a multi-scale *functional surface* descriptor. The goal of this task is to provide point descriptors for the functional surfaces of proteins. These descriptors summarize the values of various properties of the functional surface in a small neighborhood around a particular point as a vector of numbers.

I then describe an algorithm which uses these descriptors to investigate the binding of small molecules (i.e. *ligands*) on the surface of a protein. I approach this as a machine learning problem: under the assumption that the shape and chemical microenvironment surrounding a specific atom in a ligand is unique to that atom, I train a classifier using a corpus of examples of that bound atom to recognize its microenvironment 'signature'. This training process is described in detail in Section 6.3.

While signatures for specific atoms provide limited information about ligand binding as a whole, they serve as a key building block for considering these larger regions. In the *prediction* phase, described in Section 6.4, I describe a method that first uses each signature to construct an atom-binding probability function, defined on sample points distributed over the entire surface. In Section 6.4.3, I describe how to combine these *atom predictors* into a *moiety predictor*, by combining atom-binding functions into a moiety-binding function, using the known structure of the moiety itself as a guide.

6.1.1 Contributions

The contribution in this chapter is an approach to ligand binding prediction that leverages the shape and electrochemical properties of the functional surface at multiple scales. This approach involves several novel contributions:

- **A functional descriptor**, in the form of a feature vector definition, which characterizes the environment surrounding a point on the functional surface at multiple, biologically-relevant scales. This descriptor combines the electrochemical properties contained Altman's FEATURE descriptor [8] with multi-scale curvature, similar to Olson [41]. In contrast to these methods, however, my descriptor is derived from an explicit tessellation of the solvent-excluded surface, and not from the implicit functional surface formed by surface atoms. I thus approach the problem of surface representation in a new way.

- **A method to train atoms, using this descriptor**: this uses my functional descriptors to train a classifier on the microenvironment surrounding an atom. Similar to Nassif [87], my method uses a SVM classifier operating on an encoding of the multi-scale features on the surface. While their method, and indeed all training methods that I am aware of, train on the binding partner as a whole, mine trains on each atom using a separate classifier tuned to that atom. This approach is described in Section 6.3.

- **A method for automatically generating a diverse corpus of training examples**, using a moiety as an exemplar. In this step, I use a simple subgraph-isomorphism technique to

locate similar moieties in a corpus, taking care both to prune homologous proteins from the resulting set, as well as to return only those moieties that are completely bound to the surface.

- **A method for combining atomic predictions into moiety predictions**, using known structural information about the moiety. This method shares similarities with blind-docking systems such as AutoDock [134, 135] in that it is aimed at finding the location of small ligand binding, and is designed to take into account ligand flexibility. We differ in approach: blind docking runs an exhaustive search of the energy landscape of a protein, which can take many hours. In contrast, my method uses the predicted locations of atom binding, with respect to the entire ligand, to perform a much less exhaustive search of the potential locations (but not orientations) of ligand-binding.

6.1.2 Method Overview

My algorithm for predicting the location of ligand binding is separated into two tasks, *training* and *prediction*.

In the training phase, ligands are first broken into moieties, and a training corpus is built for each moiety (Section 6.3.1). Each moiety is then further broken into atoms, and a classifier is built for each atom based on the various microenvironments found to surround that atom in the corpus (Section 6.3.2).

Prediction proceeds in the opposite direction: first each of the atomic-classifiers that were built during training are run on samples which are evenly distributed over the surface of a query protein. This produces the probability, for each sample, that that atom could bind there (Section 6.4.2). To reduce false positives, atomic predictions are then grouped into moiety predictions, using the structure of the moiety as a model for grouping (Section 6.4.3). Finally, moiety predictions are merged into a final ligand prediction (Section 6.5).

6.2 The Functional Surface Descriptor

6.2.1 Descriptor Scales

As described in Section 5.2, my multi-scale descriptor is designed to characterize the curvature and anisotropy of a disc-shaped region of the surface of a triangle mesh. The radius of this disc determines which points are used, and it is these points to which a quadratic is fit. The descriptor may therefore be fit to any scale by varying the radius R of this disc. Practically, of course, the larger R is, the larger the potential residual of the fit will be.

The *functional surface descriptor* introduced here, in contrast, is a "feature vector", which contains a fixed set of dimensions. Each dimension represents a particular feature of interest (be it shape, or an electrochemical property) at a particular scale (disc radius). Multi-scale properties, then, are encoded in the feature vector by first deciding on a fixed set of radii, and then adding a dimension for each combination of feature and disc radius. For instance, if we decide to use 3 radius values (say, 1Å, 2Å and 3Å), and we desire to add curvature as a feature, then there will be 3 dimensions for curvature, one sampled at each radius value.

Having a compact descriptor is of paramount importance: with more features comes larger storage requirements, slower comparison, and potentially the curse of dimensionality. Further, multi-scale features often change slowly (or not at all) across a wide range of patch radii. Thus a balance must be struck between robustly accounting for the shape variations across scale for a point on the surface (by sampling at a large number of radii), and having a compact descriptor (and thus a minimal number of radii in the descriptor).

Striking this balance is equivalent to finding the largest possible set of scales that, across all desired features, exhibits minimal covariance between scales. To find this set, I evaluated the covariance of curvature and electrostatics across 20 scales. The smallest scale, 1.6Å, was chosen to be small enough to account for atomic features, but large enough that every patch of this size contained at least five points (not counting the center point): the minimum number to avoid an under-constrained fit. The largest size, 8Å was empirically chosen to capture pockets of approximately the size of the largest moiety to be characterized.

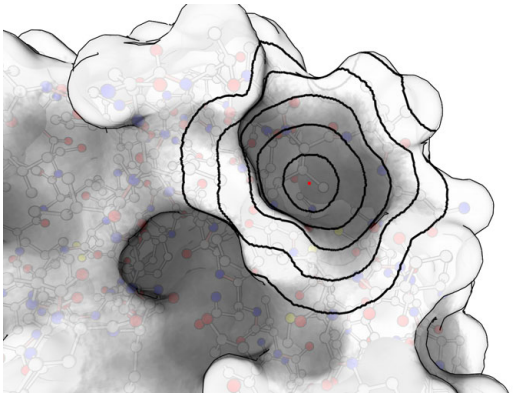


Figure 6.1 Shown here are, for one sample point, the disc-shaped patches of each radii used in the functional surface descriptor: 1.6Å, 3.2Å, 4.8Å, 6.4Å and 8Å.

Remaining scales were removed if their statistical correlation with any other scale approached 1.0, which indicated that they conveyed little new information. I ultimately chose the following set of 5 scales: 1.6Å, 3.2Å, 4.8Å, 6.4Å and 8Å. These also map to the scale of important biological features, the first to the size of an atom, the next two to the size of a residue, and the last two to the size of small pockets (see Figure 6.1 for a depiction of these sizes).

6.2.2 Descriptor Features

In addition to its shape, a molecule induces a variety of physical and chemical effects in the neighborhood surrounding it which strongly affect binding specificity. Many of these are relevant at multiple scales: for instance, hydropathy and electrostatic potential may affect binding directly on the surface (at a small distance and thus a small scale), or further away, where we may think of the potential as a blend of the potentials on the nearby surface.

These features, therefore, may be modeled at multiple scales. For a given patch, the weights are constructed as in Section 5.2.1.1. For sample i, if electrostatic potential is computed as $elec(v)$, then the weighted average electrostatic potential for all vertices v in the neighborhood of radius R around i, $N_R(i)$ becomes:

$$elec_R(p) = \sum_{v \in N_R(i)} \widehat{W_i(v)} \cdot elec(v) \qquad [6.1]$$

Refer to equation 5.2 for a definition of $\widehat{W_i(v)}$. Hydropathy is computed similarly.

Other features, such as the presence of hydrogen bond donors or acceptors, make less sense when considered at multiple scales, as these interactions are highly localized. Instead, these appear in the feature vector as one value: the distance between the sample point and the nearest available hydrogen bond donor/acceptor. Other facets of the atoms and residues of the underlying protein are treated similarly: the distance to the nearest non-polar backbone/sidechain atoms, and to specific, biologically meaningful polar backbone/sidechain atoms.

Refer to Table A.1 for a complete functional surface descriptor definition. In the remainder of this document, all samples/features will follow this definition.

6.2.3 Normalization

Because each feature spans a different range of values, samples must be normalized so that the scale of the distributions for each feature are similar. In my initial implementation, samples on the surface of a protein were normalized against all the other samples on that protein. Unfortunately, this introduced a per-protein bias which made comparison between proteins inconsistent (as features may have quite different statistics from one protein to the next).

In this document, I use samples drawn over a set of proteins randomly selected from the entire PDB (100 in all) to compute an overall mean and standard deviation for each feature. Each feature, then, is normalized against these values before being used.

6.3 Per-Atom Training

Before predicting the binding behavior of a ligand, the learning algorithm must first be trained on the specifics of that ligand's binding behavior. This step has two phases: first, a corpus of training examples is found, drawn from the Protein Data Bank (Section 6.3.1). Then, that corpus is used to train a machine-learning algorithm to recognize its specific binding preferences (Section 6.3.2).

This learning process is broken down to the level of the atoms themselves, rather than the entire ligand. I base this decision on two fundamental assumptions, which are drawn from experimental observations. First, I assume that *the preferred microenvironment for each atom in a ligand is*

different. This difference may be small, as in the case of two neighboring atoms of similar size and polarity, or large, as in, for instance, the difference between the predominantly negatively-charged phosphate chain in ATP and the positive adenine moiety. Intuitively, this assumption is justified by the lock-and-key principle: each atom in a moiety has a particular property (polarity, shape, etc.) which is different from all other atoms in the moiety. Thus complementarity implies that the matching surface for each of these atoms will also have different, complementary properties.

The second assumption is that, for a given atom, *the microenvironment surrounding that atom is consistent across all proteins to which it binds*. For this to be true, in the samples in close proximity to a given binding atom, the feature vectors for each sample must form one or more clusters in feature space (i.e. its 'signature'). This implies that samples binding a given atom may be distinguished from samples that don't, given a classifier trained on a corpus large enough to encompass all of the possible binding microenvironments of an atom. This assumption is key to the success of my work: if this were not true, then my binding prediction algorithm would be impossible, as separating positive and negative examples would itself be impossible.

6.3.1 Building Training Examples

If a specific binding microenvironment to which a particular atom might bind is not seen during training for that atom, then a classifier would be unlikely to recognize it in a tested protein (except perhaps by chance). It is therefore essential that a large, diverse corpus be used during training. Unfortunately, for many of the ligands that we may wish to predict, there are only a few examples available in the PDB to use. Thus if we limit the scope of our search to only those ligands, we may miss infrequent yet legitimate binding configurations.

Fortunately, nature reuses many of the same building blocks, or *moieties*, in a number of different ligands (for instance, the adenine moiety in CoenzymeA, ATP, $NADP^+$, etc). I take advantage of this fact by training and classifying surfaces based on moieties, not ligands. To start, the ligand to be trained must manually broken into moieties. Each of these moieties is then compared against all ligands in the Protein Databank. In this comparison, any ligand that contains within it the exact

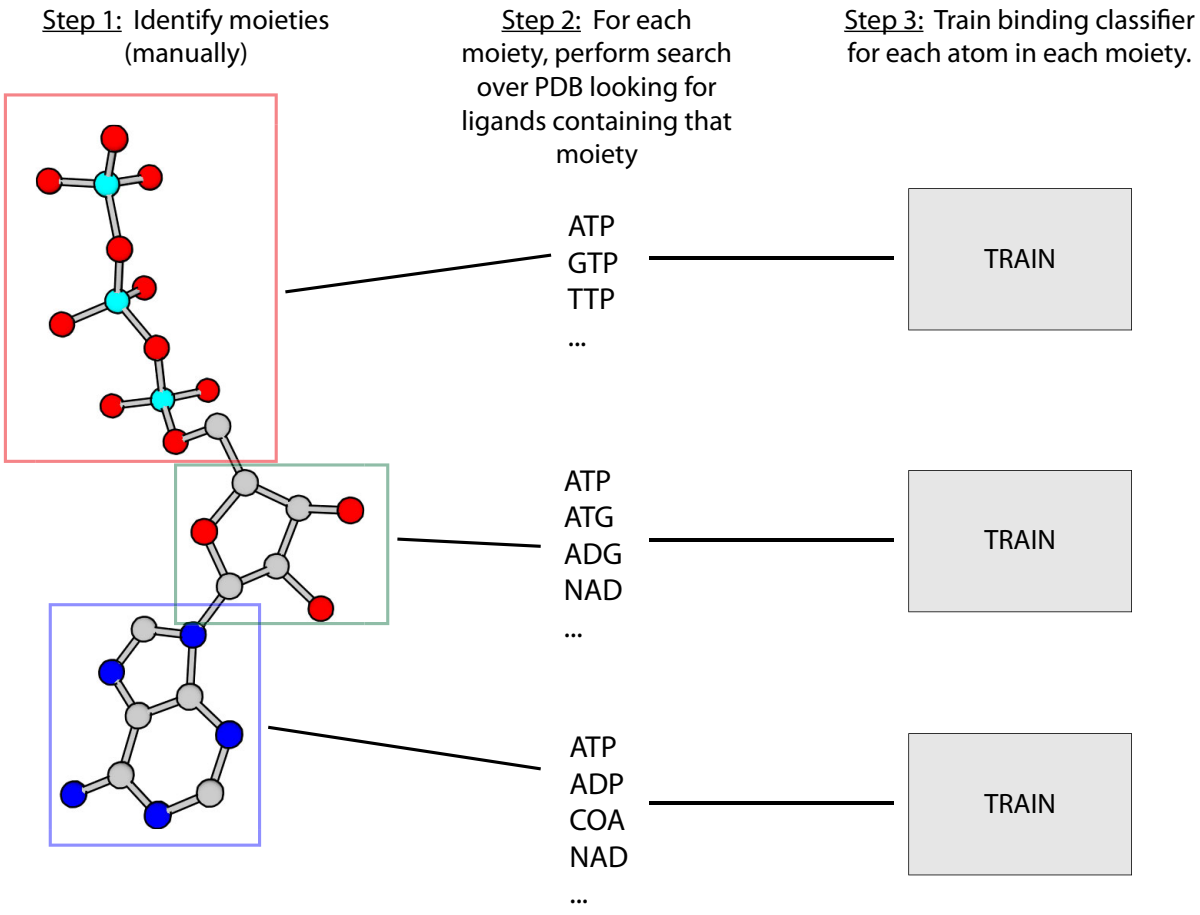


Figure 6.2 Shown here is ATP, broken up by its moieties. Ligands are found within the PDB which contain each moiety, and these examples are used to train a classifier.

