University of Alberta

Bioinformatics Tools for Structural Biology

By

Rajarshi Maiti ©

A thesis submitted to the Faculty of Graduate Studies and research in partial fulfillment of the requirements for the degree of

Master of Science

In

Pharmaceutical Sciences

Faculty of
Pharmacy and Pharmaceutical Sciences
Edmonton, Alberta
Fall 2005

Abstract

Structural biology lies at the heart of many current efforts in drug discovery and drug development. Indeed, it is through structural biology efforts spanning the past 40 years scientists have learned a great deal about the mode of action and molecular behavior of many important protein drugs and drug targets. However, the size and complexity of protein structures, along with the intractable behavior of many enzymes and proteins still makes their structure determination and structure characterization difficult. Even with the advent of experimental techniques such as high-throughput X-ray crystallography and NMR spectroscopy or the development of computational methods such as homology modeling and molecular dynamics, structural biologists still struggle for months or even years to fully characterize or "solve" most protein structures. Clearly new tools are needed to assist in the structure/dynamic characterization of proteins and enzymes. In this thesis I will describe several computational tools that I have developed that could have a significant impact on the characterization, comparison or prediction of protein structures and protein dynamics. These tools include a new approach for rapid 3D protein structure prediction that employs a computational method called distance geometry; a robust, rapid and highly general approach for 3D structure comparison; and finally, a web-based tool that can be used to rapidly predict, compare and display the molecular motions and movements of proteins.

Acknowledgements

I would like to thank Dr. David Wishart for his support and guidance for the past years in making this project a successful endeavor. I would also like to thank Haiyan Zhang and Dr. Gary Van Domselaar for their contribution towards this thesis. Thanks also go to Robert Yang and to Philip Luk; summer students in Dr. Wishart's laboratory. Robert wrote the code for the simplex algorithm, while Philip wrote the initial coordinate mapping algorithm for the structure generation program. Both of these programs were well written and spared me endless hours of debugging. Haiyan helped me a lot in the initial stages, patiently debugging my code and offering advice whenever I had programming problems. It has also been a pleasure working with Dr. Gary Van Domselaar on SuperPose and MovieMaker. Gary taught me better programming principles and practices. He contributed a lot to both SuperPose and MovieMaker and they wouldn't be successful projects without his comprehensive knowledge of perl and cgi scripting. Thanks also go to all the other members in Dr. Wishart's laboratory, for their support and thoughts and discussions that I have shared with them. Lastly, I would like to thank the Faculty of Pharmaceutical and Pharmaceutical Sciences, as well as the Protein Engineering Network of Centre of Excellence (PENCE) for financial support.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

2D	Two Dimensional
3D	Three Dimensional
AMBER	Assisted Model Building with Energy Refinement
BD	Brownian Dynamics
BLAST	Basic Local Alignment Search Tool
CE	Combinatorial Extension
CGI	Common Gateway Interface
CHARMM	Chemistry at HARvard Molecular Mechanics
CPK	Corey, Pauling and Koltun
CPU	Central Processing Unit
DD	Difference Distance
DG	Distance Geometry
DNA	DeoxyriboNucleic Acid
GA	Genetic Algorithm
GHz	Giga Hertz
GIF	Graphics Interchange Format
GUI	Graphical User Interface
HTML	HyperText Markup Language
MB	Mega Byte
MC	Monte Carlo
MD	Molecular Dynamics

MHz	Mega Hertz
MPEG	Moving Pictures Expert Group
NMR	Nuclear Magnetic Resonance
PDB	Protein Data Bank
PERL	Practical Extraction and Report Language
PNG	Portable Network Graphics
PSI-BLAST	Position Specific Iterative Basic Local Alignment Search Tool
RAM	Random Access Memory
RCSB	Research Collaboratory for Structural Bioinformatics
RMS	Root Mean Square
RMSD	Root Mean Square Deviation
RNA	RiboNucleic Acid
VADAR	Volume Area Dihedral Angle Reporter

Chapter 1: Introduction

1.1 Introduction

This thesis combines both hypothesis driven research with conventional software development in an effort to devise new strategies that will aid research and researchers in the field of structural biology. To gain a broad level of experience and programming expertise, I have selected three areas of protein structure research and have attempted to address or partially address several key problems in this area by using a wide range of computational tools and techniques. These three key problems include: 1) 3D structure prediction; 2) 3D structure comparison and 3) modeling or predicting protein dynamics. The motivation for examining these problems is the fact that existing methods (experimental or computational) are either inadequate or unacceptably slow.

