ALGORITHMS EXPLOITING THE CHAIN STRUCTURE OF PROTEINS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Itay Lotan
August 2004

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Jean-Claude Latombe
(Principal Adviser)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Leonidas J. Guibas

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Michael Levitt

Approved for the University Committee on Graduate Studies.

Preface

Proteins are large molecules (from a few hundred to several thousand atoms) that affect almost all properties of living organisms. Gene expression, catalysis, storage and transport, transmission of signals, defense against intruders and many other vital functions are all performed by proteins. Since their functions are largely dependent on their spatial structure, the study of the 3-D shapes of proteins is central to the field of structural biology. Various computational methods have been developed to enhance our understanding of proteins, from molecular simulation techniques, to structure determination tools, to comparison and classification methods. Their ubiquitous use in research today, along with the rapidly growing number of proteins of known sequence and structure, create a need for efficient algorithms that can handle the complexity of proteins. This thesis presents three new algorithms that address fundamental problems in computational structural biology. The common thread of these algorithms is their use of concepts and techniques from robotics and computational geometry to exploit the long chain kinematics of proteins (a protein is a sequence of amino-acids that form a long flexible backbone with short protruding side-chains).

The first algorithm is an automatic method for completing partial models of protein structures resolved using X-ray crystallography. It was developed in collaboration with the Joint Center for Structural Genomics at SSRL. A protein fragment of known sequence needs to be optimally fitted into an electron density map between two anchor points. The generation of the missing fragment is approached as an inverse kinematics problem by treating the protein backbone as a redundant serial linkage. First, a large set of candidates is sampled that meet the closure constraint and then the most promising candidates are refined to improve the fit. The method successfully generates fragments for gaps at high as well as medium-low resolutions. Preliminary results indicate its usefulness in generating alternative backbone conformations in ambiguous regions in the density map.

The second algorithm deals with the computation of structural similarity. It speeds up this computation at the expense of introducing a small error in the similarity measure. It exploits the fact that the Cα representation of a protein conformation contains redundant information, due to the chain topology and limited compactness (caused by van der Waals forces) of proteins. This redundancy is reduced by approximating sub-chains by their centers of mass. This method can be used in applications where small errors can be tolerated, or as a fast filter when exact measures are required. It has been successfully tested for comparing large collections of conformations of the same protein, as well as for comparing native structures of different proteins using the program STRUCTAL.

The third algorithm is a method for speeding up Monte Carlo simulation of proteins without changing its outcome. It exploits the chain kinematics of the protein backbone and the fact that only a few degrees of freedom are changed at each simulation step to quickly detect pairs of interacting atoms (i.e., atoms that are within a given cutoff distance of each other). It is based on a novel data structure, called the ChainTree, that captures both the kinematics and the shape of a protein at successive levels of detail. The ChainTree also makes it possible to identify partial energy sums left unchanged at each simulation step.

Acknowledgements

First and Foremost, I would like to thank my adviser, Prof. Jean-Claude Latombe for his continuing support and encouragement throughout the four years we have worked together. He allowed me to find my own path, for which I am very grateful, always listening to my ideas and offering sound advice. I have learned much from him on how to be a scientist and how to conduct research.

I would like to thank Prof. Leo Guibas for many helpful discussions, for being supportive and attentive, and for reading this thesis.

I would like to thank Prof. Michael Levitt for his kindness and patience and for agreeing to read this thesis. His feedback and comments were invaluable in establishing the biological relevance of my work.

I owe a special thanks to Prof. Vijay Pande, who taught me much of the biology I know and let me into his group. His help was essential in getting me started and working in a field I knew very little about.

I would like to thank the members of the Pande and Levitt group during my tenure at Stanford for their kindness and patience in answering my numerous questions.

I would like to thank my colleagues during these five — sometimes arduous — years, Daniel Russakoff and Rachel Kolodny, for being my support group, always attentive and encouraging.

I would like to thank Dr. Fabian Schwarzer for his collaboration in two of the projects that make up this thesis. He helped was crucial to my getting started and making progress. I enjoyed working with him very much.

I would like to thank Dr. Henry van den Bedem for his collaboration in the third project I worked on. Our work together was very enjoyable and interesting for me.

I would like to thank Prof. Dan Halperin for his collaboration on the first project I worked on. His friendliness and willingness to listen and help were very important to my

success.

I would like to thank my parents, Ilana and Amos Lotan, for their love and support throughout this long journey, and all the years preceding it. I am very grateful to them for helping me to accomplish this goal.

I would like to thank my grandmother, Maria Buchweiz, for supporting me in this undertaking from the very beginning.

I would like to thank my parents in-law Rivka and Amichai Kuperfish for their support and kindness.

Finally, I would like to thank my wife, Sharon Kuperfish, who stood by me, continuously encouraging, supporting and loving. She is my strength and joy, and I love her dearly.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A protein is a linear copolymer made up of a concatenation of amino acids. There are 20 naturally occurring amino acids that make up all known proteins. The central dogma of molecular biology states that the genes found in the DNA are transcribed to mRNA and then translated by the ribosome into a sequence of amino acids. Each sequence of three bases codes for one amino acid, thus the translation from mRNA to protein is unique. Proteins vary by length and sequence. Once synthesized, a protein molecule in physiological conditions (aqueous solution, 37°C, pH 7, atmospheric pressure) *folds* to a unique, compact and relatively stable 3-D structure, which determines the specific biological function of the protein. The sequence of the protein is believed to completely determine its folded 3-D structure, which arguably corresponds to the free energy minimum of the molecule.

All 20 amino acids have similar chemical structure. Each has a bonded sequence of three atoms, a nitrogen and two carbons, that concatenates with the same atom-triplets in other amino acids to form the *backbone* of the protein. Each also has a side-chain stemming out of the middle carbon (the Cα) that has between 0 and 10 heavy atoms. The amino acids are linked together by *peptide* bonds that form between the C$'$ atom of one amino acid and the N atom of the following amino acid in the sequence. Due to resonance, this bond has a partial double-bonded character and the six atoms surrounding it have a strong tendency to be coplanar (see Figure 1.2(a)). When part of a protein, the amino acids are referred to as *residues*. Some examples of amino acid residues can be seen in Figure 1.1.

The rest of this chapter is structured as follows. First, in Section 1.1, the representations of protein structure used by the algorithms in this thesis are introduced. Then, in Section 1.2, a number of key computational problems in structural biology are surveyed.

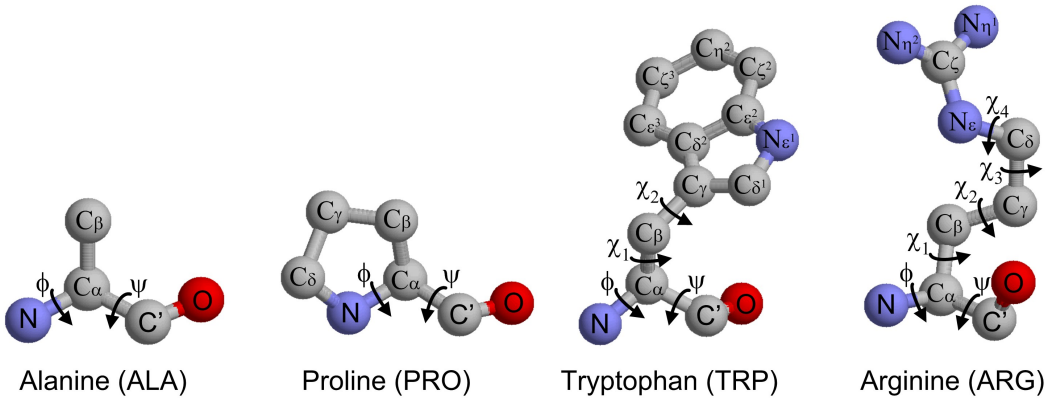


Figure 1.1: Some examples of amino acid residues (hydrogen atoms not shown).

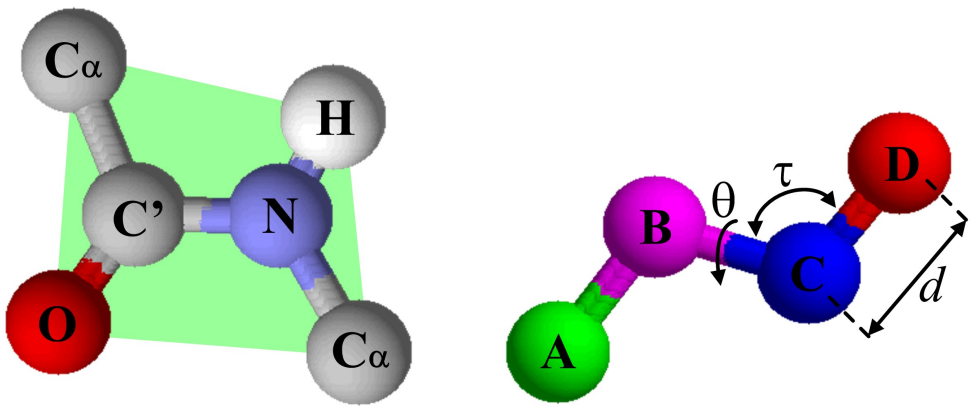
Finally, in Section 1.3, I explain how the chain structure, implicit in the protein representations we use, is exploited in the algorithms presented in this thesis. The main terms and abbreviations used in this thesis can be found in the Glossary.

1.1 Representations of protein structure

When scientists study the properties of proteins *in silico* they use a wide range of representations at various levels of detail. The most detailed is the *all-atom* model, where all atoms of the protein are explicitly represented. A small simplification is the *united atom* model, where non-polar hydrogen atoms are incorporated into the heavy atoms, to which they are bonded [207]. The side-chains are sometimes approximated as single spheres with varying radii and chemical properties in what is called the *lollipop model* [12, 123]. An even coarser representation models each residue as a single sphere in what is sometimes called a *bead model* [23]. Finally vectors have been used to represent entire secondary structure elements [180]. Below we discuss in more detail the protein models relevant to this thesis.

1.1.1 Cartesian coordinates representation

The structure of a protein is completely described by the 3-D positions of all of its atoms. In this representation a protein molecule has three times the number of its atoms' degrees of freedom (DOFs), three cartesian coordinates for each atom specifying its position in



(a) (b)

Figure 1.2: (a) The coplanarity of the atoms around the peptide bond and (b) the internal coordinates parameters.

space. The protein is regarded as a collection of particles (atoms) without explicit topology. Namely, the underlying covalent structure is not specified. Thus when this representation is used, e.g. in molecular dynamics simulation, the covalent bonds need to be explicitly maintained by stiff springs that keep bond lengths close to their ideal values. This representation makes physical sense, since bonds can stretch and even break at high temperatures. Moreover, interactions between non-bonded atoms such as van der Waals and electrostatic interactions, depend on the distances between pairs of atoms, which are easily computed using this representation. However, since proteins are predominantly studied in physiological conditions, where bonds are extremely unlikely to break, incorporating the covalent structure of the protein into the model can be beneficial.

1.1.2 Internal coordinates representation

This representation describes the position of each atom in terms of three other atoms that are one, two and three covalent bonds away, and three parameters: a *bond length*, a *bond angle* and a *dihedral angle*. See Figure 1.2(b) for an illustration. The position of atom D is uniquely specified by its distance from atom C (the length of the B-C bond), the angle that is formed between bonds BC and CD (the B-C-D bond angle) and by the dihedral angle between the plane defined by atoms A, B and C and the plane defined by atoms B, C and D. This representation uses only $3n - 6$ parameters, since the position of the first

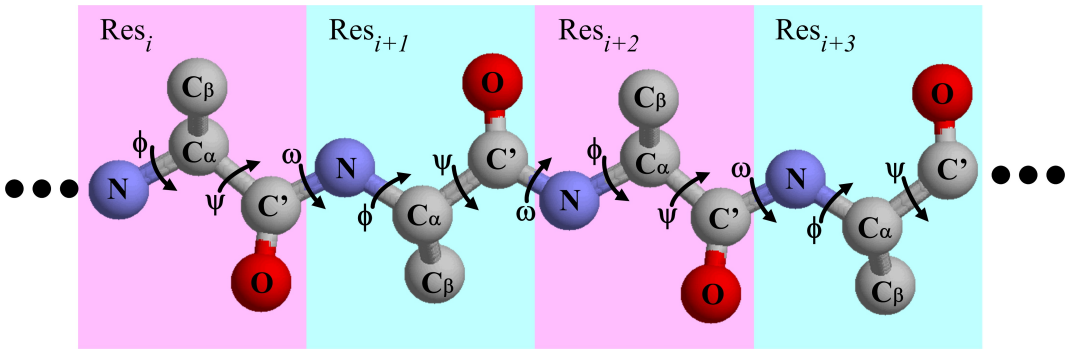


Figure 1.3: The DOFs of the protein backbone when bond lengths and angles are kept fixed. A poly-alanine chain with all hydrogens hidden

atom is arbitrary, only a bond length is specified for the second atom, and no dihedral angle is specified for the third atom. The 6 DOFs that are saved describe the global position and orientation of the protein. Imposing constraints on bond lengths is straightforward using this model, and all the topological information is included in the representation (bonds that close loops need to be added). However, computing the relative positions of non-bonded atom pairs requires additional non-trivial calculations.

1.1.3 Torsion angles representation

Analysis of experimentally determined protein structures deposited in the Protein Data Bank (PDB) [15, 16] has shown that bond lengths and bond angles exhibit small variance [73]. The flexibility of the protein molecule is thus due almost completely to rotation around covalent bonds. The backbone of the protein has two bonds and thus two DOFs per residue (also denoted *torsion* angles) with an additional DOF for every peptide bond between two neighboring residues. These are the ϕ angle which denotes rotations around the N-Cα bond, the ψ angle rotation around the Cα-C$'$ bond and the ω angle rotation around the peptide bond (See Figure 1.3 for illustration). Since the peptide bond has a partial double-bond character, rotation around it is restricted and to a good approximation its value can be fixed at 180° (the trans conformation is overwhelmingly favored due to the energetics of the non-bonded interactions). Thus only two DOFs per residue are needed to closely approximate backbone conformations. In this representation the side-chains have between 0 (e.g., Glycine and Alanine) and 4 (e.g., Arginine) torsion DOFs, which are denoted $\chi_1, \ldots, \chi_4$ (See some examples in Figure 1.1).

Using this representation, the conformation of a protein is described by at most seven parameters per residue: the triplet (ϕ, ψ, ω) for the backbone segment of the residue and a subset of the 4-tuple $(\chi_1, \ldots, \chi 4)$ for the residue's side-chain.

1.1.4 Cα representation and common metrics in structure space

When comparing two protein structures, one is most interested in determining the similarity between the backbone geometries. As mentioned above, the backbone is a concatenation of atom triplets (N, Cα, C$'$), one for each residue in the protein. Due to the planarity of the peptide bond and the small variance of bond lengths and angles, the backbone geometry is completely determined by the positions of the Cα atoms. Therefore the protein structure can be represented as a sequence of 3-D points, specifying the centers of all $C\alpha$ atoms. Connecting these points in order yields a chain, which closely follows the protein backbone.

Given this representation, the similarity between two structures is computed by summing the deviations between corresponding Cα positions. In order to make the similarity independent of the position and orientation of each structure, the two compared structures can be optimally aligned using a rigid-body transform [107]. Two metrics are commonly used for the comparison. They are described below.

1.1.4.1 Coordinate root mean square deviation

Given two sequences of points in 3-space describing the Cα positions of two protein structures $P = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$ and $Q = (\mathbf{q}_1, \ldots, \mathbf{q}_n)$, the root mean square deviation (RMS) between the coordinates of the corresponding Cα ($cRMS$) is defined as

$$cRMS(P,Q) = min_T \sqrt{\frac{1}{n} \sum_{i=1}^{n} \parallel \mathbf{p}_i - T\mathbf{q}_i \parallel^2} \tag{1.1}$$

where $\| \cdot \|$ is the Euclidean L_2-norm and T is a rigid body transform (rotation and translation). A closed form solution for T yields the optimal transform [87, 97].

1.1.4.2 Distance root mean square deviation

Another common RMS shape similarity measure is based on comparing intra-molecular distance matrices, i.e., the matrix of distances between all Cα atoms in each structure. For

a point set P, this matrix is defined as

$$\left(d_{ij}^{P}\right) = \|\mathbf{p}_i - \mathbf{p}_j\| \,. \tag{1.2}$$

The distance matrix RMS deviation ($dRMS$) of P and Q is then defined as

$$dRMS(P,Q) = \sqrt{\frac{2}{n(n-1)} \sum_{i=2}^{n} \sum_{j=1}^{i-1} (d_{ij}^{P} - d_{ij}^{Q})^2}. \tag{1.3}$$

By using internal distance matrices the need to optimally align the two conformations is removed, however the computation time becomes quadratic in the length of the correspondence between the two proteins.

1.2 Key computational problems in structural biology

1.2.1 Structure determination and prediction

There are a number of experimental methods to determine protein structures, such as X-ray crystallography and NMR spectroscopy. Considerable attention has also been given to the development of computational methods to predict the structure of a protein from its amino acid sequence.

1.2.1.1 Structure determination

X-ray crystallography This is currently the main method of protein structure determination [48]. An X-ray beam is diffracted by a crystallized protein onto a detector. The diffraction pattern is then Fourier-transformed into an electron density map. The interpretation of the density map yields the 3-D structure of the protein. The main problem in this process, besides difficulties in the experimental setup, is the recovery of phase information. The diffraction pattern that is obtained contains only the intensities of the rays that hit the detector. The phase of these rays is lost and can be recovered only through an iterative refinement procedure, coupling the transformation and interpretation stages of the determination process. The automation of the entire process, going from diffraction pattern to a final, refined 3-D structure is still a work in progress.

NMR spectroscopy Nuclear magnetic resonance (NMR) spectroscopy is another experimental method for structure determination [210]. It can be used to obtain both structural and dynamic information about proteins in solution. The protein is placed inside a strong magnetic field and irradiated with radio-frequency pulses. The atom nuclei emit radiation whose spectrum depends on their chemical neighborhood. Nuclear Overhauser effects (NOEs) are used to measure distances between neighboring hydrogen atoms. These distances are used as restraints, together with constraints such as the primary structure and reference protein geometry, to calculate the 3-D structure of the protein. The result is an ensemble of solution structures that are compatible with the NOE data.

1.2.1.2 Structure prediction

Since the native structure of proteins is completely determined by their amino acid sequence, it should be possible to compute the 3-D structure from the sequence alone (taking into account the laws of physics and chemistry as well as known protein structures). This problem has 3 levels of difficulty corresponding to CASP[1] categories [143].

New fold prediction This is the most difficult case. The target sequence has no similarity to any protein of known structure and the fold is expected to be novel. Knowledge-based methods are extensively used in this case to identify structural fragments from the PDB that can be used to build the new structure. Secondary structure and native contacts prediction tools are often used. Sophisticated search tools such as Monte Carlo, molecular dynamics and genetic algorithms are used as well. Some of the more successful methods are ROSETTA [18] and TOUCHSTONE [181].

Fold recognition The fold of the target sequence is expected to be similar to a known protein structure even though no obvious sequence similarity exists. Here, sophisticated sequence comparison techniques are used and known structural elements are searched for compatibility with parts of the target sequence (an approach called *threading*). Some of the more successful methods are described in [36, 67, 160].

Homology modeling When the target sequence is similar to proteins whose structure is already known an approach called homology (comparative) modeling can be used [138].

[1] Critical Assessment of Techniques for Protein Structure Prediction: `http://predictioncenter.llnl.gov/`

It exploits the fact that, in general, sequence similarity entails structural similarity. The prediction process then consists of finding the expected fold of the target sequence based on sequence similarity, aligning the target sequence to a template sequence and using the alignment to build a model using large pieces of the known related structures. These models can be as good as medium resolution crystallographic structures. As the number of structures in the PDB increases so will the usefulness of this approach. Some of the more successful methods are [166, 197].

1.2.2 Structural similarity and classification

Once the structures of a large number of proteins is known, the next step is to compare and classify them. Structurally similar proteins are expected to have similar functions, and structural differences may hint at differential functionality.

1.2.2.1 Computing structural similarity

In order to compute the structural similarity between two protein structures, it is necessary to decide which parts of both structures correspond and should be compared. However, in order to decide correspondence, we need to know which parts are similar. Thus it is necessary to solve both aspects of the problem simultaneously. For a given superposition of the two structures the best correspondence can be computed efficiently and for a given correspondence the optimal superposition can be computed in closed form. Existing algorithms either use heuristic search methods to find a good correspondence and then deduce the superposition (e.g., DALI [85] and CE [173]), or iterate between computing correspondences and superpositions until convergence (e.g., STRUCTAL [65, 188]). A survey of existing methods is given in [107] and some of the more popular methods are evaluated in [150, 177].

1.2.2.2 Classifying protein structures

Based on structural similarity and perhaps other factors such as sequence similarity, evolutionary relation or common function the proteins can be hierarchically classified. Current classifications span the range from completely manual (e.g. SCOP [146]), through semi-automatic (e.g. CATH [152]) to completely automatic (e.g. FSSP [86]). A comparison of some of the more popular classifications can be found here [77].

1.2.3 Conformational sampling and dynamic simulation

A powerful tool in the study of proteins in silico is computer simulation. The two most popular methods are Monte Carlo simulation and molecular dynamics simulation. While the latter generates physically meaningful trajectories, that admit kinetic interpretation, the former is more efficient at sampling the conformation space.

1.2.3.1 Monte Carlo simulation

This family of simulations is most often used to compute ensemble properties for a given protein or to find low energy conformations, and in particular its native structure. The general Monte Carlo framework consists of a random walk in conformation space. At each step, a new conformation is proposed through some changes to the current conformation, and it is accepted based on probability that depends on the change in energy. In order for the sampling to converge to the Boltzmann distribution, the acceptance probability must obey *detailed balance* (also called *microscopic reversibility*):

$$p_i \Pi_{ij} = p_j \Pi_{ji}, \tag{1.4}$$

where p_i is the probability of being in state i and Π_{ij} is the probability of transition from state i to state j.

Many variants and extensions of the classical Metropolis Monte Carlo [140] have been developed. Some are mentioned in Section 5.6. See [142, 167] for more examples. These methods attempt to improve the sampling efficiency of the method by dealing better with the local minima problem and by using more sophisticated moves. Once the sampling method is fixed, it is also crucial to compute the simulation steps efficiently, since even the smallest proteins require very long simulations to adequately sample the conformation space.

1.2.3.2 Molecular dynamics simulation

This method allows for the detailed and physically accurate simulation of the dynamics of a single protein molecule in space and time. It affords access to deformations of the molecule from small time-scale events such as hydrogen bond formation to larger time-scale phenomena such as folding. It can be used to explore ensemble properties of the studied protein as well as its folding kinetics. It consists of simulating the motion of the atoms of the

protein under the influence of a specified force-field, by following the atoms in time according to the laws of Newtonian mechanics. The equations of motion are integrated using very small time steps (on the order of a femtosecond) to accurately simulate bond vibrations. As a result, the folding of most proteins, which occur on the 1 microsecond to 10 second time-scale, is still out of the reach of even the fastest computers. It is, therefore, crucial to speed up the simulation. One approach is to use clever integrators that can employ longer time-steps [167]. Another approach simulates the dynamics in torsion angle space, instead of the cartesian space, which enables the use of larger time-steps [76, 186]. The efficient calculation of the energy and forces at each time-step is also instrumental. Detailed description of different molecular dynamics simulation methods can be found in [63, 119, 167].

1.2.4 Molecular docking

Given the 3-D structures of two molecules, we would like to predict their bound association. In its most general form, the binding site and the bound conformation of the two molecules are not known in advance. The simpler problems — assuming known binding sites and known bound structures — are still challenging. A comprehensive survey can be found in [80, 208].

1.2.4.1 Protein – protein interactions

Interactions between proteins are computed by docking one molecule onto the other [182]. Most docking methods have two stages. First, the space of orientations is searched for plausible docked configurations. This is done, for example, using fast Fourier transforms, geometric hashing or genetic algorithms. Second, a scoring function picks out the correct orientations. Here one uses measures of shape complementarity, which requires a geometric description of the molecular surface of both proteins, as well as measures of electrostatic and desolvation effects. While the conformations of some pairs of proteins change very little upon docking, the conformations of other pairs may change considerably. Therefore it may be important to introduce flexibility when searching for the bound complex, in the side-chains as well as the backbone. Some of the more popular methods are BiGGER [153], and DOT [133].

1.2.4.2 Small ligand docking

In this context the second molecule, called the *ligand*, is much smaller then the first, which is called the *receptor*. We want to compute the binding conformation and affinity of the ligand to the receptor. Docking algorithms need to take the ligand flexibility into account to find its tightest-binding conformation. Including a limited measure of receptor flexibility is desirable as well but is often very computationally expensive. Most often we would like to scan a large database of ligands (potential drugs) and pick those that are most likely to bind tightly to the target enzyme. The ligands are scored by their geometric and chemical fit to the binding site. This is known as *virtual screening* [175]. Some of the more popular docking algorithms are DOCK [53], FlexX/FlexE [33, 165] and GOLD [93].

1.3 Exploiting the chain-structure of proteins

The problems surveyed above are computationally intensive, often to the point that the biologist cannot afford to wait for the calculation to end. This is due to a number of reasons:

- The inherent complexity of the problem. The structure prediction problem, for example, is known to be NP-hard for random sequences [148]. A recently developed view of protein folding claims the biological sequence actually directs the chain to fold along energetically favorable pathways that greatly reduce the combinatorial search [47]. However, computing these pathways may still prove extremely difficult.

- The number of times the problem needs to be solved (e.g. computing similarity between each pair in a database of thousands of proteins [86]).

- The lengths of simulations. Monte Carlo and molecular dynamics simulations are often run for millions of steps, simulating only a few nanoseconds of physical time. In order to get at slower events such as protein folding, these simulations would need to be run even longer [119]).

This issue can be addressed along two complementary approaches:

- The development of simplified models of protein molecules, for which the computations are faster, yet without significant loss of accuracy or biological relevance.

- The development of algorithms that take advantage of specific properties of a given model in order to reduce the computation time, without losing any accuracy.

While the first approach has been explored exclusively by biophysicist and biochemists because of the deep understanding of physics and chemistry that is required, computer scientists have recently explored to some success the applicability of tools and techniques from fields such as robotics and geometry for the second approach. These two approaches should eventually be combined so that simplified protein models are developed together with the algorithms that take the most advantage of their specific properties.

An example of the first approach is the advent of the so-called *minimalist* models [23] for the study of folding kinetics. The simplification of both the chemistry and the physics of the model results in molecular dynamics simulations that consistently reach the folded state (e.g., [171, 172, 183]). This is due to the shorter time to compute each simulation step as well as the longer physical time that can be assigned to it. The resulting trajectories can thus be used to speculate about the kinetics of the studied protein. Recall that even the all-atom models and Newtonian mechanics-based molecular dynamics simulations are in effect simplifications of the much more elaborate quantum mechanics models.

This thesis presents work along the second approach — the development of novel algorithms for existing protein representations using model-specific properties. The algorithms that are presented take advantage of specific properties of proteins to improve running times and offer novel solutions for a diverse set of problems. The properties that will be exploited are all derived from the inherent chain-structure of the protein backbone [42]. The algorithms are largely based on techniques from robotics and computational geometry that were designed for computations involving physical chains and linkages.

1.3.1 The protein backbone as a kinematic chain

The torsion angles model described in Subsection 1.1.3 makes explicit the protein's chain-like structure, built as a concatenation of amino acids. The three parameters used by this model — the (ϕ, ψ, ω) triplet — are the predominant sources of backbone conformational variability in proteins. To simplify matters without compromising much in way of accuracy, it will be henceforth assumed that the ω angles are kept fixed in their trans position ($\omega = 180°$). The rotatable bonds, around which the (ϕ, ψ) angles are defined, cut the protein backbone into rigid groups of atoms. There are two types of rigid groups. The first, referred

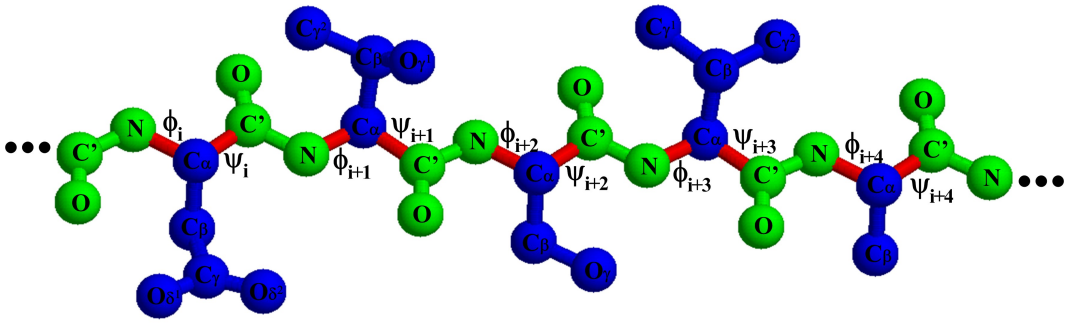


Figure 1.4: The protein backbone represented as a kinematic chain. The peptide groups are in green and the side-chain groups are in blue. The links (rigid groups) are separated by rotatable bonds constituting the joints of the chain

to as the *peptide group* include the C'_{i-1}, O_{i-1} and N_i atoms. The second, called the *side-chain group* includes $C\alpha_i$ and all side-chain heavy atoms attached to it. Although the second group may not be rigid due to variation of the χ angles, this motion has no direct effect on the conformation of the backbone. These rigid groups alternate along the chain. The peptide group is found between the ψ_{i-1} and ϕ_i bonds and the side-chain group is found between the ϕ_i and the ψ_i bonds. See Figure 1.4 for an illustration.

The rigid atom-groups can be thought of as *links* and the rotatable bonds make up the *joints*. Thus the backbone of the protein can be regarded as a long kinematic chain (serial linkage) with revolute joints between its links. This property will be exploited in this thesis to achieve significant gains in computational efficiency for a number of diverse applications in structural biology.

1.3.2 Properties of chains

In the general case, each joint can have up to 6 DOFs. Since the subject of this thesis is protein chains, only 1-DOF revolute joints corresponding to rotatable bonds will be considered. The number of DOFs of a kinematic chain can be computed using Grubler's formula [41] for manipulators in 3-D:

$$N_{DOF} \geq 6(n-1) - 5N_{joint}, \tag{1.5}$$

where l is the number of links and n_{joint} is the number of 1-DOF joints. Thus a chain with n links and $n-1$ joints will have $n-1$ *internal* DOFs. It will also have 6 *external* DOFs

that define its position and orientation in space.

1.3.2.1 Properties of an open chain

A chain is called *open* if at most one of its ends is fixed. A chain that is not fixed (a *free-floating* chain) has all its *external* DOFs as well as all its *internal* DOFs. A chain that is fixed at one end has no external DOFs. Some properties of open chains are:

Local changes have global effects: Changing a single DOF causes all links beyond this DOF, all the way to the end of the chain, to move. Any computation that requires knowing the absolute position of every link must perform linear work in the length of the protein after each change to the chain, even when the number of changed DOFs is a small constant.

Small angular changes may cause large motions: The displacement of a link caused by a change to a DOF depends not only on the angular variation, but also on the distance between the DOF axis and the link (radius of rotation). So, after applying a change, a link with a large radius of rotation undergoes a large displacement, which cannot be bound as a function of the magnitude of the change alone.

Large sub-chains remain rigid at each step: When only a few DOFs are perturbed, large contiguous fragments of the chain remain rigid. Consequently, there cannot be any new self-collisions inside each of these fragments and the distances between all pairs of atoms inside them remain fixed.

A protein backbone undergoing a Monte Carlo simulation is in fact such an open chain. It is free-floating in a force-field and deforms through changes applied to a few of its DOFs at each simulation step. The algorithm described in Chapter 4 for maintenance and self-collision detection in kinematic chains is designed to take advantage of these properties. A novel data structure — called *ChainTree* — is proposed for representing the deforming chain in a way that makes updates and self-collision queries efficient to compute. This data structure represents the chain geometry and kinematics at successively decreasing levels of resolution. In Chapter 5 this method is extended to efficiently update the energy of a protein during Monte Carlo simulations. The ChainTree is augmented to efficiently find all pairs of atoms closer than a specified cutoff distance. This is crucial when calculating the energy of protein conformations, a computation dominated by pairwise atomic interactions.

A companion data structure — called the *EnergyTree* — is introduced to efficiently reuse partial energy sums, exploiting the third property from the list above — that long contiguous fragments remain rigid between steps.

1.3.2.2 Properties of a closed chain

A closed chain is fixed at both endpoints. In effect the first and last links are identified, combined to be a single link, so the chain has only $n-1$ links. As a result, by applying Equation 1.5, the closed chain has only $n-7$ DOFs. These DOFs define an $(n-7)$-dimensional manifold — the *self-motion manifold* — embedded in the $(n-1)$-D configuration space of the chain. More formally, let $\mathbf{x} = f(\mathbf{q})$ be the forward kinematics function of the chain, where $\mathbf{x}$ is the 6-dimensional vector of end-point position and orientation and $\mathbf{q}$ is the vector of joint parameters. The self-motion manifold for a given end-point position and orientation $\bar{\mathbf{x}}$ is the pre-image of $\bar{\mathbf{x}}$ in f, namely $\{\bar{\mathbf{q}} | f(\bar{\mathbf{q}}) = \bar{\mathbf{x}}\}$. The self-motion manifold may consist of several disconnected components.

Searching for a conformation of a protein fragment that bridges a gap in a protein structure can be posed as an inverse kinematics problem. As such it becomes a search through the self-motion manifold of the fragment. A practical method for this search in the context of X-ray Crystallography is described in Chapter 2. The processing of a diffraction pattern obtain by shooting X-rays through a protein crystal generates a 3-D electron density map. The known sequence of the protein is built into the electron density map in order to determine its structure. Automatic model-builders may leave gaps in the resolved structure due to localized disorder in the electron density map. The method described in Chapter 2 automatically builds the missing fragment into the gap despite the deficient density information by limiting the conformational search to the fragment's self-motion manifold. A two-step algorithm is proposed that first rapidly generates a large set of candidate closed conformations, from which a small subset is chosen for refinement against the electron density map.

1.3.2.3 Properties of distances between the links of a chain

When computing similarity between protein structures, it is a common to use the Cα model described in Subsection 1.1.4. Thus, similarity is computed between chains of Cα atoms. In these chains, the distance between two consecutive Cα atoms is about 3.8Å and the angle formed by three consecutive Cα atoms is almost always between 90° and 130° [123].

Two properties of distances between links in these chains (Cα atoms) can be observed:

Neighboring links along the chain are spatially close Since the distance between two consecutive atoms in this chain is small and fixed, the distance between two links that are close along the chain is guaranteed to be small in all possible conformations of the chain. Another way of stating this is to say that the centroid of a small sub-chain is guaranteed to be close to all links in the sub-chain

Distant links along the chain are spatially distant (on average) This property is the complement of the previous one. The mean distance between two links over all conformations of the chain will be larger the farther apart these two links are along the chain. It is also true for sets of protein conformations (not necessarily random) due to the effect of excluded volume caused by van der Waals repulsion between atoms [74].

These observations motivate the algorithm presented in Chapter 3 that speeds up the computation of similarity between protein structures. The Cα representation of the backbone is compressed by averaging the coordinates of short sub-chains of length m. Thus each sub-chain is represented by a single 3-D coordinate vector. This shorter representation can then be used to approximately compute the structural similarity between two proteins. The choice of m allows a tradeoff between the error that is introduced and the speedup that is gained. This method is applicable to tasks that require the computation of similarity for all pairs of structures in a large data set, both sets of conformations of the same protein and native folds of different proteins.

1.4 Summary of contributions

In this thesis I make the following contributions:

1. Model completion in X-ray crystallography.
   (a) A method for sampling gap-closing fragments biased towards the electron density map.
   (b) A refinement method for improving a fragment's fit to density without breaking closure.
   (c) A fully automatic fragment completion software that can be used as part of an automatic structure determination pipeline.

2. Similarity computation for large sets of protein structures.

   (a) A uniform simplification scheme of protein structures for similarity computation.

   (b) A method based on this scheme that speeds-up existing similarity measures such as cRMS and dRMS.

   (c) The simplification scheme offers a trade-off between speed and precision of similarity computation.

   (d) A method for efficient computation of nearest neighbors in large sets of protein conformations.

   (e) A method for approximate classification of protein structures based on the program STRUCTAL.