Algorithm 1: The corpus-building algorithm

Input: All proteins in the PDB (clustered by sequence identity), an exemplar moiety M
Output: A non-homologous training set of functional surfaces, each with a binding ligand that contains M

```
foreach Protein P in the PDB do
    if The cluster C containing P has not been used yet then
        if P contains a ligand L that matches M, whose atoms bind to surface S then
            Add S and L to returned set
            Remove cluster C from consideration
        end
    end
end
```

same configuration of atoms and bonds as the moiety is considered a match; variations in atom type, bond type, or the number of bonds connected to an atom result in a negative match.

After removing homologous matches, these PDB files are used as a training corpus for that moiety. Because a moiety may appear in a range of ligands, this algorithm helps to increase both the number and diversity of training examples. This is illustrated in Figure 6.2 and described in detail in Algorithm 1.

In this algorithm, homology is based on the clusterings produced by BLASTCLUST (using data available at ftp://resources.rcsb.org/sequence/clusters/). Proteins are considered homologous if they contain greater than 95% sequence identity. This leaves 27551 clusters (out of 66961 proteins).

6.3.2 Training an Atom Learner

Once a corpus has been built, the training process begins.

6.3.2.1 Sampling the Surface

As a first step, feature vectors are built for points on the surface. For the sake of brevity, I call these *samples*. Because my multi-scale descriptor is tolerant to noise and tessellation, and because electrochemical properties do not change quickly as a function of surface distance, I need

not densely sample the surface. Nevertheless, it is important that samples are distributed closely enough to capture important features.

As a convenient solution, I use the vertices formed during tessellation with MSMS [108], with the sampling density set at approximately 3 points per $Å^2$. This means, for instance, that a carbon atom, having a radius of 1.74 Åand thus a surface area of about about 10.7 Å, will contain around 30 samples. By visual inspection, this appears to be enough to represent all geometric detail at the smallest scale.

For the learning phase, the entire set of points is used. However, classification can be significantly sped up by intelligently reducing the number of samples on the surface. This will be described in more length in Section 6.4.1.

6.3.2.2 The Training Algorithm

In this step, a classifier is trained for each atom in a moiety. At a high level, the algorithm proceeds as follows: first, for every atom in the moiety to be trained, all samples are found from all proteins in the training corpus that come within 1.6 Å that atom. These are classified as positive examples. Negative samples were randomly chosen from the remainder of the surface. Though there are many more negative samples on a surface than positive, adding more negative examples does not necessarily improve predictive accuracy [136]. Further, classification time increases with the number of samples, as noted in Section 6.6.3.2. The negative set is therefore chosen to be the same size as the positive set. Both sets are then used to train a classifier to recognize that atom. This process is repeated for every atom in the moiety.

To decide on a classifier to use, I experimented with the Naive Bayes, Bayesian Network [137] and Support Vector Machine (SVM) classifiers contained in Weka [138, 139], using the test described in 6.6.2.4. I found that for the atomic recognition task, a SVM classifier with a radial basis kernel [140] tended to have the best classification accuracy.

Refer to Algorithm 2 for a detailed description of this process, and to Figure 6.3 for a visual depiction.

Algorithm 2: The algorithm for training a set of classifiers C_A on a specific moiety M given a test corpus containing M.

Input: The training set, along with a moiety M
Output: A set of classifiers C_A, one per atom A in M
foreach *Surface S in the training set* **do**
    Sample the entire surface: one sample per vertex in S
    Construct the feature vector for each sample
end
foreach *Atom A in M* **do**
    Construct a classifier C_A
    foreach *Surface S in the training set* **do**
        Find the set of samples P_{pos} nearest to A in S
        Randomly pick an equal number of points P_{neg} on the rest of S
        Classify P_{pos} as positive examples, P_{neg} as negative
        Add both sets to C_A
    end
end

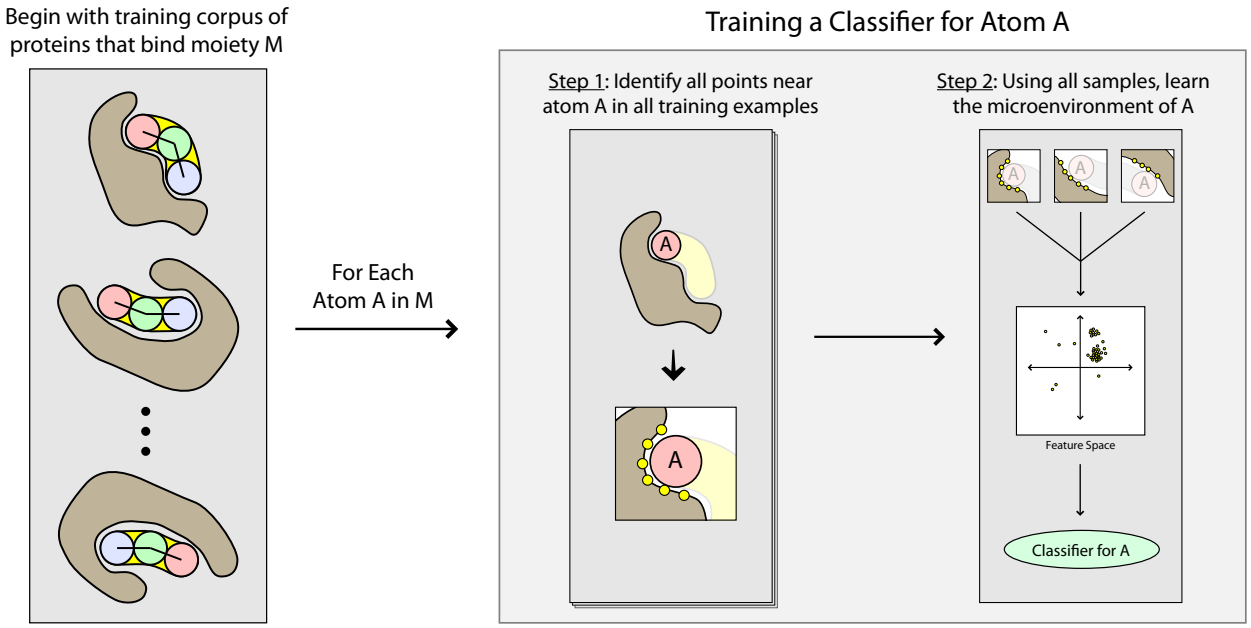


Figure 6.3 A visual depiction of Algorithm 2, for training a classifier to recognize the environment surrounding a specific atom, given a corpus of examples of that atom's binding.

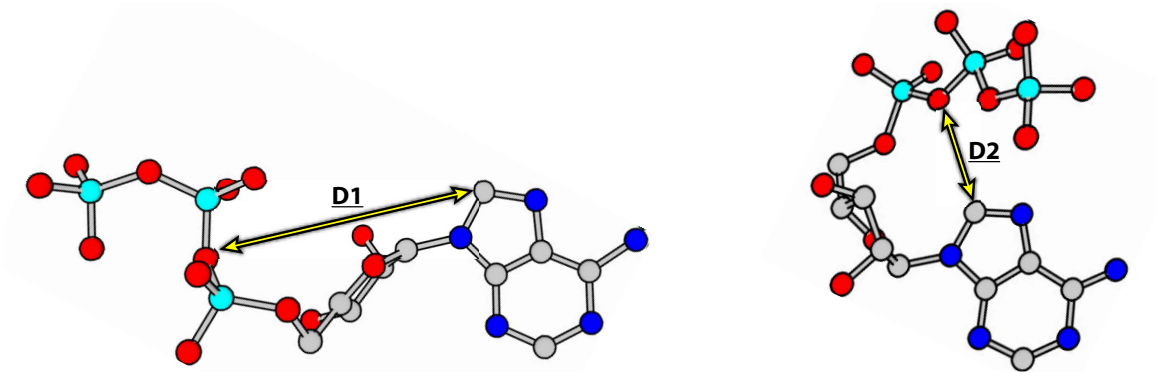


Figure 6.4 Shown here are two possible conformations for Adenosine Triphosphate (ATP). Note that the distances between atoms within the rigid adenine moiety do not change. Distances between non-rigid components, such as those between the 'C8' and 'O3A' atoms, may change dramatically. As these will be used later to combine atomic predictions, the observed minimum and maximum distances between each pair of atoms are stored during the training phase.

6.3.2.3 Atomic Distance Measurement

As will be described in more detail in Section 6.4.3, my algorithm uses structural knowledge about the moiety as a guide for combining atomic surface predictions. Specifically, to combine a prediction for atom A with a prediction for atom B, it needs to know the allowable range of distances that A and B can be from one another with respect to the moiety that contains them both; it would make no sense to combine predicted locations for both atoms that were either too close or too far from one another to be physically plausible.

Thus the final aspect of training for a moiety, separate from building the per-atom classifiers themselves, is the computation of all-pair distances between atoms in that moiety. My algorithm does this by finding the minimum and maximum bound for the distance between each pair of atoms in all ligands encountered during training.

For rigid structures, the bounds for these distances will be quite tight, as any two atoms will appear at similar distances from one another across the corpus. But for flexible moieties, this process accounts for their structural flexibility (see Figure 6.4).

The computational time taken in this step is negligible.

Algorithm 3: Overview of the moiety-prediction process: using trained classifiers to predict the presence and location of a moiety on the surface of a molecule.

Input: Trained classifiers for a moiety M, and a test surface X
Output: For each sample on X, the probability that M binds to that sample

```
Sample the entire surface as in Section 6.3.2.1
Build out feature vector for each sample
foreach Atom A in M do
    Cull samples according to the statistics of A to speed up prediction (Section 6.4.1)
    foreach Sample S in culled set do
        Using C_A, predict the probability of A binding at S (Section 6.4.2)
    end
end
Combine atom-binding probabilities P_A(X) into moiety-binding prediction P_M(X)
(Section 6.4.3).
```

6.4 Per-Moiety Prediction

In this section, I describe how to use the classifiers that were built in Section 6.3 to predict the binding of the classified moiety on new surfaces. Refer to Algorithm 3 for a description and Figure 6.6 for a visual depiction of this process.

6.4.1 Reducing Sample Count

As was noted in Section 6.3.2.1, a full set of samples is used in the training phase. This is done because during training, nothing is yet known about the statistics of the surface features, so there is not yet enough information to determine which features are important for discrimination and which are not. Therefore, the learning algorithm cannot identify 'redundant' points — those which are both physically adjacent on the surface and also close in feature space along the features which matter.

This culling could be performed after all samples have been found across the corpus, and a future version of this algorithm may benefit from such culling. Fortunately, however, the training phase is neither time nor space-critical: training is performed once, often as an off-line process, and the resulting classifiers are cached for future use.

Algorithm 4: A greedy algorithm for grouping samples on the surface according to feature distance

Input: A set of samples S_{in} on surface X
Output: A non-redundant set of discs D_{out}

Set initial disc size to $R = 4$Å, distance threshold $T = .5$

while $R \geq .5A$ **do**
  $DiscSet$ = []
  foreach *Sample* S *in* S_{in} **do**
    Build disc D of radius R centered at S
    if *All samples in* D *are in* S_{in} **then**
      Compute the average Feature Distance ($AvgFD$) between samples in D
      if $AvgFD < T$ **then**
        Add [D, $AvgFD$] to $DiscSet$
      end
    end
  end
  Sort $DiscSet$ by $AvgFD$ (increasing)
  foreach *[*D*,* $AvgFD$*] in* $DiscSet$ **do**
    if *All samples in* D *are in* S_{in} **then**
      Add D to D_{out}
      Remove all samples in D from S_{in}
    end
  end
end
repeat
  foreach *Sample* S *in* S_{in} **do**
    foreach *Disc* D *in* D_{out} **do**
      if S *neighbors* D *and* $AvgFD$ *between* S *and* $D < T$ **then**
        Add S to D
      end
    end
  end
until *A sample* S *has been added to a disc* D ;

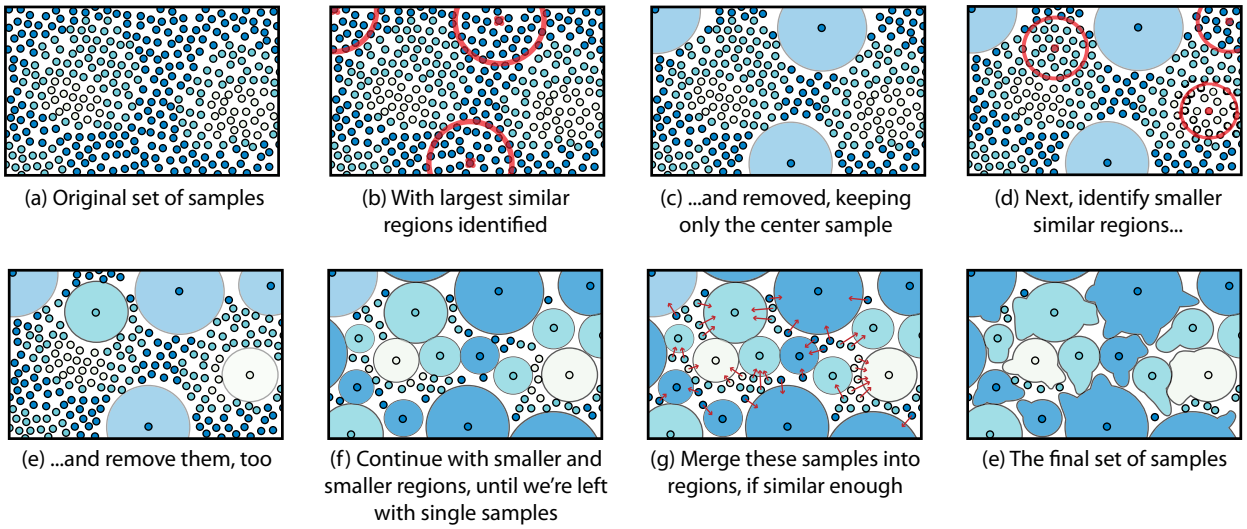


Figure 6.5 An illustration (in 2D) of how my method for grouping samples on the 3D surface works. In this illustration, each circle represents a sample; samples having similar values in feature space are given the same color. The algorithm proceeds as follows: starting with a radius R, identify discs of radius R that have minimum average distance (in feature space) between elements in the disc. Replace the best non-overlapping discs with the sample in the center of each disc. Repeat, each time reducing the size of the disc. When complete, there will still be samples not contained in a disc. Merge those into neighboring discs if their distance from the center sample is less than a threshold T. The resulting center samples are used for surface prediction. In all results, $R = 4$Å and $T = .25$

Though high performance was not one of the goals of this project, there is a clear need to have classification finish as quickly as possible while still producing accurate results. As the runtime for classification is highly dependent on the number of samples (see Section 6.6.3), I have developed a fast sample-grouping algorithm to perform this reduction, which is described in Algorithm 4, and depicted in Figure 6.5.