Experimentally, 3D structure determination of proteins by X-ray crystallography or NMR spectroscopy is time consuming and laborious. Usually, a time span of 6 – 12 months is required to deduce a protein structure from experimental data. In an effort to shorten the time required or reduce the tediousness of the structure determination process, many computational methods have been tried and developed to predict the 3D protein structures from sequence or secondary structure information (Monge et al., 1994; Alexandrov et al., 1995; Pedersen, 1996; Rohl and Baker, 2002; Hung and Samudrala, 2003; Rohl et al., 2004). To date, no suitable algorithm has been found which can accurately or rapidly predict the 3D structure of proteins from sequence or secondary structure information. In Chapter two of this thesis, I will describe a computational method based on distance geometry (Crippen 1981, Crippen and Havel 1988) that can be used to rapidly generate a library of possible protein structures from known or predicted secondary structure information. This approach, which treats protein folding or protein structure prediction as a “low resolution” packing problem, could potentially accelerate the protein structure prediction process by many orders of magnitude.

The second area of structural biology research that I have chosen to address concerns protein structure comparison or structure superposition. Structural superposition has long been used to study the similarities or dissimilarities between proteins and other macromolecules (McLachlan, 1982; Kearsley, 1990; Diamond, 1992; Flower, 1999). However, most approaches and most available software packages do not support the range of superposition options or address the needs that many modern structure biologists have frequently expressed. These include the need to compare non-homologous or highly dissimilar structures, the need to compare “open” and “closed” forms of enzymes or metal binding proteins, the need to rapidly and accurately superimpose more than two structures or the need to extract far more quantitative detail about a structure superposition or comparison. Furthermore, many standalone molecular comparison packages are so detailed or complex that users have to spend hours of training to become even modestly familiar with their many options or hard-to-use interfaces. To address these issues I have developed a simple, web-accessible superposition server that supports a wide range of structure comparison and structure superposition options. The complete software tool is described in Chapter three.

The fourth chapter in this thesis tackles the problem of predicting and visualizing protein dynamics. While structural biologists are primarily concerned with describing or solving the static structure of proteins or other macromolecules, there is a growing awareness that protein motions are perhaps just as important as protein structures in understanding protein function and protein-ligand binding. Conventional methods, such as molecular dynamics (MD) or Brownian dynamics (BD) are computationally expensive and are only capable of revealing motions occurring over relatively short (<1 ns) time scales or distances (<5 Angstroms). Given that many important dynamic events occur over much longer periods or length scales, a new and computationally faster approach needs to be found. Recently, Krebs and Gerstein (2000) have pointed out the power of using existing protein structures and existing B-factor or ensemble information to extend the time period and length scales with which protein motions could be simulated. Using this concept, and incorporating some simplifications of our own, we were able to develop a web server that rapidly and accurately generates large (and small) scale motions for proteins, using just one or more static PDB structures as input.

Collectively, these efforts represent a relatively modest contribution to structural proteomics research, but I believe they could serve as the basis for more far reaching developments in the field of structural proteomics.

1.2 Structural Proteomics in Pharmaceutical Research

A major challenge of biomedical research today involves the characterization of the properties and biological functions of not only genes, but proteins. The study of many proteins at once or in a high-throughput manner is often called proteomics. Proteomics is derived from the word proteome which refers to the proteins encoded by a genome. Specifically, proteomics can be defined as the large scale study of proteins via biochemical methods (Pandey and Mann, 2000). It is through proteomics research that we can learn how proteins and peptides are involved in human diseases and how they can be used for industrial or pharmaceutical applications.

Proteomics research can be divided into three key areas: 1) Functional Proteomics; 2) Expressional or Classical Proteomics and 3) Structural Proteomics. Functional proteomics is primarily concerned with identifying the functions, locations or pathways of newly identified proteins or protein targets. Expressional proteomics is concerned with monitoring or measuring the protein expression changes that happen among proteins in response to an external perturbation (a drug, a disease or a pathogen). Structural proteomics is concerned with determining the 3D structures of the more interesting proteins or protein targets that have been identified or isolated by functional or expressional proteomics. By knowing something about the shape, size, or the atomic detail of a protein's active site it is often possible to rationally design or modify small molecule drugs or drug leads. Specifically, the active site of a protein allows one to define a pharmacophore model which then can be used to design a complimentary ligand to bind at the active site. In fact, structural proteomics and structural biology actually serve as the basis to an entire field of pharmaceutical research known as rational or structure-aided drug design (Ringe, 1995; Klebe, 2000; Veselovsky and Ivanov, 2003; Thiel, 2004).