3. Energy computation in Monte Carlo simulation of proteins.

   (a) An algorithm and data structure for efficient maintenance and self-collision detection for general kinematic chains.

   (b) An efficient algorithm for the computation of pair-wise interactions in Monte Carlo simulation of proteins.

   (c) An efficient caching scheme for partial energy sums during Monte Carlo simulation.

   (d) A Monte carlo simulation software using the above methods, freely available for download.

Chapter 2

An Inverse Kinematics Approach to Model Completion in X-Ray Crystallography*

2.1 Introduction

As outlined in Subsection 1.2.1, X-ray crystallography and nuclear magnetic resonance (NMR) spectroscopy are two widely used experimental techniques to obtain atomic coordinates of macromolecular structures. NMR spectroscopy enables structural biologists to observe dynamic properties of proteins, but is limited in protein size. X-ray crystallography has no such limitation, but has the drawback that it requires crystallizing protein samples. Once a sequence of X-ray diffraction images is collected from a protein in crystalline form, a distribution of electronic charge in $\mathbb{R}^3$ of the atoms constituting the molecule can be calculated. Coordinates of each atom are obtained by interpreting this 3-D *electron density map* [48]. This consists of placing the atoms in 3-D space to best match the electron density. The process, from synthesizing the protein in the lab to depositing the obtained coordinates of its folded structure in the Protein Data Bank (PDB) [15] may take several months, and sometimes years.

The Protein Structure Initiative (PSI), a National Institute of General Medical Sciences

*This chapter describes work done jointly with Henry van den Bedem and Ashley M. Deacon and published in [131, 199]

program in the US, aims to reduce the time and associated costs to determine a three-dimensional protein structure. Stimulated in part by funding initiatives such as the PSI, tremendous advances in automated model building have been achieved over just the past few years. Indeed, various programs are now capable of building a protein model into an experimental electron density map without human intervention [158, 192, 122, 90].

The degree of completeness of these initial models varies with the quality of the experimental data, yet rarely reaches 100%. Accurately determining the atomic coordinates of mobile fragments in the molecule, for instance, remains a challenge. Flexible loops may lead to disorder in the crystal, which, in turn, leads to electron density of locally poorer quality, rendering interpretation difficult. Manually completing a partial protein model, i.e. fitting backbone and side-chain atoms to the density to build the missing residues is generally feasible, but is time-consuming and easily leads to inaccuracies.

In practice, often a large portion of the molecule has been resolved, and the N- and C-termini of a missing fragment in the initial model are known. The missing main-chain fragment can be modeled as a kinematic chain, with rigid groups of atoms as links, and rotatable bonds as joints (see Subsection 1.3.1). Fitting a fragment can thus be cast as an inverse kinematics problem [136, 137]: Given the position and orientation of the endpoints of a kinematic chain, the corresponding joint angles need to be determined, which optimize its fit to the electron density map.

Exploiting this observation, we have combined an inverse kinematics algorithm with real space, torsion angle refinement in a two stage approach to fit a poly-alanine chain to the electron density between two anchor points. The first stage aims to sample a large number of closing conformations, guided by the electron density. These candidate conformations are ranked according to density fit. Top-ranking conformations are then subjected to torsion angle, real space sub-chain refinement in the second stage. Optimization steps are projected onto the null space of the sub-chain, thus preserving rigid geometry and closure.

2.2 Description of problem

Rapid protein structure determination depends critically on the availability of software that can automatically generate a protein model from an electron density map. At high resolution, existing programs may build over 90% of the protein backbone [13]. At medium to low resolution levels ($2.3\text{Å} \leq d < 2.9\text{Å}$), the model resulting from these programs is

typically a gapped chain, and only about 2/3 completeness is attained.

In a crystallographic experiment, the phase angle of the diffracted beam is lost, and only the magnitudes are recorded [48]. The phase angle is recovered at a later stage, but may have a substantial error associated with it, leading to systematic errors in the electron density map. The resolution at which diffraction data was collected also affects the interpretability of the map. High-resolution data allow the crystallographer to distinguish detail at the atomic level, whereas the electron density map appears "blurred" at lower resolution. It is often difficult or even impossible to obtain high-resolution electron density maps of some proteins. A protein crystal contains many replicas of the protein, which all contribute to the resulting electron density map. If the structures of these replicas are not identical, localized disorder in the electron density map may result. Temperature-dependent atomic vibrations and the existence of mobile regions in the protein are among the causes of local disorder.

Initial protein models, whether obtained manually or by automated procedures, are only approximately correct. Their coordinates typically serve as initial values for standard maximum likelihood algorithms, which improve the model coordinates using appropriate statistical techniques [145]. Iterating model building and refinement steps to improve the completeness and quality of atomic models has clear advantages [158, 193]. Phase information from the updated model is combined with experimental phases to improve the electron density map that is used to generate a new model. Missing fragments in a model hinder the ability of this procedure to converge to the correct phases and model, since significant parts of the electron density map remain unaccounted for by the model. Thus, filling in the gaps with fragments whose coordinates are within the radius of convergence of maximum likelihood algorithms may significantly improve the electron density map at an early stage in the process.

The input to our algorithm is the electron density map, the partial structure resolved by the automatic model builder, and the two anchor residues that need to be bridged (denoted N-anchor and C-anchor). Since in the majority of partially-resolved structures the amino acid sequence is correctly assigned, we assume the residue sequence of the missing fragment is known. Our goal is to propose candidate structures for the missing fragment that fall within the radius of convergence of existing refinement tools (1 - 1.5Å RMSD [145]).

The electron density map is likely to have systematic errors due to erroneous phases. Consequently the parts that are solved — including the anchor residues of the missing fragment — may also be misplaced. Moreover, the density in the gap is somewhat disordered

and thus not entirely reliable. Our protein model assumes *idealized geometry*, namely all bond lengths and bond angles are fixed to their idealized values [52]. These parameters vary very little in proteins [73], and their variance falls within the error margins of our input. Thus, the only DOFs in our model are the backbone torsional angles denoted the ϕ and ψ angles (one pair per residue). To simplify slightly our problem we will not have any side-chain DOFs and therefore no side-chain atoms in our protein model, with the exception of the Cβ atom (See Figure 1.3). Side-chains can be built onto the model once the backbone is fully in place. All Cβ and backbone oxygen atoms are included because they help orient the backbone while not adding any DOFs.

When computing the missing fragment for a given gap, the two anchor residues will be incorporated at the ends of the fragment. We denote these copies of the anchors the *mobile* anchors. The anchors that are attached to the resolved structure are denoted the *stationary* anchors. Having mobile anchors provides us with a measure of closure. Closure is measured at both ends as the RMS distance of the three backbone atoms of the mobile anchor residue from their positions on the stationary anchor residue.

2.3 Related Work

2.3.1 Exact inverse kinematics solvers

The problem of fitting a protein backbone fragment between two anchor points is closely related to the inverse kinematics problem in robotics [41]. It is known that for manipulators in a 3-D workspace there are a finite number of solutions to the inverse kinematics problem when the number of DOFs does not exceed six. However, there is no analytical method that can compute these solutions for all types of manipulators. In the case of a 6-revolute-joint manipulator, which is the most relevant to this work, an analytic solution can always be obtained and the number of unique solutions is at most 16 [164]. An efficient algorithm was derived in [135] which was later applied to the manipulation and conformational analysis of small molecular chains [136, 137]. For hyper-redundant robots, i.e. robots with a very large number of regularly distributed joints, methods based on curve approximations can be used [31]. Biologists have also developed their own specialized inverse kinematics tools. A method for computing conformations of ring molecules, when rotation is possible around all backbone bonds, was introduced as early as 1970 [69]. An inverse kinematics solution for three consecutive residues having ideal geometry was later devised [206]. Most

recently, a new formulation was developed that extends the domain to any three residues (not necessarily consecutive) with arbitrary geometry [39].

2.3.2 inverse kinematics solutions by optimization

Optimization techniques have also been applied to the inverse kinematics problem. A method for bounding the exact inverse kinematics solutions within small intervals in the context of drug design was proposed in [216]. When the number of DOFs exceeds six, no analytical solution is known. Currently, only optimization-based solutions exist. The *random tweak* method [60, 170] closes a loop by iteratively changing its DOFs until the desired distances between its two terminals are reached. It uses the Jacobian matrix of these distances (with respect to the torsional DOFs) to compute the DOF changes at each iteration. The *cyclic coordinate descent* [205] method closes a kinematic chain by iteratively adjusting its DOFs until a desired closure tolerance is reached. This method has been applied to protein chains [27].

2.3.3 Motion planning for closed loops

Searching for the fragment that best matches an electron density map has much in common with techniques used in roadmap-based motion planning for sampling robot configurations and connecting them. A number of works offer methods for planning the motion of closed kinematic loops. Common to many of them is the use of the probabilistic roadmaps framework [98]. The work in [212] samples configurations by first ignoring the closure constraint and then enforcing the constraint through gradient descent. Nearby configurations are connected by the chaining of local steps that are generated in the null space of the Jacobian matrix. In [81], configurations are sampled by breaking the loop into an *active* part, for which forward kinematics sampling techniques are used, and a *passive* part, for which an exact inverse kinematics solution is computed. Nearby samples are connected by using a local planner for the active part and letting the passive part follow the motion. An extension of this method [38] samples the active part of the chain one DOF at a time, making sure its endpoints are always reachable by the passive part. This method was recently applied to sampling conformations of mobile protein regions and to studying their flexibility [37]. Another extension to [81] is proposed in [211]. A roadmap for solving a slightly different problem, known as *point-to-point inverse kinematics* for redundant robots, described in [7],

also takes into account joint constraints and obstacle avoidance. A polynomial-time complete planner for the reconfiguration of closed loops with spherical joints and no obstacles is proposed in [196].

2.3.4 Methods for redundant manipulators

The refinement method used in this work exploits the redundancy of the protein chain to minimize a target function without breaking chain closure. It is closely related to the planning of instantaneous motion of redundant manipulators. Here the redundant DOFs are used for concurrent optimization of additional criteria beyond the completion of the main task (e.g., minimizing torque or kinetic energy) or the execution of additional lower priority tasks. Some examples include [25, 30, 100, 101].

2.3.5 The loop closure problem in biology

Our problem is closely related to the *loop closure* problem in structural biology, where loops are predicted based on sequence information alone. The methods proposed for this problem can be roughly divided into two types: *ab initio* methods and *database search* methods. The ab initio methods search the loop conformation space, while database methods screen the PDB for known structures that closely meet the requirements of the desired loop.

2.3.5.1 Ab initio methods

For loops that have six DOFs or less, exact methods that enumerate all possible solutions can be used, e.g. [39, 69, 135, 137, 206]. It is also possible to use the exact solver hierarchically as in [206] and thus handle longer loops. Examples of search methods include sampling biased by the database distribution of the ϕ/ψ angle pairs [144], uniform conformational search [24], sampling from a small set of ϕ/ψ pairs [43, 44] or sampling from a small library of short representative fragments [109]. Other algorithms sample conformations randomly and then enforce the closure constraints, e.g. the *random tweak* method [60, 170] or a method based on the cyclic coordinate descent algorithm [27]. Various methods exist for optimization of candidate loops, such as molecular dynamics [24, 61, 220] and Monte Carlo [2, 35] simulations.

2.3.5.2 Database search methods

Work on extracting loop candidates from the PDB started with [95]. Loops are chosen that have the right length and meet the required geometric constraints. The applicability of this approach was initially limited to loops of length 4 [59], but later work suggested an upper limit of 9 [202]. A recent study claims the limit should be raised to 15 [49].

2.3.5.3 Crystallography

A variety of techniques have found successful application and widespread use in automated interpretation of electron density maps. The program *ARP/wARP* [158], for instance, iteratively adds pseudo-atoms to a partial model that it subsequently refines in reciprocal space. The program *TEXTAL* [90] employs local pattern recognition techniques to select regions in a database of previously determined structures similar to those in the unknown structure. Some automated systems targeting lower resolution levels, notably *RESOLVE* [192] and *MAID* [122], start by identifying larger secondary-structure elements using sophisticated template matching techniques, and then connect these 'fits' through loop regions.

Relying on unambiguous experimental data and elementary stereochemical constraints, areas of poorly defined electron density remain a challenge for these approaches. For instance, exposed, mobile loop regions typically have poorly resolved side chain density, or show discontinuous main-chain density even at low contour levels. Patterns in the density may go unnoticed to template matching techniques for a variety of reasons. Nevertheless, at high resolution, these programs may provide over 90% of the protein main chain of the final model [13]. At medium to low resolution levels ($2.3\text{Å} \leq d < 2.9\text{Å}$), the initial model resulting from these programs is typically a gapped polypeptide chain, and only about 2/3 completeness is attained. In the majority of cases, the amino acid sequence is correctly assigned, so gap lengths and the identity of their residues are known.

In practice, to complete a model, the crystallographer manually builds the missing residues onto the partially completed structure using an interactive graphics program. These programs, such as the *X-BUILD* package in *QUANTA*, *Insight II* (both Accelrys, Inc.), and *O* [94] provide a variety of semi-automated tools and techniques to assist the model building and refinement steps. In *O*, database fragments straddling a gap can be refined against the density using torsion angle refinement based on grid summation [96]. Oldfield [151] developed a method combining a random search of conformation space with grid- and

gradient-based refinement techniques to close loops. Insight II employs the *random tweak* algorithm [60, 170] to build fragments.

2.4 Methods

The algorithm proceeds in two stages: candidate generation and refinement. In the first stage, candidate loops are built using the cyclic coordinate descent algorithm, while putting additional constraints on the DOFs to take the electron density and collision avoidance into account. Next, initial conformations are ranked according to density fit. Top-ranking initial conformations are refined by minimizing a standard real-space target function [45, 29, 111]. An optimization protocol based on simulated annealing and Monte Carlo + minimization [125] searches for the global minimum of the target function while maintaining loop closure. Each candidate is optimized 6 times and the best scoring loops are returned.

Deficient density information is compensated for by taking advantage of the loop closure constraint to guide the loop to its correct positioning in space. In the first stage, the closure constraint enables the generation of loops that lie within 2Å RMSD of the true solution. The approximate enforcement of the closure constraint during loop refinement prevents the search from diverging and limits the searched space to motions that preserve loop closure.

The implementation of the algorithm uses the following software packages: *Clipper* [40], the *CCP4 Coordinate Library* [112] and the exact inverse kinematics solver of [39].

2.4.1 Stage 1: Generating loop candidates

We start by constructing a protein fragment $\mathcal{C}$ of length L in a random conformation, where residue 0 is a copy of the N-anchor (the residue preceding the gap), and residue $L-1$ is a copy of the C-anchor (the residue after the gap). This fragment is attached to either the N- or C-anchor, thus determining the *closing direction*.

Upon starting the procedure, the position of the mobile anchor will not coincide with the stationary anchor. The algorithm adjusts each backbone dihedral angle in turn such that the distances between the backbone atoms of the mobile anchor and the corresponding atoms of the stationary anchor are minimized. A total of 2000 iterations are allowed for closure up to a preset tolerance distance d_{closed}. Chains that do not close are discarded. A cross-correlation density score is calculated for all conformations, and the 99-th percentile is

passed on to stage two. Longer loops (9 or more residues) are split in the middle. Each half-chain is attached to its corresponding stationary anchor, and its terminal residue alternates between acting as stationary and mobile anchor in subsequent iterations.

2.4.1.1 Random Initial Conformations

A total of 1000 starting conformations are calculated to start the procedure. For each starting configuration, ω_i is considered to be a fixed, $N(180, 5.8)$ random variable for all i. Half of the starting configurations are obtained by adjusting each $(\phi, \psi)_i$ in turn to optimize agreement with the electron density while stereochemical constraints are observed. The remaining five hundred starting conformations are purely random, and obtained from sampling $(\phi, \psi)_i, i = 0 \ldots L-1$ angle pairs from PDB-derived distributions. A finite mixture of bivariate normal distributions was thereto fitted to frequencies calculated from the Top500 database [132] of non-redundant protein structures using the program EMMIX [139]. We obtained distributions for each of the 20 amino acids, and an additional distribution for residues immediately preceding proline in the amino acid sequence. The angles ϕ_0 and ψ_{L-1} remain fixed at their initial values.

2.4.1.2 Electron Density Constraints

A change to the DOFs of a residue is calculated as follows: A distance minimizing value is proposed for dihedral angle ϕ_i of residue i by the cyclic coordinate descent method, and based on ϕ_i, a minimizing ψ_i is proposed. (In our implementation, we change each DOF in turn, although this is not strictly necessary.) Thus, a proposed angle pair $(\phi, \psi)_i^p$ is obtained. To guide the loop, a heuristic electron density constraint has been added to the cyclic coordinate descent algorithm. For each pair i, consider the set of atoms $\mathcal{A}_i$ that is subject to change by angle pair i, and not affected by changes in angle pair $i+1$. Hence, $\mathcal{A}_i = \{C\beta_i, C_i, O_i, N_{i+1}, C\alpha_{i+1}\}$, where $C\beta_i$ is excluded whenever residue i is a Glycine. Electron density scores corresponding to trial positions in a square neighborhood $U_{(\phi,\psi)^p}$ about $(\phi, \psi)^p$ in conformation space are calculated. A simple local scoring function is adopted; the sum of the electron density values at atom center positions of $\mathcal{A}_i$. The angle pair $(\phi, \psi)_i$ is set to the trial position with maximum density score, i.e $(\phi, \psi)_i = \arg\max_{(\phi,\psi)\in U_{(\phi,\psi)_i^p}} S(\phi, \psi)$, where $S(\phi, \psi) = \sum_{A_j \in \mathcal{A}_i} \rho(A_j(c))$, and $A_j(c)$ denotes the center of atom A_j. At this point, overlaps of van der Waals surfaces of atoms in $\mathcal{A}_i$ and the rest of the protein structure are determined. If no overlaps occur, the new $(\phi, \psi)_i$ pair is accepted, otherwise the pair is

accepted with a probability inversely related to the amount of overlap. The size of $U_{(\phi,\psi)^p}$ is reduced linearly in the number of cyclic coordinate descent iterations to allow closure of the chain.

2.4.2 Stage 2: Refining loop candidates

A candidate fragment is refined by minimizing the least squares residuals between the electron density map ρ^o and the density calculated from the model ρ^c. This is a standard procedure used in crystallography for refinement of protein models [29, 45]. The target function sums the squared differences between the observed density and the calculated density at each grid point in some volume V around the fragment:

$$T(q) = \sum_{g_i \in V} \left[S\rho^o(g_i) + k - \rho^c(g_i) \right]^2 . \tag{2.1}$$

The calculated density at each grid point is a sum of contributions of all atoms whose center lies within a cutoff distance from this point. The calculated density contribution of an atom is a sum of isotropic 3-D Gaussians [204]. The factors S and k scale ρ^o to ρ^c and are computed once at initialization using the partial model.

2.4.2.1 Optimization with closure constraints

Our method is based on the approach described in [25, 100] for optimizing an objective function while performing a task by taking advantage of robot manipulator redundancy. The redundant DOFs define a subspace of configuration space termed the *self-motion manifold*. Motions on this manifold do not influence the main task and thus can be used to move the manipulator configuration towards a minimum of the objective function. Since this manifold may be very complex, these motions are in general difficult to compute. It is therefore common to locally approximate the self-motion manifold by its linear tangent space. This space is defined as the null-space of the linearized instantaneous kinematic relation between the linear and angular velocities of the end-effector (a frame attached to the end of the chain) and the joint velocities, also known as the chain's Jacobian matrix [41]. For an n-DOF chain in $\mathbb{R}^3$ at configuration q, the Jacobian $J(q)$ is a $6 \times n$ matrix satisfying the equation:

$$\dot{x} = J(q)\dot{q}. \tag{2.2}$$

Thus, $J(q) = df(q)/d(q)$ where $f(q)$ is the chain's forward kinematics function mapping joint coordinates to end-effector position and orientation. The rank of the Jacobian in $\mathbb{R}^3$ is at most 6 and thus the dimensionality of its null space is at least $n - 6$.

In general, an instantaneous change in the configuration corresponding to a desired small change in end-effector position is computed by inverting Equation 2.2. We get:

$$\dot{q} = J^{\dagger}(q)\dot{x} + N(q)N^{T}(q)y, \tag{2.3}$$

where $J^{\dagger}$ is the pseudo-inverse of the Jacobian and $N(q)$ is an orthonormal basis for the null-space. The null space can now be used to optimize some objective function without affecting the motion of the end effector. When optimizing an objective function for a closed loop, the instantaneous change in position and orientation of the end, $\dot{x}$, is set to zero and y is taken to be the gradient vector of the objective function. Projecting y onto the null space of the Jacobian produces a motion that minimizes the objective function while not disturbing the chain closure. In general, the step size of the motions (the magnitude of the vector $\dot{q}$) needs to be small since the Jacobian approximation of the self-motion manifold is valid only in a small neighborhood about the current point in configuration space.

2.4.2.2 Implementation details

Computing the null space: We use the *atom group local frames* method [217] to represent the protein as a kinematic chain and the method prescribed in [28] to efficiently compute the Jacobian. The null space is computed by applying the Singular Value Decomposition [71] to the Jacobian matrix giving the decomposition $J(q) = U\Sigma V^{T}$, where V is an orthonormal basis for the row-space of $J(q)$. The null-space basis $N(q)$ is simply the set of vectors of V that correspond to singular values (the diagonal elements of Σ) that are equal to zero.

Target function gradient: We derive an analytical expression for the gradient of the target function with respect to the torsional DOFs of the fragment. The gradient is computed using a recursive method [4], which is linear in the number of DOFs of the chain. The naïve approach in this case has quadratic complexity.

Minimization protocol: A gradient descent search for the minimum of the target function is prone to get stuck in local minima. To increase the radius of convergence, we use the

Monte Carlo + minimization approach [125]. At each step, a large random move in conformation space is proposed, the new conformation is then minimized by gradient descent and the resulting local minimum is accepted or rejected using the Metropolis criterion [140]. The effect of minimization is to increase the acceptance probability of a trial move, enabling the search to make more progress. This comes at the cost of increasing the time of each simulation step. To further improve the ability to escape from local minima, we envelop the Monte Carlo + minimization method with a simulated annealing (SA) protocol [104], which controls the probability of acceptance of uphill moves by varying the pseudo-temperature of the search.

Random moves: Two methods are used for generating large random closure-preserving moves for the Monte Carlo + minimization method. The first is to choose a random direction in the null-space of a randomly chosen sub-chain, and taking a step whose magnitude depends on the current temperature of the SA protocol. A similar method for motion on the self-motion manifold was described in [212]. Before performing minimization, we make sure the closure tolerance has not been exceeded. A second method for generating random steps is the use of an exact inverse kinematics solver [39]. One of the solutions is chosen at random as the proposed move. The use of an exact solver allows jumping between unconnected parts of the self-motion manifold, which the Jacobian-based moves are unable to do.

Approximate closure: The closure constraint is relaxed during this stage. A maximum RMSD of 0.5Å is allowed at both ends of the fragment. Since there are errors in the positions of the anchor residues, and because the resulting fragment will be subsequently refined together with the rest of the protein, this tolerance is acceptable. By allowing approximate closure, larger steps can be taken in the null space of the Jacobian.

Refinement search protocol: The refinement protocol is composed of three nested loops (see Figure 2.1). The inner loop performs Monte Carlo + minimization using the two methods described above for generating trial moves. The middle loop performs the SA protocol by gradually reducing the pseudo-temperature of the Monte Carlo + minimization. The outer loop enhances the SA protocol by restarting the search at decreasing pseudo-temperatures. Decreasing levels of smoothing are applied to the density after each

```
for start_temp = high_start_TEMP downto low_start_TEMP {
    temp = start_temp;
    SmoothDensity(start_temp);
    for SA_steps = 1 to 8 {
        for MCM_steps = 1 to NUM_ITERS {
            M = ProposeRandomMove(temp);
            MinimizeMove(M);
            AcceptMove(M);
        }
        temp *= TEMP_dec_factor;
    }
}
```

Figure 2.1: Pseudo-code for refinement search protocol

restart. The magnitude of attempted null-space moves is reduced together with the pseudo-temperature of the simulation to increase the chance that these moves will be accepted.

Density smoothing: The density map is smoothed by convolving it with an isotropic 3-D Gaussian kernel. The convolution is done using three 1-D kernels exploiting the separability of the Gaussian kernel. Since the convolution of a Gaussian with a Gaussian can be computed analytically by summing the means and variances of the two functions, computing ρ^c does not require any convolution. The variance of the summed Gaussians in ρ^c is simply augmented by the variance of the desired kernel.

2.5 Experimental results

The performance of the algorithm was evaluated at three different scenarios. In the first scenario, described in Subsection 2.5.1, fragments of varying lengths are removed from two resolved structures and completed using the final refined electron density map deposited in the PDB. The gaps in this scenario are artificial and are meant to test the performance of the method under ideal conditions. In a second scenario, described in Subsection 2.5.2, the method's ability to close gaps at various lengths using the experimental electron density map and initial models was tested. This scenario represents well the conditions under which the method would be used in practice. The experimental data was provided by the Joint Center for Structural Genomics (JCSG). Finally, in a third scenario, described in

Subsection 2.5.3, the algorithm's ability to identify alternative conformations in a disordered region was evaluated.

In all the tests that are described below, the missing fragments were built in the following way:

1. 1000 candidate fragment conformations are generated in the first stage of the method.
2. The 6 top-ranking candidates are selected for further refinement.
3. Each top-ranking candidate is refined 6 times.
4. The 2 top-scoring fragments of each refinement set are saved.

This gives a total of 12 fragments. The program also writes a log file containing the full cross-correlation electron density score for each fragment.

Fragments are evaluated using all-atom cRMS, which is the RMSD between all atoms of the generated poly-alanine fragment and their corresponding atoms in the known PDB structure of this fragment, after the fragments have been optimally aligned in space.

2.5.1 Completing artificial gaps

Artificial gaps were generated in two protein structures taken from the PDB: TM1621 (PDB code 1O1Z, SCOP classification a/b), a 234-residue protein structure resolved at a resolution of 1.6Å, and TM0423 (PDB code 1KQ3, SCOP classification multi-domain a/b) a 376 residue protein. The results for TM1621 are described below in detail. A summary of the results for TM0423 is also given.

2.5.1.1 Tests on TM1621.

A set of 103 structurally diverse fragments was obtained by creating gaps of length 4, 8, 12, and 15 at each even numbered residue of the protein. This protein consists of one chain, with 34% of the residues in 10 alpha helices, and 19% in 9 beta sheets. To evaluate the performance at various resolution levels, three $2mF_o - DF_c$[1] electron density maps were calculated at 2.0, 2.5, and 2.8Å, using structure factors obtained from the PDB. Since the low resolution electron density was obtained by truncating a high resolution data set, the RMSDs in this section are not typical for their resolution levels.

[1] A standard formula for computing the final refined EDM, where F_o is the observed structure factors and F_c is the structure factors computed from the model.

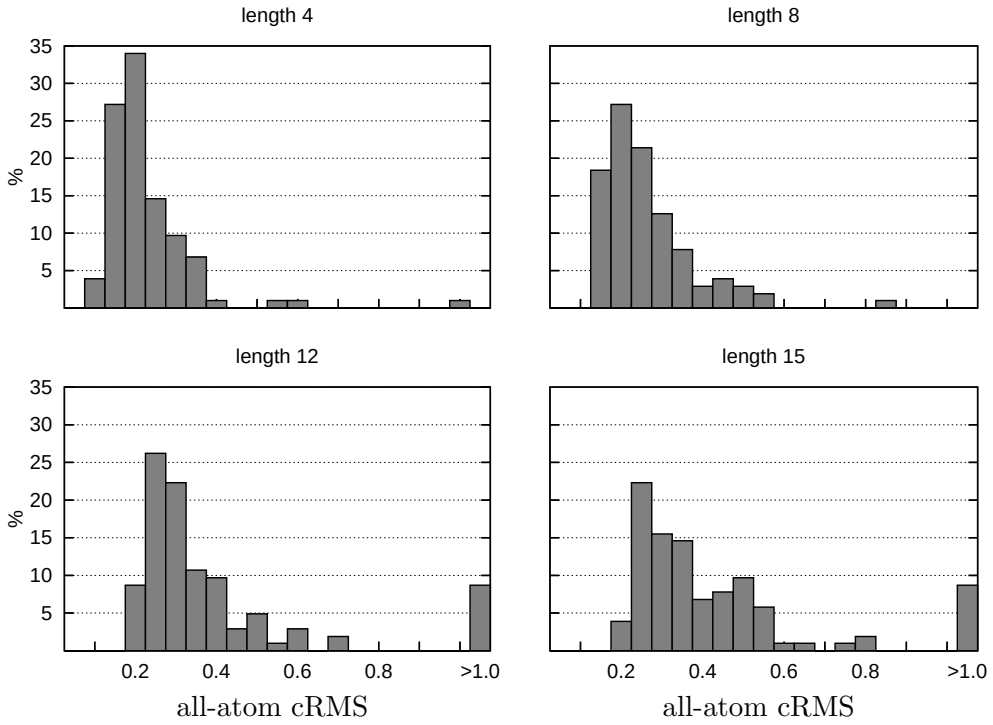


Figure 2.2: The all-atom cRMS-distribution of 103 fragments with lengths 4, 8, 12, and 15 residues of TM1621 at a resolution of 2.0Å. A total of 9% of 12-residue and 9% of 15-residue fragments have an all-atom cRMS > 1.0Å.

At a resolution of 2.0Å, the algorithm successfully closed all 103 gaps of length 4 to within 1.0Å, and all length 8 gaps to within 0.85Å, as shown in Figure 2.2. Wider gaps are more difficult to close; a total of nine 12-residue and nine 15-residue fragments were found to have an all-atom cRMS greater than 1.0Å.

To evaluate the effect of secondary structure on RMSD, all 12- and 15-residue fragments were classified as helix, strand or 'other.' A fragment is considered a helix or strand only if at least 2/3 of its residues are classified as such. A total of fourteen 12-residue fragments and eight 15-residue fragments met our criteria for helices. Three 12-residue fragments and no 15-residue fragments were classified as strands. The maximum all-atom cRMS for the 12-residue strands over all resolutions was 0.3Å. Four percent of non-helical, 12-residue fragments were found to have an all-atom cRMS > 1.0Å, compared to 36% of helical fragments. For 15-residue fragments, these numbers are 4% and 63% respectively.

At a resolution of 2.5Å, all gaps of length 4 and 8 were closed to within 1.0Å all-atom

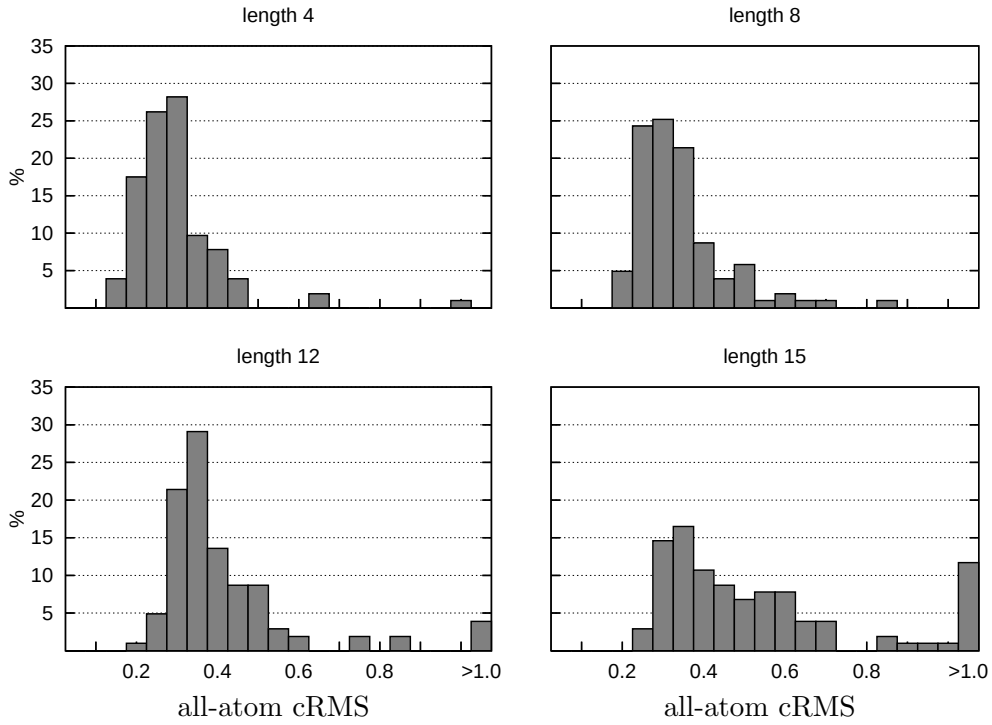


Figure 2.3: The all-atom cRMS-distribution of 103 fragments with lengths 4, 8, 12, and 15 residues of TM1621 at a resolution of 2.5Å. A total of 4% of fragments of length 12, and 12% of fragments of length 15 have an all-atom cRMS > 1.0Å.

cRMS and 0.85Å all-atom cRMS respectively, whereas four 12-residue fragments and twelve 15-residue fragments deviated by more than 1.0Å all-atom cRMS. The results are depicted in Figure 2.3. One percent of non-helical, 12-residue fragments were found to have an all-atom cRMS > 1.0Å, compared to 21% of helical fragments. For 15-residue fragments, these numbers are 7% and 63% respectively.

At a resolution of 2.8Å, all gaps of length 4 and 8 closed to within 1.05Å all-atom cRMS and 0.75Å all-atom cRMS respectively. Four 12-residue fragments and eighteen 15-residue fragments deviated by more than 1.0Å all-atom cRMS. The results are depicted in Figure 2.4. Two percent of non-helical, 12-residue fragments were found to have an all-atom cRMS > 1.0Å, compared to 14% of helical fragments. For 15-residue fragments, these numbers are 12% and 88% respectively.

Table 2.1 summarizes the performance at three resolution levels.

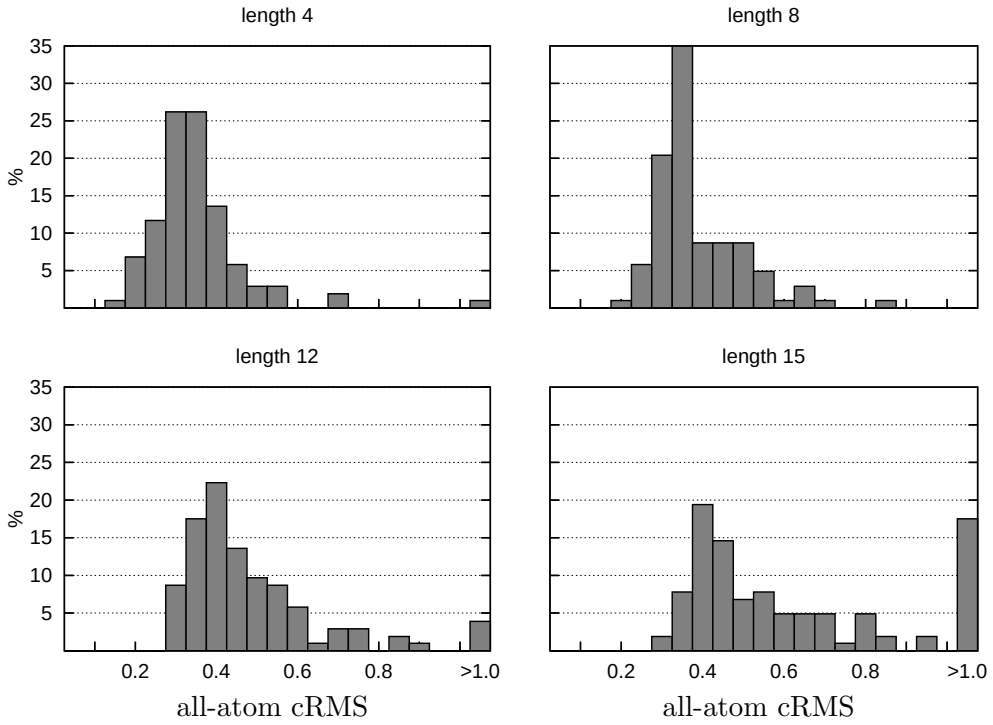


Figure 2.4: The all-atom cRMS-distribution of 103 fragments at lengths 4, 8, 12, and 15 residues of TM1621 at a resolution of 2.8Å. A total of 4% of fragments of length 12, and 17% of fragments of length 15 have an all-atom cRMS > 1.0Å.