The algorithm is designed to repeatedly find the largest possible disc (less than 4 Å in radius) such that all samples in this disc have an average feature distance less than some threshold T (in this case .5, or equivalent to half of the standard deviation over the corpus) from one another. When no more discs of a particular size can be found, the search begins again with a smaller disc, down to a radius .5 Å. Finally, since some samples were not assigned, those are merged into any

neighboring discs if the feature distance between that sample and the sample in the center of the disc is less than T.

Note that because each atom in a moiety has a slightly different microenvironment, each atom has a different set of 'important' features. I define important features to be those that, over the training set, have less variance than background, which, because each feature is normalized, equals 1. It is only these features that are used in the distance computation. Thus, this algorithm is run for every atom, and every atom gets its own set of non-redundant samples.

By grouping similar samples and culling all but one representative sample for each group, this algorithm significantly reduces the number of samples on the surface, usually anywhere from 5x to 20x, depending on overall sample similarity. This has a significant impact on prediction speed, as described in Section 6.6.3.2. Note that grouping samples in this way could result in an overestimation of the distance between samples. My algorithm compensates for this effect during each range search by relaxing the inter-atom distance constraints by the radius of the patch.

6.4.2 Predicting For an Atom

In my algorithm, predicting the location of a moiety begins with predicting the location of each atom in that moiety. In this phase, each atom classifier is run over the non-redundant samples produced for that atom on the test surface. For each sample, the classifier produces a probability that the atom on which it was trained binds to that sample.

For a surface X, the end result is a function over the samples on X, $P_A(X)$, indicating the likelihood that an atom A would bind at each sample. This is Step 1 in Figure 6.6.

6.4.3 Combining Atom Predictions

Once a set of probability functions $P_A(X)$ have been computed over the set of samples, with one function for each atom A in moiety M, the next step is to combine these predictions into a single function predicting the likelihood that M binds at each sample, which will be called $P_M(X)$. Note that since the culled set of samples is different for each atom, $P_M(X)$ will be over the *complete* (i.e. non-culled) set of samples.

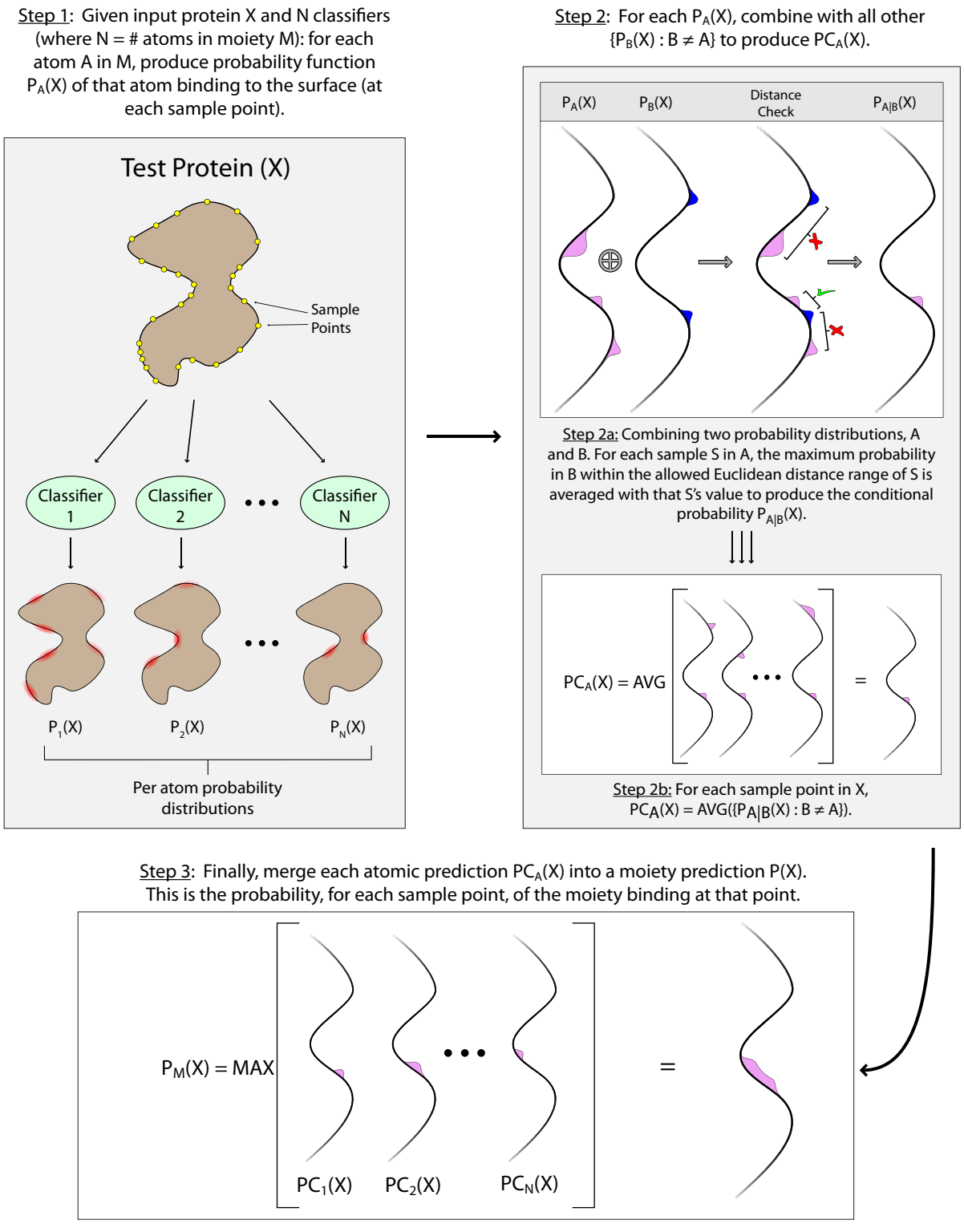


Figure 6.6 Prediction Phase: combining atom surface functions to predict a ligand.

Algorithm 5: A method for combining a set of probability functions $P_A(X)$, which indicate the likelihood of an atom A binding at samples on surface X, into set of probability functions $PC_A(X)$, which indicate the likelihood of A binding at each sample *con rmed by all other probability functions*.

Input: Probability functions $P_A(X)$, one for each atom A in moiety M
Output: Combined probability function $PC_A(X)$, one per atom
Let $D = [min(A, B)...max(A, B)]$ represent the range of allowable distance from atoms A to B
foreach *Atom* A *in* M **do**
    $PC_A(X) = P_A(X)$ **foreach** *Sample* S *in* $P_A(X)$ **do**
        foreach $P_B(X)$ *where* $B = A$ **do**
            Find all samples S_B in $P_B(X)$ within D of S
            Add average probability over these samples to $PC_A(X)[S]$
        end
    end
    $PC_A(X) /= M$
end

As shown in Step 2 of Figure 6.6, as an intermediate step toward this goal, the algorithm first computes for each atom A, $PC_A(X)$: the probability that A binds to each sample S in X, given the probabilities computed for all other atoms in M. In this step, the distance information that was computed in Section 6.3.2.3 is used when combining the probability functions for two atoms.

To recap: in a given ligand, any two atoms A and B only appear within a specific range of distances from one another. Therefore, if A has a high probability of binding to surface X at a point S, then we can *further con rm the legitimacy* of that prediction by seeing if there is a similarly high probability of B binding to X within the correct distance. Thus, $PC_A(X)$ refers to the probability of A binding at each sample *con rmed by all other probability functions*. This is described in detail in Algorithm 5.

Note that in merging individual probabilities $P_A(X)$ into conditional probabilities $PC_A(X)$, neighboring probability functions are averaged within the allowed distance window. This was chosen instead of using the 'maximum' neighboring probability so that spurious high probability scores do not have undue influence on the final conditional result.

Finally, as shown in Step 3 in Figure 6.6, all $PC_A(X)$ are merged into a final $P_M(X)$ by taking the maximum probability over all $PC_A(X)$. The intuition here is that false positives have already been accounted for by the previous step, so each $PC_A(X)$ should contain a good prediction of the binding location of A within moiety M. Therefore, merging these predictions will produce a final prediction of the locations where M will bind.

6.5 Merging Moiety Predictions

For those ligands that have been broken into multiple moieties, we now have multiple predictions $P_M(X)$ over the surface for the locations of moieties. In many proteins, only a subset of the moieties may bind to the surface of the protein, with the rest floating off the surface. Further that subset is not consistent: ATP, for instance, has three moieties, and within its training corpus, all possible combinations of these three moieties are bound to the surface.

The prediction that any one moiety binds, therefore, may be enough to predict ligand binding. I adopt a heuristic for merging moieties that simply returns back the highest probability over all moiety predictions as the final probability. In other words, for each sample S, $P(X)[S] = \max_{M \in L} P_M(X)[S]$. This method is robust against partial matches, as described above, but as a consequence is likely to confuse ligands that share many similar moieties (ADP and ATP, for instance). I discuss this limitation in more detail in Section 7.1.3.

6.6 Evaluation

I begin my discussion of the different tests I ran to gauge the performance of my methods by noting that I have performed only limited direct, quantitative evaluations with previous methods for ligand-binding prediction. In Section 6.7, I qualitatively compare several methods for binding site prediction with mine. These comparisons are meant to be illustrative, but will likely form the basis for a full comparison in the future.

In lieu of a direct comparison with previous methods, I instead set out to prove two specific hypotheses: 1) that by characterizing the surface in such a way that supervised learning techniques

could be brought to bear, I can predict potential locations of binding activity for specific partners; and 2) that incorporating multi-scale features into the surface description improves predictive accuracy and specificity.

I first evaluated my predictor by testing its performance on a simple example: ion binding. This test stressed only the per-atom prediction component of the predictor. I considered FEATURE [8] as a basis for comparison in this task, using their test corpus of 11 proteins, trained on 100 examples. To further evaluate the relationship between training corpus size and prediction performance, I ran this same test on a range of corpus sizes. This is described in Section 6.6.2.1.

I next expanded testing to the full algorithm, evaluating both how well a ligand predictor identified the presence of the ligand on which it was trained, as well as how likely it was to confuse the binding of another ligand with the one it was meant to predict. This test was composed of 4 test ligands: Adenosine Triphosphate (ATP), Glucose, Heme, and Dihydrogen Testosterone (DHT). For each ligand, 10 test cases were held aside from the training corpus for each moiety, chosen from among those proteins that fully bound that ligand, for a total of 40 test proteins. A ligand classifier was then trained on all remaining examples. Finally, each classifier was tested against all ligands (4x40 tests). Results for the full ligand test are described in detail in Section 6.6.2.4.

6.6.1 Training

All training examples were chosen from the Protein Data Bank, as described in Section 6.3.1. For ATP and Calcium, because so many examples were found, I randomly selected 25% from the found corpus as the actual training corpus. For the others, I used all found examples, except those chosen as test cases, as a training corpus.

Table 6.1 lists each of the moieties contained within the 4 test ligands. In addition, it lists information about how each moiety was trained, including a partial listing of which ligands were found to contain each moiety, and the total number of training examples found.

After collecting the corpus for each test case, atom classifiers were built for each moiety, as described in Section 6.3.2.

Ligand	Moiety	Training Examples	#
ATP	Phosphate Chain (PA O1A O2A O3A PB O1B O2B O3B PG O2G O1G O3G)	ATP, 5FA, CH1, CSG, CTP, D3T, DCT, DGT, DTP, GTP, TTP ...	45
	Ribose (C1' C2' C3' C4' O2' O3' O4')	ATP, 5GP, ACP, ADN, ADP, AMP, ANP, AP0, APC, ATG, C5P, FAD, GDP, GNP, GTP, NAD, NAP, NDP, RIB, SAH, SAM, SSA, UDP, ADP, ...	302
	Adenine (C2 C4 C5 C6 C8 N1 N3 N6 N7 N9)	ATP, ACO, ACP, ADP, AMP, ANP, ATG, CMP, COA, FAD, NAD, NAP, NDP, SAH, SAM, ...	268
Glucose	Glucose	GLC	88
HEM	oxygenated end (O1A O2A CGA CBA CAA C2A CMA C3A C1A CHA C4A NA CHB)	HEM, DHE, FDE, FDD, HAS, HCO, HDD, HEA, HEB, HEC, HEV, HFM, HIF, VEA, VER, ...	338
	non-oxygenated end (1) (CHB C1B NB CMB C2B C4B CHC C3B CAB CBB)	HEM, HDD, HDM, HEA, HEB, HEC, HFM, HKL, ...	338
	non-oxygenated end (2) (CHD C4C NC CAC C3C C1C CHC C2C CBC CMC)	HEM, HDD, HDM, HEA, HEB, HEC, HFM, HKL, ...	338
DHT	(O3 C1-10 C19)	DHT, AE2, AND, C0R, CLR, CPQ, DXC, FFA, HC2, HCY, STR, TES ...	65
	(O17 C8 C9 C11-18)	DHT, AE2, AND, ASD, EST, FFA, TES, WZA ...	65

Table 6.1 Listed are the ligands to be used as test cases, the moieties they contain, the ligands found during training which match each moiety, and the total number of proteins used as training examples.

PDB Code	1ANX	1AYP	1CGV	1CLM	1OMD	3CLN	1SAC	2SCP	3ICB	3PAL	5CPV
ROC Area	0.94	0.85	1.00	0.99	0.91	0.98	0.98	0.98	0.97	0.97	0.99

Table 6.2 Here are results from a test of how well my atomic predictor finds the binding location of calcium ions. Shown are the 11 proteins used as test cases by Altman et. al [8]. To test these, I first trained a predictor on 100 examples of calcium binding. I then evaluated each of the above protein surfaces using this predictor, generating an ROC curve for each test. The number below each PDB code is the area under its respective ROC curve. 1.0 indicates a perfect score, with all true positives found and no false positives.