Clearly if we knew (or could know) the structure of most or all proteins in the human proteome, drug discovery and drug development could be made much more robust and much more cost-efficient. Indeed, knowing the structure of most human proteins would allow a large number of highly effective drugs to be designed on a computer without a great deal of experimentation – saving considerable time and money. Given that the time it takes from the initial identification of a protein target to the market launch of a viable drug can take up to ten years and cost in excess of $800,000,000 (Kara Bio Annual report, 2000), any saving in time or effort would be greatly welcomed by the pharmaceutical industry.

While structural proteomics offers some tantalizing opportunities for the pharmaceutical industry, it is still important to remember that not all proteins are experimentally amenable to structure determination. In fact, the vast majority of pharmaceutically interesting proteins are not. Consequently, a second branch of structural proteomics has emerged – one that is not based in experimental techniques such as X-ray crystallography or NMR spectroscopy, but rather one that is based on using computers to solve or "predict" 3D protein structures.

1.3 3D Structure predictions of proteins

The fundamental tenet of structural biology is that function follows form. This concept had its roots in the monograph by C.B. Anfinsen, *The Molecular Basis of* Evolution (Anfinsen, 1959), wherein he stated "Protein chemists naturally feel that the most likely approach to the understanding of cellular behavior lies in the study of structure and function of protein molecules". Over the last forty years, the work of hundreds of X-ray crystallographers and NMR spectroscopists has repeatedly confirmed this view, as the description of structure and function of proteins is now frequently understood at the atomic level. The classical experiments of Anfinsen and co-workers (Anfinsen et al., 1961; Anfinsen, 1973) demonstrated that the enzyme ribonuclease could be denatured and refolded without loss of enzymatic activity. This proved that a protein's amino acid sequence contains sufficient information to define its three-dimensional structure in a particular environment. The acceptance of this tenet has led to various efforts to predict the conformation (3D shape) of proteins based on sequence and secondary structure information.

Three dimensional protein structure prediction methods can be broken down into two different classes or methods. The first of these is called comparative (homology) modeling or threading. Both homology modeling and threading rely on the similarity between the target sequence (the sequence of the protein to be modeled) and at least one known 3D structure. The second approach, called *de novo* or *ab initio* structure prediction is aimed at predicting the structure of a target protein from its sequence alone, without depending on any similarity between the target sequence and any known structure.

Comparative or homology modeling is the most straightforward and accurate way to predict the 3D conformation of a protein. First, using sequence searching methods one or more known structures (i.e. templates) that are closely related (> 30% sequence identity) to the target sequence are found. The target sequence is then aligned with one or more of the template structures; and then the model is built and evaluated (Figure 1.1). The templates for modeling are normally found by sequence comparison methods, such as BLAST or PSI-BLAST (Altschul et al., 1997). The accuracy of a comparative model is related to the percentage of sequence identity on which it is based (Renom et al., 2000; Sanchez and Sali, 1998; Koehl and Levitt, 1999). High accuracy (1 Å RMS error for the main chain atoms) comparative models typically require more than 50% sequence identity to their templates. Medium accuracy models (1.5 Å RMS error for main chain atoms) are based on 30-50% sequence identity. Finally, low accuracy models are based on less than 30% sequence identity. The accuracy and reliability of models produced by *de novo* methods is much lower than that of comparative models based on alignments with more than 30% sequence identity.