2.5.1.2 Tests on TM0423.

An equivalent analysis on TM0423, a protein with a helical domain, gives similar results, see Table 2.2. TM0423 consists of one chain, with 46% of the residues in 16 helices, and 11% in 8 beta-sheets. The longest helix has length 17, and, if a single Glycine classified as a hydrogen bonded turn is included, its length is 26.

2.5.1.3 Discussion of results

Clearly, the algorithm performs more modestly when fitting longer fragments. In addition to an increasing median all-atom cRMS, a larger proportion of fragments deviates by more than 1.0Å as fragment length increases, particularly when a large number of residues are in alpha-helical conformation. It has been observed in previous studies that accurately modeling secondary-structure elements may require specialized sampling algorithms [91]. Our

	2.0Å			2.5Å			2.8Å		
length	$\tilde{x}$	$\bar{x}$	p	$\tilde{x}$	$\bar{x}$	p	$\tilde{x}$	$\bar{x}$	p
4	0.13	0.14	0	0.18	0.19	0	0.31	0.32	0
8	0.16	0.18	0	0.23	0.23	0	0.33	0.36	0
12	0.28	0.51	9	0.34	0.41	4	0.41	0.52	4
15	0.33	0.53	9	0.43	0.63	12	0.49	0.76	17

Table 2.1: Median ($\tilde{x}$) and mean ($\bar{x}$) all-atom cRMS of fitted fragments to corresponding regions in TM1621 at resolutions 2.0, 2.5, and 2.8Å, and percentage of fragments deviating by more than 1.0Å (p).

	2.0Å			2.5Å			2.8Å		
length	$\tilde{x}$	$\bar{x}$	p	$\tilde{x}$	$\bar{x}$	p	$\tilde{x}$	$\bar{x}$	p
4	0.18	0.19	0	0.24	0.25	0	0.32	0.32	0
8	0.20	0.22	0	0.28	0.29	0	0.35	0.38	0
12	0.29	0.55	26	0.33	0.50	19	0.40	0.56	19
15	0.34	0.96	38	0.43	0.92	29	0.52	1.19	29

Table 2.2: Median ($\tilde{x}$) and mean ($\bar{x}$) all-atom cRMS of 174 fitted fragments to corresponding regions in TM0423 at resolutions 2.0, 2.5, and 2.8Å, and percentage of fragments deviating by more than 1.0Å (p).

current implementation lacks such targeted approaches, yet gives acceptable performance for fragments up to length 12 across all resolutions.

Interestingly, lowering the resolution of the data only mildly affects performance. We believe that this is the true strength of the algorithm; lack of structured, well-defined electron density information is compensated by maintaining a closed conformation.

2.5.1.4 Run times

The run time of the algorithm depends on the length of the fragment to be fitted, as well as on the resolution of the diffraction data. Run times vary from about 30 minutes for short fragments to just under 3 hours for the longest fragments at high resolution. Table 2.3 summarizes average run times calculated while generating the 103 fragments of TM1621 used in this section. All tests were performed on a 2.66GHz Intel P4 Xeon running RedHat 9. The source code was compiled using gcc 3.2.

length	2.0Å	2.5Å	2.8Å
4	40	29	28
8	92	63	58
12	134	82	73
15	178	105	95

Table 2.3: Average run times (in minutes) on a 2.66GHz Intel P4 Xeon at various fragment lengths and resolution levels. The average is calculated over 103 fragments.

2.5.2 Completing true gaps

In this section, we present three examples of protein model completion by inserting poly-alanine fragments into a gapped, initial model at high and medium-to-low resolution. Rather than closing a few selected gaps, we aim to fully complete each model. Thus, we calculate all missing fragments less than 16 residues in length. In one instance, the protein TM1586, the algorithm was actively used to complete the model, and detailed results will appear in a separate publication. The remaining two structures had been completed and refined prior to testing the algorithm and we use their PDB structure to verify our results. All initial models were obtained from common crystallographic model building programs.

It was found that residues anchoring a gap in partial models do not always fit the density correctly. In these cases, the gap was widened at its N and/or C end until the new anchors fit the density satisfactorily. Furthermore, missing fragments of length < 4 are extended to length 4 in this section, again by widening the gaps at both ends.

The electron density score of generated fragments and RMSD to the final, refined structure cannot be expected to perfectly correlate in areas of poor density. In an extreme case, it may happen that conformations attain a higher score by jumping over to a neighboring, empty stretch of density (a beta-sheet, for instance) for a few residues. In this section, in addition to the all-atom cRMS of the best scoring fragment, we therefore report the lowest achieved all-atom cRMS.

2.5.2.1 Completing TM1586 at 2.0Å

An initial model for the 206-residue hypothetical protein TM1586 was obtained from Xsolve, a fully automated crystallographic data processing and structure solution software suite under development at the JCSG [209]. At the time of processing this data, Xsolve only supported RESOLVE v2.06 for model building.

Gap	Length	Secondary Structure	all-atom cRMS (Å) (Top Score)	all-atom cRMS (Å) (Lowest)
19-29	9	·	2.43	0.95
53-58	4	·	2.06	0.49
95-105	9	S··STTTTS	0.95	0.79
113-123	9	·····	0.49	0.38
147-157	9	HT·GGGGG·	0.47	0.45

Table 2.4: RMSD of fitted fragments in TM1586 and corresponding regions in the final, refined structure.

The model was obtained from MAD data collected at 2.0Å, and showed gaps in between residues 92-104, 113-123, and 148-156. Furthermore, 66 residues were missing at the N terminus of the molecule. Overall completeness was reported to be 51%. After widening the gap 148-156 by one residue at each end, this gap and gap 113-123 were easily closed to within 0.5Å all-atom cRMS using the experimental map from RESOLVE. The gap in between residues 92-104 proved to be more difficult. After combining the RESOLVE model and our fragments with an ARP/wARP model, a more complete model was obtained after various rounds of phase improvements. The N terminus was now largely complete, with gaps remaining in between residues 19-29, 55-58, and 95-105. The gap in between residues 19-29 was widened to span residues 19-32. Once fragments for these final gaps were generated and inserted, the complete model was refined using a maximum likelihood, torsion angle dynamics protocol in CNS. Table 2.4 summarizes the all-atom cRMS of top-scoring fragments to this CNS-refined structure.

The best scoring fragments for the gaps 19-29 and 53-58 are likely not within the radius of convergence of standard refinement programs. In both cases, a different fragment was manually selected from the 12 candidates and inserted into the model prior to refinement. Figure 2.5 shows residues 19-32 of the CNS-refined structure, together with the fragment that was inserted into the model. Note that the main-chain density is discontinuous at the displayed contour level of 0.8 σ, and that side-chain density is poorly defined.

2.5.2.2 Completing TM1742 at 2.4Å.

MAD data for the 271-residue, putative Nagd protein TM1742 (PDB code 1VJR) was collected at a resolution of 2.4Å. An initial electron density map of good quality was obtained using the program SOLVE v2.03 [194] at a resolution of 2.5Å. Iterative model building

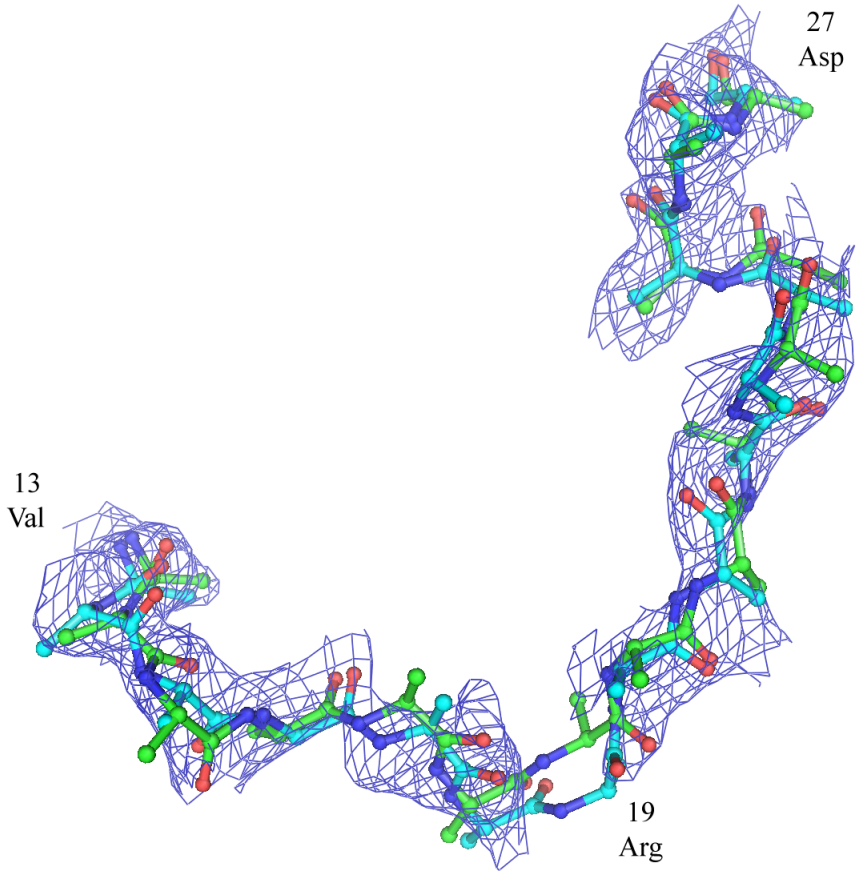


Figure 2.5: Residues 19-32 of TM1586. The fragment inserted into the model is shown in cyan, and the corresponding CNS-refined fragment in green. The all-atom cRMS between the two fragments is 0.95Å. The electron density map is shown contoured at 0.8 σ, and is discontinuous around the Arginine 25.

Gap	Length	Secondary Structure	all-atom cRMS (Å) (Top Score)	all-atom cRMS (Å) (Lowest)
17-25	7	ETTEE·T	0.72	0.66
56-62	5	HHHHT	0.78	0.78
126-132	5	HHHHH	0.36	0.36
146-148	1	·	0.44	0.40
191-202	10	HHHHHT··GG	0.43	0.43
228-233	4	SSS·	0.22	0.22

Table 2.5: RMSD of fitted fragments in TM1742 and corresponding regions in the final, refined structure obtained from the PDB.

using Terwilliger's `resolve_build` script resulted in an 88% complete model, with gaps in between residues 17-25, 56-62, 129-132, 146-148, 229-231. Furthermore, the region in between residues 191-202 had been built incorrectly. The RESOLVE model was independently completed and refined. Table 2.5 summarizes the all-atom cRMS of top-scoring fragments built with our algorithm to the final, refined structure.

2.5.2.3 Completing TM0542 at 2.6Å.

MAD data for the 376-residue protein TM0542 (Malate Oxidoreductase) was collected at a resolution of 3.0Å, and a native data set was obtained at 2.6Å. An electron density map was calculated with phase extension using the program SOLVE. Iterative model building using RESOLVE revealed that the unit cell contains four NCS related molecules. Molecule A was the most complete of this set of four with 56% of residues placed, and gaps in between residues 12-89, 134-142, 212-227, 256-266, 272-285, and 318-324. This RESOLVE starting model was independently completed and refined. The refined model was used to calculate RMSDs for our automatically generated fragments.

The algorithm successfully closed all gaps, but for the first 76-residue one. Table 2.6 summarizes the results.

Fitting a poly-alanine fragment into the density is rather sensitive to residues being flipped along the chain. This problem is exacerbated by the fact that exposed loop regions typically have poorly resolved side chains in the electron density. Figure 2.6 shows an example of a fragment where two consecutive residues are flipped. While the all-atom cRMS is relatively high at 0.9Å for this fragment, the $C\alpha$-trace is in excellent agreement with the manually built fragment. The flipped residues are easy to identify and correct for

Gap	Length	Secondary Structure	all-atom cRMS (Å) (Top Score)	all-atom cRMS (Å) (Lowest)
134-142	7	HHHHHHH	0.93	0.78
212-227	14	BS··SSGGGGG·HH	0.91	0.90
256-266	9	ES·SS·SHH	0.87	0.87
272-285	12	·SSEEEEEE·SS	1.15	1.15
318-324	5	HHHHH	0.72	0.72

Table 2.6: RMSD of fitted fragments in molecule A of TM0542 and corresponding regions in the manually built structure.

a trained crystallographer.

2.5.3 Identifying alternative main-chain conformations

Binding of ligands to a protein or protein-protein interactions are typically facilitated by mobile regions in the macromolecule. Such flexible loops are often poorly resolved in the electron density due to disorder in the crystal, making it difficult for the crystallographer to identify multiple conformations. Our methods in principle support identification and refinement of multiple conformations, even at sub-atomic resolution.

A model for the 398-residue hypothetical protein TM0755 was determined from a 1.8Å MAD data set using ARP/wARP. The structure was completed manually, apart from a short fragment around residue A320. The electron density from residue A317 to A323 suggested that this fragment crystallized in two distinct conformations. Furthermore, a structurally similar dioxygen reduction enzyme, Rubredoxin Oxygen: Oxidoreductase (PDB code 1e5d), binds a Flavin Mononucleotide at the corresponding residues, providing additional evidence for the presence of multiple conformations at this site.

While one conformation was clearly visible in the electron density, the main-chain trace of the alternative conformation was much less obvious. From residue A320 to A323 the density was particularly ambiguous; the alternative conformation was difficult to identify and initially not modeled. To complete the model, it was decided to build two conformations at half occupancy each. The algorithm was slightly modified; half occupancy was hard-coded, and density-smoothing was disabled to narrow the radius of convergence of the refinement stage. Runs at four different lengths were attempted. The N-anchor was kept fixed at Serine A316, and the C-anchor ranged from Alanine A320 to Histidine A323. In

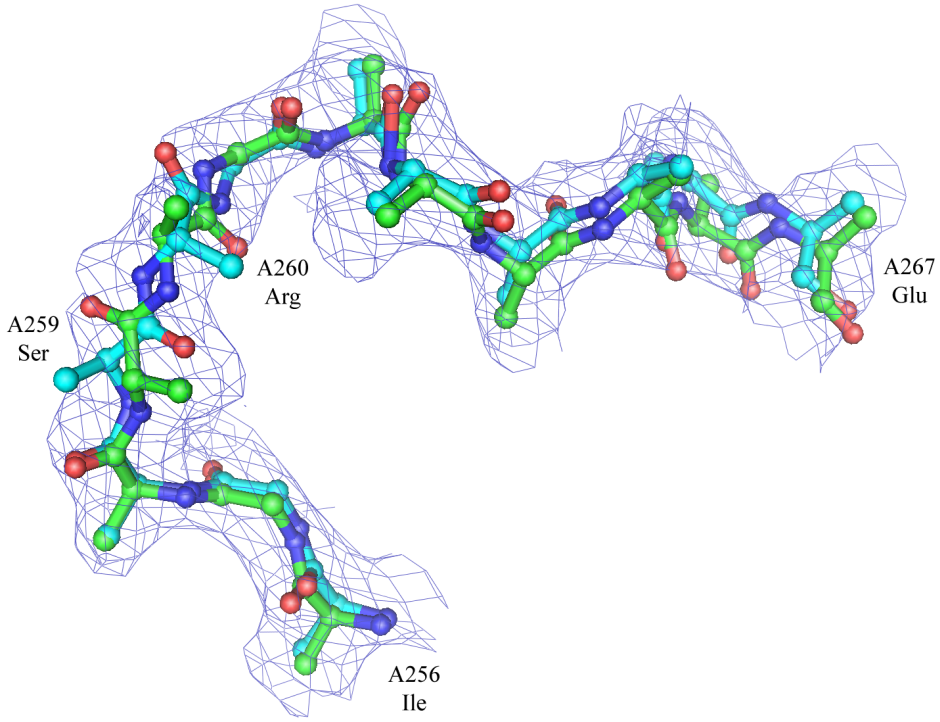


Figure 2.6: Residues A256-A267 of TM0542. The top-scoring fragment is shown in cyan, and the corresponding manually completed and refined fragment in green. The all-atom cRMS between the two fragments is 0.87Å. The fragment is largely correct, apart from residues A259 (Serine) and A260 (Arginine) being flipped. The electron density map is shown contoured at 1.0 σ.

the final run, four out of the final twelve fragments assumed configuration 'A', another three assumed conformation 'B', and the remaining five fragments did not fit the density meaningfully. Figure 2.7(a) shows the two alternative conformations for residues A316-A323. Side chains were added manually to the poly-alanine chains. Figure 2.7(b) and (c) show residues A316 to A320 of both conformations in the electron density. The fit of Tyrosine A318 is particularly telling in each case.

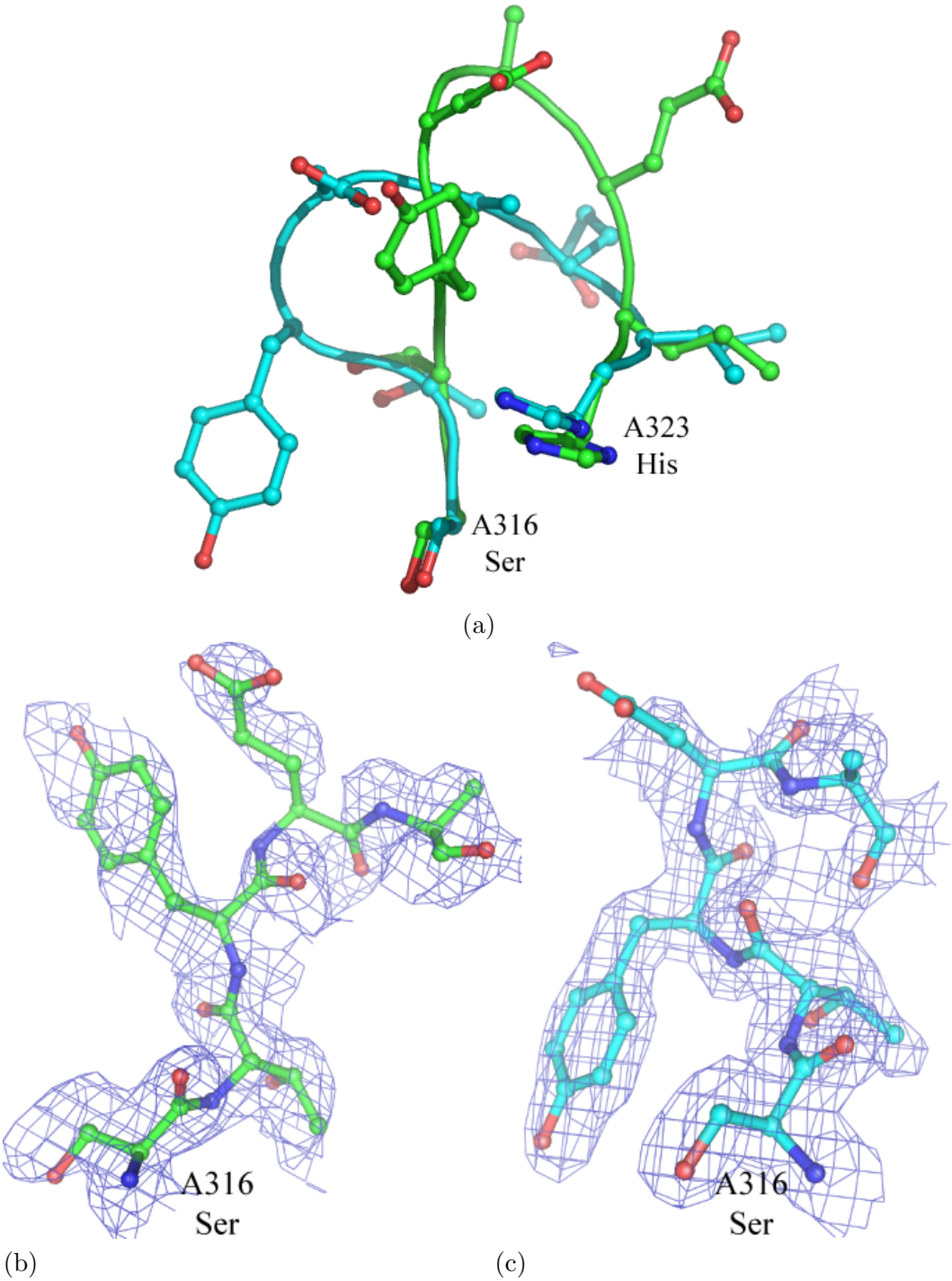


Figure 2.7: (a) The two alternative conformations for residues A316-A323 in the hypothetical protein TM0755. A total of 33% of fragments output by the algorithm converged to conformation 'A' (in green), while another 25% assumed conformation 'B' (in cyan). Side chains were added manually. (b) Residues A316-A320 of conformation 'A'. (c) Residues A316-A320 of conformation 'B'. For clarity, residues A321-A323 are omitted in figures (b) and (c).

Chapter 3

Approximation of Protein Structure for Fast Similarity Measures*

3.1 Introduction

Automatic protein structure comparison is an important problem in computational structural biology, e.g., in structural databases and classification [66, 86, 146, 152], structure prediction [54, 57, 108, 156, 176, 178], analysis of trajectories through conformational space generated by molecular dynamics and Monte Carlo simulations [116, 174, 214], graph-based methods for evaluating ensemble properties [8, 9, 10, 11], etc.

In contrast to sequence matching, structural matching requires a similarity measure that is based on spatial atom coordinates. Nevertheless, it is still important whether the structures that are compared have the same amino acid sequence, or not. For conformational samples from the same sequence there are no ambiguities about correspondences. In this case, similarity measures such as cRMS or dRMS are commonly used [107]. These measures are defined as the root mean square (RMS) of either distances between corresponding atoms in the two compared structures (cRMS) or their corresponding intra-molecular distance matrix entries (dRMS). Comparing protein structures that derive from different sequences is more difficult because it is generally not obvious which features (atoms) of one structure

*This chapter describes work done jointly with Fabian Schwarzer and published in [127, 168]

correspond to which features from the other structure. Moreover there is a tradeoff between the length of the correspondence that is chosen and the quality of the similarity that results. Numerous methods for finding a set of correspondences have been proposed, e.g., [58, 65, 70, 85, 107, 114, 173, 191].

Many similarity measures are based on a rather fine granularity feature selection. Typically, the coordinates of all Cα atoms and sometimes even those of additional atom centers are considered. For large proteins, the number of considered features greatly affects the efficiency of the comparison. For example, the size of the intra-molecular distance matrices and the complexity of the dynamic programming algorithm in [65] are quadratic in the number of residues. While this is not a real problem when comparing a single pair of structures (many proteins have less than 1000 residues), it becomes an important issue when querying a large database for similar structures or clustering a large set of conformations.

In this chapter, it is shown that the description length of a protein conformation can be reduced while introducing only a small error in the computed similarity measure. The backbone is uniformly sub-divided into contiguous sub-chains of fixed length and represent each of the sub-chains by the average coordinates of its atom centers. Similarity measures can then be computed on these "averaged conformations." Although this simplification introduces some error, theoretical and experimental evidence is provided, demonstrating that enough information about relative similarity is retained to discriminate structures in practical applications. In particular, the derived similarity measures using the averaged protein representation are highly correlated to their original full-atomic counterparts. If high accuracy is a concern, approximate similarity measures are still useful as a fast filter to considerably reduce the number of pairs that need to be passed to the exact similarity measure.

While we cannot give bounds on the error that is introduced, it is shown through wavelet analysis of protein structures and random chains that averaging is a reasonable method for reducing the dimensionality of structure descriptors. Our analysis reveals two properties of proteins, which make averaging work. The first property is the chain topology of proteins, which forces atoms that are nearby along the chain to also be spatially close in any conformation the protein chain assumes. The second property is the fact that van der Waals forces limit the compactness of the protein structures. Since atoms can only have small overlap, on average the positions of atoms, which are far apart along the chain, are also spatially distant.

Reducing the computational complexity of similarity measures significantly accelerates many tasks that involve structural matching. In our experiments, we observed decreases in running times by large factors, typically from days to hours or even minutes. For very large sets of proteins, both the efficiency of structure comparison of a single pair and the number of such pairs that are actually evaluated are important. Many approaches require evaluation of all pairs ("brute-force approach") even if the task is to identify only a small constant number of nearest neighbors for each conformation in the set. Using our averaged representation, we show that we can avoid this quadratic cost of examining all pairs in a k nearest-neighbor application. In automatic structural classification, the complexity of a previous algorithm [65, 188] that matches pairs of structures grows quadratically with the number of residues. In this case as well, a small reduction in the number of features results in substantial savings.

3.2 Shape similarity and approximation of chains

Given two sequences of points in 3-space $P = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$ and $Q = (\mathbf{q}_1, \ldots, \mathbf{q}_n)$, there are two RMS measures that can be used to compute the similarity between them: cRMS and dRMS. They were both defined in Subsection 1.1.4. When applying these similarity measures to proteins, it is common practice to use the Cα atom centers, ordered along the backbone, as defining points. (Sometimes, additional atoms are included or C$_\beta$ atoms are used instead.) Due to the special geometry of proteins the positions of these atoms completely determine the shape of the backbone. However, in the case of dRMS, the intra-molecular distance matrices grow quadratically with the length of the protein, which significantly slows down the dRMS computation for large proteins.

3.2.1 Approximate similarity measures

We reduce the number n of sample points in P and Q as follows. In each sequence, we replace contiguous subsequences of points by their centroids. That is, we uniformly partition the sequence P of length n into contiguous subsequences of length m each. (If n/m is not an integer, some subsequences will be chosen to be longer by one.) For each subsequence, we then replace its points by their centroid, which we denote by $\overline{\mathbf{p}}_j$ for subsequence j. For example, if subsequence j spans points $(\mathbf{p}_a, \ldots, \mathbf{p}_b)$ where $b - a + 1 = m$ then

$$\overline{\mathbf{p}}_j = \frac{1}{m} \sum_{i=a}^{b} \mathbf{p}_i. \tag{3.1}$$

Based on these averaged subsequences we define the m-averaged representation $\overline{P}_m$ of P as the sequence of $r = \lfloor n/m \rfloor$ points $(\overline{\mathbf{p}}_1, \ldots, \overline{\mathbf{p}}_r)$. (For Q, we proceed in the same way.) We can now define the simplified RMS measures for $\overline{P}_m$ and $\overline{Q}_m$ analogously to the above RMS measures on the original sequences. That is, in the defining formulas (Equations 1.1, 1.2 and 1.3), we replace $\mathbf{p}_i$ ($\mathbf{q}_i$) by $\overline{\mathbf{p}}_i$ ($\overline{\mathbf{q}}_i$) and n by r. We call these measures *m-averaged measures* and denote them by $\overline{c}_m RMS$ and $\overline{d}_m RMS$.

Obviously, the error of these simplified similarity measures continuously approaches zero as the two compared point sets P and Q become more similar, i.e., $\lim_{Q \to P} |\overline{c}_m RMS(P,Q) - cRMS(P,Q)| = 0$ and the same holds for $\overline{d}_m RMS$. For general point sets, the error introduced by this approximation can be quite substantial. However, for proteins the error is small because of their chain topology and the limited compactness of their conformations (due to van der Waals forces). In the following section, we explain why this is the case using random chains and wavelets analysis.

3.2.2 Random chains and Haar wavelets

We use random chains and the Haar wavelet transform to argue that averaging is a reasonable method for reducing the size of the representation of a protein for computing structural similarities.

A random chain $C = (\mathbf{c}_0, \ldots, \mathbf{c}_{n-1})$ in 3-D is an ordered set of points in space defined as follows:

$$\begin{aligned} \mathbf{c}_0 &= \mathbf{0}, \\ \mathbf{c}_{i+1} &= \mathbf{c}_i + \mathcal{S}_2 \cdot l \quad i = 0, \ldots, n-2 \end{aligned} \tag{3.2}$$

where $\mathbf{0}$ is the origin, $\mathcal{S}_2$ is a random 3-D vector uniformly distributed on the unit 3-D sphere, and l is the fixed Euclidean distance between two consecutive points of the chain. $\mathcal{S}_2$ is sampled as follows:

$$\mathcal{S}_2 = \begin{bmatrix} \sin\phi\cos\theta \\ \sin\phi\sin\theta \\ \cos\phi \end{bmatrix} \tag{3.3}$$

where $\theta \sim U[0, 2\pi]$ and $\cos\phi \sim U[-1, 1]$.

Computing the covariance matrix of $\mathcal{S}_2$ reveals that the off-diagonal elements are identically 0, and, as a result, the three dimensions of each random step are uncorrelated. Since each step is independent of all other steps, the distributions of the three dimensions of any point on the chain are uncorrelated. Consequently, the random chain described above can be approximated well by replacing $\mathcal{S}_2$ with a 3-vector sampled from the normal distribution $\mathcal{N}(\mathbf{0}, \frac{1}{3} \cdot \mathcal{I})$, where $\mathcal{I}$ is the 3×3 identity matrix, for sufficiently large values of n (typically greater than 10). Moreover, the fact that the three dimensions are uncorrelated enables us to perform three independent one-dimensional Haar wavelet transforms as described below, instead of the more complicated three-dimensional transform.

The Haar wavelet transform of a chain is a recursive averaging and differencing of the coordinates of the points. The transform recursively smoothes the chain while keeping the *detail* coefficients needed to reconstruct the full chain from the smoothed out version. We define the full resolution chain to be of level 0: $C = C^0$. We recursively create smoothed versions of the chain by averaging pairs of consecutive points:

$$\mathbf{c}_i^j = \frac{1}{\sqrt{2}} \left(\mathbf{c}_{2i}^{j-1} + \mathbf{c}_{2i+1}^{j-1} \right) \quad \begin{cases} j = 1, \ldots, \log n \\ i = 0, \ldots, \frac{n}{2^j} - 1 \end{cases} . \tag{3.4}$$

As each level of resolution is created, we also compute the *details* that are smoothed out by the averaging:

$$\mathbf{d}_i^j = \frac{1}{\sqrt{2}} \left(\mathbf{c}_{2i}^{j-1} - \mathbf{c}_{2i+1}^{j-1} \right) \quad \left\{ \begin{array}{l} j = 1, \ldots, \log n \\ i = 0, \ldots, \frac{n}{2^j} - 1 \end{array} \right\} \tag{3.5}$$

Note that the averages and details are multiplied by a scale factor of $\sqrt{2}$ at each level. Given C^j, the smoothed chain at level j, and D^j, the detail coefficients of level j, it is possible to reconstruct C^{j-1} exactly by inverting the formulas of Equations 3.4 and 3.5. The Haar wavelet transform of a chain C is thus defined as:

$$\hat{C} = \left(C^{\log n}, D^{\log n}, D^{\log n - 1}, \ldots, D^1 \right). \tag{3.6}$$

The length of $\hat{C}$ is the same as C and $C^{\log n}$ is the centroid of the entire chain. Since C can be exactly reconstructed from $\hat{C}$, no information is lost in the transform. This representation can then be compressed by setting to 0 all coefficients in $\hat{C}$ smaller than some threshold.

Since the coefficients are scaled, the square of the L_2 error of approximation in this case would be equal to the sum of the squares of the removed coefficients [187].

Given the normal approximation of the random chain construction, we can analytically determine the *pdf* (probability density function) of each of the coefficients in $\hat{C}$ by adding, subtracting and scaling independent normally distributed variables. The *pdf* of each detail coefficient in level j is:

$$\mathbf{d}^j \sim \mathcal{N}\left(\mathbf{0}, \frac{4^j+2}{36} \cdot \mathcal{I} \cdot l\right), \tag{3.7}$$

and the *pdf* of their squared L_2 norm is thus:

$$\|\mathbf{d}^j\|_2^2 \sim \chi^2(3dof) \cdot \frac{4^j+2}{36} \cdot l \tag{3.8}$$

with a mean of $\frac{4^j+2}{12} \cdot l$ and variance of $\frac{(4^j+2)^2}{216} \cdot l^2$. Since the *pdf*s of the detail coefficients are centered at the origin and their variance is decreasing by a factor of roughly 4, they are expected to be ordered (in absolute value) in $\hat{C}$, from largest to smallest. More precisely, the detail coefficients of level i are expected to be larger by roughly a factor of 4 than the detail coefficients of level $i-1$, and this holds true for all levels of the transform. Note that the $\sqrt{2}$ scaling during the construction of the coefficients would account for an average growth of a factor of 2 in their variance from one level to the next. The special geometry of random chains accounts for the second factor of 2. Hence, as a general policy, it is best to remove coefficients starting at the lowest level and climbing up. These coefficients have the lowest variance and thus contain the least information for determining structural similarity. The effect of averaging as described in Section 3.2.1 for $m = 2^v$ is the same as that of removing the lowest v levels of coefficients. Since these are expected to be the smallest coefficients, we can conclude that using an m-averaged chain will give the smallest expected error for a representation that uses only m Haar detail coefficients for each dimension. The wavelet analysis allows us to estimate the approximation mean squared error introduced by removing all coefficients of level j. It can be computed to be $\frac{l}{12}\left(2^i + \frac{1}{2^{i-1}}\right)$. Therefore for an m-averaged approximation the mean squared error is expected to be on the order of $(m \cdot l)/6$.

As the above analysis shows, we can remove quite a few levels of coefficients without introducing a large error. This is due to the large *ratio of the average variances* (henceforth called RAV) between two successive levels of detail coefficients, which was shown to be

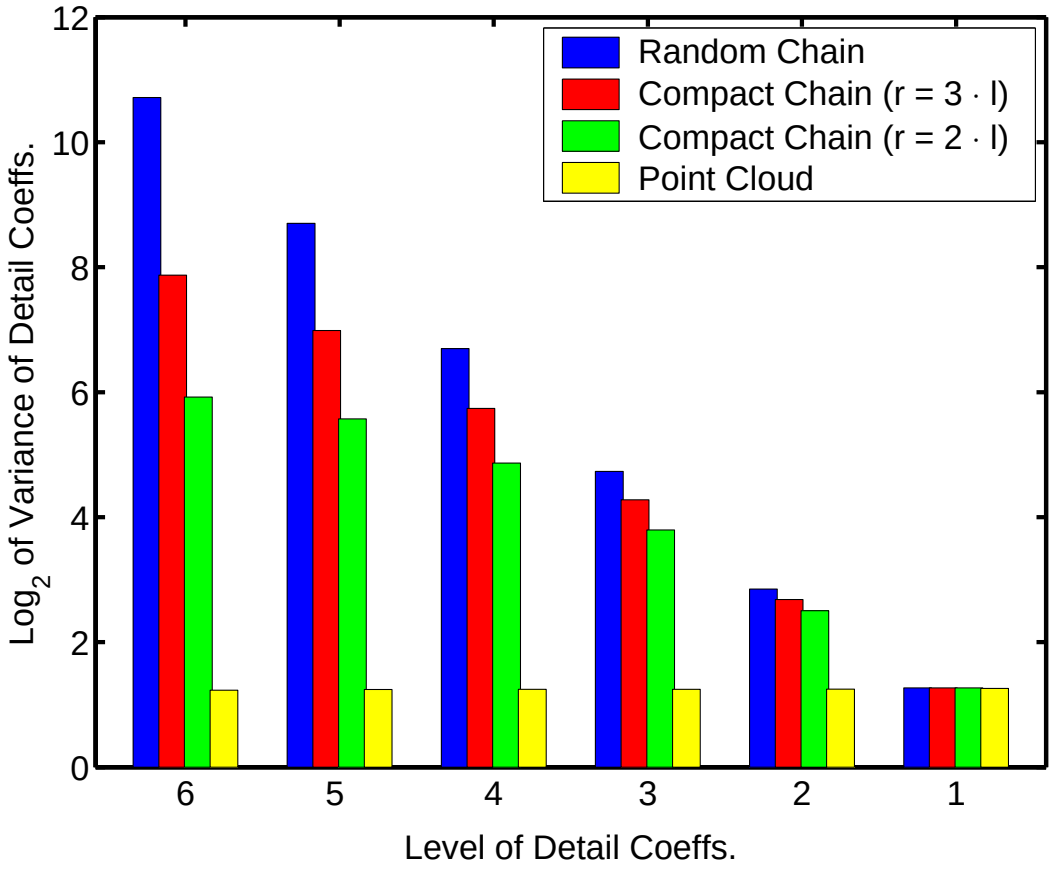


Figure 3.1: Average variance of the Haar detail coefficients for random chains, compact random chains and random point clouds.

approximately 4 for all levels. This behavior of the Haar detail coefficients is a result of the fact that on the average random chains grow further and further away from their starting point. The expected distance of the nth point from the origin is on the order of $\sqrt{n} \cdot l$. We see this behavior in Figure 3.1 where we compare the average variances of the coefficients of random chains to those of very compact random chains (chains forced to reside inside small spheres) and to those of point clouds sampled randomly from inside a sphere of radius $\sqrt{n} \cdot l$. All chains are of length $n = 64$. The RAV of the unconstrained random chain is significantly larger than that for the compact random chains, which decreases as the compactness of the chain is increased. The worst case is for a point cloud, where the average variances of all levels of detail coefficients have the same magnitude making all levels have the same importance.