6.6.2 Results

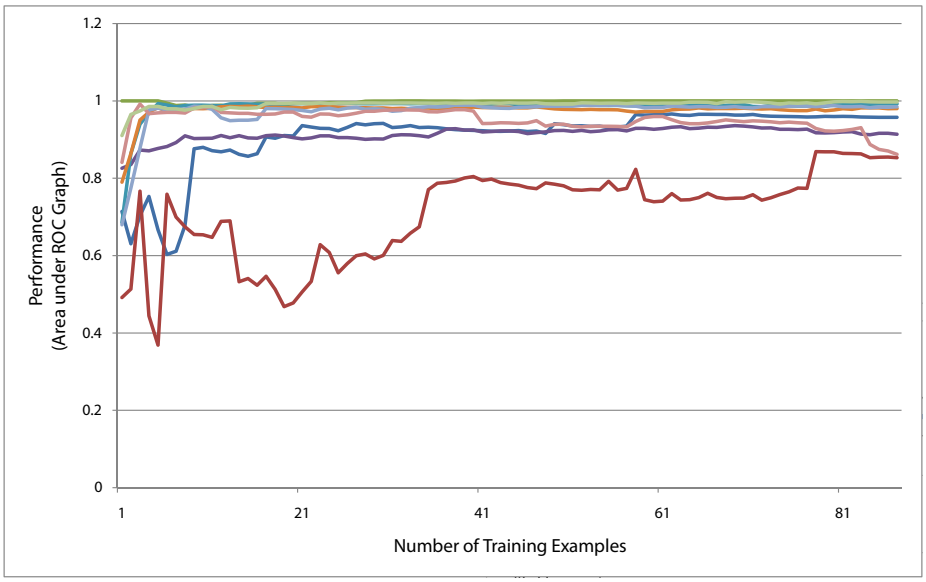
All tests in this section evaluate prediction performance against a surface. I take as positive examples those samples on the surface that come within 1.6Å of any atom in the ligand of interest which binds to that surface. Prediction performance is then evaluated against those samples: true positives are those samples near the ligand that have a prediction probability above a particular threshold, false negatives are those that have a probability below that threshold, and so on.

6.6.2.1 Per-Atom Predictor Test: Calcium Ion Binding

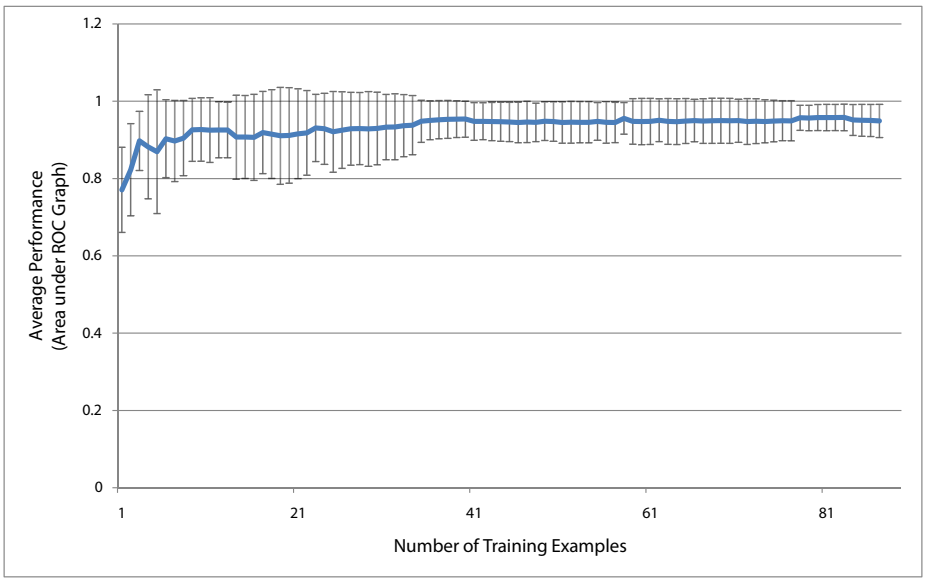
The performance of the full algorithm is highly dependent of the performance of individual atomic predictors. Therefore, evaluating these predictors forms an important first step toward validating the algorithm as a whole. In this section, I test my atomic predictor in isolation on calcium ion binding. This choice was not arbitrary: because this type of binding was used as a test subject for FEATURE [8], it serves as a basis of comparison with my predictor.

I tested atom prediction on the 11 proteins that formed their test corpus, listed in Figure 6.2. Training was done using a corpus of proteins which bind to calcium. To make the test fair, none of the tested proteins appeared within training set. Further, none were found to be homologous (at 95% sequence identity) to any protein in this set.

Without detailed statistics on the performance of FEATURE as its discrimination threshold is varied, it is difficult to make a full comparison. They show 91% sensitivity at 100% specificity in independent tests, which tracks very closely with my results of 92% sensitivity at 99% specificity.



(a) One test case per line



(b) Average + 95% conf. interval

Figure 6.7 Shown here is the performance over all test cases of calcium binding (Table 6.2) as a function of the size of the training corpus. Performance is measured by the area under the ROC graph produced by each example. On the top, each test case is shown as a separate line, each with a different color. Note that while with only a few training examples, the correct pocket is found in most tests, a few harder test cases require more training before they can be reliably predicted. On the bottom is the same data, but averaged, with error bars indicating 95% confidence intervals.

6.6.2.2 Testing the Impact of Training Corpus Size

In supervised learning tasks, proper training is essential to achieving maximal prediction performance. Having too many examples can result in a predictor that is at best needlessly complicated, and at worst over-fit, thus sacrificing generality. Having too few may also result in poor performance: since we can only identify binding configurations that look like something we have already seen, it is essential that those areas of feature space be adequately sampled.

Figure 6.7 demonstrates this property. Both graphs depict predictive performance as a function of the size of the corpus. As can be seen, with only a few examples, many sites of calcium binding are completely misidentified. As more examples are added, test performance goes up. Around 80-100 examples are necessary, in the case of calcium binding, to accurately predict the location of binding in all test cases. Interestingly, after about 90 examples, the performance of a few test cases starts going down, possibly indicating overtraining.

6.6.2.3 Small Molecule Test: Glucose

I next compared my predictor to that of Nassif, et al. [87], to determine our relative performance. In this test, I used both their positive training set (29 surfaces) as well as their positive testing set (14 surfaces). They show that by using feature selection, they are able to get an average of 89% sensitivity with a 93% specificity. In my test of the same set, I was able to get a comparable 86% sensitivity at 93% specificity.

My method did have a few anomalies: at 93% specificity, my method had very poor sensitivity on several examples: 1Z8D and 2F2E, at 17% and 26% respectively. I do not yet know the reasons for this poor performance, especially given how well my method was able to classify all other test cases. Unfortunately, the Nassif paper does not break their results out by protein, so I was not able to confirm if they, too, were successful at classifying these examples.

Finally, when they do not use feature selection, their performance drops to well below mine, at 76% sensitivity and 84% specificity. As I do not currently use feature selection, this indicates to me that my method could potentially benefit from removing unnecessary features.

Ligand	Test Cases (PDB codes)
ATP	1A0I 1A82 1ASZ 1E8X 1FMW 1G5T 1GN8 1JWA 1XEX 2BUP
GLC	1AC0 3ACG 1FAE 1GWM 1H5V 1J0K 1JLX 1K9I 2J0Y 2ZX3
DHT	1AFS 1DHT 1I37 1I38 1KDK 2PIO 2PIP 3KLM 3L3X 3L3Z
HEM	1AOQ 1B0B 1CC5 1D0C 1DLY 1EW0 1SOX 1MZ4 2HBG 2Z6F

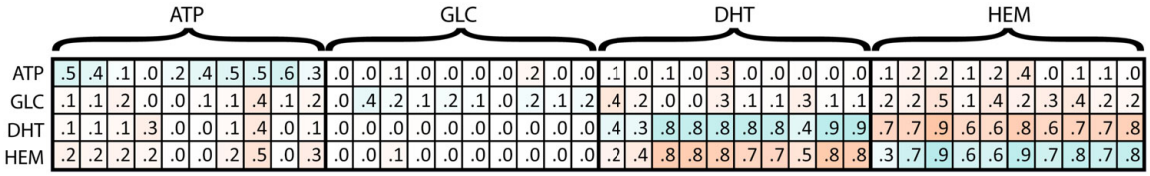
Table 6.3 A list of the four ligands used during training, and, in each row, a set of 10 test proteins to which that ligand binds. None of these proteins were included in any training corpus.

6.6.2.4 Ligand Prediction Test

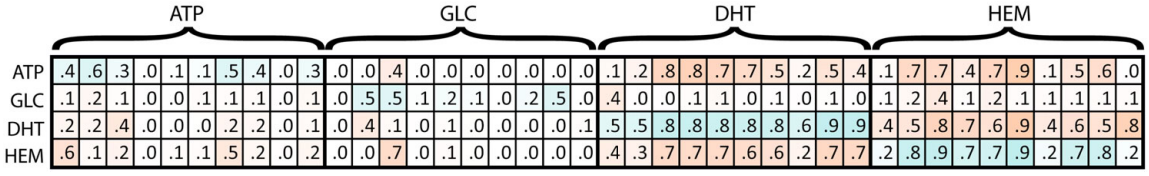
Here, I test the complete algorithm, including the moiety prediction steps described in Section 6.4. For this test, I manually curated four sets of 10 proteins, chosen so that each protein in a set binds to the same ligand, but differently (as much as is possible) than all other proteins in the set. Difference, in this case, was determined by visually assessing the shape and electrostatic distribution on the surface. As before, none of these test cases share more than 95% sequence identity with any protein in their respective training corpus. The PDB codes for proteins in each of these four sets are listed in Table 6.3.

Figure 6.8 shows a confusion matrix over all test sets. Here, all 40 test proteins are tested against the four predictors; the value in each cell is the area under the resulting precision/recall (PR) curve. Note that, unlike the above tests, I do not use ROC area. The reason: because the ratio of positive (binding) samples to negative (non-binding) samples varied widely with both the size of the ligand and the size of the protein (and was generally quite low), ROC area did not allow for intuitive comparison between test cases. Precision/recall curves better account for this disparity.

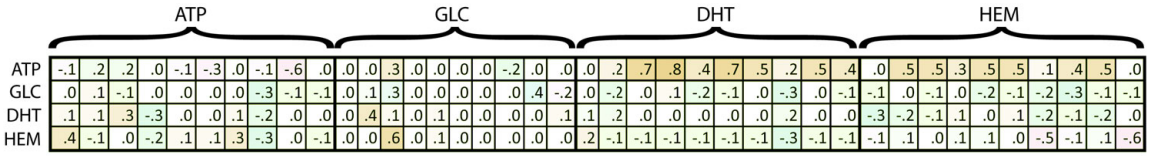
Figures 6.9 and 6.10 show the performance of each test case (ligand) in isolation: each curve represents the ROC and PR curves resulting from a test of a specific ligand, using a binding predictor trained on that ligand. For the ROC curves, lines along the diagonal represent a prediction that is no better than chance; lines above the diagonal show predictive utility — the further above, the better. For PR curves, lines closer to the upper right corner represent a better-performing prediction.



(a) Confusion matrix, using full descriptor. The value in the cell is the area underneath the precision/recall curve produced from that test. A higher value indicates a better match. Green cells indicate positive results: the predictor found the ligand it was trained for. Red represent negative: the predictor found a potential binding location for a ligand that it was not trained to predict.

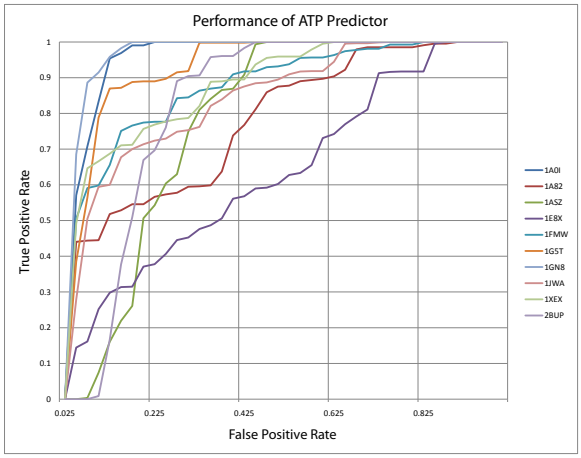


(b) Confusion matrix, using only local scales (removing features larger than 1.6Å). Graph colors and values have the same meaning as in (a)

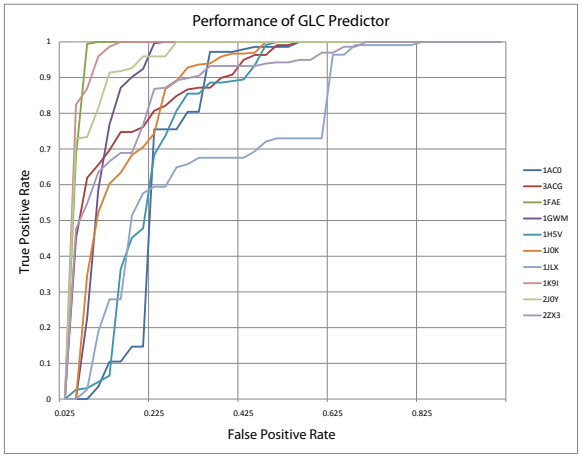


(c) The matrix for (b)-(a). Cells colored in red represent tests where using only local scales resulted in *worse* performance. Cells colored in green represent those that performed better without larger-scale features.

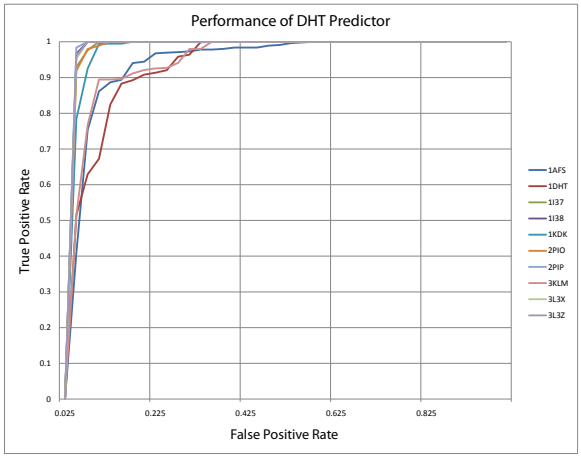
Figure 6.8 Shown here are two confusion matrices, the top tested using the full feature vector description (listed in Table A.1), the bottom using only the most local features. Each matrix is constructed using the following setup: each row represents training done on a different corpus, all of whose members bind to the specified ligand. Each column represents a specific test protein; these are grouped by the actual ligand that binds with that protein. Note that while results along the diagonal for (b) and (c) are about the same (except for a few cases that perform much worse), the loss of large-scale features increases confusion, especially between ATP and DHT.