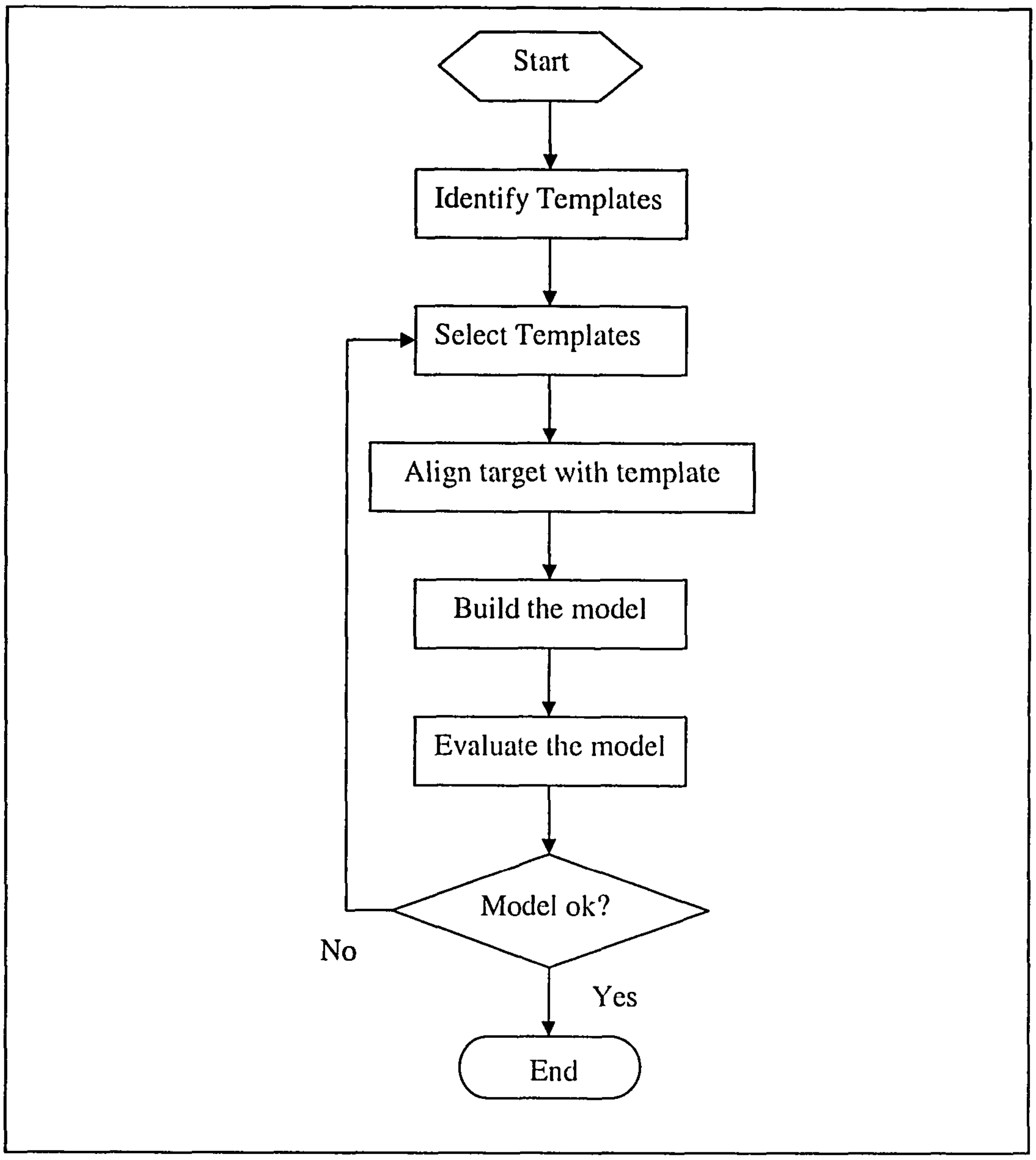


Figure 1.1 *An outline for a general algorithm used for Comparative Modeling.*

Unlike homology modeling, which generates relatively accurate 3D models, threading is limited to generating approximate models (~5 Å RMSD). In threading, fold assignment and alignment are obtained by threading the unknown sequence through each of the structures in a library of known folds (Torda, 1997). Threading gets its name as it superficially resembles the action of threading a tube down a hollow plumbing pipe system, during which the tube takes the shape of the pipe. If we view the backbone of a protein to resemble a highly contorted piping system, and we threaded a completely different (probe) protein sequence through this backbone (pipe) then the sequence similarity would determine how well the "probe" fit in the "pipe". To obtain the best fit, this process can be automated and different probe sequences can be run through the protein backbone pipe and the measure of fit evaluated by some empirical energy term or a measure of packing efficiency (Wishart, 2004).

There are two different approaches to threading: 3D threading, which is classified as a distance-based method (Novotny and Bruccoleri, 1984; Jones et al., 1992; Bryant and Lawrence, 1993) and 2D threading, which is defined as a prediction-based method (Sheridan et al., 1985; Rost et al., 1997). In 3D threading, coordinates are calculated for the probe sequence and the energy term is evaluated based on these coordinates. Generally 3D threading is a computationally expensive process that requires structure alignment, structure generation and structure/energy evaluation. 3D threading also requires complete structure (PDB) databases (>2 GBytes) to conduct these alignments/comparisons. On the other hand, 2D threading is relatively fast and simple and requires only modest computational resources and relatively small databases (<1 MByte). Instead of using 3D coordinates, 2D threading uses (predicted) secondary structure information as the primary evaluation criterion. While 3D structures or folds predicted from these threading techniques are not of high quality (RMSD > 3 Å), they can reveal the approximate shape and fold of proteins that seem to have no known structural homologues (Madej et al., 1995).