3.2.3 Application to proteins

While it is a well-known fact that the positions of neighboring Cα atoms along the backbone of native protein structures are highly correlated (e.g., see [110]), it was shown in [99] that treating the position of each Cα atom as uniformly distributed on a 3-D sphere centered at the center of the previous Cα atom yields a very good approximation of the average global behavior of native protein structures.

We performed the same wavelet transform on sets of conformations of actual proteins of length 64 residues (only the Cα atoms coordinates were used to describe the conformations) taken from *decoy sets* (conformations which are expected to be similar to the native conformation) generated by Park and Levitt [155, 156], containing 10,000 conformations each. We obtained results similar to those of random chains, namely the detail coefficients are ordered and have a RAV similar to that of random chains. The results for a few sets of proteins are presented in Figure 3.2. One important difference from random chains is that the RAV for decoy sets decreases considerably for the top levels. We explain this as follows. Small pieces of a protein cannot be highly packed because of steric clashes, while intermediate size pieces are often elongated (secondary structure helices and strands) and hence the variance of the coefficients grows considerably from one level to the next at the low and intermediate levels. The tight packing of the secondary structure of the native-like conformations makes the high-level coefficients considerably smaller than in the random chain model. We would have liked to give results for decoy sets of proteins significantly longer than 64 residues, however, no such sets were available to us.

Random protein conformations (generated as described in Section 3.2.4), on the other hand, are considerably less compact than the decoy sets, and hence behave much more like random chains at *all* levels of detail coefficients. As can be seen in Figure 3.2 the RAV at the lower levels is even bigger than what is observed for random chains. This is due to the limit on the packing density as a result of the space taken up by each residue. In our random chain model the points have no volume and the chain is allowed to cross itself. In the random protein conformations, however, atoms are not allowed to have any overlaps. Their behavior is actually modeled better by random chains in which the next step is sampled uniformly from a *hemisphere* defined by the direction of the previous step.

We thus conclude that, while decoy sets cannot be compressed as much as random sets, it is possible to remove the first few levels and still get a very good approximation.

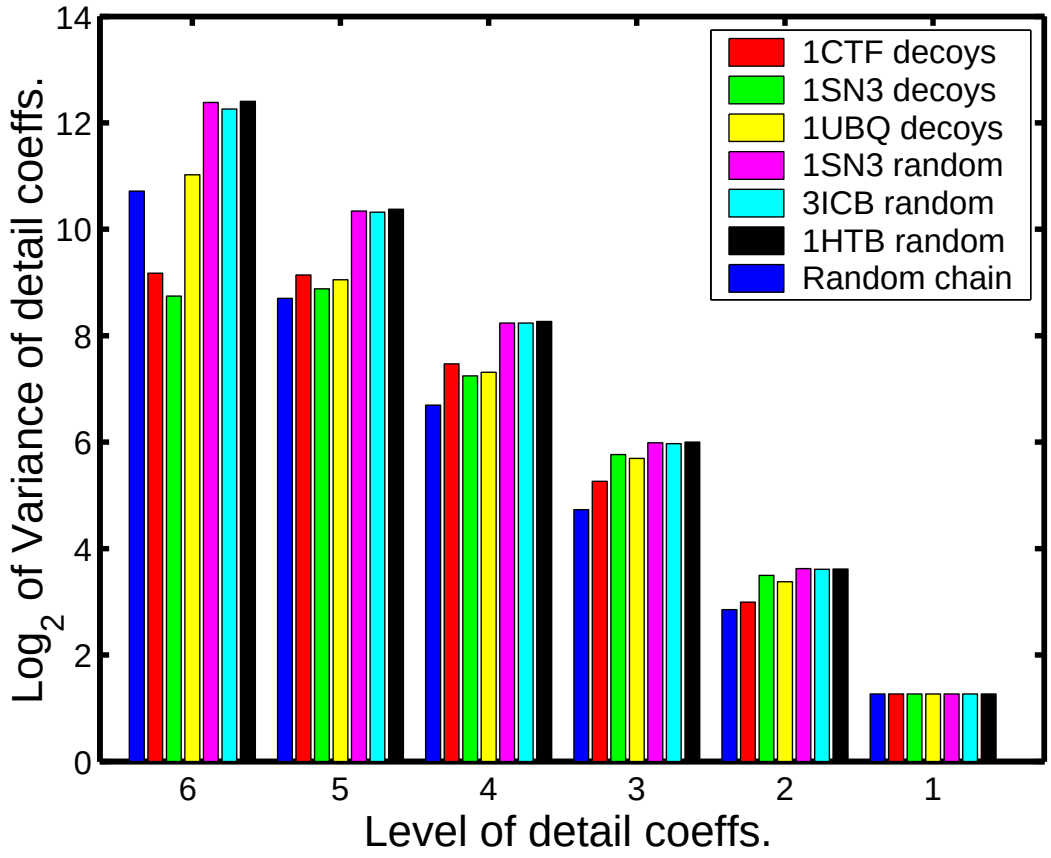


Figure 3.2: Average variance of the Haar detail coefficients for random chains, decoy sets and randomly generated conformations.

3.2.4 Correlation of approximate and exact similarity measures

When using the cRMS measure to compute similarity between random chains of length 64 we find that the approximate m-averaged versions yield very good results for m as large as 9. For $m = 4, 6, 9, 12$ and 16, the correlation of the approximate measure to the true one is 0.99, 0.95, 0.89, 0.76 and 0.60, respectively. When using the dRMS measure to compute similarity between random chains the approximate m-averaged versions is highly correlated for m as large as 6. The correlation values obtained are 0.94, 0.81, 0.67, 0.55 and 0.46, respectively.

In order to verify that the analogous behavior of the detail coefficients of protein sets and random chains carries over to approximate similarity measures we chose 8 structurally diverse proteins used by Park and Levitt in [155, 156] (1CTF, 1R69, 1SN3, 1UBQ, 2CRO,

3ICB, 4PTI, 4RXN) having between 54 (4RXN) and 76 (1UBQ) residues. For these proteins we obtained (1) decoy sets generated by Park and Levitt [155, 156], containing 10,000 conformations each and (2) randomly generated conformation sets using the program FOLDTRAJ[1] [57], containing 5000 structures each. A decoy set is a collection of conformations, which are native-like in structure. As a result they span only a small portion of the entire conformation space and are thus expected to have significant pairwise similarity. On the other hand, the conformations in the random sets sample a much larger space and as a result have on average lower pairwise similarity.

For each set, we randomly chose between 1000 and 4000 pairs whose true dRMS distance was less than 5Å (10Å for the random sets which have larger structural diversity) and computed their m-averaged distances for different values of m. The correlation of the m-averaged cRMS and dRMS measures to the true cRMS and dRMS measures for the different decoy sets can be found in Table 3.1(a). For m as large as 9, the approximate cRMS measure is still highly correlated with the true cRMS measure, which means that a reduction factor of 9 still yields a very good approximation for proteins of this size. For dRMS, high correlation is still achieved when $m = 6$, which means a reduction factor of 6 (since the complexity of dRMS is quadratic, the actual gain is by a factor of 36). We note that the correlation values obtained are quite similar to those computed for random chains.

The correlation of the m-averaged cRMS and dRMS measures to the true measures for the different random sets can be found in Table 3.1(b). Here too, a reduction factor of 9 still yields a highly correlated approximation of the cRMS measure. For dRMS, high correlation is still achieved when $m = 9$, which means a reduction factor of 9 and an actual gain of 81. Here the correlation values are in fact better than those computed for random chains as would be anticipated from the higher growth ratio of the variance of the detail coefficients of random protein sets in comparison to those of random chains (see Figure 3.2). When examining all pairs and not only those whose dRMS distance is smaller than 5Å (10Å for the random sets), the computed correlation is even larger.

The analysis of random chains in the previous section entails that the approximation error caused by averaging depends only on the averaging factor m. The better correlation we get for the random sets over the decoy sets thus requires some explanation. First, the larger RAV of the random sets means that the low-level coefficients are less important than those in decoy sets. Second, the distance between a pair of random conformations is larger

[1] `http://bioinfo.mshri.on.ca/trades/`

	1CTF		1R69		1SN3		1UBQ		2CRO		3ICB		4PTI		4RXN	
m	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*
3	0.99	0.97	0.99	0.96	0.99	0.98	0.99	0.98	0.99	0.98	0.99	0.97	0.99	0.97	0.99	0.98
4	0.99	0.95	0.99	0.95	0.99	0.97	0.99	0.97	0.99	0.97	0.99	0.95	0.98	0.94	0.99	0.96
6	0.98	0.91	0.98	0.91	0.97	0.90	0.98	0.91	0.99	0.93	0.99	0.91	0.96	0.87	0.92	0.78
9	0.86	0.71	0.96	0.82	0.89	0.73	0.83	0.71	0.95	0.87	0.98	0.96	0.90	0.72	0.81	0.65
12	0.81	0.67	0.84	0.64	0.71	0.54	0.75	0.52	0.86	0.66	0.92	0.69	0.74	0.59	0.54	0.57
16	0.39	0.46	0.66	0.47	0.55	0.44	0.58	0.48	0.75	0.53	0.81	0.54	0.42	0.46	0.49	0.49

(a)

	1CTF		1R69		1SN3		1UBQ		2CRO		3ICB		4PTI		4RXN	
m	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*
3	0.99	0.99	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99
4	0.99	0.98	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.98
6	0.99	0.96	0.98	0.96	0.98	0.95	0.99	0.95	0.98	0.95	0.99	0.95	0.98	0.96	0.99	0.97
9	0.94	0.87	0.95	0.89	0.95	0.88	0.95	0.85	0.95	0.87	0.96	0.86	0.92	0.88	0.93	0.90
12	0.91	0.81	0.84	0.76	0.84	0.75	0.90	0.73	0.84	0.74	0.90	0.73	0.86	0.81	0.87	0.84
16	0.70	0.62	0.69	0.64	0.68	0.61	0.82	0.61	0.71	0.62	0.83	0.64	0.73	0.70	0.73	0.74

(b)

Table 3.1: The correlation coefficient for different m values evaluated for cRMS and dRMS of (a) decoy sets and (b) randomly sampled conformations of various proteins.

than the distance between a pair of decoy conformations and, as a result, the approximation errors have a smaller effect on the computation of similarity.

We wanted to see how well averaging scales up as we increase the size of the protein. As we did not have access to decoy sets of larger proteins we examined only random conformation sets. We generated 1000 random conformations for the two proteins 1LE2 (144 residues) and 1HTB (374 residues). We chose 1100 pairs of 1LE2 conformation with a dRMS less than 18Å, and 1100 pairs of 1HTB conformations with dRMS less than 24Å. For 1LE2 the correlation of cRMS was above 0.9 for m as large as 27, and for dRMS it was above 0.9 for m as large as 15. For 1HTB the correlation of cRMS was above 0.9 for m as large as 40, and for dRMS it was above 0.9 for m as large as 30. Because the pairwise distances for these larger proteins were significantly larger than the distances of the smaller proteins, a larger approximation error can be tolerated without compromising performance. Thus larger values of m could generally be used for larger proteins, making the speed-ups that can be achieved larger when they are most needed.

3.3 Application 1: Nearest-neighbor search

Simulations and other conformational sampling methods generate large sets of conformations of a particular protein. For example, the project Folding@Home[2] runs parallel molecular dynamics simulations on thousands of computers across the Internet and then centrally analyzes the obtained data. An important step in evaluating such data, e.g. for clustering and related tasks, is the following: given a set of conformations of the same protein, find the k nearest neighbors (NNs) for each sample in the set. Typically, k is a small constant while the size of the set can be very large.

The straightforward ("brute-force") approach is to evaluate the similarity measure (cRMS or dRMS) for all pairs and then report the k NNs for each sample. However, the quadratic complexity makes this approach scale poorly. Spatial data structures such as the *kd-tree* [14] can avoid this complexity under certain circumstances [5, 32, 89, 105, 134, 161]. Note that these data structures allow for exact search, i.e., they return the same NNs as would the brute-force search. However, most of them require a Minkowski metric space of rather small dimensionality. Unfortunately, cRMS is not a minkowski metric. Although dRMS is a Minkowski metric, the dimensionality of the space of intra-molecular distance matrices is far too high, since typically, for dimensions higher than a few tens, none of the nearest-neighbor data structures performs better than brute-force search. Therefore, in order to use a spatial data structure to speed up NNs search, we chose to use the dRMS measure, but find a way to significantly reduce the dimensionality of the structure descriptors below the averaged conformations we presented in Section 3.2.

3.3.1 Further reduction of distance matrices

We use the singular value decomposition (SVD) [71] to further compress the intra-molecular distance matrices of averaged proteins, that is to further reduce the number of parameters involved in computing $\overline{d}_m RMS$. SVD is a standard tool for principal components analysis (PCA) and computes directions of greatest variance in a given set of high-dimensional points. When considering the set of intra-molecular distance matrices these directions correspond to directions that contain the most dissimilarity information. These directions are called principal components (PCs). The SVD can be used to linearly map a set of high-dimensional input vectors (data points), stored in a matrix A, into a lower-dimensional

[2] `http://folding.stanford.edu`

subspace while preserving most of the variance. Such a transform can be found by decomposing the matrix A of the input vectors into $A = USV^T$, the SVD of A, where U and V are orthogonal matrices and S is diagonal with the singular values of A along the diagonal.

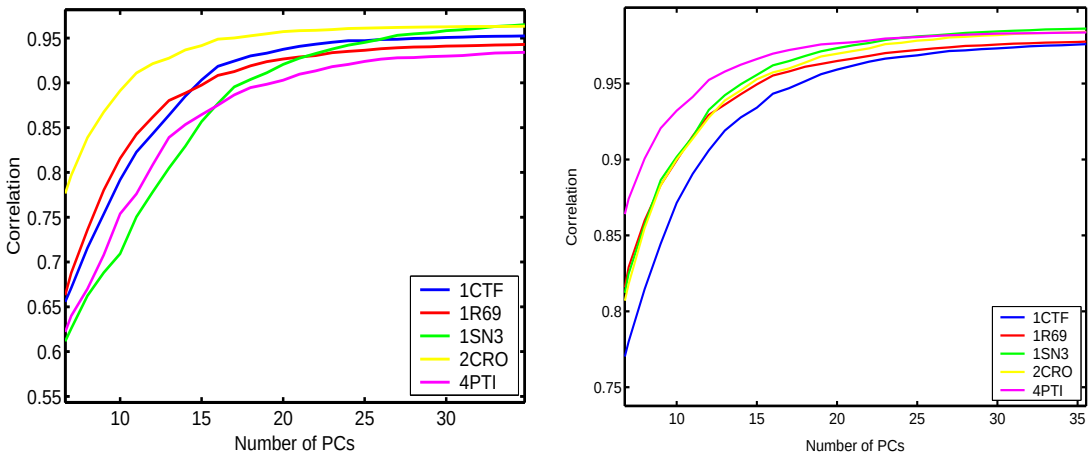
Our reduction algorithm has two steps:

1. Compute an m-averaged representation of each conformation in the set.
2. Use the SVD of the entire set to further reduce the description length of each conformation.

Efficient algorithms exist that compute the SVD in time $O(s^2t)$ where s is the smaller and t the larger dimension of A (rows or columns). Note that while in principle, SVD could be applied to intra-molecular distance matrices without first averaging sub-chains, the quadratic dependency on the smaller dimension of A shows the important advantage of averaging: usually, the larger dimension t will reflect the size of the conformational sample set while the smaller dimension s will correspond to the size of a single intra-molecular distance matrix. Reducing the distance matrix size by using averaged conformations, as described in Section 3.2, is therefore key to performing SVD in practice.

To perform SVD on a set of intra-molecular distance matrices derived from m-averaged conformations, each of these distance matrices is rewritten as an $r(r-1)/2$ dimensional vector (where $r = \lfloor n/m \rfloor$) and then SVD is applied to the matrix that contains all these vectors. Taking the resulting U matrix and removing all columns that correspond to small singular values (the directions of little variance), we have the linear map that takes the set of distance matrices into a lower-dimensional Euclidean space while preserving a high percentage of the variance and thus distance information. We found that in practice, a relatively small output dimensionality between 12 and 20 is sufficient to maintain about 90% of the variance of the distance matrices. In what follows, we will denote the dRMS measure obtained from an SVD compressed set of m-averaged distance matrices by $\overline{d}_m^{PC}RMS$. (PC stands for the number of principal components that are used after compression.)

Since the cost of the SVD is quadratic in the length of the protein (assuming a very large set of conformations), its dependence on m, the averaging factor, is quartic. Therefore, one would like to use a large m when averaging. On the other hand, averaging is lossy, and thus it would be best to apply SVD to a representation that has the least amount of averaging possible. Figure 3.3 shows the effectiveness of the SVD for reducing the description length of protein conformations. Figure 3.3(a) presents the correlation between dRMS and $\overline{d}_4^{PC}RMS$



(a) (b)

Figure 3.3: Correlation between dRMS and $\overline{d}_4^{PC}RMS$ for different number of PCs on (a) decoy sets and (b) random sets of some proteins.

on some of the decoy sets as the number of PCs that is used increases. With as little as 16 PCs a correlation of 0.9 is achieved. Figure 3.3(b) presents the correlation on some of the random sets as the number of PCs that is used increases. Here as little as 12 PCs suffice to get a correlation of 0.9. For comparison, we present in Figure 3.4 the correlation when SVD is applied after averaging with a factor of $m = 9$. Here the same number of PCs yields lower correlation than was achieved for $m = 4$. The computation time would be more than 16 times faster though.

3.3.2 Evaluation of approximation errors for nearest-neighbor search

Using an approximate measure to find the k NNs of a conformation entails that some true NNs will be missed while other conformations would be chosen instead, which are not true NNs. As described above, the approximate measure we propose to use is $\overline{d}_m^{PC}RMS$. We tested the usefulness of this measure on both the decoy sets and the random sets used in Subsection 3.2.4.

Given a set of conformations S and a query conformation q from that set, we define two orderings of the elements in S: S_1 and S_2. S_1 is the ordering of the conformations in S by their exact dRMS distance from q. S_2 is ordered by the $\overline{d}_m^{PC}RMS$ distance of the conformations in S from q. Thus, S_2 is the approximation of S_1 using our reduced similarity

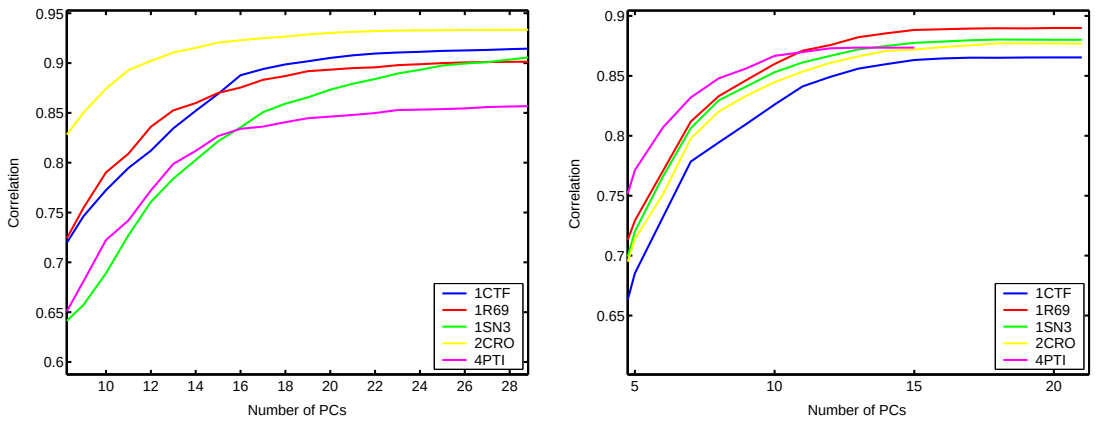


Figure 3.4: Correlation between dRMS and $\overline{d}_9^{PC} RMS$ for different number of PCs on (a) decoy sets and (b) random sets of some proteins.

measure. In our test we used $m = 4$ for both random and decoy sets. We used 16 PCs for the decoy sets but only 12 for the random sets. Based on these two sets we compute two error measures and two parameters to help evaluate the performance of the approximate similarity measure for the k NNs application.

The two error measures we use are:

$\mathbf{err_1}$ Given the dRMS distance of the kth conformation in S_2 (the approximate kth NN) and the dRMS distance of the kth conformation in S_1 (the true kth NN), how much is the former larger than the latter. This error is reported as percentage of the dRMS distance of the kth conformation in S_1.

$\mathbf{err_2}$ Given the average dRMS distance of the first k conformations in S_2 (the average distance of an approximate NN) and the average dRMS distance of the first k conformations in S_1 (the average distance of a true NN), how much is the former larger than the latter. This error is reported as percentage of the average dRMS distance of the top k conformations in S_1

We also computed two other parameters that are indicative of how well the approximate measure finds true NNs:

NN1 The number of elements among the top k of S_1 found in the top k of S_2. In other words, the number of true NNs found by the approximate measure.

	$k = 10$				$k = 25$				$k = 100$			
	err_1	err_2	*NN1*	$NN2$	err_1	err_2	*NN1*	*NN2*	err_1	err_2	*NN1*	*NN2*
1CTF	11%	3%	7.8	20	12%	3%	19	67	10%	2%	81	260
1SN3	14%	5%	7.5	23	15%	4%	18	69	12%	2%	76	301
1UBQ	15%	7%	7.0	38	14%	5%	18	102	13%	3%	77	332
2CRO	12%	4%	8.3	15	13%	3%	20	48	11%	2%	82	212
3ICB	13%	5%	7.8	23	13%	4%	19	68	12%	2%	78	292
4PTI	15%	6%	7.5	24	17%	5%	18	80	14%	3%	73	343
4RXN	13%	5%	7.9	19	16%	4%	19	62	16%	3%	78	295

Table 3.2: Mean values of err_1, err_2, *NN1* and *NN2* for 250 queries of k nearest neighbors using the decoy sets.

NN2 The maximal index in S_2 of an element from the top k in S_1. In other words, how many approximate NNs do we need to find in order to have all the true NNs.

We first looked at the decoy sets of size 10,000 for each of the eight proteins. We used $m = 4$ and 16 PCs ($\bar{d}_4^{16} RMS$ measure) to find approximate NNs. For each set, we randomly chose 250 query conformations and evaluated err_1, err_2, *NN1* and *NN2* for each of them. Table 3.2 shows the mean values of these parameters over the 250 queries (standard deviations were generally small).

The err_1 error was about 15% for all proteins, which translates to an error of no more than 1.5Å in the furthest NN that is returned. The err_2 error is always smaller than 7%, which means that on average the ith approximate nearest neighbor is no more than 0.7Å further away than the ith true NN. The *NN1* parameter reveals that at least 70% of the true NNs are returned by the approximate measure, and the *NN2* parameter indicates that almost always it is enough to find $3k$ approximate NNs in order to guarantee that all true k NNs are found.

Next we looked at the random sets, each containing 5000 conformations and performed the exact same calculations as we did for the decoys. However, here we used $m = 4$ and 12 PCs ($\bar{d}_4^{12} RMS$ measure) to find approximate NNs. The results are shown in Table 3.3. Here the err_1 error was less than 10%, which means the approximate furthest NN was less than 1.5Å further away than the true furthest NN. Note that the random sets cover a larger part of the conformation space and thus the distance between a pair of conformations would be larger than what we find for decoy sets, which span only a small portion of conformation space. The err_2 error is in general smaller than 3% which translates to about 0.5Å. While here the approximate method found somewhat fewer true nearest neighbors when $k = 10$ than for the decoy sets (*NN1* ≥ 6.5) its performance was equivalent to the decoy sets for

	$k = 10$				$k = 25$				$k = 100$			
	err_1	err_2	*NN1*	*NN2*	err_1	err_2	*NN1*	*NN2*	err_1	err_2	*NN1*	*NN2*
1CTF	8%	3%	6.8	20	8%	2%	18	61	9%	1%	82	203
1SN3	8%	3%	6.8	23	9%	2%	18	62	9%	1%	82	197
1UBQ	8%	3%	6.9	38	9%	2%	18	60	9%	<1%	83	190
2CRO	6%	2%	7.8	15	6%	1%	21	41	5%	1%	91	134
3ICB	9%	3%	6.6	23	10%	2%	18	67	9%	1%	81	217
4PTI	8%	3%	6.9	24	9%	2%	19	58	8%	1%	83	194
4RXN	8%	3%	6.9	19	9%	2%	19	58	9%	1%	82	198

Table 3.3: Mean values of err_1, err_2, *NN1* and *NN2* for 250 queries of k nearest neighbors.

	$k = 10$				$k = 25$				$k = 100$			
	err_1	err_2	*NN1*	*NN2*	err_1	err_2	*NN1*	*NN2*	err_1	err_2	*NN1*	*NN2*
1CTF decoys	20%	9%	7.1	39	22%	8%	17	132	20%	5%	67	571
2CRO decoys	22%	10%	7.1	43	24%	9%	17	156	21%	6%	65	656
1CTF random	15%	7%	4.6	85	16%	6%	12	192	17%	4%	60	615

Table 3.4: Mean values of err_1, err_2, *NN1* and *NN2* for 250 queries of k nearest neighbors.

$k = 25$ and superior ($NN1 \geq 81$) for $k = 100$. In general, about $2.5k$ approximate NNs were enough to ensure that all true k NNs are found.

Finally, we looked at some larger sets of 100,000 conformations each. We ran the same test as we did for the smaller conformation sets. Approximate nearest neighbors were computed using the $\overline{d}_4^{16}RMS$ measure for the decoy sets and the $\overline{d}_4^{12}RMS$ measure for the random set. The results are displayed in Table 3.4. In general the results are slightly worse than what we observed for the smaller sets in terms of the error percentages and the number of true NNs found by the approximate measure. This is due to the fact that the SVD is less effective for the larger sets. Namely, more PCs are needed to capture well enough the variance of all the conformations. Indeed when we used $\overline{d}_4^{20}RMS$ instead of $\overline{d}_4^{16}RMS$ for the 1CTF decoys, the results were comparable to what we saw with the smaller decoy set (10,000 conformations). Still the *NN2* statistic is very promising, requiring only about $6k$ approximate NNs to capture all true k NNs.

3.3.3 Running time

We now consider the running times in a concrete nearest-neighbor task: given a set of $N = 100,000$ random conformations of protein 1CTF, find $k = 100$ nearest neighbors for each sample in the set. The reported times in this section refer to a sequential implementation in C running on a single 1GHz Pentium processor on a standard desktop PC.

We first compare the running times for a brute-force (all-pairs) implementation using

N	cRMS	$\overline{c}_4RMS$	dRMS	$\overline{d}_4RMS$
1,000	18.6s	12.4s	31.0s	2.2s
2,000	74.4s	50.0s	137.5s	8.0s
5,000	464.8s	312.0s	759.8s	43.4s
100,000	∼52h	∼35h	∼84h	∼4.8h

Table 3.5: Brute-force search using cRMS vs $\overline{c}_4RMS$ and dRMS vs $\overline{d}_4RMS$

both cRMS and dRMS, and their corresponding averaged similarity measures $\overline{c}_mRMS$ and $\overline{d}_mRMS$. All measures were used to find the $k = 100$ nearest neighbors for each of the N samples. Table 3.5 shows that the latter measures already result in a notable speed-up. For $\overline{d}_4RMS$, a significant speed-up over dRMS is obtained due to the quadratic down-scaling of intra-molecular distance matrices by averaging proteins. For $\overline{c}_4RMS$, the improvement over cRMS is smaller. This is because the reduction by averaging affects the number of involved points only linearly, and the main effort in computing cRMS comes from finding an optimal rigid-body alignment of two point sets.

Note that the increase in running times agrees quite well with the expected quadratic scaling of the brute-force nearest neighbor approach. The times for $N = 100,000$ samples were therefore extrapolated from the actual running times measured for the smaller values of N. In fact, for dRMS using all Cα atom coordinates, we could not store all intra-molecular distance matrices. This problems does not occur with averaged proteins and $\overline{d}_mRMS$.

We next address the quadratic scaling problem of the brute-force approach. To be able to apply a kd-tree, we first further reduced the 120-dimensional space of $\overline{d}_4RMS$ for 1CTF using SVD and retained 16 principal components. This further compression took about one minute for the complete set of 100,000 samples. Building the kd-tree for the resulting 16-dimensional data took only about 4 seconds.

We then ran both the brute-force and the kd-tree approach using $\overline{d}_4^{16}RMS$ as the similarity measure. Table 3.6 shows the running times for finding $k = 1$ and $k = 100$ nearest neighbors for each sample in the full set of 100,000 samples. The obtained total speed-up of our NN search (approximate similarity measures and a kd-tree) over the current best approach (brute-force search using all Cα coordinates) is several orders of magnitude (∼84 hours down to 19 minutes).

In general, the speed-up obtained by using a kd-tree can be expected to increase with increasing sample set size N. Due to its quadratic scaling, brute-force search will become

k	Brute-force	kd-tree
1	30min	4min 10sec
100	41min	19min

Table 3.6: Brute-force vs kd-tree search for k nearest neighbors.

very slow for larger sample sets. On the other hand, the sub-quadratically scaling kd-tree approach should allow to process much larger sample sets within a few hours without parallelization on a standard desktop PC.

3.4 Application 2: structural classification

Given a set of native protein structures each having a different amino acid sequence, such as the Protein Data Bank (PDB)[3] [15, 16], we would like to classify the structures into groups according to their structural similarity. This task has been performed manually in the SCOP (structural classification of proteins) database[4] [146], where protein structures are hierarchically classified into *classes*, *folds*, *superfamilies* and *families*. It has also been done semi-automatically in the CATH[5] [152] database, where protein structures are hierarchically classified into *classes*, *architectures*, *topologies* and *homologous superfamilies*. An automatic classification can be found in the FSSP (Fold classification based on Structure-Structure alignment of Proteins) database[6] [86], where the DALI program is used to classify all structures in the PDB into families of similar structures.

The major difficulty in automatically classifying protein structures lies in the need to decide, given two protein structures, which parts of both structures should be compared, before it can be determined how similar these parts are. There is an inherent trade-off between the length of the compared parts and the level of similarity that is found. The longer the compared parts the less similar the two structures will be, and vice-versa. This *correspondence problem* does not arise when different conformations of the same protein are compared because in that case the correspondence is trivially determined. For this reason, computing the similarity between structures of different proteins requires considerably more computation than the methods described in Section 3.2.

Several algorithms have been proposed for structural classification. Some of the more

[3] `http://www.rcsb.org/pdb/`
[4] `http://scop.mrc-lmb.cam.ac.uk/scop/`
[5] `http://www.biochem.ucl.ac.uk/bsm/cath/`
[6] `http://www.ebi.ac.uk/dali/fssp/`

popular ones are briefly described here. For a survey of other methods see [107].

DALI [7] The internal distances matrices of both proteins are computed. Then all pairs of similar sub-matrices of small fixed size (one from each protein distance matrix) are found and a Monte Carlo algorithm is used to assemble the pairs into larger consistent alignments. For more details see [85].

PROSUP [8] Similar fragments are identified in both proteins. They are iteratively expanded to create alignments. A dynamic programming algorithm is then used to iteratively refine each alignment and finally insignificant alignments are removed. For more details see [114].

CE [9] Each structure is cut into small fragments and a matrix of all possible aligned fragment pairs (AFPs) is created. Combinations of AFPs are selectively extended or discarded leading to a single optimal alignment. For more details see [173].

STRUCTAL [10] The backbones of the two protein structures are directly matched by iteratively cycling between a dynamic programming algorithm that computes optimal correspondence given the current orientation of the two structures, and a least-square fitting that optimally orients the structures to minimize coordinate difference between the corresponding parts. For more details see [65, 188].

3.4.1 STRUCTAL and m-averaging

All the above methods stand to gain in performance by using our averaging scheme. In order to verify this claim, we tested the speed-up and accuracy obtained by using the STRUCTAL method on averaged protein structures. We could not test our approach on PROSUP, DALI and CE because these servers did not accept our averaged structures since they are not valid protein structures. These algorithms were too involved for us to implement ourselves reliably.

[7] `http://www2.ebi.ac.uk/dali`
[8] `http://lore.came.sbg.ac.at/CAME/CAME_EXTERN/PROSUP`
[9] `http://cl.sdsc.edu/ce.html`
[10] `http://bioinfo.mbb.yale.edu/align`

The STRUCTAL algorithm starts with an initial alignment of the backbone Cα atoms of the two structures according to one of a number of possible heuristics (aligning the beginnings, the ends, random segments, by sequence similarity, etc). Then a two step process is repeated until convergence. First, a dynamic programming algorithm analogous to the Needleman and Wunsch sequence alignment algorithm [147] finds the correspondence between the two structures that yields the highest score. Scoring is based on assigning a cost to each possible corresponding pair, which is inversely proportional to the distance between Cα positions, and a gap penalty for every gap in the alignment (for more details see [65]). Computing the best correspondence thus requires $O(n_1 n_2^2 + n_2 n_1^2)$ time (n_1 and n_2 are the number of residues in each structure). Second, an optimal relative orientation is computed for the two structures, based on the previously computed correspondence, by using the method in [87, 97]. The score of the final correspondence is returned together with the cRMS distance of the corresponding parts and the length of the correspondence. Since the result is sensitive to the initial alignment, the algorithm is usually run a number of times for each pair of structures, each time using a different initial alignment. The results of the highest scoring run is kept.

STRUCTAL also computes a p-value for the final score, which can be used to determine the statistical significance of the similarity that was computed. It gives the probability that a similarity of a given score and length could occur at random. As described in [124], this value is a function of the STRUCTAL alignment score and the length of the correspondence. It was computed based on structural comparison of the entire SCOP database. In what follows we will call a statistically significant alignment ($p < 0.005$) a *good* match and all other alignments *bad* matches.

We investigate the performance of STRUCTAL when used together with m-averaging to compute structural similarity among the native structure of different proteins in terms of the tradeoff between the gain in speed and the amount of error that results. We use STRUCTAL as a black box and input m-averaged structures as if they were true protein structures. An m-averaged structure is created by cutting the protein into sub-chains of length m starting at its N-terminal and computing the average of all Cα atoms in each fragment. The sub-chain at the C-terminal could have between 1 and m Cα atoms. It is possible to shift the sub-chains by t residues, where $1 \leq t \leq (N \bmod m)$, and get a different m-averaged structure. The choice of t_1 and t_2 for a pair of m-averaged structures could affect the computed similarity score, however we expect this effect to be small. In what

follows we use $t = 0$ (no shift) when computing all m-averaged structures. Since both n_1 and n_2 are reduced by a factor of m, the complexity of the dynamic programming, which is the main part of the STRUCTAL, is reduced by a factor of m^2. The complexity of the optimal orientation procedure is reduced by a factor of m.

3.4.2 Experimental results

In this section we evaluate our method based on how well it is able to pick out the good matches from the bad matches (as defined in the previous subsection). We cannot test m-averaging in this context by evaluating its performance in finding NNs since it is difficult to impose a strict ordering on a set of proteins, by measuring their similarity to one structure in the set. This is due to the aforementioned inherent tradeoff between the length of the correspondence and the quality of the similarity that is achieved for a given pair.

For our tests we used the latest version of STRUCTAL (courtesy of Michael Levitt). This program inputs two protein structures in the form of PDB files, and outputs their alignment and its score.

The set of structure pairs on which we tested our method was constructed as follows. We examined the clustering results based on STRUCTAL comparisons presented by Erik Sandelin on his PROTOFARM web site[11] and picked out 3,691 pairs of structures, most of which were found to be good matches. These pairs all have a match correspondence of at least 150 Cα atom pairs and are taken from a set of 256 different protein structures of length between 180 and 420 residues each. We then randomly picked 6,375 more pairs from our set of 256 structures. Altogether we examined 10,066 pairs of structures, of which 4052 where found by STRUCTAL to be good matches and 6,014 were found to be bad matches. For each of the structures we generated an m-averaged approximation for $m = 3, 4$ and 5. Finally we ran STRUCTAL on the set of pairs using the 3 sets of averaged structures and recorded the results.