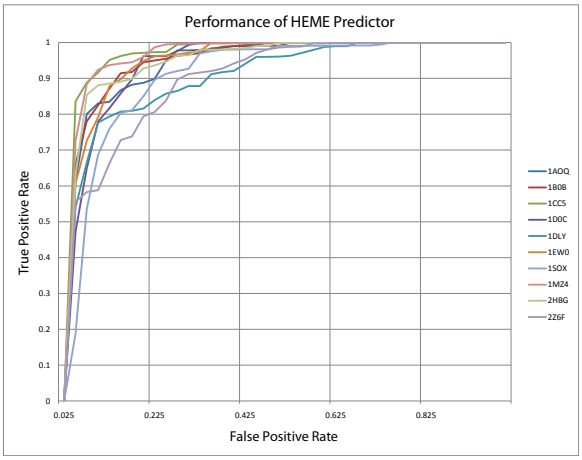
(a) ATP



(b) GLC

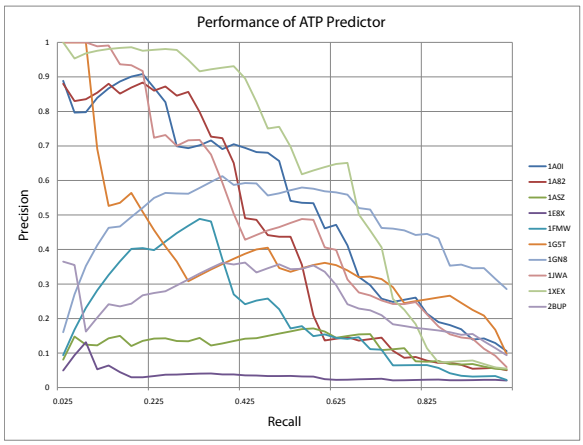


(c) DHT

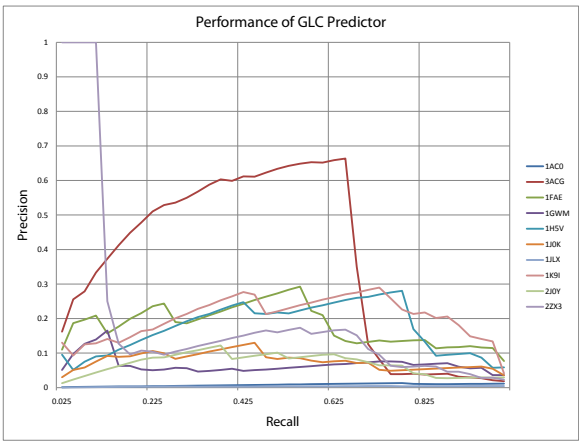


(d) HEM

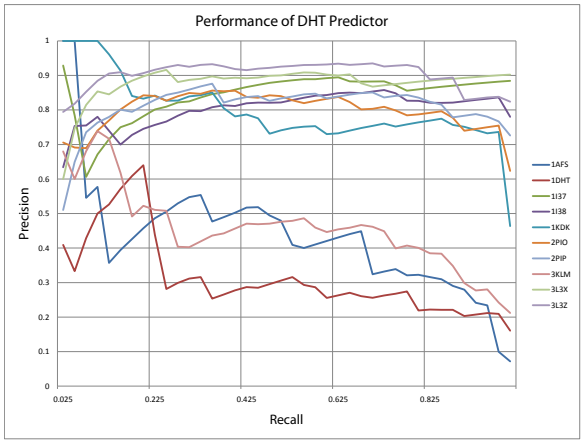
Figure 6.9 Shown here are graphs of prediction performance: for each of the above graphs, a predictor was trained on the corpus binding a specific ligand. The task: to use that predictor to find the site of binding on the 10 proteins in 6.3 known also to bind that ligand. Each graph contains ROC curves for each protein in the set.



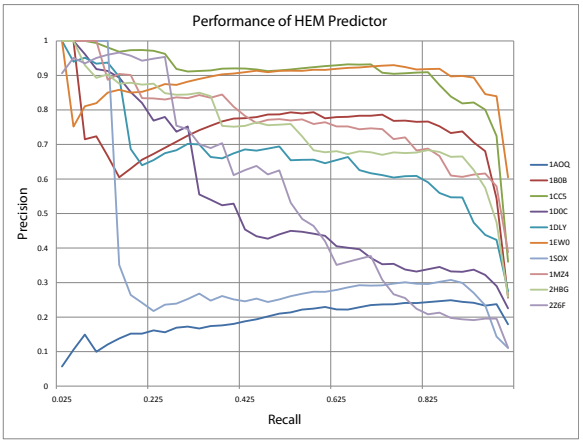
(a) ATP



(b) GLC



(c) DHT



(d) HEM

Figure 6.10 Shown here are the precision/recall curves produced for each ligand when tested against training data for that ligand. This is the same test as in Figure 6.9.

With this in mind, the graphs show that the predictions my algorithm made for DHT and HEM were very accurate, the majority having between 60 to 90% precision at every recall rate. ATP was less accurate: though the actual binding site did appear in areas of the highest probabilities on the surface, so did other sites that should not have. Finally, glucose performed very poorly, which was especially evident in the precision/recall curves. I explain why this may have happened in the discussion below (Section 6.8).

6.6.3 Run-Time Performance

In the following sections, I discuss the running cost of my testing, training and grouping algorithms.

6.6.3.1 Training Run-Time

The time needed to build samples is dominated by three steps: surface tessellation (MSMS), computing electrostatic potential (APBS), and multi-scale shape description. On a 2.8Ghz Core I5 computer with 4gb of RAM, the total time needed to build samples was found to take from 15 seconds for a small protein (1B7V - 70 residues) to 3 minutes for a large protein (1N1H - 1260 residues). The APBS step often dominated, taking well over half the total time. Though I did not pursue this, it is worth considering if using a simpler electrostatics method may produce adequate results at lower runtime cost.

Samples, once built, are cached for future use. Each classifier is also serialized and saved to disc, allowing it to be reused. I have not yet taken advantage of distributed computing platforms, such as Condor. This task, however, is well suited for a distributed approach: surfaces can be independently computed, so each one (or a bundle of them) could be sent out to a separate computer.

6.6.3.2 Testing Run-Time

Classification performance is highly dependent on both the number of atoms in the training moiety, as well as the size of the protein. Sampling density stays fixed as size is increased; therefore, the number of samples grows linearly with the overall surface area of the geometrical surface of a protein.

Having a large number of surface samples becomes especially problematic during the surface-combination phase, shown in step 2 of Figure 6.6. This step requires n^2 surface function combinations, where n is the number of atoms in the moiety. Each combination requires iterating over every point on one surface and comparing it to a small set of points on another. It's easy to show that the size of this set is a function of both the areal sampling density and the radius of the search. The former is fixed, and the latter is constrained by the physical size of the moiety.

For performance reasons, the surfaces for two atoms that are more than 15Å apart are not combined. Thus the overall time to to combine two surfaces is proportional to the number of samples times the search time. I use the Approximate Nearest Neighbor library [141], to accelerate this search process: finding k points out of n samples with this library takes $O(k \cdot log(n))$. Since I have a fixed upper bound for k, this means that combining two surfaces is an $O(n \cdot log(n))$ operation, and combining all surfaces is $O(n^2 \cdot logn)$.

Surface grouping, therefore, is essential to making my algorithm run fast; with an average 10-fold reduction in the number of samples comes an over 100-fold reduction in run-time. On the system described above, this reduces the time to classify a mid-sized protein from over an hour to less than a minute. In the final algorithm, grouping usually accounts for about one-third of the total classification time; thus, the resulting speedup more than compensates for its cost.

Besides the cost of classifying large numbers of samples, the time it takes to classify a given sample using Weka's LibSVM classifier increases as the training corpus grows. I am currently investigating methods for ameliorating this problem, either by using a simpler classifier, or by intelligently reducing the number of samples used to train.

6.7 Comparison to Existing Methods

In this section, I select several seminal or interesting methods for characterizing small-ligand binding and discuss their results, comparing with mine when possible. Of course, this is not intended to be a comprehensive assay, as there are many hundreds of methods for pocket detection and binding-site prediction, but rather illustrative of the current state of research into binding-site prediction.

6.7.1 Thornton, Spherical Harmonics

Kahraman and Thorton [81] describe a method for assessing the similarity of binding pockets based on a comparison of the spherical-harmonics decomposition of each pocket. The authors use this descriptor to test a diverse set of binding pockets, including those for ATP, Androgen (similar to DHT), HEM and glucose.

Their primary conclusion is that the shape of the binding pockets for a given ligand varies more than the shape of the ligand itself. This means that the shape variation of a pocket is greater than can be accounted for by ligand flexibility, and thus complementarity is neither necessary nor sufficient for predicting ligand binding.

To support this conclusion, they show that, using a spherical-harmonic description of the pocket, the shape of ATP, AMP and steroid pockets are easily confused with one another. Glucose pockets, on the other hand, are not often confused with anything but phosphate, thus indicating that size does matter for pocket comparison. Interestingly, in contrast to our method, which readily confuses DHT and HEM, their research shows that the shape of HEM pockets is unlike that of any other pocket.

They further show that rigid ligands, like DHT and HEM, are perfectly classified using just shape, whereas flexible ligands are not. My results, especially the ROC curves shown in Figure 6.9, substantially agree with their conclusions.

6.7.2 Kihara, Real Time Search

The method recently introduced by Chikhi and Kihara [89] is designed to allow for extremely quick lookup of protein structures, using a query structure as a template. Their method requires a few minutes per protein of preparation time to build the surface, identify pockets and compute 3D Zerninke moments. But after preprocessing, they claim that their database can return back the results of a query against hundreds of proteins in a few seconds.

So while the preprocessing cost for a single protein in their method is comparable to mine, theirs is several orders of magnitude faster for search. The question is: all else being equal, does the extra time taken by my method result in better performance? The answer: a qualified 'yes'.

Unfortunately, the statistics they provide for their results are not easily comparable to mine. They evaluate the percentage of queries in which the searched-for pocket is returned back as the top result or is in the top 3 results. They test on several datasets, including the Kahraman set [81], which contains three of the same ligands as mine: ATP, GLC and HEM. For these, on average, their method (using both pocket shape/size and electrostatic potential) finds the correct pocket on the first try 57%, 40% and 50% of the time, respectively. When expanding to the top three picks, their method improves to 92%, 100% and 86%.

Because my method is less pocket-centric, I am not able to account for my results in the same way, so I will have to estimate. For the sake of comparison, let the top 1 pocket metric be equivalent to a 95% sensitivity (given the relative proportion of surface are to pocket area of most proteins, this is about right) and top 3 be equivalent to 85% sensitivity. Then my method finds the 'top 1' binding site 60%, 55% and 83% of the time. For top 3, my method improves to 82%, 71% and 97% of the time.

Again, these methods use neither the same data set, nor the same model for evaluation. Thus, this test this is far from conclusive. But given my assumptions, it appears that my method is more likely to find the correct binding site in the first try. I get mixed results for the top 3: less likely for ATP and Glucose, and more likely for HEM.

6.8 Discussion

As can be seen in the above results, this method may aid in the prediction of surface binding. In many cases, it also can discriminate between binding partners (GLC and ATP, for instance). Unfortunately, it does not reliably discriminate in all cases (e.g, HEM and DHT). I discuss this issue and several steps to take to potentially ameliorate it in Section 7.1.3.

As shown in Figure 6.8, by including multi-scale features in the feature vector definition, I gain a small improvement in prediction reliability, especially in ATP and HEM, where having multi-scale information seems to be crucial to the detection of a few test proteins. In glucose, however, multiscale information seems to *lower* predictive performance. This could be the result of having such a small binding interface: any differences in features at large scales would be irrelevant to binding preference, diluting the influence of more-important features and perhaps even harming overall prediction.

On the other hand, having multi-scale features seems to be crucial for my method to correctly discriminate between ATP and both DHT and HEM. This could be the result of there being too much noise at the smallest scales to allow for a clean separation between positive and negative samples. Or perhaps this is because at the smallest scales, the microenvironment surrounding atoms in one ore more of the moieties of ATP looks much like that of atoms in the moieties of DHT and HEM; differences only become obvious at larger scales.

Though my approach to binding-site prediction seems promising, there are significant limitations that need to be addressed before it can be useful. One major limitation is demonstrated by the poor performance of glucose in the full ligand prediction test (Section 6.6.2.4): as expected, my atomic-predictors are not reliably able to discriminate between the putative binding of a single atom and its binding with respect to the ligand. Thus, when looking at the probability of a single atom's binding, there appear to be many false positives. Combining these predictions using the structural information contained in the moiety does help a great deal to remove these false positives, but unfortunately, the smaller the amount of structural information, the less effective surface

combination becomes. Glucose, it appears, is too small to be reliably predicted by my current method.

Interestingly, however, my method performed much better in the more limited glucose-binding test in Section 6.6.2.3. It is possible that Nassif curated a better training corpus than I did, that its smaller size allowed it classification to generalize better, or that their testing examples were either easier, or more closely related to their training examples.

Another limitation is that in order to achieve a robust classification, the ligand trainer may require a large training corpus. Unfortunately, how large this set must be appears to be dependent on the variations in binding that a ligand may undergo. Further, even when broken into moieties, some ligands do not appear enough times in the database to be reliably trained (or to serve as a test case, for that matter).

Yet another issue with my method is its runtime: even with sample reduction, classification takes much too long to be applied at a large scale (both for large proteins and large numbers of proteins). This limitation could be addressed by using a large network, such as Condor, to distribute the computing load. Nevertheless the technique itself does not scale well. This may be solved by combining a more intelligent sampling algorithm (both for training and testing) with a surface-grouping algorithm that operates on a smaller, more selective group of samples.