While comparative modeling and threading are limited to protein families with at least one known structure, *de novo* methods do not make any such assumptions. *De novo* methods start from the assumption that the native state of a protein is at the Gibbs free energy minimum (The Gibbs free energy minimum occurs when the enthalpic and the entropic contributions are at their lowest values). This method then carries out a large-scale search of conformational space for acceptable tertiary structures that are particularly low in free energy for the given amino acid sequence. The key components of these *de novo* methods are the procedure for efficiently carrying out the conformational search and the free energy function used for evaluating possible conformations. Many *de novo* methods have been developed which use simplified models of proteins, simplified free energy functions, and coarse-grained search strategies to find the optimum potential function and to reduce the conformational search space (Simons et al., 1997; Samudrala et al., 1999). In simplified models, contiguous segments of secondary structures can be represented as rigid bodies or the residues can be modeled as beads. A conformational search is then done to find the most energetically favorable arrangement of these elements.

Such coarse-grained representations can significantly reduce the size of the conformational search space. Residues can also be categorized into hydrophobic and hydrophilic monomers, arranged onto a 3D lattice and the free energy of the resulting conformation calculated (Chan and Dill, 1993). In this case, a simplification is made by reducing the search space to a discrete grid. Other approaches to reduce the search time or optimize the search space have employed distance geometry. Huang et al., (1998) used inter-alpha carbon distances as input constraints for metric matrix distance geometry to generate the structures for small helical proteins. This method was found to be particularly accurate and fast for the relatively small number of proteins tested. Most recently, (Rohl et al., 2004) have used a method that builds proteins from fragments. This algorithm, called Rosetta is by far the most successful *de novo* structure prediction algorithm. In Rosetta, short segments of the protein chain flicker between different local structures which are consistent with their local sequence, and folding to the native state occurs when these local segments are oriented such that the free energy of the protein is a minimum.

Predicted protein structures can be used in a variety of applications. High and medium accuracy comparative models are helpful in refining functional predictions that have been based on a sequence match alone as ligand binding is more directly determined by the structure of the binding site than by its sequence. The utility of low accuracy comparative models can be found in their ability to help reveal common structural principles or possible binding/active sites. Low accuracy or low resolution models have also been used in molecular reconstruction efforts that combine the atomic-scale predictions of protein structures with electron microscopy information.

This was recently shown in the determination of a molecular model of the yeast ribosome, whose construction was achieved by fitting comparative models of many ribosomal proteins into the electron microscopy map of a ribosomal particle (Spahn et al., 2001).

However, the accuracy of *de novo* models is generally too low for problems requiring high resolution structure information. Instead, the low resolution models can reveal structural and functional relationships between proteins not apparent from their amino acid sequences and provide a framework for analyzing spatial relationships between evolutionarily conserved residues or between residues shown experimentally to be functionally important (Bonneau et al., 2001).

Improvements in the accuracy of models produced by both *de novo* and comparative modeling approaches will obviously require newer and better approaches. These may include methods that more finely sample the conformational space or the development of better free energy or scoring functions that are sufficiently accurate to distinguish native structures from the nonnative conformations. Despite many years of progress in molecular simulation methods, attempts to refine models that are relatively close to the native structure have met with relatively little success (Wedemeyer and Baker, 2003). This failure can be attributed to the inaccuracies in the potential functions used in the simulations, particularly in the treatment of electrostatics and solvation effects. Improvements in sampling strategies may also be required as will improvements in our understanding of other aspects of physical chemistry, thermodynamics, and water structure (McDonald et al., 1997; Wang et al., 2001).

1.4 Bioinformatics and Protein Structure Analysis

The number of solved protein structures along with the quantity of related structural information has been growing at an exponential rate for more than 30 years (Figure 1.2). This is due mainly to the near-continuous technological progress in X-ray crystallography, NMR spectroscopy and computer technology. As protein structures are being solved and deposited into the PDB at an ever increasing rate, there is a growing need for better and faster processing techniques to analyze or compare 3D structures to one another – not just their sequences. This need for biological data management and