We restrict our analysis to the STRUCTAL score that is computed for each pair, since we do not have a method for computing p-values for evaluating the statistical significance of a match between two m-averaged structures. Developing such a method would require in depth analysis of the distribution of scores of m-averaged pairs (see [124]), which is beyond the scope of this work.

In Figure 3.5 we show the structural comparison results in four graphs. Each depicts two

[11] `http://csb.stanford.edu/sandelin/protofarm/html/`

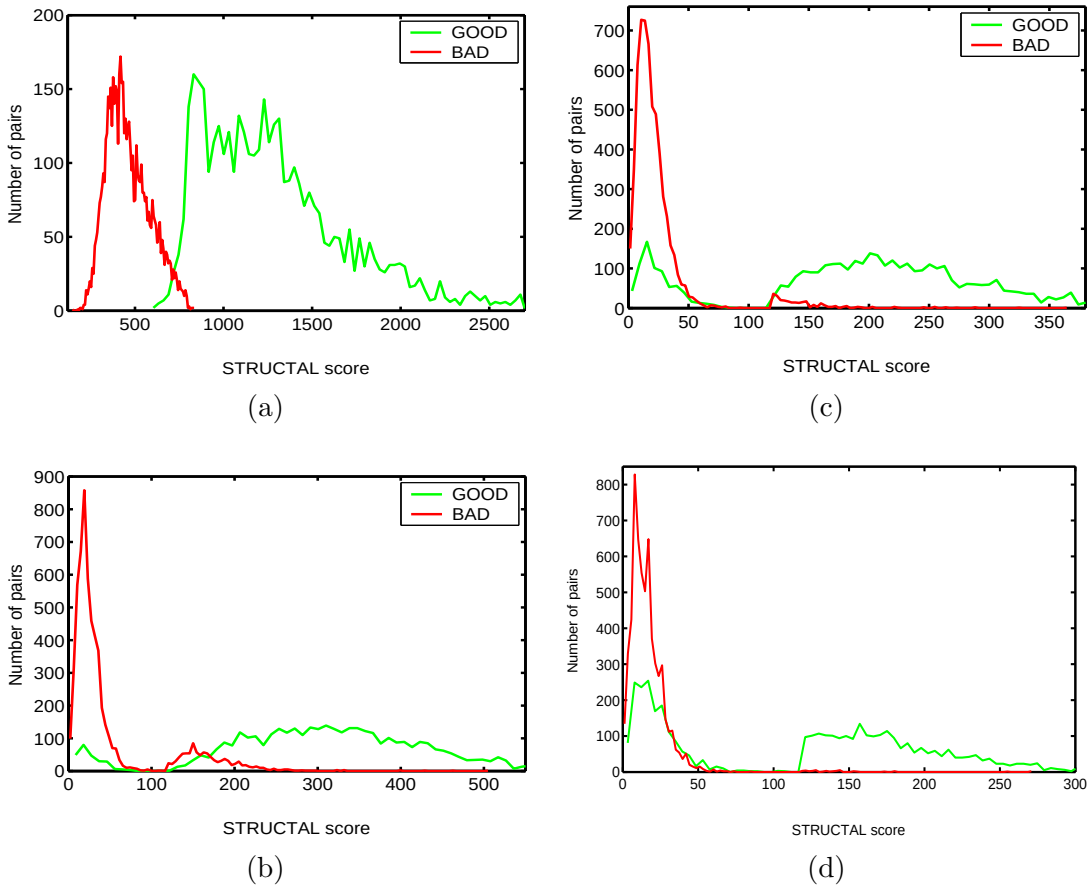


Figure 3.5: Histogram of scores obtained for both the good and the bad matches using STRUCTAL on (a) the full structures, (b) m-averaged structures with $m = 3$, (c) m-averaged structures with $m = 4$ and (d) m-averaged structures with $m = 5$.

histograms, one for the score distribution of the good matches and one for score distribution of the bad ones. The results are binned by the score of each aligned pair. Since the p-value is based on both the score and the length of the correspondence (see [124] for details), the score alone is not enough to completely separate the two histograms even when using the full structures (Figure 3.5(a)). Thus some matches will inevitably be misclassified. Figures 3.5(b), 3.5(c) and 3.5(d) show the histograms obtained for the m-averaged structures with $m = 3, 4$ and 5 respectively. One can still see that the bulk of the matches are classified well by the STRUCTAL score, with the amount of overlap between the histograms increasing as m grows larger.

The exact numbers of misclassified pairs can be found in Table 3.7. The numbers are

	$p \leq 0.005$			$p \leq 0.001$		
m	Total	FP	FN	Total	FP	FN
1	157 (1.6%)	48	109	130 (1.3%)	39	91
3	765 (7.6%)	331	434	646 (6.6%)	199	447
4	966 (9.6%)	205	761	782 (7.8%)	238	544
5	1572 (15.6%)	87	1485	1107 (11%)	136	971

Table 3.7: The total number of false positives (FP) and false negatives (FN).

.

given for both the STRUCTAL p-value cutoff of $p \leq 0.005$ and a more stringent cutoff of $p \leq 0.001$. These numbers are computed by finding the value of the STRUCTAL score that minimizes the amount of total misclassifications for each value of m. Less than 7% of the pairs are misclassified when we use m-averaging with $m = 3$, less than 10% of the pairs are misclassified when $m = 4$ and about about 15% misclassified when $m = 5$. The number of false negatives clearly increases with m, since averaging may obfuscate the similarity. Moreover, the scoring function that was optimized for protein structures becomes less and less effective as the m-averaged structures grow less and less protein-like as m increases, and as a result matching parts can be missed. For the more stringent cutoff of $p \leq 0.001$ we observe in Table 3.7 that there is a considerable drop in the number of misclassified pairs compared to the first p-value used. This indicates that a large number of misclassifications are borderline matches.

When precision is important, m-averaged structures can be used as a fast filter to quickly separate a small set of potentially good matches from the significantly larger amount of bad matches. Then the full structures can be used to weed out the false positives that snuck in. In this scenario, the false negatives are the only true errors since the false positives can be removed in the second stage. Their number can be significantly reduced by shifting the cutoff score down. Thus the number of false negatives will be reduced at the cost of allowing more false positives. For example, when $m = 3$, it is possible to reduce the number of false negatives by 40% while increasing the total number of misclassified pairs by less than 40%.

Another indication that m-averaging introduces only a small error is the correlation between the STRUCTAL score using the full structures and the score using the m-averaged structures. It is 0.924 when $m = 3$, 0.908 when $m = 4$ and 0.85 when $m = 5$. This entails that the m-averaged scores are a good predictor of the true score and could also be used for classification into more than just 2 classes (good or bad matches).

3.4.3 Discussion

The test results presented in the previous subsection show the effectiveness of m-averaging for speeding up structural comparison using the STRUCTAL algorithm. It can be used both as a fast method for structural classification with a small amount of error, or as a fast filter that is able to remove most bad matches, leaving only a small subset of pairs to be evaluated using full protein structures.

In this work, we treated STRUCTAL as a black box. However, a scoring function that is better suited for m-averaged structures can be devised for STRUCTAL. Such a function would take into account the fact that the distance between neighboring points along the backbone chain in an m-averaged structure is different — in fact, larger — than the fixed 3.8Å between Cα atoms in real proteins.

We propose a hierarchical use of STRUCTAL with m-averaging, in which the value of m is decreased gradually. At first a high value is used to quickly remove the very bad matches. Then as m is decreased more and more bad matches are discovered and removed. In this scheme, the alignment that is computed for a pair of structures, for some value of m, could serve as an initial alignment for aligning the two structures at subsequent iterations with smaller values of m, in addition to the other heuristics that are used. Indeed, in most good matches that were detected using the m-averaged structures, the rotation and translation that was used for the alignment (the transformation that superimposes the two structures) was very similar to the one computed using the full structures.

Another good measure for structural alignment that can be used instead of the STRUCTAL score, is the ratio of the cRMS distance between the corresponding Cα atoms of the two structures to the length of the correspondence. This ratio is highly correlated with the STRUCTAL score and yields only a small increase in the number of misclassifications. It is also independent of the comparison algorithm that is used, and can be obtained from other alignment algorithms such as DALI or CE.

3.5 Conclusion

Two general properties of proteins, their chain topology and limited compactness, are exploited to uniformly reduce the number of features for structural similarity computations. Substantial savings in terms of storage and running time are attained with small errors.

In applications in which no approximation error is tolerable, our approach can be used

as a first step to filter a small subset of pairs that are within some tolerance band around the desired similarity. More expensive exact similarity measures can then be used on the reduced set of pairs.

Two possible applications were presented: finding k nearest neighbors in large sets of conformations of the same protein and classification of different proteins using the STRUCTAL algorithm. For the first application the introduced error was low and the correlation to the true similarity measure was very high. For the second application the number of misclassified matches increased only slightly when using m-averaged structures. In both applications, the running times were reduced from days to hours or even minutes.

In general, applications that input very large sets of proteins or that employ computationally intensive algorithms on large proteins stand to benefit from approximation of structures as suggested in this work.

Chapter 4

Efficient Maintenance and Self-Collision Testing for Kinematic Chains*

4.1 Introduction

A powerful and popular computational tool for the study of proteins is computer simulation. The motion of a single protein molecule is simulated as it deforms in time, changing its conformation. When the laws of physics guide the motion, the protein is undergoing a molecular dynamics simulation. When special probabilistic criteria are used to accept proposed motions, it is undergoing Monte Carlo simulation [119]. A key component of the algorithms that perform these simulations is the efficient maintenance of proximity information between the different particles of the system — atoms in the case of proteins. This information is needed when simulating any physical system, not just proteins, in order to detect steric clashes (collisions) between particles. In molecular simulation proximity information is also required to compute the energy of the simulated system, which largely depends on short-range interactions between pairs of atoms.

During the simulation, the conformation of the protein changes, altering the proximity relations between its atoms. It is essential to the computational performance of the simulation that updating the proximity information after each conformational change is done

*This chapter describes work done jointly with Fabian Schwarzer and Dan Halperin and published in [128, 129]

efficiently. The conformational variability of the protein backbone at physiological conditions is well approximated using the torsion-angle model introduced in Subsection 1.3.2. This model represents the protein backbone as a kinematic chain with many DOFs. This property is exploited in developing an efficient algorithm for maintaining the proximity information of a protein molecule undergoing Monte Carlo simulation. A novel data structure — the *ChainTree* — is designed for representing deforming kinematic chains. It takes advantage of specific properties of kinematic chains and Monte Carlo simulation of proteins in order to efficiently update proximity information as the protein deforms during the simulation.

In this chapter the ChainTree is introduced in the context of general kinematic chains. Its performance in detecting steric clashes (self-collision) in a deforming kinematic chain is analyzed both theoretically and in practice. In Chapter 5, the ChainTree is extended and used to compute all pairs of interacting atoms in a protein molecule undergoing a Monte Carlo simulation. A novel scheme for efficient reuse of unchanged partial energy sums is presented, which speeds up energy computation at each step of the simulation.

4.2 Related work

The problems described above are closely related to the detection of collision among moving objects and checking a deforming object for self-collision. Collision detection has been extensively studied in robotics [26, 55, 56, 62, 126, 163, 169], computer graphics [22, 34, 64, 72, 88, 92, 106, 115, 200] and computational geometry [6, 46, 75, 128], to only cite a few works. Most research, however, has been conducted in environments made of rigid objects, few of them moving.

4.2.1 Feature tracking

The collision-detection methods in [6, 126] rely on tracking object features (e.g., closest features) to compute minimal separation distance. They require the identity of the tracked features to change rarely, the so-called *temporal/spatial coherence* assumption. In particular, this assumption implies that during any small time step the placements of the objects undergo small changes. A long kinematic chain does not satisfy this assumption since a small DOF change may cause relatively large displacements of parts of the chain.

4.2.2 Space-partition methods

Other collision-detection methods partition the space in which an object moves, for example, into an octree [55, 64], a regular 3-D grid [79], or a set of projections onto subspaces [34]. Usually, these approaches do not lend themselves to incremental updating to handle a deformable object or many objects moving simultaneously. Exceptions include [34, 64], but then the temporal/spatial coherence assumption must be satisfied. It is worthwhile explaining in more detail one such method – referred to below as the *grid algorithm*. Used under various forms [79, 84, 159, 162, 201, 213] this method reduces the complexity of finding self-collision to asymptotically linear time in non-pathological cases by indexing the basic fragments of the object in a regular grid. This approach assumes that all object fragments have similar size and that the centers of their bounding sphere do not come within a small fixed distance. In this case it is formally shown in [79] that the number of collisions of one fragment is bounded by a constant. The 3-D space occupied by the object is divided into cubes whose sides are set to the diameter of the largest bounding sphere. For each fragment, the cube that contains the center of its bounding sphere is computed, and indexed using a hash-table. This data structure is re-computed after each step in $\Theta(n)$ time. Determining which fragments intersect any given fragment then takes $O(1)$ time. Hence, finding all collisions takes $\Theta(n)$ time.

4.2.3 Bounding volume hierarchies

The most popular approach to collision detection pre-computes a bounding-volume hierarchy (BVH) for each object. This hierarchy captures spatial proximity between small components of the object at successive resolutions. The hierarchies are then used to expedite collision tests by quickly discarding pairs of components contained in non-overlapping bounding volumes (BVs) [22, 56, 72, 88, 106, 115, 163, 200]). Various types of BVs have been used, e.g., spheres and bounding boxes. The performance of a bounding box hierarchy was analyzed theoretically in [221, 222]. These techniques have been extended in [22, 200] to handle deformable objects by exploiting the facts that neighboring elements in the meshed surface of an object always remain in proximity to each other when the object deforms. For each deformable object, a balanced BV tree is pre-computed by successively grouping topologically close features of the meshed surface and enclosing them in a BV. When the object deforms, the topology of the tree stays unchanged; only the sizes and locations of

the BVs are updated in a bottom-up fashion. However, in general, BV techniques lose efficiency when applied to detecting self-collision in a deformable object, because they cannot avoid detecting the trivial self-collision of each object component with itself. They also lose efficiency when many objects (rigid or not) move independently. They were, nonetheless, successfully applied to detecting self-collision in a rope during knot-tying simulation [21].

The method described below borrows from previous work on BVHs. It uses a BVH based on the invariance of the chain topology. But it combines it with a transform hierarchy that makes it possible to efficiently prune the search for self-collision, when few DOFs change simultaneously.

4.2.4 Collision detection for kinematic chains

Only a limited amount of previous research has been dedicated to collision detection for kinematic chains. A general scheme is proposed in [78] to efficiently update a representation of a kinematic chain designed to efficiently detect collision with fixed obstacles. however, this scheme does not support self-collision detection. The work in [75] addresses a similar problem – testing self-collision in a deformable necklace made of spherical beads. Like the method described here, it builds a BVH based on the chain topology. However, it assumes that all DOFs change simultaneously at each step and does not attempt to take advantage of rigid sub-chains. When all DOFs change simultaneously our algorithm does about the same amount of work as the algorithm in [75] to detect self-collision.

The problem of deciding whether a torsion angle change causes a self-collision in a 3D polygonal chain has been studied in [184, 185]. It was shown in [185] that determining whether a rotation around a bond causes self-collision anywhere along its path takes $\Omega(n \log n)$ time, when n is the number of links in a chain. It is further conjectured in [184] that no amount of preprocessing enables performing n such rotations in worst case sublinear time per rotation. However, in this work, we only care whether the conformation at the end of the rotation is collision-free and not whether there is a collision during the motion.

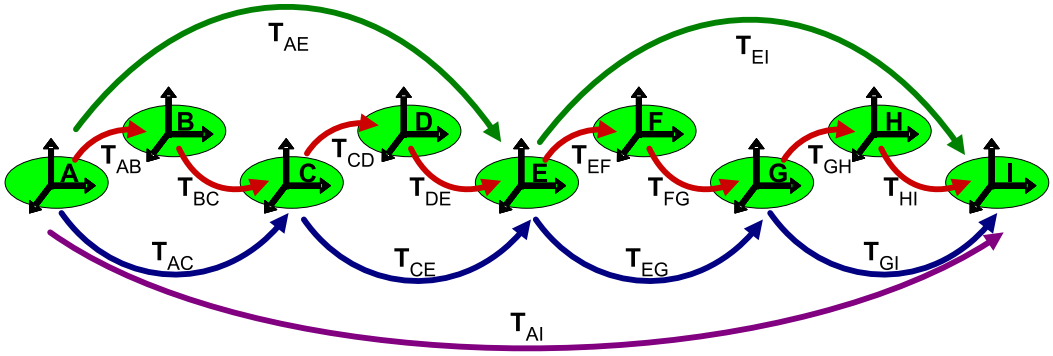


Figure 4.1: Transform hierarchy: grey ovals depict links; $T_{\alpha\beta}$ denote rigid-body transforms between the reference frames α and β.

4.3 The ChainTree

In this section, the ChainTree, the data structure used to represent a protein, is described. It is designed to take advantage of the open chain properties that were described in Subsection 1.3.2.1. First, the two hierarchies that make up the ChainTree are described. The *transform* hierarchy that approximates the kinematics of the backbone is introduced in Subsection 4.3.1 and the *bounding-volume* hierarchy that approximates the geometry of the protein is presented in Subsection 4.3.2. Next, the ChainTree is presented as the combination of the two aforementioned hierarchies into a single balanced binary tree (Subsection 4.3.3). Finally, the update scheme of the ChainTree is describe(Subsection 4.3.4).

In the following the algorithm that updates the ChainTree is referred to as the *updating* algorithm and the algorithm that tests self-collision or finds interacting pairs is referred to as the *testing* algorithm.

4.3.1 Transform hierarchy

A reference frame is attached to each link of the protein's backbone and each DOF is mapped to the rigid-body transform between the frames of the two links it connects. The transform hierarchy is a balanced binary tree of transforms. See Figure 4.1, where ovals and labeled arrows depict links and transforms, respectively.

At the lowest level of the tree, each transform represents a DOF of the chain. Products of pairs of consecutive transforms give the transform at the next level. For instance, in

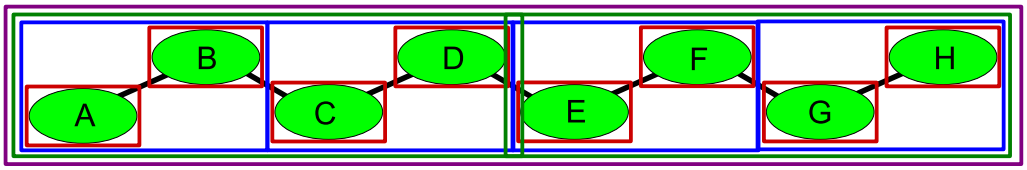


Figure 4.2: The BVH: each box (OBB) approximates the geometry of a chain-contiguous sequence of links.

Figure 4.1, T_{AC} is the product of T_{AB} and T_{BC}. Similarly, each transform at every level is the product of two consecutive transforms at the level just below. The root of the tree is the transform between the frames of the first and last links in the chain (T_{AI} in the figure).

Each of the $\log n$ levels of the tree can be seen as a chain that has half the links and DOFs of the chain at the level just below it. In total, $O(n)$ transforms are cached in the hierarchy. We say that each intermediate transform $T_{\alpha\beta}$ *shortcuts* all the transforms that are in the subtree rooted at $T_{\alpha\beta}$.

The transform hierarchy is used from the top down by the testing algorithm to propagate transforms defining the relative positions of bounding boxes (from the other hierarchy) that need to be tested for overlap. It also allows computing the relative position of any two links or boxes in $O(\log n)$ time, but this property is not used by our algorithm.

A structure similar to our transform hierarchy was introduced in [184]. However, it was only used for theoretical analysis and was not implemented.

4.3.2 Bounding-volume hierarchy

The bounding-volume (BV) hierarchy is similar to those used by prior collision checkers (see Section 4.2.3). As spatial proximity in a deformable chain is not invariant, the BVH used here is based on the proximity of links along the chain. See Figure 4.2. Here, for simplicity, the BVH is introduced in the context of self-collision detection. In Section 5.3.2 the extensions that allow for self-collision detection *and* finding interacting pairs using a single hierarchy are described.

Like the transform hierarchy, the BVH is a balanced binary tree. It is constructed bottom up in a "chain-aligned" fashion. At the lowest level, one BV bounds each link. Then, pairs of neighboring BVs at each level are bounded by new BVs to form the next

level. The root BV encloses the entire chain and each level contains a chain with half the BVs as the chain at the level below it. Each chain of BVs encloses the geometry of the chains of BVs at all levels below.

The type of BV chosen in this work is the oriented bounding box (OBB) [72], a rectangular bounding box at an arbitrary orientation. OBBs were selected because they bound well both globular objects (single atoms, small groups of atoms) and elongated objects (chain fragments). In addition, unlike simpler axis-aligned bounding boxes [22, 200], but like spheres, OBBs are invariant to a rigid-body transform of the geometry they bound. In the ChainTree, this property eliminates the need to re-compute the BV of a rigid sub-chain, even when this sub-chain has moved. Finally, OBBs can be efficiently computed and tested for intersection. Spheres are another frequently used BVs [75, 163] that would meet our needs. However, in a chain-aligned hierarchy, they are expected to bound poorly elongated sub-chains, which are likely to occur.

Each intermediate OBB is constructed to enclose its two children, thus creating a *not-so-tight* hierarchy (in contrast to a *tight* hierarchy where each BV tightly bounds the links of the sub-chain it encloses). In the Appendix (Lemma 2), it is proven that the size of these OBBs does not deteriorate too much as one climbs up the hierarchy. The major advantage of using this not-so-tight BVH is the efficiency with which each box can be updated. Indeed, the shape of the OBB stored at each intermediate node depends only on the 16 vertices of the two OBBs held by this node's children.

4.3.3 Combined data structure

The ChainTree combines both the transform and the BV hierarchies into a single binary tree as the one depicted in Figure 4.3. The leaves of the tree (labeled A through H in the figure) correspond to the rigid atom-groups (links) of the protein's backbone as described in Subsection 1.3.1. Each leaf holds both the bounding box of the corresponding link and side-chain and the transform (symbolized by a horizontal arrow in the figure) to the reference frame of the next link.

Each internal node (nodes J through P) is associated with the frame of the leftmost link in its sub-tree. It holds both the bounding box of the boxes of its two children and the transform to the frame of the next node at the same level, if any. Computing the relative position of a box and its left child box requires no coordinate transform, while computing the position of a box relative to its right child box requires one transform, which is stored

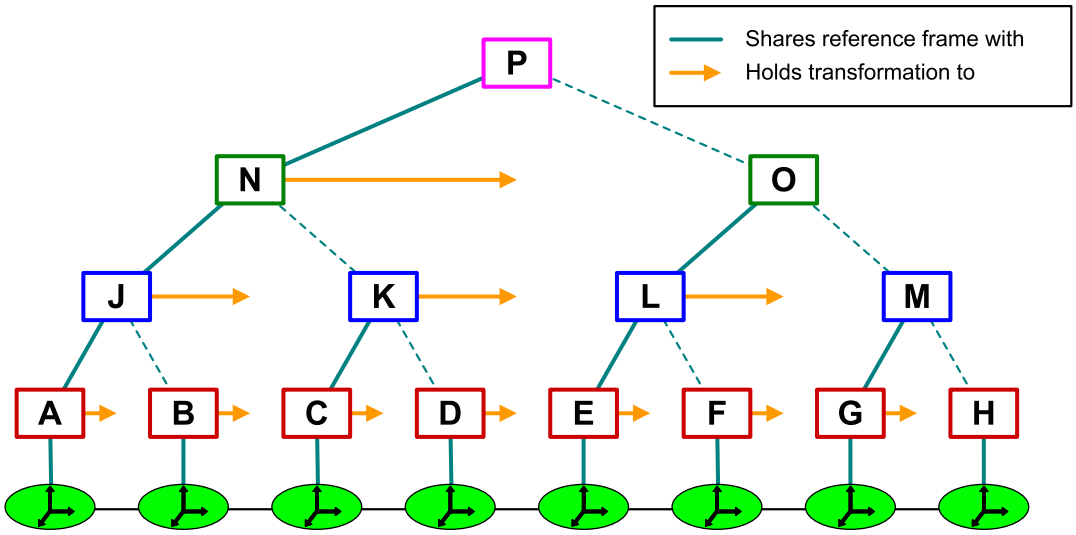


Figure 4.3: A binary tree that combines the transform and BV hierarchies.

at the left child node. This mechanism is used by the testing algorithm to propagate down the relative positions of pairs of boxes that are to be tested for overlap.

The ChainTree contains both pointers from children to parents, which are used by the updating algorithm to propagate updates from the bottom up, as described below, as well as pointers from parents to children, which are used by the testing algorithm (Section 4.4).

4.3.4 Updating the ChainTree

When a change is applied to a single arbitrary DOF in the backbone, the updating algorithm re-computes all transforms that shortcut this DOF and all boxes that enclose the two links connected by this DOF. It does this in a bottom-up fashion, by tracing the path from the leaf node immediately to the left of the changed DOF up to the root. A single node is affected at each level. If this node holds a transform, this transform is updated. If it holds a box that contains the changed DOF, then the box is re-computed. For example, in Figure 4.3, when the DOF between the links associated with nodes F and G is changed, the transforms stored at F and L and the boxes at O and P are re-computed. Since the shape of an OBB is invariant to a rigid-body transform of the objects it bounds, all other boxes remain unchanged.

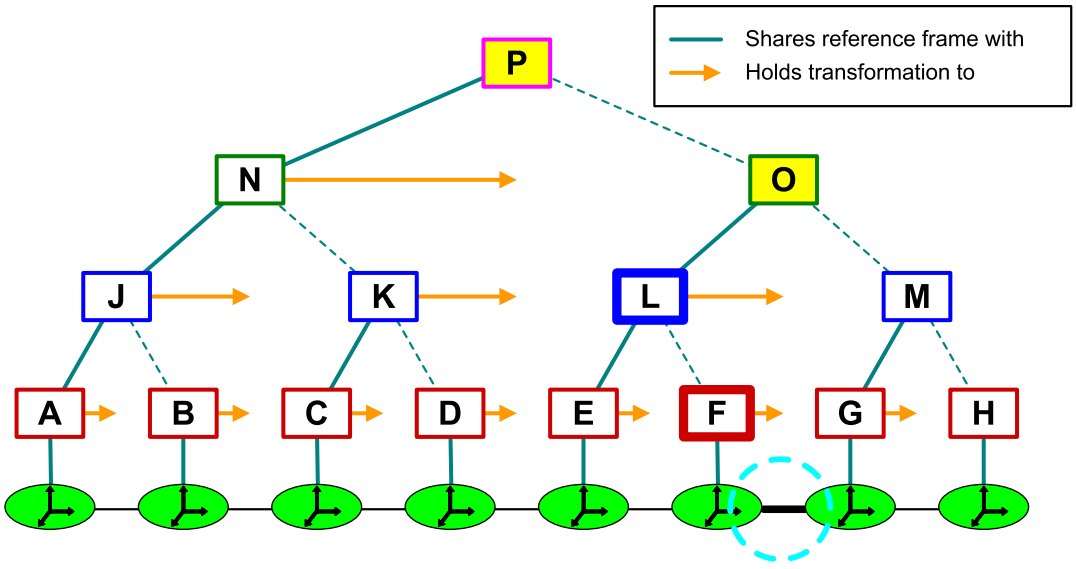


Figure 4.4: The ChainTree after applying a 1-DOF perturbation. Highlighted nodes were updated

When multiple DOFs are changed simultaneously, the ChainTree is updated one level at a time, starting with the lowest level. Hence, all affected transforms and boxes at each level are updated at most once before proceeding to the level above it.

The updating algorithm marks every node whose box and/or transform is re-computed. This mark will be used later by the testing algorithm. Figure 4.4 shows the marked nodes in the ChainTree of Figure 4.3, after a change in the DOF between the links associated with nodes F and G. The nodes with bold contours (F and L) are those whose transforms were updated. The nodes in yellow (O and P) are those whose boxes were re-computed. In general, however, marked nodes may have had both their transforms and boxes updated.

4.4 Self-collision detection

4.4.1 Using a BVH

BVHs have been widely used to detect collision between pairs of rigid objects, each described by its own hierarchy [72, 88, 106, 163, 200]. Given the hierarchies of two objects, the algorithm first checks whether the root boxes overlap. If they do not, it can safely return

that the two objects do not collide. If they do overlap, then the algorithm descends one level in both hierarchies and tests all four pairs of children. It continues this process iteratively. When the lowest level of one hierarchy is reached, the algorithm continues its descent through the other hierarchy, testing the leaf boxes of one hierarchy against boxes at the newly reached level in the other hierarchy. When it reaches leaves in both hierarchies, it tests the actual components of the objects for overlap and returns a collision when they overlap. The algorithm may stop as soon it has found a collision, or it may run until it has found all collisions. In the former case, it is best for the algorithm to search for a collision in a depth-first manner to reach overlapping leaves as quickly as possible.

The algorithm terminates quickly when the two objects are well separated, because the search then ends near the top of both hierarchies and avoids dealing with the possibly complicated geometry of the actual objects. If one is only interested in detecting whether a collision occurs, the algorithm also terminates quickly when the overlap is large, because the depth-first search is then expeditious in finding a pair of overlapping leaf nodes. The algorithm takes longer when the objects are close but do not collide, or when there are many distinct collisions and one wants to find them all.

A simple variant of this algorithm detects self-collision by testing the BVH of the object against itself. This variant skips the test of a box against itself and proceeds directly to testing the box's children. However, it takes $\Omega(n)$ time, since all leaves will inevitably be visited (each leaf is trivially in collision with itself). The ChainTree avoids this lower bound by exploiting the third property stated in Section 1.3.2.1— large sub-chains remain rigid between steps.

4.4.2 Using the ChainTree

When only a small number k of DOFs are changed simultaneously, long sub-chains remain rigid at each step. These sub-chains cannot contain new self-collision. So, when the BVH contained in the ChainTree is tested against itself, the branches of the search that would look for self-collision within rigid sub-chains are pruned.

There are two distinct situations where pruning occurs:

1. When the algorithm is about to test a box against itself and this box was not updated after the last DOF changes, the test is pruned.

2. When the algorithm is about to test two different boxes, and neither box was updated

after the last DOF changes, and no DOF between those two boxes was changed, the test is pruned.

The last condition in this second situation – that no DOF between the two boxes was changed – is slightly more delicate to recognize efficiently. Two nodes at the same level in the ChainTree are dubbed *separated* if there exists another node in between at the same level that holds a transform modified after the last DOF changes. This node will be named a *separator*. They are also separated if the transform held by the leftmost of the two was modified. Hence, if two nodes are separated, a DOF between them has changed. Note that:

- when two nodes at any level are separated, any pair consisting of a child of one and a child of the other is also separated.
- When two nodes at any level are *not* separated, a child of one and a child of the other are separated if and only if they are separated by another child of either parent.

Hence, by pushing separation information downward, the testing algorithm can know in constant time whether a DOF has changed between any two boxes it is about to test. The algorithm also propagates transforms from the transform tree downward to compute the relative position of any two separated boxes in constant time before performing the overlap test.

To illustrate how the testing algorithm works, consider the ChainTree of Figure 4.4 obtained after a change of the DOF between F and G. F and L are the only separators. The algorithm first tests the box stored in the root P against itself. Since this box has changed, the algorithm examines all pairs of its children, (N, N), (N, O) and (O, O). The box held in N was not changed, so (N, N) is discarded (i.e., the search along this path is pruned). (N, O) is not discarded since the box of O has changed, leading the algorithm to consider the four pairs of children (J, L), (J, M), (K, L), and (K, M). Both (J, L) and (K, L) satisfy the conditions in the second situation described above; thus, they are discarded. (J, M) is not discarded because J and M are separated by L. The same is true for (K, M), and so on.

4.5 Complexity analysis

Two fundamental tradeoffs must be made when using a BVH for self-collision detection:

1. Between the number of box overlap tests needed to detect or rule out self-collision and the cost of one such test: In general, the more complex the BVs, the tighter they bound a given geometry and the fewer tests are required; but the more expensive each test becomes.

2. Between the time needed to update the BVH and the time to detect or rule out self-collision: In general, reducing updating time (e.g., by using not-so-tight boxes or keeping the topology of the hierarchy fixed) impairs the performance of self-collision detection.

The analysis below is aimed at clarifying the choices made in the ChainTree and comparing the updating and testing algorithms to current state-of-the-art algorithms. It is based on the worst-case asymptotic behavior of the algorithms when n grows arbitrarily large. But this behavior is not always meaningful in practice, since for many proteins n is too small and/or the worst case is unlikely. For this reason, Section 4.6 will complement this analysis with experimental tests.

Throughout this section we ignore the side-chains. Since the size of each one is small and bounded by a constant, the side chains only affect the asymptotic complexity bounds by a constant factor.

4.5.1 Updating the ChainTree

Updating the ChainTree after a 1-DOF change requires re-computing transforms and boxes held in nodes along the path from the leaf node immediately to the left of the changed DOF up to the root of the tree. By doing the updates from the bottom up, each affected transform is re-computed in $O(1)$ time. By using not-so-tight OBBs — thus, trading tightness for update-speed — re-computing each box is also done in $O(1)$ time. Since the ChainTree has $\lceil \log n \rceil$ levels, and at each level at most one transform and one box are updated, the total cost of the update process is $O(\log n)$.

When a k-DOF change is made, affected transforms and boxes are updated one level at a time. This ensures that no transform or box is re-computed more than once when the converging update paths merge. The total updating time is then $O(k \log (n/k))$. When k grows this bound never exceeds $O(n)$.

The efficient updating time for small values of k derives from the fact that the topology of the ChainTree is never modified. However, this topology is not always optimal to detect

self-collision, as it only imperfectly represents spatial proximity among links.

4.5.2 Detecting self-collision

All known algorithms to detect self-collision in an n-link chain take $\Theta(n^2)$ time in the worst case, if no further assumption is made about the chain. Hence, they do not behave better than a brute-force algorithm.

In this subsection, it is assumed that the chains are *well-behaved*. Each link of a chain is associated with its minimal enclosing sphere. Given two positive constants γ and δ, a chain is well-behaved if it verifies the following two properties:

1. The ratio of the radii of the largest and smallest enclosing spheres is smaller than γ.
2. The distance between the centers of any two enclosing spheres is greater than δ.

It is not hard to convince oneself that proteins for example form well-behaved chains for some γ and δ. The first property follows from the fact that there are only 20 different types of amino acids. The second property is verified if we exclude conformations where atoms overlap almost completely. These conformations are physically impossible because of the van der Waals repulsion. It is shown in [79] that in well-behaved chains of arbitrary length n the number of links that overlap a given link is bounded by a constant. Thus, there are at most $O(n)$ overlaps between the links of a well-behaved chain.

The motion model assumed in this work, that of changing only a few DOFs at time, cannot cause a well-behaved chain to degenerate as long as new configurations that violate the second condition above are never accepted. For example, in Monte carlo simulation of proteins, a step that causes a pair of atoms to interpenetrate, will be rejected because of the huge energetic penalty incurred by a steric clash. Thus, no cumulative effect during successive steps will degenerate the chain.

Our algorithm performs two tasks for every pair of nodes it examines. First, it must decide whether to prune this search path. This requires testing if the nodes have been updated after the last DOF change and, if not, whether they are separated. Second, if this search path is not pruned, the two boxes are tested for overlap. Both tasks require constant time. Thus, the complexity of the algorithm is governed by the number of pairs of nodes that are examined, which is proportional to the number of overlapping boxes at all levels of the ChainTree in the worst case.

In the Appendix the following theorem is proved:

	Update	Detection
Brute force	$\Theta(n)$	$\Theta(n^2)$
Grid	$\Theta(n)$	$\Theta(n)$
Spatially-adapted hierarchy	$O(n \log n)$	$\Theta(n)$
Chain-aligned hierarchy	$O(n)$	$\Theta(n^{\frac{4}{3}})$
ChainTree	$O(k \log \frac{n}{k})$	$\Theta(n^{\frac{4}{3}})$

Table 4.1: Comparison of complexity measures for updating and detecting self-collision in well-behaved chains.