Yet despite these limitations, the results I have shown above give evidence that my original hypotheses, outlined in the beginning of this section, are correct: that binding recognition for a ligand may be trained using supervised machine learning techniques, and that predictive accuracy is improved by incorporating features at multiple scales into a surface description. These results serve as a validation of the descriptor itself, and of the idea of leveraging atomic-predictors to predict ligand binding.

Chapter 7

Discussion

The problem of visualizing and analyzing the surface of a protein is a difficult one not only because of their inherent complexity, but also because of their foreignness. With no analogue in the real world and no proper orientation, it becomes very easy to get lost in the details. While many tools exist both to display and characterize the protein structure, few are optimized for handling the molecular surface itself. Fewer still are able to present this surface in a way that is intuitive and understandable. This dissertation presents a set of tools and methods to address these problems. Common to all of them is the idea that we may gain new insights by interacting with complex objects at multiple scales.

To aid in visualization, I have presented a complementary view of the molecular surface, called a *molecular surface abstraction*. With this method, a researcher can gain a quick appreciation of the gestalt of a complicated protein, including the gross shape and charge distribution over the surface, before switching back to a full-detail representation. By adjusting the level of smoothing, these abstractions may also operate at multiple scales, to emphasize features of varying sizes.

I have constructed a web server, called *GRAPE*, as a means to let biochemists explore molecular surface abstractions as easily as possible, overcoming their natural reluctance to try yet another complicated system. By browsing to the GRAPE website, they may quickly view an abstracted representation of their proteins. My hope is that this site will both give researchers new insights, as well as accelerate the adoption and use of abstracted molecular surfaces.

I have also shown the utility of multi-scale representations for binding prediction. I have started by creating a curvature descriptor that is able to operate at various scales relevant to binding. I then

expanded this representation into a full molecular surface descriptor, incorporating physical and electrochemical features, each sampled on the surface at multiple scales.

Finally, this full-scale molecular surface descriptor was then used to characterize the binding of small ligands, allowing their binding affinity to be predicted in yet-unclassified proteins. Though my work in this task is only in its inception and many challenges remain, I have demonstrated their potential usefulness for describing the functional surface. I further believe that my binding prediction algorithm is useful as an additional tool to help annotate the surface of previously unclassified proteins. While it is unfortunately not currently able to discriminate between regions of the surface that bind certain classes of ligands that look similar at the moiety level (such as HEM and DHT), it does show the ability to predict binding locations (in general) for medium to large ligands. This in itself could be a useful tool to apply to unclassified proteins, to point out new potential areas of study.

7.1 Issues and Limitations

There remain many issues and limitations both in my visual abstraction and numerical abstraction methods. Some of these could be considered future work, as they are not fundamental to the overall usefulness of the methods, and some are flaws that should be addressed in order to make these methods as robust as possible.

7.1.1 Visual Abstraction

Evaluation of Abstractions: To date, I have only performed limited evaluations of the utility of my visual abstractions. To justify their existence, I could go in one of two directions: first, I could perform a full controlled user-study, using members of the structural biology community as test subjects. This test would have to be carefully structured to avoid biasing users toward abstract representations, and to compensate for their potentially wide range of comfortability with viewing functional surface models. The second type of test

could be more informal: by allowing users to give feedback both on how useful abstraction tools were and on what discoveries they may have facilitated, I can gain insight into how visual abstractions are used by actual researchers. This might let me uncover which specific aspects of my method are the most useful, and which are unnecessary (or even counterproductive).

Ideal Scale of Abstraction: The abstractions in this document were all done using scales that I chose empirically, often on the basis of how pleasing the resulting visualization was. An ideal scale, however, would be found on the basis of utility, and would vary per-protein. Quantifying how useful an abstraction is to a user might be difficult, and perhaps different for each user. It would be an interesting study, however, to see how much the scale chosen determines utility, and if there exists a heuristic for finding the ideal scale.

Abstracting Larger Molecules: Besides the matter of finding the appropriate scale of abstraction, large proteins present their own abstraction challenges. My method is currently topology-preserving. This choice made sense for small proteins, as topology was often an informative property of the functional surface. For larger proteins, however, topological detail might get in the way. Thus, volumetric methods, which better handle topological changes, might need to be employed. Of course, finding the size at which a protein gets 'too large' presents its own challenges.

Bad Parameterizations with Exponential Maps: Exponential maps, being local and greedy, do a great job of parameterizing small surface patches with simple topology. However, they break down for larger patches, or for patches with vertices of high curvature. I have implemented techniques, such as splitting up poorly parameterized patches, for mitigating these effects, but these are not always completely reliable. Adopting global parameterization may result in a better surface parameterization for large stickers.

De ciencies in the GRAPE Viewer: While GRAPE was designed to be a simple, light-weight viewer, several features need to be implemented before it can be truly useful. At a minimum, a future revision should: allow identification of which regions of the surface are

proximal to specific amino acids in the sequence; support stick-and-ball, and perhaps ribbon visualizations; and allow for certain parameters (such as the color maps for electrostatic potential and the degree of abstraction) to be adjusted.

7.1.2 Curvature

Errors in Moment-Based Curvature Computation: Computing curvature using moments is much faster than, and often agrees with, my more-expensive quadratic method. In several specific cases, however, moments failed to agree with quadratic fitting: on edges, where quadratics were better able to fit in the absence of data, and in regions with significant foldover. I should investigate methods to compensate for these scenarios.

Evaluating the Need for Shape Description: While shape complementarity is presumed to be a useful guide to binding affinity, I have not yet evaluated whether incorporating these features is, in fact, useful, especially when viewed at different scales. I have some evidence that this is true: for instance, by looking at the information gain in individual descriptors. Resolving this question would require a study of binding complementarity at multiple scales, over a diversity of interfaces.

7.1.3 Binding Prediction

Confusion: My proposed method tends to confuse the binding surfaces for ligands that contain the same or similar moieties. Many of the following improvements may help to address this issue, but first a study of *why* they are being confused should be done and a targeted approach taken to reduce confusion without lowering predictive performance.

Surface Topology: Because my descriptor uses surface distance as an underlying metric, it is unable to capture phenomena that happen at distances farther apart than the radius of the largest neighborhood. In some proteins, however, the surface may 'fold over' onto itself, putting areas of the surface close together that are geodesically far apart along the surface. This could affect binding recognition if that foldover is biologically relevant. Study should

be done to see how frequently this occurs, in what cases binding prediction is affected, and if by adopting a volumetric approach, I may mitigate this problem.

Sampling Density: I chose a sampling density of 3 samples per $Å^2$ based on visual observations. A more thorough examination of sampling density could benefit my method. I might start by testing a range of values for performance, and choosing the one that had the best speed/performance ratio.

Sample Culling: Currently I use all samples on the surface during training. Many are likely redundant, but at this time I do not have a method to identify and remove them. It is possible that by simply evaluating their similarity in feature space, I could remove a large portion. As both the testing and training speed of Weka's SVM classifier is dependent on the number of training samples, this would help both parts of my binding prediction algorithm. I would need to be careful to make sure that I do not bias prediction, however.

Corpus Size: I have not yet come up with a way to determine the optimal number of proteins to include in the corpus (or if, indeed, such a determination could be made *a priori*). This would clearly help both the speed of training, as fewer proteins need to be trained. In addition, for those ligands that have few training examples, being able to determine training coverage would allow my method to give some indication of the confidence in which it is making its predictions.

Combining Atom-Location Predictions: The methods I use for combining atomic predictions to predict moiety binding are designed to be simple, and to represent a first step along the path toward an accurate prediction method. Fixing their flaws could improve prediction performance. First, the method that I chose to combine surfaces combines pairs independently. Thus, any knowledge gained about the conditional location of an atom does not benefit any other pairing. By iterating over pairings, I could further refine my results, taking advantage of any reduction of false positives in the previous iteration to better predict likely atom locations in the current iteration. Secondly, atom distance bounds are represented independently. In a real ligand, however, these bounds are highly dependent on one another, as the

ligand is constrained by its connectivity to take on a finite set of shapes. Utilizing these constraints would likely further improve predictive performance.

Combining Moiety Predictions: The simple method I have presented for merging moiety-binding predictions into a prediction of ligand binding assumes that when it predicts that a moiety binds in a particular location, this means that the full ligand also binds in that location. This is clearly not always the case, and might be one culprit in the poor performance that my method showed in binding confusion tests. On the other hand, in some cases, binding only happens for a subset of moieties. Thus, limiting a ligand prediction to only those areas where *all* moieties bind might miss this type of binding. More work needs to be done to find a balance between these two issues.

Using Energy Minimization: At no point during computation do I take advantage of the energy landscape to guide binding. This is by design, as such computations are expensive to perform over the entire molecule. As a final step to my algorithm, however, I could place the ligand into its predicted location (and potentially in a predicted orientation) and use energy minimization to guide it into a final position. The energy of binding at this position would give me a further indicator of the probability that a given prediction is, in fact, accurate. Culling inaccurate predictions, according to this metric, would likely improve my specificity.

Feature Selection: Currently, I do not perform any feature selection on the samples sent to each atom-classifier. As shown by Nassif, et al. [87], performing an intelligent reduction of the set of features, especially using a small number of training examples, can dramatically improve predictive performance. I should experiment with using random forests, as they do, or some other feature reduction technique, both to improve performance, as well as to discover which features (especially larger-scale features) contribute to good performance.

Prediction-Performance Metrics: While the metrics I use to describe the performance of my system do capture exact matches to the binding interface, they do not give any indication of how spatially close false positives were to the true positives. It may be possible that many of the false positives are very close to the true binding interface, which would paint my

results in a better light as these false positives are, presumably, less of an issue than those that are far away. Unfortunately, there remains the issue of deciding how far to expand the allowable region before I begin to bias my results. This would be an interesting experiment to run.

7.2 Future Work

As much of this work is still in its infancy, there are many future directions in which it could go. Here are a few.

PyMOL Plugin: While the GRAPE web server has given my work more exposure to potential users, it does not integrate well into the tool-chains that they are already using. A big step in this direction would be the construction of a turn-key Pymol plugin for molecular surface abstraction. In creating this, we would simplify their adoption for many users, hopefully expanding their utility. At the same time, we would also leverage a robust molecular visualization infrastructure.

Abstraction of Large Proteins: A limitation of my surface abstraction method is that it requires the entire protein be loaded into memory, including the chemical structure, surface triangulation and electrostatic field. Because of this, some proteins may be too large to be abstracted. Unfortunately, it is precisely these proteins that would benefit from abstraction. In the future, I would like to address this limitation, either by adapting the algorithm to work out-of-core, or by performing abstraction on a reduced-detail version of the protein.

Molecular Dynamics: Though the information coming from protein crystallography is static and frozen in time, proteins themselves are incredibly dynamic and subject to both thermal vibration and conformational changes. Adapting abstraction to account for these dynamics would be a monumental task, but an important next step in molecular visualization.

The Molecular Descriptor Format: Further tuning and analysis of the molecular descriptor format could improve the performance of my classifier. In particular, now that a pipeline is in place, it would be interesting to see if any features are completely redundant, or if others

are so powerful that they dominate the learning process. This study would also give more insight into binding morphology, as it would allow us to discern the range of importance for various features in the binding of a ligand.

Performance Improvements: As described above, there are still many opportunities for improving the performance of my prediction algorithm, both in terms of speed and accuracy.

Ultimately, I hope that my work on the multi-scale visualization and analysis of proteins will lead to new insights in molecular biology, and will inspire others to develop new ways of looking at complicated objects such as these.

LIST OF REFERENCES

[1] J. N. Abelson, M. I. Simon, and R. F. Doolittle, *Computer Methods for Macromolecular Sequence Analysis, Volume 266 (Methods in Enzymology)*. Academic Press, 1996.

[2] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool.," *Journal of molecular biology*, vol. 215, pp. 403–10, Oct. 1990.

[3] J. D. Thompson, D. G. Higgins, and T. J. 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, pp. 4673–80, Nov. 1994.

[4] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The Protein Data Bank," *Nucleic acids research*, vol. 28, pp. 235–242, Jan. 2000.

[5] W. L. DeLano, "The PyMOL Molecular Graphics System," 2002.

[6] D. E. Koshland, "The Key-Lock Theory and the Induced Fit Theory," *Angewandte Chemie International Edition in English*, vol. 33, pp. 2375–2378, Jan. 1995.

[7] G. Cipriano, G. Wesenberg, T. Grim, G. N. Phillips, and M. Gleicher, "GRAPE: GRaphical Abstracted Protein Explorer," *Nucleic acids research*, pp. 1–7, May 2010.

[8] L. Wei and R. B. Altman, "Recognizing protein binding sites using statistical descriptions of their 3D environments," *Paci c Symposium on Biocomputing. Paci c Symposium on Biocomputing*, pp. 497–508, Jan. 1998.

[9] M. L. Connolly, "Molecular Surfaces: A Review," 1996.

[10] J. Tate, *Molecular Visualization*. Methods of Biochemical Analysis, Hoboken, NJ, USA: John Wiley & Sons, Inc., Feb. 2003.

[11] D. S. Goodsell, "Visual methods from atoms to cells," *Structure (London, England : 1993)*, vol. 13, pp. 347–354, Mar. 2005.

[12] S. I. O'Donoghue, D. S. Goodsell, A. S. Frangakis, F. Jossinet, R. A. Laskowski, M. Nilges, H. R. Saibil, A. Schafferhans, R. C. Wade, E. Westhof, and A. J. Olson, "Visualization of macromolecular structures," *Nature methods*, vol. 7, pp. S42—-55, Mar. 2010.

[13] R. A. Laskowski, "SURFNET: a program for visualizing molecular surfaces, cavities, and intermolecular interactions," *Journal of molecular graphics*, vol. 13, pp. 323–30, 307–8, Oct. 1995.

[14] F. Glaser, R. J. Morris, R. J. Najmanovich, R. A. Laskowski, and J. M. Thornton, "A method for localizing ligand binding pockets in protein structures," *Proteins*, vol. 62, pp. 479–488, Feb. 2006.

[15] E. F. Pettersen, T. D. Goddard, C. C. Huang, G. S. Couch, D. M. Greenblatt, E. C. Meng, and T. E. Ferrin, "UCSF Chimera–a visualization system for exploratory research and analysis.," *Journal of computational chemistry*, vol. 25, pp. 1605–1612, Oct. 2004.