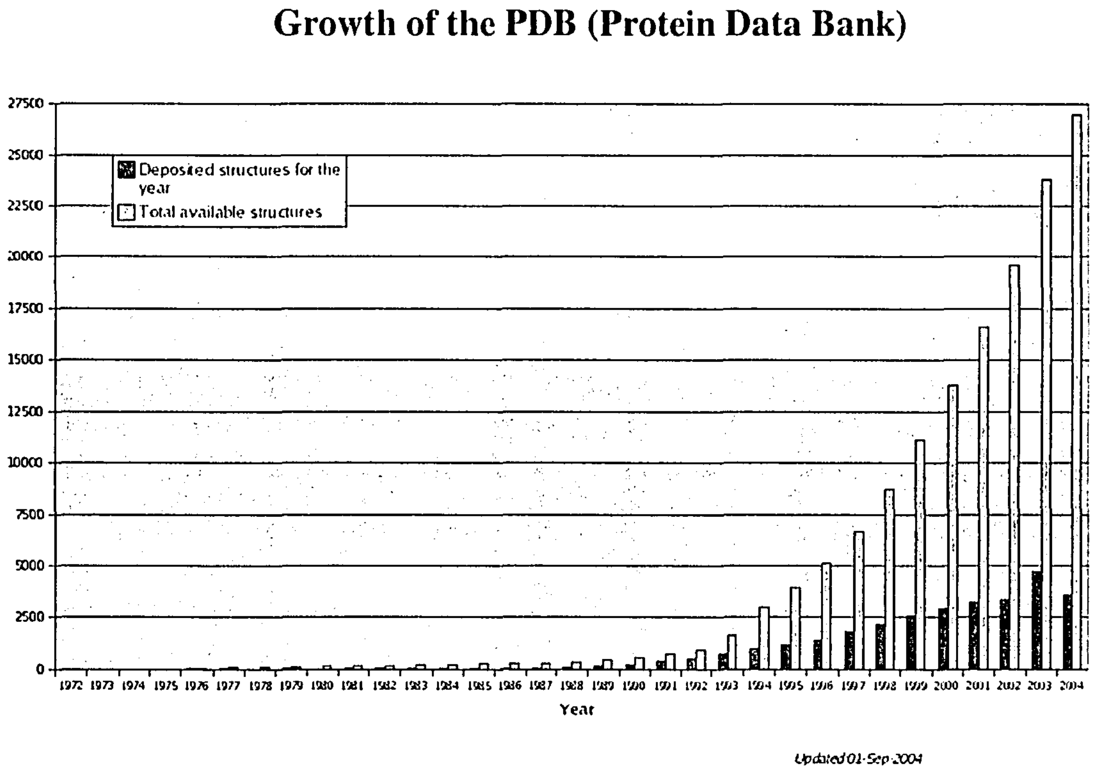


Figure 1.2 *The exponential growth in the number of total protein structures deposited in the PDB (Protein Data Bank) (http://www.rcsb.org/pdb/holdings.html).*

processing has led to the development of a new and highly specialized area of life science called bioinformatics. Simply stated, bioinformatics is a field of information technology which endeavors to improve the storage, management and analysis of biological data. Bioinformatics has found its way into many areas of life science, including medical and pharmaceutical research. Bioinformatics is also playing an increasingly important role in structural biology and structural proteomics. One way in which bioinformatics is having an impact in structural biology is in the way that far more information and substantially better analytical services for protein structures are being made available through the web. These developments in bioinformatics motivated us into creating several web-based tools to facilitate structure analysis and structure/dynamic interpretation. These are described in more detail in Chapters 3 and 4 of this thesis.

Specifically, in Chapter 3, I describe a web server that can be used to robustly compare and superpose multiple protein structures of widely differing shapes or sequence similarity. In Chapter 4, I describe another web server which is able to calculate and display realistic larger-scale dynamics and state transitions between 2 or more protein structures. The motivation for these works is described in the following two sections.

1.5 Superposition of 3D protein structures

In the same way that sequence comparisons can provide tremendous insight into the origins, function, location, interactions and activity of a protein, so too can structure comparisons. In fact, because structure is actually much more conserved than sequence, structure comparisons allow us to look even further back into biological prehistory than sequence comparisons (Orengo, Pearl and Thornton, 2003). For homologous proteins (similar ancestry), structure comparison essentially provides the "gold standard" for sequence alignment and can clearly elucidate the common ancestry of any given pair of proteins. For non-homologous proteins, structural comparisons allow us to identify common substructures of interest. These structure comparison methods also allow us to classify proteins into clusters or groups, thereby permitting the development of a consistent and evolutionary meaningful structural taxonomy. In terms of drug discovery, structure comparisons can also be used to identify structurally similar receptor cavities or they can be used to facilitate docking studies among structurally similar proteins.