Theorem 1 *The maximum total number of overlapping boxes at all levels of the ChainTree of a well-behaved chain of n links is $\Theta(n^{\frac{4}{3}})$.*

Therefore, the testing algorithm runs in worst-case $\Theta(n^{\frac{4}{3}})$ time. This bound, which is similar to the one established in [75], holds whether the algorithm stops after detecting the first collision or keeps running to detect all self-collisions. Indeed, the fact that two boxes overlap does not imply that they contain colliding links. So, to detect self-collision or its absence, the algorithm may have to eventually consider all pairs of overlapping boxes.

Note also that the bound is not affected by the pruning of search paths. So, when all DOFs change at each simulation step, and as long as the chain remains well-behaved, self-collision detection still takes $\Theta(n^{\frac{4}{3}})$ time in the worst case, after $O(n)$ updating.

The experimental results of Section 4.6 show however that when few DOFs change at each step, the algorithm behaves much better in practice than the above bound suggests.

4.5.3 Comparison with other methods

There exist several methods applicable to the problem of detecting self-collision in a chain. The asymptotic worst-case complexity of the most important ones is given in Table 4.1.

At each step, the brute-force algorithm first re-computes the position of every link that has moved, which is done in worst-case linear time. It then detects self-collision by testing all pairs of links for overlap, resulting in $\Theta(n^2)$ tests in the worst case (e.g., when there are no collisions).

Under the assumption that the chain is well-behaved, the grid algorithm described in Section 4.2 reduces the worst-case complexity of both updating and testing to linear time [79].

Although the grid algorithm is optimal in the worst case, BVH methods are intended to do better on average. For a chain, two types of BVHs may be used:

Spatially-adapted hierarchy: A BVH as described in [72, 106] which is based on a spatial partitioning of each chain conformation to optimally encode spatial proximity between links.

Chain-Aligned hierarchy: A BVH as the one in the ChainTree (see [75] for another example) that encodes the *chain-wise* proximity of links along the chain. Links are said to be in chain-wise proximity if they are not separated by more than a few links along the chain.

Ideally, a BVH should be such that the depth of any box in the tree is a good indicator of the spatial proximity of the links bounded by this box. This is precisely what a spatially-adapted hierarchy is intended to achieve. Instead, a chain-aligned hierarchy only encodes chain-wise proximity. If two links are chain-wise close, they are also spatially close. But the reverse is not true: in a given chain conformation, pairs of links that are chain-wise far apart may be spatially very close (e.g., this happens in folded protein conformations). Consequently, testing self-collision with a spatially-adapted hierarchy is more efficient than with a chain-aligned one. But, because spatial proximity varies as the chain deforms, while chain-wise proximity does not, updating a spatially-adapted hierarchy is more expensive.

A spatially-adapted hierarchy of a chain can be tested for self-collision in $\Theta(n)$ time in the worst case. This bound is a special case of a theorem proven in [222]. However, it is expected to do better in practice. Building a new hierarchy at each step takes $O(n \log n)$ time [72]. One could attempt to reduce updating time to $\Theta(n)$ by not changing the topology of the hierarchy and only updating the size and location of the BVs. But a single DOF change may then corrupt the hierarchy so as to raise the number of BV overlap tests required to detect self-collision to $\Theta(n^2)$. Figure 4.5 illustrates such a scenario. In 4.5(a), the hierarchy is spatially-adapted with no overlap between boxes. A change in the middle DOF creates the conformation in 4.5(b). The boxes were updated to bound the new conformation without changing the topology of the hierarchy. All pairs of boxes at the displayed level now overlap.

In contrast, despite its fixed topology, a chain-aligned BVH guarantees an $O(n^{\frac{4}{3}})$ self-collision test, with an $O(n)$ update when not-so-tight OBBs are used and an $O(n \log n)$ update when tight spheres are used [75]. Thanks to its transform hierarchy, the ChainTree reduces further the updating cost after a k-DOF change to $O(k \log (n/k))$.

The worst-case bound on self-collision detection with the ChainTree hides the practical speed-up allowed by search pruning. To evaluate this speed-up, we need to perform empirical

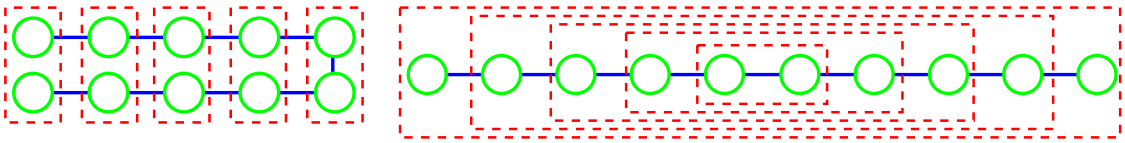


Figure 4.5: A 1-DOF change corrupting a spatially-adapted hierarchy. (a) Before change. (b) After 1-DOF change. Only second level from bottom is shown ($\frac{n}{2}$ boxes).

tests.

4.6 Test results for self-collision detection

We conducted various tests with our implementation of our method – called here **ChainTree** – and the following three methods:

Grid - This is the grid method as described in Section 4.2.2. The length of a side of each grid cell is the diameter of the largest atom. Both updating and testing take $\Theta(n)$ time.

1-OBBTree - Here, a tight spatially-adapted OBB hierarchy is re-computed at each step, and then tested against itself for self-collision. Updating and testing take $O(n \log n)$ and $O(n)$ worst-case time, respectively.

K-OBBTree - At each step, this algorithm computes a tight spatially-adapted OBB hierarchy for each rigid sub-chain. It then tests each pair of hierarchies for collision. Updating and testing take $O(n \log n)$ and $O(n)$ worst-case time, respectively.

Both 1-OBBTree and K-OBBTree are based on the PQP library [72, 115] from UNC. The function of ChainTree testing pairs of OBBs for overlap is also from this library. Our experiments were run on a single 400 MHz UltraSPARC-II CPU of a Sun Ultra Enterprise 5500 machine with 4.0 GB of RAM.

In a first series of tests, chains of n spheres of radius 1 unit, spaced 4 units apart by a torsion joint were constructed, with $n = 1000$, 2500, 5000 and 10000 spheres in initial compact cube-like conformations. Each simulation consisted of 100,000 steps, each changing a single DOF picked uniformly at random. The magnitude of the change was chosen uniformly at random between 0° and 30°. If a collision was detected, the step was rejected,

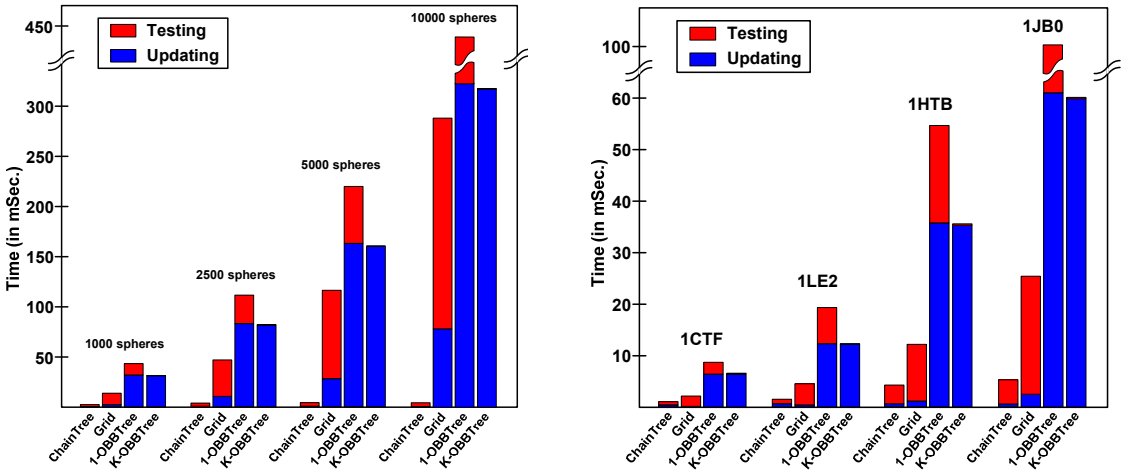


Figure 4.6: Average time per step for detecting self-collision in (a) a compact pseudo-protein chain and (b) backbones of different proteins.

i.e., the change was undone before proceeding to the next step. This requires keeping two copies of the ChainTree. For each chain, the four runs (one with each method) started at the same initial configuration and used the same seed of the random number generator. Hence, they generated the same sequence of configurations.

The average times per step are shown in Figure 4.6(a), when the algorithms stop after finding the first collision. The plots show both updating times (black) and testing times (grey). Some times are too small to be discernable. ChainTree is 5 times faster than Grid for 1000 spheres and about 60 times faster for 10000 spheres. When finding all self-collisions, ChainTree is 4 times faster than Grid for 1000 spheres and 53 times faster for 10000 spheres. In all cases, 1-OBBTree and K-OBBTree are the slowest methods because of their high updating time, although K-OBBTree has the fastest testing time of all four methods.

In a second series of tests four proteins from the Protein Data Bank[1] [15] were used. We selected these proteins – named 1CTF, 1LE2, 1HTB, and 1JB0 – to span different sizes. See Table 4.2. For each protein, we ran the same simulation as in the first series of tests, starting at the folded (native) state of the protein. However, only the protein's backbone was considered and the side-chains were ignored. At each step the four algorithms stopped after finding the first collision. The results are summarized in Figure 4.6(b). ChainTree is

[1] `http://www.rcsb.org/pdb`

Protein	#amino acids	#Atoms	# Backbone atoms	#Links
1CTF	68	487	204	137
1LE2	144	1167	432	289
1HTB	374	2776	1112	749
1JB0	755	5878	2265	1511

Table 4.2: Proteins used in experiments.

twice as fast as Grid for the smallest protein (1CTF) and 5 times faster for the largest one (1JB0). Note that the chains in this benchmark are significantly shorter than those in the first benchmark.

The effect of the number k of simultaneous DOF changes at each step on the performance of the algorithms was also examined. We let k vary from 1 to 121 and, for each value, measured the average time per step to update and find all collisions. In order to eliminate the effect of compactness on the results, 11 conformations of 1HTB were chosen, each having a different *radius of gyration* distributed uniformly between 20Å (the radius of the native conformation) and 85Å. For each such conformation and each value of k, we ran a simulation of 10,000 steps starting at that conformation. Each simulation was constrained to stay at its starting conformation so its compactness would not change. Since more simultaneous changes cause more collisions, on average it takes less time to find one collision when k is large. Therefore we measured the time to find *all* collisions which is less affected by k. Figure 4.7 shows the results. The time results are averaged over all 11 conformations. In this case, ChainTree is the fastest algorithm until $k = 50$, thanks to both its logarithmic updating time and its fast self-collision detection due to search pruning. When k increases, the updating time of ChainTree deteriorates and search pruning becomes less effective as rigid sub-chains are shorter. For $k > 50$, Grid, whose performance is independent of k, becomes faster.

Figure 4.8 reveals more explicitly the effectiveness of search pruning in ChainTree. It shows the average number of box overlap tests performed by ChainTree with and without search pruning in the previous experiment (averaged over 11 conformations of 1HTB, as explained above), for increasing values of k. The numbers are significantly smaller with pruning than without when k is small. As expected, they converge as k increases.

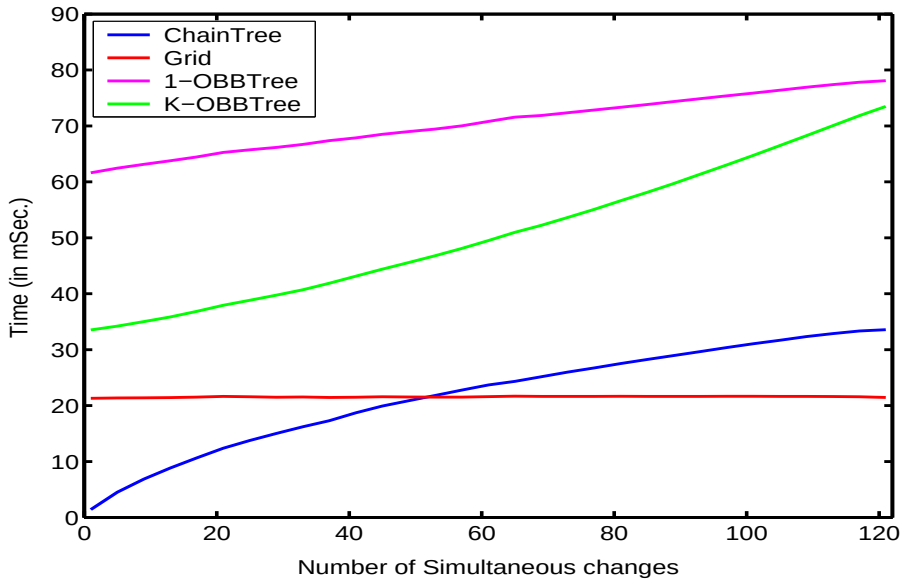


Figure 4.7: The effect of increasing the number of simultaneous DOF changes on the running time of the four algorithms.

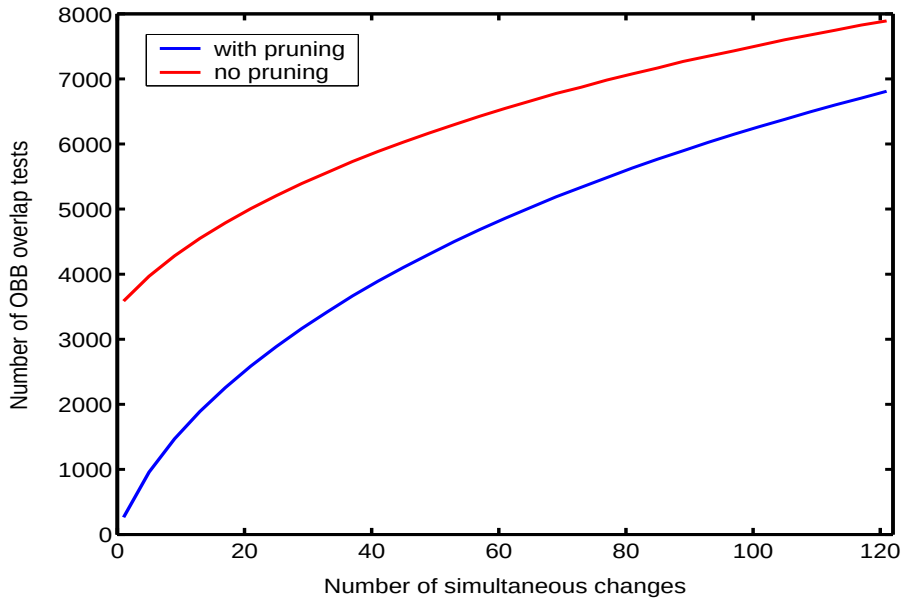


Figure 4.8: The average numbers of box overlap tests by ChainTree with and without search pruning.

Chapter 5

Efficient Energy Computation for Monte Carlo Simulation of Proteins*

5.1 Introduction

5.1.1 Monte Carlo simulation

The study of the conformations adopted by proteins is an important topic in structural biology. Monte Carlo simulation (MCS) [17] is one common methodology for this study. In this context, it has been used for two purposes: (1) estimating thermodynamic quantities over a protein's conformation space [83, 121, 218] and, in some cases, even kinetic properties [171, 172]; and (2) searching for low-energy conformations of a protein, including its native structure [2, 3, 219]. The approach was originally proposed in [140], but many variants and improvements have later been suggested [82].

MCS is a series of randomly generated *trial steps* in the conformation space of the studied molecule. Each such step consists of perturbing some degrees of freedom (DOFs) of the molecule [125, 171, 172, 218, 219], in general torsion (dihedral) angles around bonds. Classically, a trial step is *accepted* – i.e., the simulation actually moves to the new conformation – with probability $\min\{1, e^{-\Delta E/k_b T}\}$ (the so-called Metropolis criterion [140]), where

*This chapter describes work done jointly with Fabian Schwarzer and Dan Halperin and published in [129, 130]

E is an energy function defined over the conformation space, ΔE is the difference in energy between the new and previous conformations, k_b is the Boltzmann constant, and T is the temperature of the system. So, a downhill step to a lower-energy conformation is always accepted, while an uphill step is accepted with a probability that goes to zero as the energy barrier grows large. It has been shown that a long MCS with the Metropolis criterion and an appropriate step generator produces a distribution of accepted conformations that converges to the Boltzmann distribution.

Molecular dynamics simulation (MDS) is another common approach for studying protein conformations. The forces on the atoms are computed at each step, and used to calculate the atom positions at the next step. To be physically accurate, MDS proceeds by small time steps (usually on the order of a few femto-seconds), resulting in slow progress through conformation space. Most MDS techniques update the Cartesian coordinates of the atoms at each step, but recently there has been growing interest in directly updating torsion angles [76, 186], as this allows for both a more compact representation of the conformation space and for larger time steps. In general, MCS makes larger conformational changes than MDS and thus tends to explore subsets of the conformation space faster. But unlike MDS, pathways produced by MCS may not be physically valid, hence may not always provide useful kinetic information.

The need for general algorithms to speed-up MCS has often been mentioned in the biology literature, most recently in [218]. In this chapter, the algorithm and data structure introduced in Chapter 4 are extended to achieve this goal, independent of the specific energy function, step generator, and acceptance criterion. More precisely, the new algorithm reduces the average time needed to decide whether a trial step is accepted, or not, without affecting which steps are attempted, nor the outcome of the test. It achieves this result by incorporating efficient techniques to incrementally update the value of the energy function during simulation. Although this algorithm is presented in the context of classic MCS, it could also be used to speed up other kinds of MCS methods, as well as other optimization and sampling techniques. Several such applications will be discussed in Section 5.6.

5.1.2 The protein model

This algorithm uses the protein model described in Section 1.3.1. The DOFs of the protein are its torsion angles. A link, which designates a rigid part of a kinematic chain, is a group of atoms with no DOFs between them. This model has two types of links, the peptide

atom-group and the side-chain atom-group (See Figure 1.4). Each side-chain may have between 0 and 4 torsion DOFs (known as the χ angles), however these DOFs do not affect the conformation of the backbone.

It is also possible to apply this algorithm to models that include additional DOFs, such as: ω angles, bond lengths, and bond angles. At the limit, one can make each link a single atom and each joint a rigid body-transform. However, while it is theoretically possible to perform MCS in the Cartesian coordinate space, where each atom has 3 DOFs, it is more efficient to run it in the torsion-DOF space [149]. Hence, the vast majority of MCS are run in this space [125, 171, 172, 218, 219].

Due to the chain kinematics of the protein, the properties of open chains listed in Subsection 1.3.2.1 are applicable. E.g., a small change in one DOF of the backbone may cause large displacements of some atoms. Thus, in an MCS, a high percentage of steps are rejected because they lead to high-energy conformations, in particular conformations with steric clashes (self-collisions). In fact, the rejection rate tends to grow quickly with the number k of DOFs changed in a single step. This behavior has been observed before [1, 125]. Figure 5.1 illustrates this point with data gathered during an actual MCS run of 100,000 steps on an unfolded conformation of 1HTB (378 residues). The top plot shows the rejection rate as a function of k, while the bottom plot gives the total number of DOF changes in accepted steps during a run of 100,000 trial steps, also as a function of k. Note that the largest total number of DOF changes is obtained for $k = 1$. Hence, it is common practice in MCS to change few DOFs (picked at random) at each trial step [103, 113, 125, 171, 172, 218, 219].

5.1.3 Computing the energy

Various energy functions have been proposed for proteins (e.g., [68, 103, 118, 120, 190]). For all of them, the dominant computation is the evaluation of non-bonded terms, namely energy terms that depend on distances between pairs of non-bonded atoms. These may be physical terms (e.g., van der Waals and electrostatic potentials [120]), heuristic terms (e.g., potentials between atoms that should end up in proximity to each other [68]) and/or statistical potentials derived from a structural database (e.g. [103]).

To avoid the quadratic cost of computing and summing up the contributions from all pairs, cutoff distances are usually introduced, exploiting the fact that physical and heuristic potentials drop off quickly to 0 as the distance between atoms increases. The pairs of

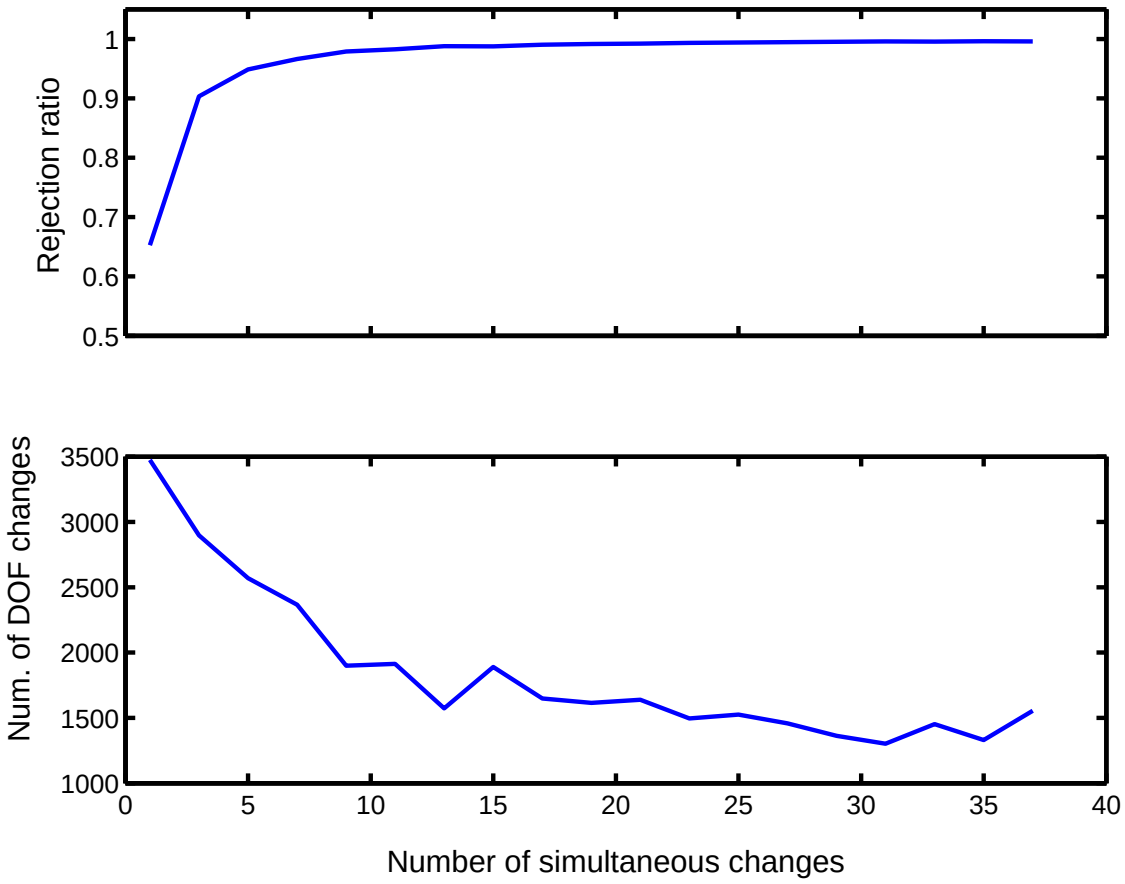


Figure 5.1: Effect of the number of simultaneous DOF changes. Rejection ratio above and total number of changes below.

.

atoms that are close enough to contribute to the energy function will be referred to as the *interacting pairs*. Because van der Waals prevent atom centers from getting very close, the number of interacting pairs in a protein is often less than quadratic [79].

Hence, one may try to reduce computation by finding the interacting pairs without enumerating all atom pairs. A classical method to do this is the grid algorithm (see Subsection 4.2.2), which indexes the position of each atom in a regular three-dimensional grid. This method takes time linear in the number of atoms, which is asymptotically optimal in the worst case. However, it does not exploit an important property of proteins, namely that they form long kinematic chains. It also does not take advantage of the common practice in MCS to change only a few DOFs at each time-step, which entails that at every step

of the simulation large fragments of the protein remain rigid, leaving many partial energy sums unaffected. The grid method re-computes all interacting pairs at each step and cannot directly identify partial sums that have remained constant. Thus it fails to efficiently sum up the contributions of the interacting pairs.

5.2 Related work

The earliest method for computing interactions is called *neighbor lists* (NLs). For each atom, a list of atoms within a distance d somewhat larger than the cutoff distance d_c is maintained by updating it every s steps [195, 203]. The idea is that atoms further apart than d will not come closer than d_c in less than s steps. There is a tradeoff between s and $d - d_c$ since the larger this difference, the larger the value of s that can be used. However, choosing d big causes the NLs to become too large to be efficient. Updating the NLs, which is required every s steps is naïvely quadratic in the number of atoms. A constant factor speed-up is achieved by first generating NLs for bonded atom groups (chemical groups) of size at most 10. The atomic NLs are then generated based on the group lists [20]. A method for updating NLs based on monitoring the displacement of each atom is described in [141]. It is important to note that all NLs methods require spatial-temporal coherence in order to be efficient.

A method for finding interaction based on a 3-D grid was first introduced in the early 70's. In [84] the side-length of each cubic grid cell is set to at least d_c. Each atom is placed in the cell where its center falls, and all atoms found in the immediate neighboring cells (27 cells altogether) are considered as interactions. It is possible to filter these interactions to find only those that are truly within d_c. In [162] the side-length of a grid cell is set to the van der Waals radius of the smallest atom. Thus each cell contains at most one atom. For each atom, all neighboring cells within a volume approximating d_c are scanned for interactions. A combined technique that uses a cell size smaller than d_c but larger than the smallest van der Waals radius was proposed in [201].

In [79] the use of a hash table to index the occupied grid cells was proposed. In the rest of this chapter this method will be referred to as the *grid algorithm*. That work also presented a formal proof that the grid-based methods have linear complexity. It is shown that in a collection B of n possibly overlapping balls of similar radii, such that no two sphere centers are closer than a small fixed distance, the number of balls that intersect any

given ball of B is bounded by a constant. The physics of atoms (i.e. the van der Waals potential) guarantees that these assumptions will hold for all molecules. This bound can be trivially extended to hold for interactions with any fixed cutoff distance by setting the radius of each sphere to $d_c/2$. Since in any interesting simulation each atom has at least one interaction, the grid-based methods are thus asymptotically optimal in the worst case.

Finally, a combination of the grid method and the NLs method was proposed in [213]. There the grid is used to efficiently update the NLs every s steps. A similar combination of grid and group-based NLs was recently proposed [159].

Linear complexity seems to be the best one can achieve in practice when all DOFs change at each step, as is the case in Molecular Dynamics simulation. All the above methods lose efficiency when used in simulations based on internal coordinates, which change only a few DOFs at each step, such as most MCSs. We are not aware of any principled attempt to exploit the chain kinematics of proteins and the small number of changing DOFs at each step to speed up MCS.

5.3 Incremental energy updates in MCS

The method described here is an extension of the testing algorithm presented in Subsection 4.4.2, which allows for incremental updating of the energy of a protein undergoing MCS. A typical energy function is of the form $E = E_1 + E_2$, where E_1 sums terms depending on a single parameter commonly referred to as *bonded* terms (e.g., torsion-angle and bond-stretching potentials) and E_2 sums terms commonly known as *non-bonded* terms, which account for interactions between pairs of atoms or atom groups closer than a cutoff distance [103, 113, 118, 125, 171, 172]. Updating E_1 after a conformational change is straightforward. This is done by computing the sum of the differences in energy in the terms affected by the change and adding it to the previous value of E_1. After a k-DOF change, there are only $O(k)$ affected single-parameter terms. So, in what follows we focus on the update of E_2.

5.3.1 Overview

At each simulation step we must find the interacting pairs of atoms and change E_2 accordingly. Finding these pairs is similar to finding all self-collisions. One may use the same algorithm, after having grown every atom by half the cutoff distance.

However, when k is small, many interacting pairs are unaffected by a k-DOF change. The number of affected interacting pairs, though still $O(n)$ in the worst case, is usually much smaller than the total number of interacting pairs at the new conformation. Therefore, an algorithm like the grid algorithm that computes all interacting pairs at each step is not optimal in practice. Moreover, after having computed the new set of interacting pairs, we still have to update E_2, either by re-computing all pairwise contributions from scratch, or by scanning the old and new sets of interacting pairs to determine which terms should be subtracted from the old value of E_2 and which terms should be added to get the new value. In either case, we perform again a computation at least proportional to the total number of interacting pairs.

Instead, this method directly finds all new interacting pairs, including the previous pairs whose distances have changed. It also detects partial energy sums unaffected by the DOF change (these sums correspond to interacting pairs where both atoms belong to the same rigid sub-chains). The energy terms contributed by the new pairs are then added to the unaffected partial sums to obtain the new value of E_2. In practice, the total cost of this computation is roughly proportional to the number of changing interacting pairs. The algorithm makes use of a new data structure, which is called the *EnergyTree*, in which partial sums computed at previous steps have been cached.

5.3.2 Finding the new interacting pairs

At each step of the simulation all new interacting pairs of atoms need to be found. These include all interacting pairs that were not interacting at the previous conformation, as well as pairs that were previously interacting but whose distances have changed. They do not include, however, previous interacting pairs that are no longer interacting.

This can be done using the ChainTree described in Section 4.3, after having grown all atom radii by half the cutoff distance. The testing algorithm of Subsection 4.4.2 finds exactly the new interacting pairs. However, changing the cutoff distance for the same molecule requires re-computing all OBBs in the ChainTree. Changing this distance can be useful, e.g., to detect all very close pairs of atoms in a first pass and thus quickly discard conformations that are very unfavorable energetically. See Subsection 5.4.3.

This drawback leads to the replacement of OBBs by RSSs (for rectangle swept sphere) and overlap tests of BVs by distance computation. An RSS is a type of BV introduced in [115] that is defined as the Minkowski sum of a sphere and a rectangle. The RSS

bounding a set of points in 3-D is created very much like an OBB. The two principal directions spanned by the points are computed and a rectangle is constructed along these directions to enclose the projection of all points onto the plane defined by these directions. The RSS is the Minkowski sum of this rectangle and the sphere whose radius is half the length of the interval spanned by the point set along the dimension perpendicular to the rectangle. In comparison, the OBB of this set of points is the cross-product of the rectangle by this interval.

The RSS hierarchy is created in the way described in Subsection 4.3.2. The algorithm of Section 4.4.2 is slightly modified to test whether pairs of RSSs are closer apart than the cutoff distance, instead of testing pairs of OBBs for overlap. The search pruning rules for rigid sub-chains are unchanged, so that interacting pairs inside rigid sub-chains are not re-computed. These have already been identified at previous steps of the MCS. The partial energy sums corresponding to these pairs are unchanged and can be retrieved from the EnergyTree as described in the next subsection. To compute the distance between two RSSs, one simply computes the distance between the two underlying rectangles minus the radii of the swept spheres. This is faster than computing the distance between two OBBs, and the BVH is independent of the cutoff distance.

The asymptotic bounds for updating and testing stated in Section 4.5 hold unchanged. For the bound on testing, this derives from the observation that an RSS may be larger than an OBB only by a constant factor. More precisely, the RSS of a set of points is larger than the OBB along its two principal directions by no more than the length of the smallest side of the OBB (the diameter of the swept sphere). Thus, the RSS could easily fit inside an OBB twice as large as the optimal OBB along each dimension. Moreover, computing the distance between two RSSs takes constant time, just like testing two OBBs for overlap.

However, the fact that the asymptotic bound for testing — $\Theta(n^{\frac{4}{3}})$ — is unchanged can be misleading. This bound relies on the fact that protein backbones are well-behaved chains, and as such, the number of atoms that may *interact* with any given atom is bounded by a constant as the protein grows arbitrary large. This constant is likely to be much larger than the constant bounding the number of atoms that may *overlap* any given atom. The situation might even be worse for small proteins, in which almost all pairs of atoms could be interacting. Hence, we may expect the testing algorithm to take significantly more time when it is used to find all new interacting pairs than when it is used to find all steric clashes. This remark will be exploited in Subsection 5.4.3 to propose a two-pass testing algorithm.

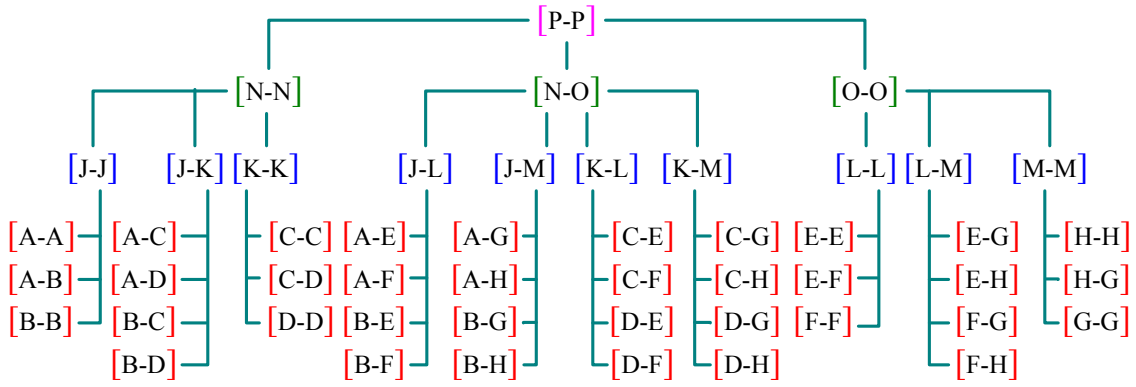


Figure 5.2: EnergyTree for the ChainTree of Figure 4.3

.

5.3.3 Updating the energy value

Recall that when the testing algorithm examines a pair of sub-chains (including the case of two copies of the same sub-chain), it first tests whether these sub-chains have not been affected by the DOF change and are contained in the same rigid sub-chain. If this is the case, the two sub-chains cannot contribute new interacting pairs, and the algorithm prunes this search path. But, for this same reason, the sum of energy terms contributed by the interacting pairs from these sub-chains is also unchanged. We would like to be able to retrieve it. To this end, we introduce another data structure, the *EnergyTree*, in which we cache the partial sums corresponding to all pairs of sub-chains that the testing algorithm may possibly examine. Figure 5.2 shows the EnergyTree for the ChainTree of Figure 4.3.

Let α and β be any two nodes (not necessarily distinct) from the same level of the ChainTree. If they are not leaf nodes, let α_l and α_r (resp., β_l and β_r) be the left and right children of α (β). Let $E(\alpha, \beta)$ denote the total energy contributed by all interacting pairs in which one atom belongs to the sub-chain corresponding to α and the other atom belongs to the sub-chain corresponding to β. If $\alpha \neq \beta$, we have:

$$E(\alpha, \beta) \quad = \quad E(\alpha_l, \beta_l) + E(\alpha_r, \beta_r) + E(\alpha_l, \beta_r) + E(\alpha_r, \beta_l). \tag{5.1}$$

Similarly, the partial energy sum $E(\alpha, \alpha)$ contributed by the interacting pairs inside the

sub-chain corresponding to α can be decomposed as follows:

$$E(\alpha, \alpha) \quad = \quad E(\alpha_l, \alpha_l) + E(\alpha_r, \alpha_r) + E(\alpha_l, \alpha_r). \tag{5.2}$$

These two recursive equations yield the EnergyTree.

The EnergyTree has as many levels as the ChainTree. Its nodes at any level are all the pairs (α, β), where α and β are nodes from the same level of the ChainTree. If $\alpha \neq \beta$ and they are not leaves of the ChainTree, then the node (α, β) of the EnergyTree has four children $(\alpha_l, \beta_l), (\alpha_r, \beta_r), (\alpha_l, \beta_r)$, and (α_r, β_l). A node (α, α) has three children $(\alpha_l, \alpha_l), (\alpha_r, \alpha_r)$, and (α_l, α_r). The leaves of the EnergyTree are all pairs of leaves of the ChainTree (hence, correspond to pairs of links of the protein chain). Each node (α, β) of the EnergyTree holds the partial energy sum $E(\alpha, \beta)$ for the current conformation in the simulation. The root holds the total sum. In the EnergyTree of Figure 5.2, we have $E(P, P) = E(N, N) + E(N, O) + E(O, O)$, $E(N, O) = E(J, L) + E(J, M) + E(K, L) + E(K, M)$, and so on.

At each step, the testing algorithm is called to find new interacting pairs. During this process, whenever the algorithm prunes a search path, it marks the corresponding node of the EnergyTree to indicate that the energy sum stored at this node is unaffected. The energy sums stored in the EnergyTree are updated next. This is done by performing a recursive traversal of the tree. The recursion along each path ends when it reaches a marked node or when it reaches an un-marked leaf. In the second case, the sum held by the leaf is re-computed by adding all the energy terms corresponding to the interacting pairs previously found by the testing algorithm. When the recursion unwinds, the intermediate sums are updated using Equations (5.1) and (5.2). In practice, the testing algorithm and the updating of the EnergyTree are run concurrently, rather than sequentially.