[16] D. P. Luebke, "A developer's survey of polygonal simplification algorithms," *IEEE Computer Graphics and Applications*, vol. 21, no. 1, pp. 24–35, 2001.

[17] T. D. Goddard, C. C. Huang, and T. E. Ferrin, "Software extensions to UCSF chimera for interactive visualization of large molecular assemblies," *Structure*, vol. 13, pp. 473–482, Mar. 2005.

[18] M. F. Sanner, "A component-based software environment for visualizing large macromolecular assemblies.," *Structure (London, England : 1993)*, vol. 13, pp. 447–462, Mar. 2005.

[19] H.-L. Cheng and X. Shi, *Quality Mesh Generation for Molecular Skin Surfaces Using Restricted Union of Balls*. IEEE, 2005.

[20] Z. Yu, M. J. Holst, Y. Cheng, and J. A. McCammon, "Feature-preserving adaptive mesh generation for molecular shape modeling and simulation.," *Journal of molecular graphics & modelling*, vol. 26, pp. 1370–80, June 2008.

[21] A. Nicholls, R. Bharadwaj, and B. Honig, "GRASP: Graphical Representation and Analysis of Surface Properties," *37th Meeting of the Biophysical Society*, vol. 64, p. A166, 1993.

[22] M. S. Chapman, "Mapping the surface properties of macromolecules," *Protein Science*, vol. 2, pp. 459–469, Mar. 1993.

[23] C. H. Lee and A. Varshney, "Representing thermal vibrations and uncertainty in molecular surfaces," in *Proceedings of SPIE*, no. January, pp. 80–90, SPIE, 2002.

[24] J. Schmidt-Ehrenberg, D. Baum, and H.-C. Hege, "Visualizing dynamic molecular conformations," *IEEE Visualization, 2002. VIS 2002.*, pp. 235–242, 2002.

[25] G. Grigoryan and P. Rheingans, "Point-based probabilistic surfaces to show surface uncertainty," *IEEE transactions on visualization and computer graphics*, vol. 10, no. 5, pp. 564–573, 2004.

[26] D. S. Goodsell and A. J. Olson, "Molecular illustration in black and white," *Journal of Molecular Graphics*, vol. 10, pp. 235–240, Dec. 1992.

[27] T. A. Larsen, A. J. Olson, and D. S. Goodsell, "Morphology of protein-protein interfaces," *Structure (London, England : 1993)*, vol. 6, pp. 421–427, Apr. 1998.

[28] M. Tarini, P. Cignoni, and C. Montani, "Ambient occlusion and edge cueing to enhance real time molecular visualization," *IEEE transactions on visualization and computer graphics*, vol. 12, no. 5, pp. 1237–1244, 2006.

[29] T. Gatzke and C. Grimm, "Estimating Curvature on Triangular Meshes," *International Journal of Shape Modeling*, vol. 12, pp. 1–29, June 2006.

[30] S. Petitjean, "A survey of methods for recovering quadrics in triangle meshes," *ACM Computing Surveys*, vol. 34, pp. 211–262, June 2002.

[31] P. Sander and S. Zucker, "Inferring surface trace and differential structure from 3-D images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 9, pp. 833–854, 1990.

[32] J. Berkmann and T. Caelli, "Computation of surface geometry and segmentation using covariance techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 11, pp. 1114–1116, 1994.

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

[34] J. Goldfeather and V. Interrante, "A novel cubic-order algorithm for approximating principal direction vectors," *ACM Transactions on Graphics*, vol. 23, pp. 45–63, Jan. 2004.

[35] S. Rusinkiewicz, "Estimating curvatures and their derivatives on triangle meshes," *Proceedings. 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004.*, pp. 486–493, 2004.

[36] R. G. Coleman, M. A. Burr, D. L. Souvaine, and A. C. Cheng, "An intuitive approach to measuring protein surface curvature," *Proteins*, vol. 61, pp. 1068–1074, Dec. 2005.

[37] Q. Li and J. G. Griffiths, *Least squares ellipsoid speci c tting*. IEEE, 2004.

[38] E. M. Stokely and S. Y. Wu, "Surface parametrization and curvature measurement of arbitrary 3-D objects: five practical methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 8, pp. 833–840, 1992.

[39] B. S. Duncan and A. J. Olson, "Shape analysis of molecular surfaces.," *Biopolymers*, vol. 33, pp. 231–8, Feb. 1993.

[40] S. R. Buss and J. P. Fillmore, "Spherical averages and applications to spherical splines and interpolation," *ACM Transactions on Graphics*, vol. 20, pp. 95–126, Apr. 2001.

[41] Q. Zhang, M. Sanner, and A. J. Olson, "Shape complementarity of protein-protein complexes at multiple resolutions.," *Proteins*, vol. 75, pp. 453–67, May 2009.

[42] M. Körtgen, G. J. Park, M. Novotni, and R. Klein, "3D Shape Matching with 3D Shape Contexts," *In the 7th Central European Seminar on Computer Graphics*, Apr. 2003.

[43] T. Gatzke, C. Grimm, M. Garland, and S. Zelinka, *Curvature maps for local shape comparison*. IEEE Comput. Soc, 2005.

[44] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 433–449, May 1999.

[45] R. Gal and D. Cohen-Or, "Salient geometric features for partial shape matching and similarity," *ACM Transactions on Graphics*, vol. 25, pp. 130–150, Jan. 2006.

[46] C. Chua and R. Jarvis, "Point Signatures: A New Representation for 3D Object Recognition," *International Journal of Computer Vision*, 1997.

[47] B. B. Goldman and W. T. Wipke, "Quadratic Shape Descriptors. 1. Rapid Superposition of Dissimilar Molecules Using Geometrically Invariant Surface Descriptors," *Journal of Chemical Information and Modeling*, vol. 40, pp. 644–658, May 2000.

[48] S. Gumhold, "Maximum entropy light source placement," in *IEEE Visualization, 2002. VIS 2002.*, (Boston, Massachusetts), pp. 275–282, IEEE, 2002.

[49] C. H. Lee, X. Hao, and A. Varshney, "Geometry-dependent lighting," *IEEE transactions on visualization and computer graphics*, vol. 12, no. 2, pp. 197–207, 2006.

[50] C. Toler-Franklin, A. Finkelstein, and S. Rusinkiewicz, "Illustration of complex real-world objects using images with normals," in *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, (San Diego, California), pp. 111–119, ACM, 2007.

[51] R. Vergne, P. Barla, X. Granier, and C. Schlick, "Apparent relief: a shape descriptor for stylized shading," *Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*, p. 23, 2008.

[52] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Moller, "Curvature-based transfer functions for direct volume rendering: methods and applications," in *VIS 03: Proceedings of the 14th IEEE Visualization 2003 (VIS 03)*, pp. 513–520, IEEE, 2003.

[53] A. Girshick, V. Interrante, S. Haker, and T. Lemoine, "Line direction matters," *Proceedings of the rst international symposium on Non-photorealistic animation and rendering - NPAR 00*, pp. 43–52, 2000.

[54] G. Gorla, V. Interrante, and G. Sapiro, "Texture synthesis for 3D shape representation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, pp. 512–524, Oct. 2003.

[55] E. Praun, M. Webb, and A. Finkelstein, "Real-time hatching," *In Proceedings of SIGGRAPH 2001*, pp. 579–584, 2001.

[56] Y. Cai and F. Dong, "Surface Hatching for Medical Volume Data," in *International Conference on Computer Graphics, Imaging and Visualization (CGIV 05)*, pp. 232–238, IEEE, 2005.

[57] E. Kalogerakis, D. Nowrouzezahrai, P. Simari, J. Mccrae, A. Hertzmann, and K. Singh, "Data-driven curvature for real-time line drawing of dynamic scenes," *ACM Transactions on Graphics*, vol. 28, pp. 1–13, Jan. 2009.

[58] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, "Suggestive contours for conveying shape," *ACM Transactions on Graphics*, vol. 22, p. 848, July 2003.

[59] A. P. Mangan and R. Whitaker, "Partitioning 3D surface meshes using watershed segmentation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 308–321, 1999.

[60] A. Sheffer and J. C. Hart, "Seamster: inconspicuous low-distortion texture seam layout," in *IEEE Visualization, 2002. VIS 2002.*, (Boston, Massachusetts), pp. 291–298, IEEE, 2002.

[61] M. Mortara, G. Patané, M. Spagnuolo, B. Falcidieno, and J. Rossignac, "Blowing Bubbles for Multi-Scale Analysis and Decomposition of Triangle Meshes," *Algorithmica*, vol. 38, pp. 227–248, Oct. 2003.

[62] F. B. Sheinerman, R. Norel, and B. Honig, "Electrostatic aspects of protein-protein interactions," *Current opinion in structural biology*, vol. 10, pp. 153–159, Apr. 2000.

[63] M. Hendlich, F. Rippmann, and G. Barnickel, "LIGSITE: automatic and efficient detection of potential small molecule-binding sites in proteins," *Journal of molecular graphics & modelling*, vol. 15, pp. 359–363,389, Dec. 1997.

[64] B. Huang and M. Schroeder, "LIGSITEcsc: predicting ligand binding sites using the Connolly surface and degree of conservation," *BMC structural biology*, vol. 6, p. 19, Jan. 2006.

[65] F. Glaser, Y. Rosenberg, A. Kessel, T. Pupko, and N. Ben-Tal, "The ConSurf-HSSP database: the mapping of evolutionary conservation among homologs onto PDB structures," *Proteins*, vol. 58, pp. 610–7, Feb. 2005.

[66] M. Weisel, E. Proschak, and G. Schneider, "PocketPicker: analysis of ligand binding-sites with shape descriptors," *Chemistry Central journal*, vol. 1, p. 7, Jan. 2007.

[67] M. Weisel, E. Proschak, J. M. Kriegl, and G. Schneider, "Form follows function: shape analysis of protein cavities for receptor-based drug design," *Proteomics*, vol. 9, pp. 451–459, Jan. 2009.

[68] R. Wang, X. Fang, Y. Lu, C.-Y. Yang, and S. Wang, "The PDBbind database: methodologies and updates.," *Journal of medicinal chemistry*, vol. 48, pp. 4111–9, June 2005.

[69] D. G. Levitt and L. J. Banaszak, "POCKET: a computer graphics method for identifying and displaying protein cavities and their surrounding amino acids," *Journal of molecular graphics*, vol. 10, pp. 229–234, Dec. 1992.

[70] T. Kawabata and N. Go, "Detection of pockets on protein surfaces using small and large probe spheres to find putative ligand binding sites," *Proteins*, vol. 68, pp. 516–529, Aug. 2007.

[71] T. Kawabata, "Detection of multiscale pockets on protein surfaces using mathematical morphology," *Proteins*, vol. 78, pp. 1195–1211, Apr. 2010.

[72] G. P. Brady and P. F. Stouten, "Fast prediction and visualization of protein binding pockets with PASS," *Journal of computer-aided molecular design*, vol. 14, pp. 383–401, May 2000.

[73] D.-S. Kim, C.-H. Cho, Y. Cho, C. I. Won, and D. Kim, "Pocket recognition on a protein using Euclidean Voronoi diagram of atoms," *Computational Science and Its Applications-ICCSA 2005*, pp. 707–715, 2005.

[74] J. An, M. Totrov, and R. Abagyan, "Pocketome via comprehensive identification and classification of ligand binding envelopes.," *Molecular & cellular proteomics : MCP*, vol. 4, pp. 752–61, June 2005.

[75] P. J. Hajduk, J. R. Huth, and C. Tse, "Predicting protein druggability," *Drug discovery today*, vol. 10, pp. 1675–1682, Dec. 2005.

[76] I. D. Kuntz, J. M. Blaney, S. J. Oatley, R. Langridge, and T. E. Ferrin, "A geometric approach to macromolecule-ligand interactions.," *Journal of molecular biology*, vol. 161, pp. 269–88, Oct. 1982.

[77] R. Norel, S. L. Lin, H. J. Wolfson, and R. Nussinov, "Shape complementarity at protein-protein interfaces," July 1994.

[78] S. C. Bagley and R. B. Altman, "Characterizing the microenvironment surrounding protein sites," *Protein Science*, vol. 4, pp. 622–635, Apr. 1995.

[79] S. Shahbaz, L. F. Ten Eyck, and J. C. Mitchell, "Interfaces in Molecular Docking," *Molecular Simulation*, vol. 30, pp. 97–106, Feb. 2004.

[80] B. Ma, M. Shatsky, H. J. Wolfson, and R. Nussinov, "Multiple diverse ligands binding at a single protein site: a matter of pre-existing populations," *Protein science*, vol. 11, pp. 184–197, Feb. 2002.

[81] A. Kahraman, R. J. Morris, R. A. Laskowski, and J. M. Thornton, "Shape variation in protein binding pockets and their ligands," *Journal of molecular biology*, vol. 368, pp. 283–301, Apr. 2007.

[82] D. L. Mobley and K. A. Dill, "Binding of small-molecule ligands to proteins: "what you see" is not always "what you get"," *Structure*, vol. 17, pp. 489–498, Apr. 2009.

[83] D. Ghersi and R. Sanchez, "EasyMIFS and SiteHound: a toolkit for the identification of ligand-binding sites in protein structures," *Bioinformatics (Oxford, England)*, vol. 25, pp. 3185–3186, Dec. 2009.

[84] G. Cruciani, ed., *Molecular Interaction Fields*, vol. 27 of *Methods and Principles in Medicinal Chemistry*. Weinheim, FRG: Wiley-VCH Verlag GmbH & Co. KGaA, Oct. 2005.

[85] M. Hernandez, D. Ghersi, and R. Sanchez, "SITEHOUND-web: a server for ligand binding site identification in protein structures," *Nucleic Acids Research*, vol. 37, no. Web Server issue, p. W413, 2009.

[86] J. Hu and C. Yan, "A tool for calculating binding-site residues on proteins from PDB structures," *BMC structural biology*, vol. 9, p. 52, Jan. 2009.

[87] H. Nassif, H. Al-Ali, S. Khuri, and W. Keirouz, "Prediction of protein-glucose binding sites using support vector machines.," *Proteins*, vol. 77, pp. 121–32, Oct. 2009.

[88] B. Hoffmann, M. Zaslavskiy, J.-P. Vert, and V. Stoven, "A new protein binding pocket similarity measure based on comparison of clouds of atoms in 3D: application to ligand prediction," *BMC bioinformatics*, vol. 11, p. 99, Jan. 2010.