The most common method for 3D structure comparisons is called structure superposition. Superposition is simply the process of orienting an object until it can be overlaid (superimposed) on top of a similar object. Before the structures are overlaid, it is necessary to establish equivalence (an alignment) between the residues of the proteins. This alignment can be done based on a sequence alignment or a structural alignment. After the correspondence between the residues have been obtained, the best superposition of the matching features that minimizes the RMSD (root mean square deviation) is required. Lagrangian multipliers, quaternion methods and matrix diagonalization techniques (MacLachlan, 1982; Kabasch, 1978; Kearsley, 1990) have been used for quite some time for 3D superpositions. Throughout the

1970s and 1980s, a number of standalone computer programs were developed that employed one or more of these methods (Diamond, 1992).

Unfortunately, a large number of these standalone superposition programs are not compatible with common operating systems and compilers. This greatly limits their uses to specific operating system and web browsers. Some programs are also quite restrictive in what can be superimposed (structures must be of identical length, must have the same number of atoms), how many molecules can be superimposed (most permit just two molecules to be superimposed), how the molecules are superimposed and how different two structures can be when superimposed. Furthermore, many of these tools do not seem to address the growing list of needs of many of today's structural biologists. These include the need to compare non-homologous or highly dissimilar structures, the need to compare "open" and "closed" forms of enzymes or metal binding proteins, the need to rapidly and accurately superimpose more than two structures or the need to extract far more quantitative detail about a structure superposition or comparison. To overcome these ongoing problems, we developed a robust macromolecular superposition web server called SuperPose. SuperPose appears to overcome the underlying problems amongst standalone programs regarding complex interfaces, platform incompatibility, sequence length and limited superposition capability. Furthermore, SuperPose is sufficiently robust that it can be used to help generate state-switch models for protein dynamics (as described below).

1.6 Protein Dynamics: Its importance to drug design and understanding of proteins

Often macromolecules (proteins and nucleic acids) carry out their functions by moving key parts of their enzymatic or active site machinery in a particular way or at a particular rate. Thus motion often serves as an essential link between structure and function. In particular, protein motions are involved in numerous basic functions such as catalysis, transport of metabolites, formation of large assemblies, cellular locomotion and enzymatic activity. Often, these motions can be isolated to specific domains or loops on the surface (or active site) of a given protein or enzyme. Some examples of where these dynamic events play a key role in enzymatic processes include citrate synthase and GroEL.

Citrate synthase catalyzes the condensation of acetyl-CoA and oxaloacetate during the citric acid cycle. The free enzyme (a dimer) is in an open form with two domains that form a cleft containing the oxaloacetate binding site. When oxaloacetate binds, the smaller domain undergoes an 18° rotation which closes the cleft. This represents a substantial and absolutely critical dynamic event that relates to the function of citrate synthatse. Another example of how dynamics relate to function can be found with chaperonin proteins. The E. coli chaperonin GroEL and its co-chaperonin GroES constitute a molecular machine that assists in the folding of nascent and misfolded polypeptides. GroEL is composed of 14 identical 57-kDa subunits, while GroES consist of 7 similar10-kDa subunits. Each GroES subunit uses a mobile loop with a conserved hydrophobic tripeptide for interaction with GroEL

(Landry et al., 1993). The mobile loops are approximately 16 residues in length and undergo a transition from disordered loops to beta hairpins (Figure 1.3).