To illustrate, assume that a 1-DOF change has been applied to the chain of Figure 4.3 between F and G. In that case, the testing algorithm marks nodes (N, N), (J, L), (K, L), (L, L) and (M, M) in the EnergyTree. The above recursion re-computes from scratch the partial sums at all the unmarked leaves it encounters and updates the partial sums of all other un-marked nodes it visits as the recursion unwinds using Equations (5.1) and (5.2).

The size of the EnergyTree grows quadratically with the number n of links. For most proteins this is not a critical issue. For example, in our experiments, the memory used by the EnergyTree ranges from 0.4 MB for 1CTF ($n = 137$) to 50 MB for 1JB0 ($n = 1511$). If

needed, however, memory could be saved by representing only those nodes of the EnergyTree which correspond to pairs of RSSs closer than the cutoff distance.

5.4 Experimental results for MCS

5.4.1 Experimental setup

We extended ChainTree as described in the previous section. Since each step of an MCS may be rejected, we keep two copies of the ChainTree and the EnergyTree. RSS and distance computation routines were borrowed from the PQP library [72, 115].

We similarly extended Grid to find interacting pairs by setting the side length of the grid cubes to the cutoff distance. As we mentioned in Subsection 5.3.1, Grid finds all interacting pairs at each step, not just the new ones, and does not cache partial energy sums. So, it computes the new energy value by summing the terms contributed by all the interacting pairs.

Tests were run on a 400 MHz UltraSPARC-II CPU of a Sun Ultra Enterprise 5500 machine with 4.0 GB of RAM.

MCS was performed with the new ChainTree and Grid on the four proteins of Table 4.2. Unlike in the self-collision tests presented earlier, the side-chains were included in the models, as rigid groups of atoms (no internal DOF). So, if two leaf RSSs are within the cutoff distance, ChainTree finds the interacting pairs from the two corresponding links by examining all pairs of atoms. The energy function used for these tests includes a van der Waals potential with a cutoff distance of 6Å, an electrostatic potential with a cutoff of 10Å, and a native-contact attractive quadratic-well potential with a cutoff of 12Å. Hence, the cutoff distance for both ChainTree and Grid was set to 12Å.

Each simulation run consisted of 300,000 trial steps. The number k of DOFs changed at each step was constant throughout a run. We performed runs with $k = 1, 5$ and 10. Each change was generated by picking k backbone DOFs at random and changing each DOF independently with a magnitude picked uniformly at random between 0° and 12°. Each run started with a random, partially extended conformation of the protein. Since the van der Waals term for a pair of atoms grows as $O(d^{-12})$ where d is the distance between the atom centers, it quickly approaches infinity as d becomes small (steric clash). When a van der Waals term was detected to cross a very large threshold, the energy computation was halted (in both ChainTree and Grid), and the step was rejected.

	$k = 1$		$k = 5$		$k = 10$	
	ChainTree	Grid	ChainTree	Grid	ChainTree	Grid
1CTF	7.82	27.7	8.34	18.22	12.57	15.07
1LE2	11.16	65.05	14.31	48.84	14.29	27.12
1HTB	16.72	130.9	18.2	81.86	21.75	60.33
1JB0	21.71	271.4	22.18	130.5	29.88	133.8

Table 5.1: MCS results: average time per simulation step (in milliseconds)

	$k = 1$		$k = 5$		$k = 10$	
	ChainTree	Grid	ChainTree	Grid	ChainTree	Grid
1CTF	5,100	25,100	7,400	16,900	8,000	13,500
1LE2	5,100	48,500	6,000	36,600	7,700	23,400
1HTB	5,400	100,000	7,000	56,800	8,200	43,100
1JB0	5,900	200,000	7,000	95,600	10,300	102,000

Table 5.2: MCS results: number of interacting pairs for which energy terms were evaluated, per step.

We ran the same MCS with both methods on each protein, by starting at the same initial conformation and using the same seed of the random-number generator. Note that for any given protein conformation ChainTree and Grid compute the exact same energy value.

5.4.2 Results

The results for all the experiments are found in Tables 5.1 and 5.2. Illustrations of the average time results for $k = 1$ and $k = 5$ are presented in Figures 5.3(a) and 5.3(b) respectively. As expected, ChainTree gave its best results for $k = 1$, requiring on average one quarter of the time of Grid per step for the smallest protein (1CTF) and one twelfth of the time for the largest protein (1JB0). The average number of interacting pairs for which energy terms were evaluated at each step was almost 5 times smaller with ChainTree than with Grid for 1CTF and 30 times smaller for 1JB0.

Similar results are obtained when $k = 5$. In this case, ChainTree was only twice as fast as Grid for 1CTF and 6 times faster for 1JB0. The average number of interacting pairs for which energy terms were evaluated was twice smaller with ChainTree for 1CTF and 14 times smaller for 1JB0.

When $k = 10$, the relative effectiveness of ChainTree declined further, being only 1.2

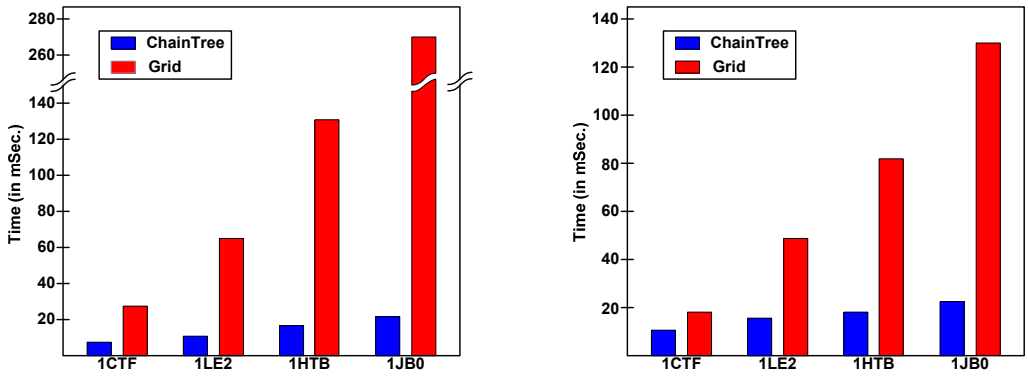


Figure 5.3: Comparing the average time per MCS step of ChainTree and Grid when (a) $k = 1$ and (b) $k = 5$.

times faster than Grid for 1CTF and 4 times faster for 1JB0. The average number of interacting pairs for which energy terms were evaluated using ChainTree was 60% of the number evaluated using Grid for 1CTF and 10 times smaller for 1JB0.

These results are consistent with those obtained for self-collision detection (Section 4.6). The larger k, the less effective our algorithm compared with Grid. When k is small, there are few new interacting pairs at each step, and ChainTree is very effective in exploiting this fact. For both ChainTree and Grid the average time per step decreases when k increases. This stems from the fact that a larger k is more likely to yield over-threshold van der Waals terms and so to terminate energy computation sooner.

In order to examine the full effect of reusing partial energy sums, the simulations are re-run for the four proteins without the van der Waals threshold for $k = 1$ and 5. The results are presented in Tables 5.3 and 5.4. Removing the van der Waals threshold does not significantly alter the behavior of the algorithms. The average time per step is of course larger, since no energy computation is cut short by a threshold crossing. The relative speed-up of ChainTree over Grid is only slightly smaller without the threshold.

5.4.3 Two-pass ChainTree

In the previous MCS the percentage of steps that were rejected before energy computation completed, due to an above-threshold van der Waals term for 1CTF, for example, rose from 60% when $k = 1$ to 98% when $k = 10$. This observation not only motivates choosing a

	$k = 1$		$k = 5$	
	ChainTree	Grid	ChainTree	Grid
1CTF	12.8	37.2	29.6	37.7
1LE2	20.9	86.5	24.6	65.4
1HTB	26.6	185	51.8	173
1JB0	40.0	401	89.1	348

Table 5.3: MCS results without threshold on the van der Waals terms: average time (in milliseconds) per step.

	$k = 1$		$k = 5$	
	ChainTree	Grid	ChainTree	Grid
1CTF	8,600	33,300	21,000	34,700
1LE2	9,900	61,900	11,400	47,500
1HTB	9,900	134,000	21,500	129,000
1JB0	12,000	280,000	30,300	248,000

Table 5.4: MCS results without a threshold on the van der Waals terms: average number of interacting pairs for which energy terms were evaluated, per step.

small k. It also suggests the following two-pass approach. In the first pass, ChainTree uses a very small cutoff distance chosen such that atom pairs closer than this cutoff yield above-threshold van der Waals terms. In this pass, the algorithm stops as soon as it finds an interacting pair, and then the step is rejected. In the second pass the cutoff distance is set to the largest cutoff over all energy terms and ChainTree computes the new energy value. We refer to the implementation of this two-pass approach as ChainTree+. Since ChainTree is much faster with a small cutoff and the first pass will often result in step rejection, we can expect ChainTree+ to be significantly faster than ChainTree. Thanks to the modifications described in Subsection 5.3.2, both passes use the same ChainTree data structure.

ChainTree and ChainTree+ are compared by running an MCS of 300,000 trial steps with $k = 5$ and measuring the average time per step. The results for the four proteins are given in Table 5.5. We ran two different simulations for each protein. One that started at a partially extended conformation and another that started at the folded state of the protein. Hence, the conformations reached in the first case were less compact than in the second case. Consequently, the rejection rate due to self-collision was higher in the second case. While ChainTree+ is faster in both cases, speed-up factors are greater (as much as 5) when starting from the folded state.

It is not clear whether a similar improvement could be obtained with Grid. Indeed,

	unfolded		folded	
	ChainTree	ChainTree+	ChainTree	ChainTree+
1CTF	8.34	2.6	15.74	6.2
1LE2	14.31	6.4	32.37	9.06
1HTB	18.2	9.23	68.92	11.35
1JB0	22.18	6.33	81.15	15.51

Table 5.5: Average running times (in mSec.) of ChainTree and ChainTree+ per step.

the resolution of the indexing grid depends on the cutoff distance. Indexing atoms in two different grids — one with small cells for detecting steric clashes and another with larger cells for computing the energy — may then considerably reduce the advantage of the two-pass approach.

5.5 MCS software

The implementation of the ChainTree algorithm was extended to include a physical, full-atomic energy function. The force-field that was chosen is *EEF1* [118]. This force field is based on the CHARMM19 potential energy function [20] with an added implicit solvent term. *EEF1* was chosen because it has been shown to discriminate well between folded and misfolded structures [117]. It is well suited for ChainTree because its implicit solvent term is pairwise and thus our algorithm can compute it efficiently. We have packaged our software into a program that runs fast MCS. It can be downloaded from `http://robotics.stanford.edu/~itayl/mcs`.

This software loads an initial structure that is described in terms of its amino acid sequence and the corresponding backbone angles of each residue. It then performs a classical MCS. The user can control some parameters of the simulation (e.g. the number of angles to change, the length of the simulation, the temperature ...) by specifying them on the command line.

Our protein model uses fixed rotamers for each amino acid, and thus a given side-chain is limited to a fixed number of conformations during the simulation. The advantage of using fixed rotamers is that the atom positions and internal energy of each rotamer can be pre-computed. Hence switching between rotamers during the simulation is done extremely fast. We used the *backbone-independent rotamer library* of the Dunbrack lab [50].

5.6 Other applications

Although we have presented the application of our algorithm to classical Metropolis MCS, it can also be used to speed up other MCS methods as well as other optimization and simulation methodologies.

For example, MCS methods that use a different acceptance criterion can benefit from the same kind of speed-up as reported in Section 5.4, since the speed-up only derives from the faster maintenance of the energy function when relatively few DOFs are changed simultaneously, and is independent of the actual acceptance criterion. Such methods include Entropic Sampling MC [121], Parallel Hyperbolic MC [218], and Parallel-hat Tempering MC [219]. MCS methods that use Parallel Tempering [83] (also known as Replica Exchange) such as [218, 219], which require running a number of replicas in parallel, could also benefit by using a separate ChainTree and EnergyTree for each replica.

Some MCS methods use more sophisticated move sets (trial step generators). Again, our algorithm can be applied when the move sets do not change many DOFs simultaneously, which is in particular the case of the moves sets proposed in [2, 3] (biasing the random torsion changes), and in [51] and [179] (moves based on fragment replacement). More computationally intensive step generators use the internal forces (the gradient of the energy function) to bias the choice of the next conformation (e.g., Force-Biased MC [154], Smart MC [102] and MC plus minimization [125]). For such step generators, the advantage of using our algorithm is questionable, since they may change all DOFs at each step. Similarly, the so-called Hybrid MC methods [19, 215] that combine MCS and Molecular Dynamics Simulation would benefit from our algorithm only when relatively few DOFs are changed at each step. The same is true for pure MDS in torsion-angle space [76, 186].

Some optimization approaches could also benefit from this algorithm. For instance, a popular one uses genetic algorithms with crossover and mutation operators [157, 189, 198]. The crossover operator generates a new conformation by combining two halves, each extracted from a previously created conformation. Most mutation operators also reuse long fragments from one or several previous conformations. For both types of operators, our algorithm would allow partial sums of energy terms computed in each fragment to be re-used, hence saving considerable amounts of computation.

Chapter 6

Conclusion

In this thesis three novel algorithms were presented that target some of the key problems in computational structural biology. These methods exploit the long chain structure of proteins as well as some application-specific properties to perform the desired computation efficiently.

1. The first algorithm, presented in Chapter 2, exploits properties of closed chains to offer an automatic procedure for completing missing fragments in protein structures resolved using X-ray crystallography. The problem of searching for a conformation of a protein fragment that bridges a gap in a protein structure is posed as an inverse kinematics problem, limiting the search space to the self-motion manifold of the fragment. A two-step algorithm is proposed that first rapidly generates a large set of candidate closed conformations, from which a small subset is chosen for refinement against the electron density map. In a test set of 103 structurally diverse fragments, the algorithm closed gaps of 12 residues in length to within, on average, 0.52Å all-atom cRMS of the final, refined structure at a resolution of 2.8Å. In an initial, 51%-complete model built at 2.6Å, it closed a 14-residue gap to within 0.9Å all-atom cRMS, thus extending automation of model building towards lower resolution levels. Preliminary results indicate that the method can also be useful in determining alternative backbone conformations in areas of ambiguous electron density.

2. The second algorithm, presented in Chapter 3, takes advantage of properties of distances between the links of a chain. These properties motivate an averaging method, which speeds up the computation of similarity between protein structures. The Cα

representation of the backbone is compressed by averaging the coordinates of short sub-chains, letting each sub-chain be represented by a single 3-D coordinate. This shorter representation is then used to approximately compute the structural similarity between two proteins. The method offers a tradeoff between the error that is introduced and the speed-up that is gained. It is applicable to tasks that require the computation of similarity for all pairs of structures in a large data set, both sets of conformations of the same protein and native folds of different proteins. It can also be used as a fast filter in cases where exact similarity computation is required. Computational tests were conducted in the context of two applications, nearest neighbor search and automatic structural classification, yielding good results.

3. The third algorithm, presented in Chapters 4 and 5, exploits properties of open chains together with properties of Monte Carlo simulation of proteins. A protein backbone undergoing a Monte Carlo simulation is a kinematic chain, free-floating in a force-field and deforming through changes applied to a few of its DOFs at each simulation step. The deforming chain is represented using the ChainTree data structure, which makes updates and self-collision queries efficient to compute. This data structure represents the chain geometry and kinematics at successively decreasing levels of resolution. The ChainTree is further extended to efficiently update the energy of a protein during Monte Carlo simulations. It is augmented to efficiently find all pairs of atoms closer than a specified cutoff distance, which is needed for calculating the energy of protein conformations, a computation dominated by pairwise atomic interactions. A companion data structure — the EnergyTree — is introduced to efficiently reuse partial energy sums, which remain unaffected between simulation steps. Computational tests on four proteins of sizes ranging from 68 to 755 amino acids show that Monte Carlo simulation with the ChainTree method is significantly faster (as much as 12 times faster for the largest protein) than with the widely used grid method. They also indicate that speed-up increases with the size of the simulated protein.

In this work existing protein representations such as the torsion angle model or the Cα representation are used and the improved efficiency and novel methods are a result of exploiting specific properties of these models. The use of concepts and techniques from robotics and computational geometry to exploit the long chain kinematics of proteins that is implicit in these models is the common thread of these algorithms and what makes

them efficient. The development of new protein models that lend themselves better to efficient computational methods would bring about even faster algorithms. In the long run, I believe, this will require a scientist that is not only proficient in the chemistry and physics of proteins, but is also an expert in physical computing, well-versed in the techniques of computational geometry and robotics. Such a scientist would be in the ideal position to suggest novel simplified protein models that allow efficient computation beyond what is currently possible without compromising physical accuracy and biological relevance.

Appendix A

Proof of Theorem 1

In this appendix we give a proof of Theorem 1 stated in Section 4.5.2. Along the way we establish successive useful lemmas.

In order to simplify our proof, we first "regularize" a well-behaved kinematic chain as follows:

1. We replace each link by an enclosing sphere whose radius is equal to that of the largest minimal enclosing sphere of any link in the chain.

2. We grow all these spheres equally until no two consecutive spheres in the chain are disjoint (of course some or many of the spheres may already be intersecting).

So, the links of the new chain are spheres of equal radius r. The new chain is also well-behaved since the regularization only change the size of the enclosing spheres by a constant factor, when the distance between the centers of the enclosing spheres of any two successive links of the original chain is lower bounded by a constant (which is the case for proteins). Therefore, the maximal number of links of the new chain that can overlap a single link is still bounded by a constant, though this constant may be greater than the one for the original chain. None of the asymptotic bounds below are affected by this change of constant.

From now on, we only consider the regularized chain. We distinguish between two types of BVH for this chain. In the *tight* hierarchy, each BV tightly bounds the links it encloses. In the *not-so-tight* one, each non-leaf BV bounds tightly the two BVs just below it. In a tight sphere hierarchy, each bounding sphere is the minimum enclosing sphere of the links it encloses.

Proposition 1 and Lemmas 1 through 3 establish the upper bound of Theorem 1. Proposition 2 then presents a chain conformation for which this bound is actually attained, hence showing that the bound is tight.

Proposition 1 *In the tight, chain-aligned sphere hierarchy of a well-behaved n-link chain, the maximum total number of overlapping bounding spheres at all levels is* $O(n^{\frac{4}{3}})$.

Proof: At level i of the hierarchy, where $i = 0, 1, \ldots, \log n$, with $i = 0$ being the lowest level, each consecutive sub-chain of $g_i = 2^i$ links is bounded by a single sphere of radius at most $g_i r$. This radius occurs when the links are arranged on a straight line. Take any such sub-chain of g_i links and let B_i denote its bounding sphere. Let C_i be the sphere concentric with B_i and having radius $3g_i r$. Any bounding sphere at level i intersecting B_i is fully contained in C_i.

Now let us bound the number of sub-chains of g_i links whose bounding spheres at level i can be contained in C_i. Since we aim for an upper bound we assume that each such sub-chain is as tightly packed as possible. Because each link can overlap at most a constant number of other links the volume occupied by a sub-chain of length g_i is bounded from below by $qg_i\frac{4}{3}\pi r^3$, where $0 < q \leq 1$ is a constant. Hence, the number of bounding spheres at level i contained in C_i is at most:

$$M_i \;=\; \frac{\frac{4}{3}\pi 27 g_i^3 r^3}{\frac{4}{3} q g_i \pi r^3} \;=\; \frac{27 g_i^2}{q} \tag{A.1}$$

M_i is therefore an upper bound on the number of bounding spheres at level i that overlap B_i. Since there are exactly n/g_i bounding spheres at level i, M_i cannot grow larger than $\frac{n}{g_i}$. The level i_{max} where this bound is reached is the smallest i defined by:

$$\frac{n}{g_i} \geq \frac{27 g_i^2}{q}.$$

Since $g_i = 2^i$, we get:

$$i_{max} = \left\lceil \frac{1}{3}\log n + \log \frac{1}{3} q^{\frac{1}{3}} \right\rceil. \tag{A.2}$$

For every $i < i_{max}$, $T_i = \frac{n}{g_i} M_i$ is an upper bound on the number of overlapping bounding spheres at level i. For every $i \geq i_{max}$, $T_i = (\frac{N}{g_i})^2$ is an upper bound on this number. In

what follows, we ignore the constant term on the right-hand side of Equation (A.2) as it has no effect on the asymptotic bound we prove. The total number of overlapping bounding spheres at all levels is therefore:

$$\begin{aligned} T &= \sum_{i=0}^{\log n} T_i \\ &= \sum_{i=0}^{\frac{1}{3}\log n} \left(\frac{27g_i^2}{q}\right)\left(\frac{n}{g_i}\right) + \sum_{i=\frac{1}{3}\log n}^{\log n} \left(\frac{n}{g_i}\right)^2 \\ &= \frac{27n}{q} \sum_{i=0}^{\frac{1}{3}\log n} 2^i + n^2 \sum_{\frac{1}{3}\log n}^{\log n} \left(2^{-i}\right)^2 \\ &= \frac{27n}{q}\left(2n^{\frac{1}{3}} - 1\right) + \frac{4}{3}\left(n^{\frac{4}{3}} - 1\right) \\ &= O\left(n^{\frac{4}{3}}\right) \end{aligned} \tag{A.3}$$

□

As a side note, this proof extends with minor changes to spaces of any dimension $d \geq 3$. We then get an upper bound of $O\left(n^{\frac{2(d-1)}{d}}\right)$.

One should note that the upper bound of Proposition 1 holds for any chain-aligned BVH (tight or not) as long as each BV at level i can be enclosed in a sphere of radius $c\,2^i r$, for an absolute constant c. We now use this remark to show that the upper bound holds for the OBB hierarchy of the ChainTree.

Lemma 1 *Given two OBBs contained in a sphere D of radius R, the OBB bounding both of them is contained in a sphere of radius $\sqrt{3}R$ concentric with D.*

Proof: We let b_1 and b_2 denote the two OBBs contained in D and B_{12} denote their OBB. Construct the bounding cube Q of D whose faces are parallel to those of B_{12}. Q contains B_{12} since along any of the main axes that define B_{12}, the faces of Q are farther out (or touching). So, the bounding sphere E of Q, which is concentric with D, also contains B_{12}. Since each side of Q has length $2R$, its diagonal has length $2\sqrt{3}R$ and the radius of E is $\sqrt{3}R$. □

Lemma 2 *At level i of the not-so-tight, chain-aligned OBB hierarchy of a well-behaved n-link chain, each OBB is contained in a sphere of radius $c\,2^i r$, where c is an absolute constant.*

Proof: Let us choose c_1 such that each OBB at levels $i = 0, 1, \ldots, 4$ of the hierarchy is contained in a sphere of radius $c_1\,2^i r$. We know c_1 exists since there are at most 16 spherical links enclosed by each OBB at these levels. We will take the constant c to be at least c_1.

We now proceed by induction, by assuming that the lemma is correct up to some level $i-1$ ($i \geq 5$). Consider 32 consecutive OBBs b_j, $j = 0, \ldots, 31$, at level $i-5$ that are bounded together in an OBB of level i. The induction hypothesis implies that the bounding sphere of each b_j has radius at most $c\,2^{i-5} r$. We take a sphere S of radius $2^i r$ that contains the sub-chain bounded by the $32 b_j$ boxes. Now each sphere bounding one of the b_j boxes must intersect S since at least one link is contained in both spheres. So, no point of box b_j is further away than $2^i r(1 + \frac{c}{16})$ from the center of S.

Let S_0 be the sphere of radius $2^i r(1 + \frac{c}{16})$ concentric with S. All boxes b_j are contained in S_0. Consider pairs which will be bounded together at the next level ($i - 4$). We apply Lemma 1 to those pairs and realize that a sphere S_1 of radius $\sqrt{3}$ times the radius of S_0 and concentric with S_0 bounds all OBBs at level $i - 4$. Continuing this line of reasoning up to level i we get that a sphere S_5 of radius $\sqrt{3}^5\, 2^i r(1 + \frac{c}{16})$ contains the OBB at level i that encloses all of the boxes b_j. We must set c such that this radius is smaller than $c\,2^i r$. Thus, c must be such that:

$$\begin{aligned} 2^i r \left(1 + \frac{c}{16}\right) \sqrt{3}^5 &\leq c\,2^i r \\ c\left(\frac{1}{\sqrt{3}^5} - \frac{1}{16}\right) &\geq 1 \end{aligned} \tag{A.4}$$

So we choose:

$$c = \max\left\{ \left(\frac{1}{\sqrt{3}^5} - \frac{1}{16}\right)^{-1}, c_1 \right\}.$$

□

Lemma 2 assures us that the OBBs in the OBB hierarchy of the ChainTree do not become too big as we climb up the hierarchy, so that the upper bound of Proposition 1 applies to the ChainTree.

Lemma 3 *In the not-so-tight, chain-aligned OBB hierarchy of a well-behaved n-link chain, the maximum number of overlapping bounding boxes at all levels is $O(n^{\frac{4}{3}})$.*

Proof: We have noted previously that the proof given for Proposition 1 holds for any chain-aligned BVH (tight or not) of a well-behaved n-link chain, as long as each BV at level i can be enclosed in a sphere of radius $c\,2^i r$, for an absolute constant c. Lemma 2 establishes that this is verified by the not-so-tight, chain-aligned OBB hierarchy of the chain. □

This completes the proof of Proposition 1, thus establishing the upper bound of Theorem 1. We now show that this bound can be attained.

Proposition 2 *There exists a well-behaved chain of n links such that the number of BV overlap tests needed to detect self-collision in certain conformations is $\Omega(n^{\frac{4}{3}})$ for any chain-aligned BVH of convex BVs.*

Proof: We prove the lemma by presenting a chain conformation that requires this much work no matter what the BVs are as long as they are convex. Given some coordinate frame, we take the first d links of the chain and place them along the X axis starting at the origin and proceeding in the positive direction. The spheres are osculating and the center of the first sphere is at the origin. The center of the dth link is therefore at $x_0 = 2(d-1)r$. We now place the next d links of the chain parallel to the Y axis, starting at $(x_0, 2r, 0)$ and going in the positive direction. Next, we place the next d links parallel to the Z axis starting just above where the previous d-link sub-chain ended. We call this sub-chain of $3d$ links a *unit*. With the next additional links of the chain we create $\frac{d}{8}-1$ more units, each in reverse order of the previous one and translated by $(2r, -2r, 0)$ relative to the previous one. We call these $\frac{d}{8}$ units a *layer*. Figure A.1 shows one such layer. Finally, with the rest of the chain, we create $\frac{d}{8}-1$ layers, each in reverse order of the previous one and translated by $(0, -2r, 2r)$ relative to the previous one. See Figure A.2.

We noticed that the convex hull of each unit contains the point $(2(d-1)r, (d-1)r, \frac{1}{4}(d-1)r)$. So, all these convex hulls are pairwise intersecting and any hierarchy of convex BVs will therefore contain that many intersecting pairs of BVs at the level where all the links of each unit are enclosed together in one BV. Since we created $\frac{d^2}{64}$ units in total, each with $3d$ links, so have $\frac{3d^3}{64} = n$. Hence, d is a small constant times $n^{\frac{1}{3}}$. The $\frac{d^2}{64}$ units yield $\Omega(n^{\frac{2}{3}})$ convex hulls and therefore $\Omega(n^{\frac{4}{3}})$ intersecting pairs of convex hulls. □

Together, Lemma 3 and Proposition 2 prove Theorem 1.

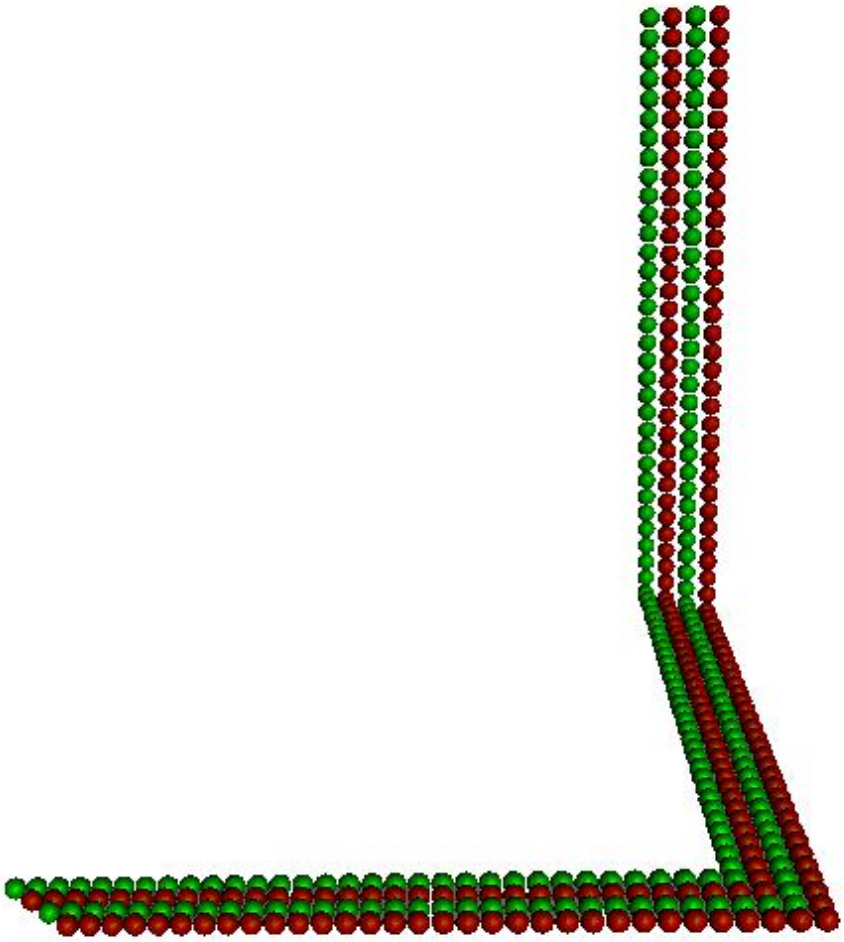


Figure A.1: One layer of the chain construction.

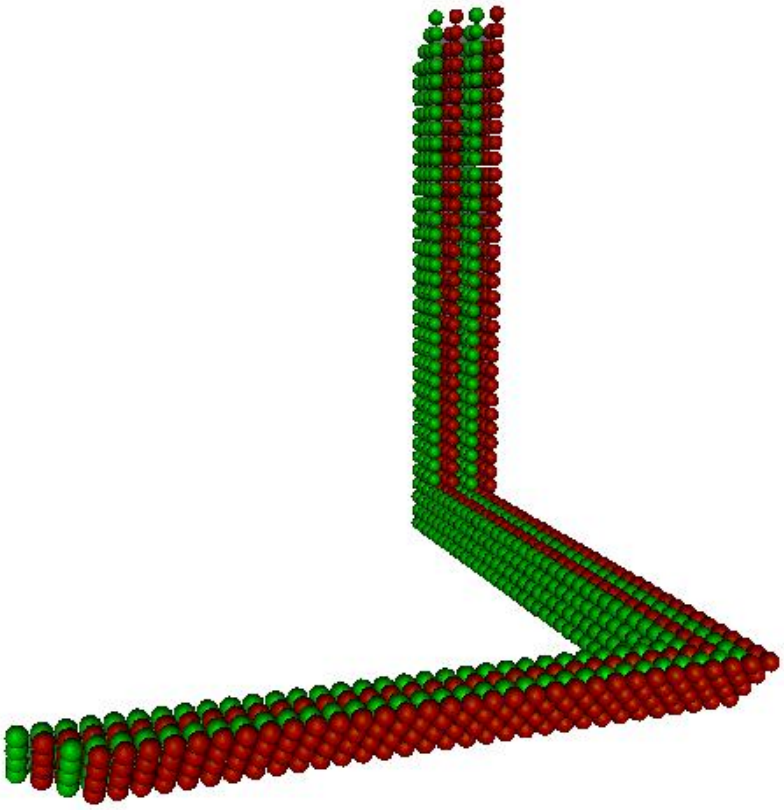


Figure A.2: The entire chain construction.

Glossary

A

amino acid The family of small molecules, which constitute the building blocks of all proteins. There are 20 naturally ocurring amino acids, which vary in size and chemical properties.

B

backbone The main chain of the protein. It is a concatrnation of atom triplets (N, Cα and C$'$), each belonging to a residue of the protein.

BVH Bounding volume hierarchy.

C

CASP Critical Assessment of Techniques for Protein Structure Prediction.

conformation The chemical equivalent term for a configuration in robotics.

cRMS Coordinate root mean squares. The RMS distances between corresponding atoms aftr the two structures are optimally aligned using a rigid-body transform.

D

DOF Degree of freedom.

dRMS Distance root mean squares. The RMS between the intra-molecular distance matrices of the two structures.

M

MCS Monte Carlo simulation.

MDS Molecular dynamics simulation.

N

NMR Nuclear magnetic resonance.

O

OBB Oriented bounding box.

P

PDB Protein data bank.

R

residue An amino acid when it is part of a polypeptide (protein) chain.

RMS Root mean squares.

RMSD RMS distance (deviation).

RSS Rectangle swept sphere.

S

self-motion manifold A lower-dimensional manifold of all closed configurations of a chain. embedded in the configration space of the chain.

side-chain A small group of atoms stemming from the Cα atom of each residue. May have between 0 and 10 heavy atoms.

V

van der Waals The repulsion between non-bonded atoms, it is very positive (replusive) at small distances, but has a slightly negative (attractive) minimum at a distance

at which the two atoms touch electron clouds. At large distances the energy goes to zero. Ususally described using a Lennard-Jones potential.

Bibliography

[1] R. Abagyan and P. Argos. Optimal protocol and trajectory visualization for conformational searches of peptides and proteins. *J. Mol. Biol.*, 225:519–532, 1992.

[2] R. Abagyan and M. Totrov. Biased probability monte carlo conformational searches and electrostatic calculations for peptides and proteins. *J. Mol. Biol.*, 235:983–1002, 1994.

[3] R. Abagyan and M. Totrov. Ab initio folding of peptides by the optimal-bias Monte Carlo minimization procedure. *J. Comp. Phys.*, 151:402–421, 1999.

[4] A. Abe, W. Braun, T. Noguti, and N. Gō. Rapid calculation of first and second derivatives of conformational energy with respect to dihedral angles for proteins. general recurrent equations. *Comput. Chem.*, 8(4):239–247, 1984.

[5] P. Agarwal. Range searching. In J. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 575–598. CRC Press, 1997.

[6] P. Agarwal, J. Basch, L. Guibas, J. Hershberger, and L. Zhang. Deformable free space tilings for kinetic collision detection. In B.R. Donald, K.M. Lynch, and D. Rus, editors, *Alg. and Comp. Robotics: New Directions (WAFR '00)*, pages 83–96. A.K. Peters, Wellesley, 2000.

[7] J.M. Ahuactzin and K.K. Gupta. The kinematic roadmap: a motion planning based global approach for inverse kinematics of redundant robots. *IEEE Trans. Robot. Autom.*, 15(4):653–669, August 1999.

[8] N. M. Amato, K. A. Dill, and G. Song. Using motion planning to map protein folding landscapes and analyze folding kinetics of known native structures. *J. Comput. Biol.*, 10(3):239–255, 2003.

[9] N. M. Amato and G. Song. Using motion planning to study protein folding pathways. *J. Comput. Biol.*, 9(2):149–168, 2002.

[10] M. S. Apaydin, D. L. Brutlag, C. Guestrin, D. Hsu., and J.-C. Latombe. Stochastic conformational roadmaps for computing ensemble properties of molecular motion. In *Workshop Algo. Found. Robot. (WAFR)*, Nice, France, Dec. 2002.

[11] M. S. Apaydin, D. L. Brutlag, C. Guestrin, D. Hsu, J.-C. Latombe, and C. Varma. Stochastic roadmap simulation: An efficient representation and algorithm for analyzing molecular motion. *J. Comput. Biol.*, 10(3):257–281, 2003.

[12] A. Aszódi, M. J. Gradwell, and W. R. Taylor. global fold determination from a small number of distance restraints. *J. Mol. Biol.*, 251(2):308–326, 1995.

[13] J. Badger. An evaluation of automated model-building procedures for protein crystallography. *Acta Cryst.*, D59:823–827, 2003.