[89] R. Chikhi, L. Sael, and D. Kihara, "Real-time ligand binding pocket database search using local surface descriptors," *Proteins*, vol. 78, pp. 2007–28, July 2010.

[90] M. C. Lawrence and P. M. Colman, "Shape complementarity at protein/protein interfaces," *Journal of molecular biology*, vol. 234, pp. 946–50, Dec. 1993.

[91] C. M. Breneman, C. M. Sundling, N. Sukumar, L. Shen, W. P. Katt, and M. J. Embrechts, "New developments in PEST shape/property hybrid descriptors," *Journal of computer-aided molecular design*, vol. 17, no. 2-4, pp. 231–40, 2003.

[92] R. J. Morris, R. J. Najmanovich, A. Kahraman, and J. M. Thornton, "Real spherical harmonic expansion coefficients as 3D shape descriptors for protein binding pocket and ligand comparisons.," *Bioinformatics (Oxford, England)*, vol. 21, pp. 2347–2355, May 2005.

[93] A. N. Jain, T. G. Dietterich, R. H. Lathrop, D. Chapman, R. E. Critchlow, B. E. Bauer, T. A. Webster, and T. Lozano-Perez, "A shape-based machine learning tool for drug design," *Journal of computer-aided molecular design*, vol. 8, pp. 635–52, Dec. 1994.

[94] T. A. Binkowski, L. Adamian, and J. Liang, "Inferring Functional Relationships of Proteins from Local Sequence and Spatial Surface Patterns," *Journal of Molecular Biology*, vol. 332, pp. 505–526, Sept. 2003.

[95] C. M. Breneman, T. Thompson, M. Rhem, and M. Dung, "Electron density modeling of large systems using the transferable atom equivalent method," *Computers & Chemistry*, vol. 19, pp. 161–173, Sept. 1995.

[96] S. Oloff, S. Zhang, N. Sukumar, C. M. Breneman, and A. Tropsha, "Chemometric analysis of ligand receptor complementarity: identifying Complementary Ligands Based on Receptor Information (CoLiBRI)," *Journal of chemical information and modeling*, vol. 46, no. 2, pp. 844–851, 2006.

[97] J. C. Mitchell, R. Kerr, and L. F. Ten Eyck, "Rapid atomic density methods for molecular shape characterization," *Journal of molecular graphics & modelling*, vol. 19, pp. 325–30, 388–90, Jan. 2001.

[98] S. J. Darnell, D. Page, and J. C. Mitchell, "An automated decision-tree approach to predicting protein interaction hot spots," *Proteins*, vol. 68, pp. 813–823, Sept. 2007.

[99] C. Hofbauer, H. Lohninger, and A. Aszódi, "SURFCOMP: a novel graph-based approach to molecular surface comparison," *Journal of chemical information and computer sciences*, vol. 44, no. 3, pp. 837–847, 2004.

[100] C. a. Orengo, A. D. Michie, S. Jones, D. T. Jones, M. B. Swindells, and J. M. Thornton, "CATH–a hierarchic classification of protein domain structures," *Structure (London, England : 1993)*, vol. 5, pp. 1093–1108, Aug. 1997.

[101] T. A. Binkowski, P. Freeman, and J. Liang, "pvSOAR: detecting similar surface patterns of pocket and void surfaces of amino acid residues on proteins," *Nucleic acids research*, vol. 32, pp. W555—-8, July 2004.

[102] M. E. Bock, C. Garutti, and C. Guerra, "Discovery of similar regions on protein surfaces," *Journal of computational biology : a journal of computational molecular cell biology*, vol. 14, pp. 285–299, Apr. 2007.

[103] S. D. Mooney, M. H.-P. Liang, R. DeConde, and R. B. Altman, "Structural characterization of proteins using residue environments," *Proteins*, vol. 61, pp. 741–747, Dec. 2005.

[104] D. Baum and H.-C. Hege, *A Point-Matching Based Algorithm for 3D Surface Alignment of Drug-Sized Molecules*, vol. 4216 of *Lecture Notes in Computer Science*, pp. 183–193. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[105] Y. Fang, Y.-S. Liu, and K. Ramani, "Three dimensional shape comparison of flexible proteins using the local-diameter descriptor," *BMC structural biology*, vol. 9, p. 29, Jan. 2009.

[106] R. Gal, A. Shamir, and D. Cohen-Or, "Pose-oblivious shape signature.," *IEEE transactions on visualization and computer graphics*, vol. 13, no. 2, pp. 261–71, 2007.

[107] B. K. Lee and F. Richards, "The interpretation of protein structures: Estimation of static accessibility," *Journal of Molecular Biology*, vol. 55, pp. 379–400, Feb. 1971.

[108] M. L. Connolly, "The molecular surface package," *Journal of Molecular Graphics*, vol. 11, pp. 139–141, June 1993.

[109] A. P. Korn and R. M. Burnett, "Distribution and complementarity of hydropathy in multisubunit proteins.," *Proteins*, vol. 9, pp. 37–55, Jan. 1991.

[110] M. N. Davies, C. P. Toseland, D. S. Moss, and D. R. Flower, "Benchmarking pK(a) prediction.," *BMC biochemistry*, vol. 7, p. 18, Jan. 2006.

[111] T. J. Dolinsky, J. E. Nielsen, J. A. McCammon, and N. A. Baker, "PDB2PQR: an automated pipeline for the setup of Poisson-Boltzmann electrostatics calculations.," *Nucleic acids research*, vol. 32, pp. W665–7, July 2004.

[112] T. J. Dolinsky, P. Czodrowski, H. Li, J. E. Nielsen, J. H. Jensen, G. Klebe, and N. A. Baker, "PDB2PQR: expanding and upgrading automated preparation of biomolecular structures for molecular simulations," *Nucleic acids research*, vol. 35, pp. W522—-5, July 2007.

[113] N. A. Baker, D. Sept, S. Joseph, M. J. Holst, and J. A. McCammon, "Electrostatics of nanosystems: application to microtubules and the ribosome," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 98, pp. 10037–10041, Aug. 2001.

[114] J. Kyte and R. F. Doolittle, "A simple method for displaying the hydropathic character of a protein.," *Journal of molecular biology*, vol. 157, pp. 105–32, May 1982.

[115] G. Cipriano and M. Gleicher, "Molecular surface abstraction," *IEEE transactions on visualization and computer graphics*, vol. 13, no. 6, pp. 1608–1615, 2007.

[116] M. F. Sanner, A. J. Olson, and J.-C. Spehner, "Fast and robust computation of molecular surfaces," *Proceedings of the eleventh annual symposium on Computational geometry - SCG 95*, vol. 6, pp. 406–407, 1995.

[117] G. Taubin, "A signal processing approach to fair surface design," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques - SIGGRAPH 95*, vol. pp, (New York, New York, USA), pp. 351–358, ACM Press, 1995.

[118] G. Taubin, "Geometric signal processing on polygonal meshes," *Eurographics State of the Art Reports*, 2000.

[119] K. Fujiwara, "Eigenvalues of laplacians on a closed riemannian manifold and its nets," *Proceedings of the American Mathematical Society*, vol. 123, no. 8, pp. 2585–2594, 1995.

[120] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth International Conference on Computer Vision*, pp. 839–846, Narosa Publishing House, 1998.

[121] R. Schmidt, C. Grimm, and B. Wyvill, "Interactive decal compositing with discrete exponential maps," *ACM Transactions on Graphics*, vol. 25, p. 605, July 2006.

[122] R. Gonzalez and R. Woods, *Digital Image Processing*. Prentice-Hall, 2002.

[123] J. W. M. Nissink, C. Murray, M. Hartshorn, M. L. Verdonk, J. C. Cole, and R. Taylor, "A new test set for validating predictions of protein-ligand interaction," *Proteins*, vol. 49, pp. 457–471, Dec. 2002.

[124] H. Landis, "Production ready global illumination." In Siggraph Course Notes, 2002.

[125] D. DeCarlo, A. Finkelstein, and S. Rusinkiewicz, "Interactive rendering of suggestive contours with temporal coherence," *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering - NPAR 04*, p. 15, 2004.

[126] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," *Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH 97*, pp. 209–216, 1997.

[127] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of molecular biology*, vol. 48, pp. 443–53, Mar. 1970.

[128] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *Journal of the Optical Society of America A*, vol. 5, p. 1127, July 1988.

[129] "Jmol: an open-source Java viewer for chemical structures in 3D."

[130] G. Cipriano, G. N. Phillips, and M. Gleicher, "Multi-scale surface descriptors," *IEEE transactions on visualization and computer graphics*, vol. 15, no. 6, pp. 1201–1208, 2009.

[131] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, and H. Hoppe, "Fast exact and approximate geodesics on meshes," *ACM Transactions on Graphics*, vol. 24, p. 553, July 2005.

[132] V. Pratt, "Direct least-squares fitting of algebraic surfaces," *ACM SIGGRAPH Computer Graphics*, vol. 21, pp. 145–152, Aug. 1987.

[133] P. Huber, *Robust Statistics*. Wiley-Interscience, Feb. 1981.

[134] G. M. Morris, D. S. Goodsell, R. Huey, and A. J. Olson, "Distributed automated docking of flexible ligands to proteins: parallel applications of AutoDock 2.4.," *Journal of computer-aided molecular design*, vol. 10, pp. 293–304, Aug. 1996.

[135] C. Hetényi and D. van Der Spoel, "Efficient docking of peptides to proteins without prior knowledge of the binding site.," *Protein science : a publication of the Protein Society*, vol. 11, pp. 1729–37, July 2002.

[136] A. Ben-Hur and W. S. Noble, "Choosing negative examples for the prediction of protein-protein interactions.," *BMC bioinformatics*, vol. 7 Suppl 1, p. S2, Jan. 2006.

[137] J. Pearl, "Bayesian Networks: A Model of Self-Activated Memory for Evidential Reasoning," in *Proceedings of the 7th Conference of the Cognitive Science Society*, (University of California, Irvine, CA), pp. 329–324, 1985.

[138] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software," *ACM SIGKDD Explorations Newsletter*, vol. 11, p. 10, Nov. 2009.

[139] C.-C. C. Lin and Chih-Jen, "LIBSVM: a library for support vector machines," 2001.

[140] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fth annual workshop on Computational learning theory - COLT 92*, (New York, New York, USA), pp. 144–152, ACM Press, 1992.

[141] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," *Journal of the ACM*, vol. 45, pp. 891–923, Nov. 1998.

APPENDIX
Molecular Surface Feature Vector De nition

Feature #	Name	Description
1	% Visibility	Percentage of outside world visible from point
2	Non-Polar Backbone	Distance to the nearest Non-Polar Backbone Atom
3	Arom. Sidechain	Distance to the nearest Aromatic Sidechain Atom
4	Aliph. Sidechain	Distance to the nearest Aliphatic Sidechain Atom
5	N Backbone	Distance to the nearest Nitrogen Backbone Atom
6	O Backbone	Distance to the nearest Oxygen Backbone Atom
7	S Backbone	Distance to the nearest Sulpher Sidechain Atom
8	Amide N Sidechain	Distance to the nearest Amide Nitrogen Sidechain Atom
9	Amide O Sidechain	Distance to the nearest Amide Oxygen Sidechain Atom
10	Trp Sidechain	Distance to the nearest Trypophan Sidechain Atom
11	Hydroxyl Sidechain	Distance to the nearest Hydroxyl Sidechain Atom
12	Charged O Sidechain	Distance to the nearest Charged Oxygen Sidechain Atom
13	Charged N Sidechain	Distance to the nearest Charged Nitrogen Sidechain Atom
		Continued on next page

A.1 (Continued from previous page)

Feature #	Name	Description
14	Anisotropy (1.6 Å)	Patch anisotropy, with radius: 1.6 Å
15	Anisotropy (3.2 Å)	Patch anisotropy, with radius: 3.2 Å
16	Anisotropy (4.8 Å)	Patch anisotropy, with radius: 4.8 Å
17	Anisotropy (6.4 Å)	Patch anisotropy, with radius: 6.4 Å
18	Anisotropy (8 Å)	Patch anisotropy, with radius: 8 Å
19	Curvature (1.6 Å)	Patch curvature, with radius: 1.6 Å
20	Curvature (3.2 Å)	Patch curvature, with radius: 3.2 Å
21	Curvature (4.8 Å)	Patch curvature, with radius: 4.8 Å
22	Curvature (6.4 Å)	Patch curvature, with radius: 6.4 Å
23	Curvature (8 Å)	Patch curvature, with radius: 8 Å
24	Curvature Var. (1.6 Å)	Variance of curvature within patch of radius: 1.6 Å
25	Curvature Var. (3.2 Å)	Variance of curvature within patch of radius: 3.2 Å
26	Curvature Var. (4.8 Å)	Variance of curvature within patch of radius: 4.8 Å
27	Curvature Var. (6.4 Å)	Variance of curvature within patch of radius: 6.4 Å
28	Curvature Var. (8 Å)	Variance of curvature within patch of radius: 8 Å
29	Hydropathy (1.6 Å)	Weighted avg. hydropathy over patch of radius: 1.6 Å
30	Hydropathy (3.2 Å)	Weighted avg. hydropathy over patch of radius: 3.2 Å
31	Hydropathy (4.8 Å)	Weighted avg. hydropathy over patch of radius: 4.8 Å
32	Hydropathy (6.4 Å)	Weighted avg. hydropathy over patch of radius: 6.4 Å

Continued on next page

A.1 (Continued from previous page)

Feature #	Name	Description
33	Hydropathy (8 Å)	Weighted avg. hydropathy over patch of radius: 8 Å
34	Charge (1.6 Å)	Weighted avg. charge over patch of radius: 1.6 Å
35	Charge (3.2 Å)	Weighted avg. charge over patch of radius: 3.2 Å
36	Charge (4.8 Å)	Weighted avg. charge over patch of radius: 4.8 Å
37	Charge (6.4 Å)	Weighted avg. charge over patch of radius: 6.4 Å
38	Charge (8 Å)	Weighted avg. charge over patch of radius: 8 Å
39	Hyd. Bond Doner	Distance to nearest potential external hydrogen bond doner
40	Hyd. Bond Acceptor	Distance to nearest potential external hydrogen bond acceptor

Table A.1 A list of each feature contained within our surface descriptor. Note that features 14 - 38 are weighted according to distance (from center vertex) and point area (i.e. the sum of the areas of all adjacent triangles).