[14] J. Bentley. multidimensional binary search trees used for associative searching. *commun. ACM*, 18:509–517, 1975.

[15] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, and et al. The protein data bank. *Nucl. Acids Res.*, 28:235–242, 2000.

[16] F. C. Bernstein, T. F. Koetzle, G. Williams, D. J. Meyer, M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The Protein Data Bank: a computer-based archival file for macromulecular structures. *J. Mol. Biol.*, 112:535–542, 1977.

[17] K. Binder and D. Heerman. *Monte Carlo Simulation in Statistical Physics*. Springer Verlag, Berlin, 2nd edition, 1992.

[18] P. Bradley, D. Chivian, J. Meiler, K. M. S. Misura, C. A. Rohl, W. R. Schief, W. J. Wedemeyer, O. Schueler-Furman, P. Murphy, J. Schonbrun, C. E. M. Strauss, and D. Baker. Rosetta predictions in CASP5: Successes, failures, and prospects for complete automation. *Proteins*, 53(S6):457–468, 2003.

[19] A. Brass, B. Pendelton, Y. Chen, and B. Robson. Hybrid Monte Carlo simulation theory and initial comparison with molecular dynamics. *Biopolymers*, 33:1307–1315, 1993.

[20] B. Brooks, R. Bruccoleri, B. Olafson, D. States, S. Swaminathan, and M. Karplus. CHARMM: a program for macromolecular energy minimizationand dynamics calculations. *J. Comput. Chem*, 4:187–217, 1983.

[21] J. Brown, J.-C. Latombe, and K. Montgomery. Real-time knot tying simulation. *Visual Comput.*, 20(2-3):165–179, 2004.

[22] J. Brown, S. Sorkin, J.-C. Latombe, K. Montgomery, and M. Stephanides. Algorithmic tools for real time microsurgery simulation. *Med. Image Anal.*, 6(3):289–300, 2002.

[23] S. Brown and T. Head-Gordon. Physico-chemical minimalist models for protein folding and design. *Curr. Opin. Struct. Biol.*, 13:160–167, 2003.

[24] R. Bruccoleri and M. Karplus. Prediction of the folding of short polypeptide segments by uniform conformational sampling. *Biopolymers*, 26:137–168, 1987.

[25] J.W. Burdick. On the inverse kinematics of redundant manipulators: characterization of the self-motion manifolds. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 1, pages 264–270, May 1989.

[26] S. Cameron. Collision detection by four-dimensional intersection testing. *IEEE Tr. Robotics Automat.*, 6(3):291–302, June 1990.

[27] A.A. Canutescu and R.L. Dunbrack Jr. Cyclic coordinate descent: A robotics algorithm for protein loop closure. *Prot. Sci.*, 12:963–972, 2003.

[28] K.-S. Chang and O. Khatib. oprerational space dynamics: efficient algorithms for modeling and control of branching mechanisms. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 850–856, San Francisco, April 2000.

[29] M.S. Chapman. Restrained real-space macromolecular atomic refinement using a new resolution-dependent electron-density function. *Acta Cryst.*, A51(1):69–80, 1995.

[30] I.D. Chen, Y.-C.; Walker. A consistent null-space based approach to inverse kinematics of redundant robots. In *IEEE Int. Conf. Robot. Autom*, volume 3, pages 374–381, May 1993.

[31] G.S. Chirikjian. General methods for computing hyper-redundant manipulator inverse kinematics. In *IEEE/RSJ Int. Conf. Intel. Robots and Sys.*, Yokohama, Japan, July 1993.

[32] K. Clarkson. Nearest neighbor queries in metric spaces. In *Proc. Symp. Theo. Comp. (SToC)*, pages 609–617, 1997.

[33] H. Claussen, C. Buning, M. Rarey, and T. Lengauer. FlexE: efficient molecular docking considering protein structure variations. *J. mol. Biol.*, 308:377–395, 2001.

[34] J. D. Cohen, M. C. Lin, D. Manocha, and M. Ponamgi. I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In *Sym. on Interactive 3D Graphics*, pages 189–196, 218, 1995.

[35] V. Collura, J. Higo, and J. Garnier. Modeling of protein loops by simulated annealing. *Prot. Sci.*, 2:1502–1510, 1993.

[36] B. Contreras-Moreira, P. W. Fitzjohn, Marc Offman, G. R. Smith, and P. A. Bates. Novel use of a genetic algorithm for protein structure prediction: Searching template and sequence alignment space. *Proteins*, 53(S6):424–429, 2003.

[37] J. Cortés, T. Siméon, M. Remaud-Siméon, and V. Tran. Geometric algorithms for the conformational analysis of long protein loops. *J. Comp. Chem.*, 25(7):956–967, 2004.

[38] J. Cortés, T. Simóon, and J.-P. Laumond. A random loop generator for planning the motions of closed kinematic chains using prm methods. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 2141–2146, May 2002.

[39] E.A. Coutsias, C Seok, M.P. Jacobson, and K.A. Dill. A kinematic view of loop closure. *J. Comput. Chem.*, 25:510–528, 2004.

[40] K. D. Cowtan. Clipper libraries, 2004. http://www.ysbl.york.ac.uk/ cowtan/clipper/clipper.html.

[41] J.J. Craig. *Introduction to robotics: manipulation nad control.* Addison-Wesley, 2nd edition, 1989.

[42] T. E. Creighton. *Proteins : Structures and Molecular Properties.* W. H. Freeman and Company, New York, 2nd edition, 1993.

[43] C.M. Deane and T.L. Blundell. A novel exhaustive search algorithm for predicting the conformation of polypeptide segments in proteins. *Proteins*, 40(1):135–144, 2000.

[44] M.A. DePristo, P.I. de Bakker, S.C. Lovell, and T.L. Blundell. Ab initio construction of polypeptide fragments: efficient generation of accurate, representative ensembles. *Proteins*, 51(1):41–55, 2003.

[45] R. Diamond. A real-space refinement procedure for proteins. *Acta Cryst.*, A27(5):436–452, 1971.

[46] D. Dobkin and D. Kirkpatrick. Determining the separation of preprocessed polyhedra – a unified approach. In *Proc. 17th Int. Colloq. Automata Lang. Program. Lecture Notes Comput. Sci.*, volume 443, pages 400–413, 1990.

[47] C. M. Dobson. Principles of protein folding, misfolding and aggregation. *Semin. Cell. Dev. Biol.*, 15(1):3–16, 2004.

[48] J. Drenth. *Principles of protein X-ray crystallography.* Springer Verlag, New York, 2nd edition, 1999.

[49] P. Du, M Andrec, and R.M. Levy. Have we seen all structures corresponding to short protein fragments in the protein data bank? an update. *Prot. Engin.*, 16(6):407–414, 2003.

[50] R. Dunbrack and F. Cohen. Bayesian statistical analysis of protein sidechain rotamer preferences. *Protein Sci.*, 6:1661–81, 1997.

[51] A. Elofsson, S. LeGrand, and D. Eisenberg. Local moves, an efficient method for protein folding simulations. *Proteins*, 23:73–82, 1995.

[52] R.A. Engh and R. Huber. Accurate bond and angle parameters for x-ray protein structure refinement. *Acta Cryst.*, A47:392–400, 1991.

[53] T. J. Ewing, S. Makino, A. G. Skillman, and I. D. Kuntz. DOCK 4.0: search strategies for automated molecular docking of flexible molecule databases. *J. Comput. Aided Mol. Des.*, 15:411–428, 2001.

[54] B. Fain and M. Levitt. A novel method for sampling alpha-helical protein backbones. *J. Mol. Biol.*, 305:191–201, 2001.

[55] B. Faverjon. Obstacle avoidance using an octree in the configuration space of a manipulator. In *IEEE Conf. on Rob. and Auto.(ICRA)*, pages 504–510, 1984.

[56] B. Faverjon. Object level programming of industrial robots. In *IEEE Conf. on Rob. and Auto. (ICRA)*, pages 1406 – 1412, 1986.

[57] H. Feldman and C. Hogue. A fast method to sample real protein conformational space. *Proteins*, 39(2):112–131, 2000.

[58] Z. Feng and M. Sippl. Optimum superimposition of protein structures: ambiguities and implications. *Fold. Des.*, 1:123–132, 1996.

[59] K. Fidelis, P.S. Stern, D. Bacon, and J. Moult. Comparison of systematic search and database methods for constructing segments of protein structure. *Prot. Engin.*, 7(8):953–960, 1994.

[60] R.M. Fine, H. Wang, P.S. Shenkin, D.L. Yarmush, and C. Levinthal. Predicting antibody hypervariable loop conformations. ii minimization and molecular dynamics studies of mcp603 from many randomly generated loop conformations. *Proteins*, 1:342–362, 1986.

[61] A. Fiser, R.K. Do, and A. Sali. Modeling of loops in protein structures. *Prot. Sci.*, 9(9):1753–73, 2000.

[62] A. Foisy and V. Hayward. A safe swept volume method for collision detection. In *The Sixth International Symposium of Robotics Research*, pages 61–68, Pittsburgh (PA), Oct. 1993.

[63] D. Frenkel and B. Smit. *Understanding molecular simultion from algorithms to applications*, volume 1 of *Computational science from theory to applications*. Academic Press, San diego, 2 edition, 2002.

[64] F. Ganovelli, J. Dingliana, and C. O'Sullivan. Buckettree: Improving collision detection between deformable objects. In *Spring Conf. on Comp. Graphics (SCCG2000)*, 2000.

[65] M. Gerstein and M. Levitt. Comprehensive assessment of automatic structural alignment against a manual standard, the scop classification of proteins. *Prot. Sci.*, 7:445–456, 1998.

[66] J. Gibrat, T. Madej, and S. Bryant. Surprising similarities in structure comparison. *Curr. Opin. in Struct. Biol.*, 6(3):377–385, 1996.

[67] K. Ginalski and L. Rychlewski. Protein structure prediction of CASP5 comparative modeling and fold recognition targets using consensus alignment approach and 3d assessment. *Proteins*, 53(S6):410–417, 2003.

[68] N. Gō and H. Abe. Noninteracting local-structure model of folding and unfloding transition in globular proteins. *Biopolymers*, 20:991–1011, 1981.

[69] N. Gō and H.A. Scheraga. Ring closure and local conformational deformations of chain molecules. *Macromolecules*, 3:178–186, 1970.

[70] A. Godzik. The structural alignment between two proteins: is there a unique answer? *Prot. Sci.*, 5:1325–1338, 1996.

[71] G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[72] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. *Comp. Graphics*, 30(Annual Conf. Series):171–180, 1996.

[73] A. Grosberg and A. Khokhlov. *Statistical physics of macromolecules*. AIP Press, New York, 1994.

[74] A. Grosberg and A. Khokhlov. *Giant Molecules: here, there and everywhere*. Academic Press, San Diego, 1997.

[75] L. J. Guibas, A. Nguyen, D. Russel, and L. Zhang. Deforming necklaces. In *Symp. on Comp. Geometry (SoCG)*, pages 33–42, 2002.

[76] P. Güntert, C. Mumenthaler, and K. Wüthrich. Torsion angle dynamics for NMR structure calculation with the new program DYANA. *J. Mol. Biol.*, 273:283–298, 1997.

[77] C. Hadley and D. T. Jones. A systematic comparison of protein structure classifications: SCOP, CATH and FSSP. *Structure Fold Des*, 7(9):1099–112, Sep 1999.

[78] D. Halperin, J.-C. Latombe, and R. Motwani. Dynamic maintenance of kinematic structures. In J.-P. Laumond and M. Overmars, editors, *Alg. for Rob. Motion and manip. (WAFR '96)*, pages 155–170. A.K. Peters, Wellesley, 1997.

[79] D. Halperin and M. H. Overmars. Spheres, molecules and hidden surface removal. *Comp. Geom.: Theory and App.*, 11(2):83–102, 1998.

[80] I. Halperin, B. Ma, H. Wolfson, and R. Nussinov. Principles of docking: An overview of search algorithms and scoring functions. *Proteins*, 47:409–443, 2002.

[81] L. Han and N.M. Amato. A kinematics-based probabilistic roadmap method for closed chain systems. In B.R. Donald, K. Lynch, and D. Rus, editors, *Workshop Algo. Found. Robot. (WAFR)*, pages 233–246, March 2000.

[82] H. Hansmann and Y. Okamoto. New Monte Carlo algorithms for protein folding. *Curr. Opin. Struct. Biol.*, 9(2):177–183, 1999.

[83] U. Hansmann. Parallel tempering algorithm for conformational studies of biological molecules. *Chem. Phys. Lett.*, 281:140–150, 1997.

[84] R. W. Hockney, S. P. Goel, and J. W. Eastwood. Quiet high-resolution computer models of a plasma. *J. Comput. Phys.*, 14(2):148–158, 1974.

[85] L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *J. Mol. Biol.*, 233(1):123–128, 1993.

[86] L. Holm and C. Sander. Mapping the protein universe. *Science*, 273:595–602, 1996.

[87] B. Horn. Closed form solution of absolute orientation using unit quaternions. *J. Optic. Soc. A*, 4(4):629–642, April 1987.

[88] P. M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Tr. on Graphics*, 15(3):179–210, 1996.

[89] P. Indyk. *High-dimensional Computational Geometry.* PhD thesis, Stanford University, September 2000.

[90] T.R. Ioerger and J.C. Sacchettini. The textal system: Artificial intelligence techniques for automated protein model building. In *Methods in Enzymology.*, volume 374, pages 244–270, San Diego, 2003. Academic Press.

[91] M.P. Jacobson, D.L. Pincus, C.S. Rapp, T.J. Day, B. Honig, D.E. Shaw, and R.A. Friesner. A hierarchical approach to all-atom protein loop prediction. *Proteins.*, 55(2):351–67, 2004.

[92] P. Jiménez, F. Thomas, and C. Torras. 3D collision detection: A survey. *Computers and Graphics*, 25(2):269–285, 2001.

[93] G. Jones, P. willett, R. C. Glen, A. R. Leach, and R. Taylor. Development and validation of a genetic algorithm for flexible docking. *J. Mol. Biol.*, 267:727–748, 1997.

[94] T.A. Jones and M. Kjeldgaard. Electron-density map interpretation. In *Methods in Enzymology*, volume 277, pages 173–230, San Diego, 1997. Academic Press.

[95] T.A. Jones and S. Thirup. Using known substructures in protein model-building and crystallography. *EMBO J.*, 5:819–822, 1986.

[96] T.A. Jones, J.-Y. Zou, and S.W. Cowtan. Ring closure and local conformational deformations of chain molecules. *Acta Cryst.*, A47:110–119, 1991.

[97] W. Kabsch. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystal. A*, 34:827–828, 1978.

[98] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.*, 12(4):566–580, 1996.

[99] K. Kedem, L. Chew, and R. Elber. Unit-vector RMS (URMS) as a tool to analyze molecular dynamics trajectories. *Proteins*, 37:554–564, 1999.

[100] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *Int. J. Robot. Autom.*, RA-3(1):43–53, February 1987.

[101] O. Khatib, L. Sentis, J.-H. Park, and J. Warren. Whole body dynamic behavior and control of human-like robots. *International Journal of Humanoid Robotics*, 1(1):29–43, 2004.

[102] A. Kidera. Smart Monte Carlo simulation of a globular protein. *Int. J. Quant. Chem.*, 75:207–214, 1999.

[103] T. Kikuchi. Inter-Ca atomic potentials derived from the statistics of average inter-residue distances in proteins: Application to bovine pancreatic trypsin inhibitor. *J. Comput. Chem*, 17(2):226–237, 1996.

[104] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[105] J. Kleinberg. Two algorithms for nearest-neighbor search in high dimension. In *Proc. Symp. Theo. Comp. (SToC)*, pages 599–608, 1997.

[106] J. T. Klosowski, J. S. B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Tr. on Visual. and Comp. Graphics*, 4(1):21–36, 1998.

[107] P. Koehl. Protein structure similarity. *Curr. Opin. Struct. Biol.*, 11:348–353, 2001.

[108] P. Koehl and M. Delarue. Building protein lattice models using self-consistent mean field theory. *J. Chem. Phys.*, 108(22):9540–9549, 1998.

[109] R. Kolodny, L. Guibas, M. Levitt, and P. Koehl. Inverse kinematics in biology: the protein loop closure problem. Submitted to IJRR, 2004.

[110] R. Kolodny, P. Koehl, L. Guibas, and M. Levitt. Small libraries of protein fragments model native protein structures accurately. *J. Mol. Biol.*, 323(2):297–307, Oct 2002.

[111] Andrei Korostelev, Richard Bertram, and Michael S. Chapman. Simulated-annealing real-space refinement as a tool in model building. *Acta Cryst.*, D58:761–767, 2002.

[112] E. Krissinel. CCP4 coordinate library project, 2004. http://www.ebi.ac.uk/ keb/cldoc/.

[113] E. Kussell, J. Shimada, and E. Shakhnovich. A structure-based method for derivation of all-atom potentials for protein folding. *Proc. Natl. Acad. Sci.*, 99(8):5343–8, April 2002.

[114] P. Lackner, W. Koppensteiner, M. Sippl, and F. Domingues. ProSup: a refined tool for protein structure alignment. *Prot. Eng.*, 13(11):745–752, 2000.

[115] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha. Fast distance queries with rectangular swept sphere volumes. In *IEEE Conf. on Rob. and Auto. (ICRA)*, 2000.

[116] S. Larson, C. Snow, M. Shirts, and V. Pande. Folding@Home and Genome@Home: Using distributed computing to tackle previously intractable problems in computational biology. In Richard Grant, editor, *Computational Genomics*. Horizon Press, 2003.

[117] T. Lazaridis and M. Karplus. Discrimination of the native from misfolded protein models with an energy funbction including implicit solvation. *J. Mol. Bio.*, 288:477–487, 1998.

[118] T. Lazaridis and M. Karplus. Effective energy function for proteins in solution. *Proteins*, 35:133–152, 1999.

[119] A. Leach. *Molecular Modelling - Principles and Applications*. Addison Wesley Longman Limited, 1996.

[120] A. Leach. *Molecular Modelling: Principles and Applications*. Longman, Essex, England, 1996.

[121] J. Lee. New Monte Carlo algorithm: entropic sampling. *Phys. Rev. Lett.*, 71(2):211–214, 1993.

[122] D.G. Levitt. A new software routine that automates the fitting of protein x-ray crystallographic electron-density maps. *Acta Cryst.*, D57:1013–1019, 2001.

[123] M. Levitt. A simplified representation of protein conformations for rapid simulation of protein folding. *J. Mol. Biol.*, 104:59–107, 1976.

[124] M. Levitt and M. Gerstein. A unified statistical framework for sequence comparison and structure comparison. *Proc. Nat. Acad. Sci.*, 95:5913–5920, 1998.

[125] Z. Li and H. Scheraga. Monte Carlo-minimization approach to the multiple-minima problem in protein folding. *Proc. Natl. Acad. Sci.*, 84(19):6611 – 6615, Oct. 1987.

[126] M. C. Lin and J. F. Canny. A fast algorithm for incremental distance calculation. In *IEEE Int. Conf. on Rob. and Auto. (ICRA)*, pages 1008–1014, 1991.

[127] I. Lotan and F. Schwarzer. Approximation of protein structure for fast similarity measures. *J. Comput. biol.*, 11(2-3):299–317, 2004.

[128] I. Lotan, F. Schwarzer, D. Halperin, and J.-C. Latombe. Efficient maintenance and self-collision testing for kinematic chains. In *Symp. Comp. Geo.*, pages 43–52, 2002.

[129] I. Lotan, F. Schwarzer, D. Halperin, and J.-C Latombe. Algorithm and data structures for efficient energy maintenance during Monte Carlo simulation of proteins. *J. Comput. Biol.*, 2004. To appear.

[130] I. Lotan, F. Schwarzer, and J.-C Latombe. Efficient energy computation for Monte-Carlo simulation of proteins. In *Workshop on Algorithms for Bioinformatics (WABI)*, Budapest, September 2003.

[131] I. Lotan, H. van den Bedem, A. M. Deacon, and J.-C Latombe. Computing protein structures from electron density maps: The missing fragment problem. In *Workshop Algo. Found. Robot. (WAFR)*, Zeist, Holand, 2004. To Appear.

[132] S.C Lovell, I.W. Davis, W.B. Arendall III, P.I.W de Bakker, J.M. Word, M.G. Prisant, J.S. Richardson, and D.C. Richardson. Structure validation by Cα geometry: ϕ, ψ and Cβ deviation. *Proteins*, 50(3):437–450, 2003.

[133] J. G. Mandell, V. A. Roberts, M. E. Pique, V. Kotlovyi, J. C. Mitchell, E. Nelson, I. Tsigelny, and L. F. Ten Eyck. Protein docking using continuum electrostatics and geometric fit. *Protein Eng.*, 14:105–113, 2001.

[134] S. Maneewongvatana and D. Mount. The analysis of a probabilistic approach to nearest neighbor searching. In *Proc. Workshop Algo. Data Struct. (WADS)*, pages 276–286, 2001.

[135] D. Manocha and J. canny. Efficient inverse kinematics for general 6R manipulator. *IEEE Trans. Robot. Autom.*, 10(5):648–657, 1994.

[136] D. Manocha and Y. Zhu. Kinematic manipulation of molecular chains subject to rigid constraints. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 2:285–293, 1994.

[137] D. Manocha, Y. Zhu, and W. Wright. Conformational analysis of molecular chains using nano-kinematics. *Comput. Appl. Biosci.*, 11(1):71–86, 1995.

[138] M. A. Martí-Renom, A. C. Stuart, A. Fiser, R. Sánchez, F. Melo, and A. Šali. Comparative protein structure modeling of genes and genomes. *Annu. Rev. Biophys. Biomol. Struct.*, 29:291–325, 2000.

[139] G.J. McLachlan, D. Peel, K.E. Basford, and P. Adams. The EMMIX software for the fitting of mixtures of normal and t-components. *J. Stat. Software*, 4(2), 1999.

[140] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1092, 1953.

[141] M. Mezei. A near-neighbor algorithm for metropolis Monte Carlo simulation. *Molecular Simulations*, 1:169–171, 1988.

[142] A. Mitsutake, Y. Sugita, and Y. Okamoto. Generalized-ensemble algorithms for molecular simulations of biopolymers. *Biopolymers*, 60:96–123, 2001.

[143] J. Moult, K. Fidelis, A. Zemla, and T. Hubbard. Critical assessment of methods of protein structure prediction (CASP)-round V. *Proteins*, 53(S6):334–339, 2003.

[144] J. Moult and M.N.G. James. An algorithm for determining the conformation of polypeptide segments in protein by systematic search. *Proteins*, 1:146–163, 1986.

[145] G. N. Murshudov, A. A. Vagin, and E. J. Dodson. Refinement of macromolecular structures by the maximum-likelihood method. *Acta Cryst.*, D53:240–255, 1997.

[146] A. Murzin, S. Brenner, T. Hubbard, and C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, 247(536–540), 1995.

[147] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.

[148] J. T. Ngo, J. Marks, and M Karplus. Computational complexity, protein structure prediction and the Levinthal paradox. In K. Merz Jr. and S. LeGrand, editors, *The proetin folding problem abd tertiary structure prediction*. Birkhäuser, Boston, 1994.

[149] S. Northrup and J. McCammon. Simulation methods for protein-structure fluctuations. *Biopolymers*, 19(5):1001–1016, 1980.

[150] M. Novotny, D. Madsen, and G. J. Kleywegt. Evaluation of protein fold comparison servers. *Proteins*, 54(2):260–70, 2004.

[151] T.J. Oldfield. A number of real-space torsion-angle refinement techniques for proteins, nucleic acids, ligands and solvent. *Acta Cryst.*, D57:82–94, 2001.

[152] C. Orengo, A. Michie, S. Jones, D. Jones, M. Swindells, and J Thornton. CATH–a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1108, 1997.

[153] P. N. Palma, L. Krippahl, J. E. Wampler, and J. J. Moura. A new (soft) docking algorithm for predicting protein interactions. *Proteins*, 39:372–384, 2000.

[154] C. Pangali, M. Rao, and B. J. Berne. On a novel Monte Carlo scheme for simulating water and aqueous solutions. *Chem. Phys. Lett.*, 55(3):413–417, 1978.

[155] B. Park, E. Huang, and M. Levitt. Factors affecting the ability of energy functions to discriminate correct from incorrect folds. *J. Mol. Biol.*, 266:831–846, 1997.

[156] B. Park and M. Levitt. Energy functions that discriminate X-ray and near-native folds from well-constructed decoys. *J. Mol. Biol.*, 258:367–392, 1996.

[157] J. Pedersen and J. Moult. Protein folding simulations with genetic algorithms and a detailed molecular description. *J. Mol. Biol.*, 269(2):240–259, 1997.

[158] A. Perrakis, T.K. Sixma, K.S. Wilson, and V.S. Lamzin. wARP: Improvement and extension of crystallographic phases by weighted averaging of multiple-refined dummy atomic models. *Acta Cryst.*, D53:448–455, 1997.

[159] R. J Petrella, I. Andricioaei, B. R. Brooks, and M. Karplus. An improved method for nonbonded list generation: Rapid determination of near-neighbor pairs. *J. Comput. Chem.*, 24(2):222–231, 2002.

[160] D. Petrey, Z. Xiang, C. L. Tang, L. Xie, M. Gimpelev, T. Mitros, C. S. Soto, S. Goldsmith-Fischman, A. Kernytsky, Avner Schlessinger, I. Y.Y. Koh, E. Alexov, and B. Honig. Using multiple structure alignments, fast model building, and energetic analysis in fold recognition and homology modeling. *Proteins*, 53(S6):430–435, 2003.

[161] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, NY, 1985.

[162] B. Quentrec and C. Brot. New method for searching for neighbors in molecular dynamics computations. *J. Comput. Phys*, 13(3):430–432, 1973.

[163] S. Quinlan. Efficient distance computation between non-convex objects. In *IEEE Int. Conf. on Rob. and Auto. (ICRA)*, pages 3324–3329, 1994.

[164] M. Raghavan and B. Roth. Kinematic analysis of the 6r manipulator of general geometry. In *Int. Symp. Robot. Res.*, pages 314–320, Tokyo, 1989.

[165] M. Rarey, B. Kramer, T. Lengauer, and G. Klebe. A fast flexible docking method using an incremental construction algorithm. *J. Mol. Biol.*, 261:470–489, 1996.

[166] I. Sasson and D. Fischer. Modeling three-dimensional protein structures for CASP5 using the 3D-SHOTGUN meta-predictors. *Proteins*, 53(S6):389–394, 2003.

[167] T. Schlick. *Molecular nodeling and simulation*, volume 21 of *Interdisciplinary applied mathematics*. Springer-Verlag, New York, 2002.

[168] F. Schwarzer and I. Lotan. Approximation of protein structure for fast similarity measures. In W. miller, M. Vingron, S. Istrail, P. Pevzner, and M. Waterman, editors, *Proc. of 7th International Conference on Research in Computational Molecular Biology (RECOMB)*, pages 267–276, Berlin, 2003.

[169] F. Schwarzer, M. Saha, and J.-C. Latombe. Exact colision detection of robot paths. In *Proc. 5th Workshop on Alg. Found. Rob.*, 2002.

[170] P.S. Shenkin, D.L. Yarmush, R.M. Fine, H.J. Wang, and C. Levinthal. Predicting antibody hypervariable loop conformation. 1. ensembles of random conformations for ring-like structure. *Biopolymers*, 26:20532085, 1987.

[171] J. Shimada, E. Kussell, and E. Shakhnovich. The folding thermodynamics and kinetics of crambin using an all-atom Monte Carlo simulation. *J. Mol. Biol.*, 308(1):79–95, April 2001.

[172] J. Shimada and E. Shakhnovich. The ensemble folding kinetics of protein G from an all-atom Monte Carlo simulation. *Proc. Natl. Acad. Sci.*, 99(17):11175–80, August 2002.

[173] I. Shindyalov and P. Bourne. protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Prot. eng.*, 11(9):739–747, 1998.

[174] M. Shirts and V. Pande. Mathematical analysis of coupled parallel simulations. *Phys. Rev. Lett.*, 86(22):4983–4987, 2001.

[175] B. K. Shoichet, S. L. McGovern, B. Wei, and J. J. Irwin. Lead discovery using molecular docking. *Curr. Opin. Chem. Biol.*, 6:439–446, 2002.

[176] D. Shortle, K. Simons, and D. Baker. Clustering of low-energy conformations near the native structures of small proteins. *Biophysics*, 95:11158–11162, 1998.

[177] M. L. Sierk and W. R. Pearson. Sensitivity and selectivity in protein structure comparison. *Protein Sci.*, 13(3):773–785, 2004.

[178] K. Simons, R. Bonneau, I. Ruczinski, and D. Baker. Ab initio protein structure prediction of CASP III targets using ROSETTA. *Proteins*, 37(3):171–176, 1999.

[179] K. Simons, C. Kooperberg, E. Huang, and D. Baker. Assembly of protein tertiary structure from fragments with similar local sequences using simulated annealing and bayesian scoring functions. *J. Mol. Biol.*, 268:209–225, 1997.

[180] A. P. Singh and D. L. Brutlag. Hierarchical protein structure superposition using both secondary structure and atomic representations. In *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, volume 5, pages 284–293, 1997.

[181] J. Skolnick, Y. Zhang, A. K. Arakaki, A. Kolinski, M. Boniecki, A. Szilágyi, and D. Kihara. TOUCHSTONE: A unified approach to protein structure prediction. *Proteins*, 53(S6):469–479, 2003.

[182] G. R. Smith and M. J. E. Sternberg. Prediction of protein-protein interactions by docking methods. *Curr. Opin. Struct. Biol.*, 12(1):28–35, 2002.

[183] J. M. Sorenson and T. Head-Gordon. Matching simulation and experiment: a new simplified model for simulating protein folding. *J. Comput. Biol.*, 7(3/4):469–482, 2000.

[184] M. Soss, J. Erickson, and M. Overmars. Preprocessing chains for fast dihedral rotations is hard or even impossible. *Comp. Geom.: Theory and App.*, 26(3):235–246, 2003.

[185] M. Soss and G. Toussaint. Geometric and computational aspects of polymer reconfiguration. *J. Math. Chem.*, 27(4):303 – 318, 2000.

[186] E. Stein, L. Rics, and A. Brünger. Torsion-angle molecular dynamics as a new efficient tool for NMR structure calculation. *J. Magn. Reson.*, 124:154–164, 1997.

[187] E. Stollnitz, T. DeRose, and D. Salesin. Wavelets for computer graphics: A primer (part 1). *IEEE Comp. Graph. Appl.*, 15(3):76–84, 1995.

[188] S. Subbiah, D. Laurents, and M. Levitt. Structural similarity of DNA-binding domains of bacteriophage repressors and the globin core. *Curr. Biol.*, 3:141–148, 1993.

[189] S. Sun. Reduced representation model of protein structure prediction: statistical potential and genetic algorithms. *Protein Science*, 2(5):762–785, 1993.

[190] S. Sun, P. Thomas, and K. Dill. A simple protein folding algorithm using a binary code and secondary structure constraints. *Protein Eng.*, 8:769–778, 1995.

[191] W. Taylor and C. Orengo. Protein structure alignment. *J. Mol. Biol.*, 208:1–22, 1989.

[192] T.C. Terwilliger. Automated main-chain model-building by template-matching and iterative fragment extension. *Acta Cryst.*, D59:34–44, 2002.

[193] T.C. Terwilliger. Improving macromolecular atomic models at moderate resolution by automated iterative model building, statistical density modification and refinement. *Acta Cryst.*, D59:1174–1182, 2003.

[194] T.C. Terwilliger and J. Berendzen. Automated mad and mir structure solution. *Acta Cryst.*, D55:849–861, 1999.

[195] S. Thompson. Use of neighbor lists in molecular dynamics. *Information Quaterly, CCP5*, 8:20–28, 1983.

[196] J.C. Trinkle and R.J. Milgram. Complete path planning for closed kinematic chains with spherical joints. *Int. J. Robot. Res.*, 21(9):773–789, 2002.

[197] Č. Venclovas. Comparative modeling in CASP5: Progress is evident, but alignment errors remain a significant hindrance. *Proteins*, 53(S6):380–388, 2003.

[198] R. Unger and J. Moult. Genetic algorithm for protein folding simulations. *J. Mol. Biol.*, 231:75–81, 1993.

[199] H. van den Bedem, I. Lotan, J.-C Latombe, and A. M. Deacon. Automated protein model completion: an inverse kinematics approach. *Acta Cryst. D*, 2004. submitted.

[200] G. van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *J. Graphics Tools*, 2(4):1–13, 1997.

[201] W. F. van Gunsteren, H. J. C. Berendsen, F. Colonna, D. Perahia, J. P. Hollenberg, and D. Lellouch. On searching neighbors in computer simulations of macromolecular systems. *J. Comput. Chem.*, 5:272–279, 1984.

[202] H.W.T. van Vlijmen and M. Karplus. PDB-based protein loop prediction: parameters for selection and methods for optimization. *J. Mol. Biol.*, 267(4):975–1001, 1997.

[203] L. Verlet. Computer "experiments" on classical fluids I: Thermodynamical properties of Lennard-Jones molecules. *Phys. Rev.*, 159(1):98–103, July 1967.

[204] D. Waasmaier and A. Kirfel. New analytical scattering-factor functions for free atoms and ions. *Acta Cryst.*, A51(3):416–431, 1995.

[205] L.T. Wang and C.C. Chen. A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *IEEE Trans. Robot. Autom.*, 7:489–499, 1991.

[206] W.J. Wedemeyer and H.A. Scheraga. Exact analytical loop closure in proteins using polynomial equations. *J. Comput. Chem.*, 20:819–844, 1999.

[207] S. J. Weiner, P. A. Kollman, D. A. Case, U. C. Singh, C. Ghio, G. Alagona, S. Profeta, and P. Weiner. A new force field for molecular mechanical simulation of nucleic acids and proteins. *J. am. Chem. Soc.*, 106(3):765–784, 1984.

[208] S. J. Wodak and J. Janin. *Advances in protein chemistry*, volume 61, chapter Structural basis of macromolecular recognition, pages 9–73. Elsevier, 2002.

[209] Guenter Wolf. Personal Communication, 2004.

[210] K. Wüthrich. *NMR of proteins and nucleic acids*. Wiley-Interscience, New York, 1986.

[211] D. Xie and N.M. Amato. Kinematics-based probabilistic roadmap method for high dof closed chain systems. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2004. To appear.

[212] J. Yakey, LaValle S.M., and L.E. Kavraki. Randomized path planning for linkages with closed kinematics chains. *IEEE Trans. Robot. Autom.*, 17(6):951–959, 2001.

[213] V. Yip and R. Elber. Calculations of a list of neighbors in molecular dynamics simulations. *J. Comput. Chem.*, 10(7):921–927, 1989.

[214] B. Zagrovic, E. Sorin, and V. Pande. β-hairpin folding simulations in atomistic detail using an implicit solvent model. *J. Mol. Biol.*, 313:151–169, 2001.

[215] H. Zhang. A new hybrid Monte Carlo algorithm for protein potential function test and structure refinement. *Proteins*, 34:464–471, 1999.

[216] M. Zhang and L.E. Kavraki. Finding solutions of the inverse kinematics problems in computer-aided drug design. Technical Report TR02-385, Rice University, 2002.

[217] M. Zhang and L.E. Kavraki. A new method for fast and accurate derivation of molecular conformations. *J. Chem. Info. Comp. Sci.*, 42(1):64–70, 2002.

[218] Y. Zhang, D. Kihara, and J. Skolnick. Local energy landscape flattening: Parallel hyperbolic Monte Carlo sampling of protein folding. *Proteins*, 48:192–201, 2002.

[219] Y. Zhang and J. Skolnick. Parallel-hat tempering: A Monte Carlo search scheme for the identification of low-energy structures. *J. Chem. Phys.*, 115(11):5027–5032, 2001.

[220] Q. Zheng, R. Rosenfeld, S. Vajda, and C. DeLisi. Loop closure via bond closure and relaxation. *J. Comput. Chem*, 14:556–565, 1992.

[221] Y. Zhou and S. Suri. Analysis of a bounding vox heuristic for object intersection. *J. of the ACM*, 46(6):833–857, 1999.

[222] Y. Zhou and S. Suri. Collision detection using bounding boxes: Convexity helps. In *8th Annual European Symposium on Algorithms (ESA 2000)*, pages 437–448, 2000.