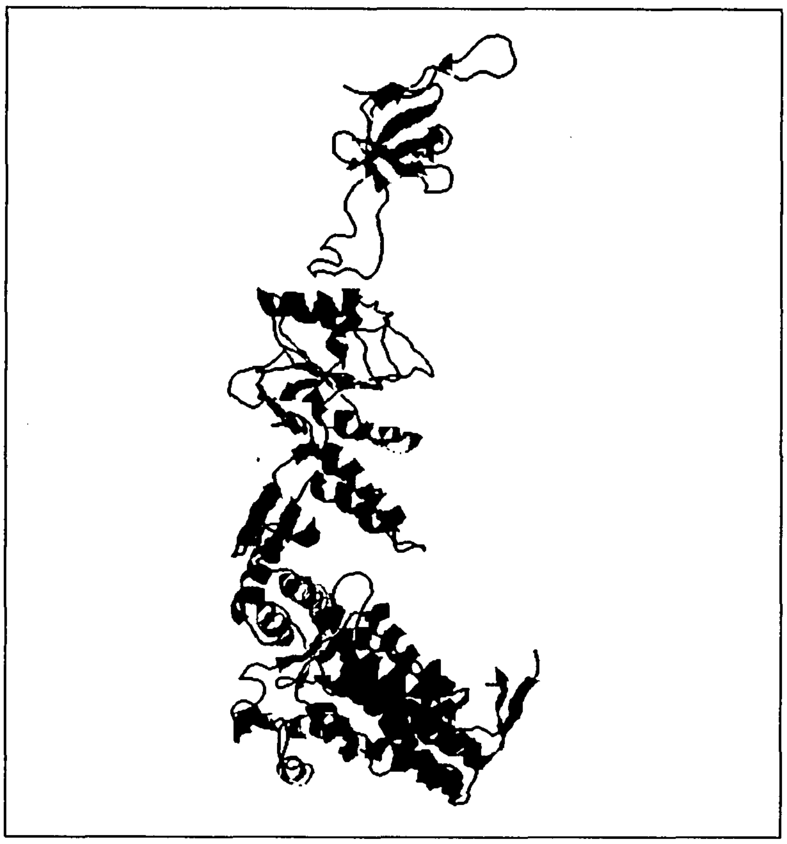


Figure 1.3 *The GROEL/GROES chaperonin complex.*

GroEL is not alone in utilizing surface loop dynamics to control its function. Surface loop movements are frequently used in substrate binding events. Some examples of such movements can be found in streptavidin and the Met repressor. A loop situated near the active site of streptavidin refolds over a biotin molecule once it is docked to the protein, thus protecting it from the environment. When the Met repressor binds to DNA, an eight residue loop changes its hairpin configuration into a conformation that wraps around the DNA phosphate backbone.

Macromolecular motions are amongst the most complicated biological phenomena that can be studied. They typically involve sub-angstrom changes occurring in thousands of atomic coordinates over periods of picoseconds or less. One of the best ways to study protein motions is through "movies", and they are particularly attractive when they are freely accessible over the web. Molecular movies may be generated using molecular dynamic (MD) simulations (via XPLOR or CHARMM) (Brünger et al., 1988; Brooks et al., 1983) and then superimposing "snapshots" of a large number of the structures calculated at different MD time steps. By running a rapid GIF rendering tool at 8-12 frames per second one can generate the illusion of motion on just about any web browser. Unfortunately, because of the computational overhead of MD simulations, these MD movies can take hours or days to generate. Alternately, cruder, perceptibly jumpy movies may be made by cycling between ensembles of NMR structures that have been superimposed on one another. These are not very visually satisfying, but they are reasonably accurate and relatively easy to generate. Finally, it is also possible to generate movies of molecular motions by interpolating between two different structures or isoforms of the same molecule. This may be done either adiabatically (Gerstein et al., 1999) using activated dynamics or it may be done via simple coordinate interpolation (vide infra).

A number of protein movies or movie programs have been described and made available on the web. In 1995 a custom movie of calmodulin moving between open and closed states was created and made web-accessible (Vonrhein et al., 1995). Similar work was done by Sawaya *et al.* (1997) who created movies of crystal structures of the beta subunit of DNA polymerase. Ray-traced 3D molecular dynamics simulation of acetylcholinesterase from mutagenesis data has also been made available (Gilson et al., 1994; Faerman. et al., 1996). Xu et al. (1997) used the technique of normal mode analysis to produce a morph movie of GroEL from structural data. More recently, Gerstein et al. developed a comprehensive database of macromolecular motions and a web server which can produce so-called dynamic morphs of proteins (Gerstein et al., 1999).

While very few servers exist on the web that display proteins morphing into each other, most of them use molecular dynamics to calculate the intermediate structures. Such methods are time consuming and computationally expensive. In Chapter 4, we describe a web server that appears to circumvent many of the shortcomings of existing motion generating tools. This server, called MovieMaker produces movies of proteins morphing into each other as well as it produces movies of other macromolecular motions such as oligomerization, docking, small scale vibrational and structure ensemble motions. For large scale motions between two proteins, the program uses a robust method of structure superposition (SuperPose) that allows it to automatically detect hinge motions or large domain motions and to appropriately superimpose the structures so that these motions can be more completely described. Secondly, the program uses coordinate interpolation rather than dynamic simulation (via XPLOR or other MD simulation software) to rapidly generate the ensemble of progressively changing coordinates needed to create a realistic and accurate molecular movie. The server is unique in its ability to automatically and rapidly calculate motions for a single chain, multiple chains, and protein pairs with considerable structural differences. It is also unique in its ability to let the user control a significant number of viewing and rendering options (on/off side chains, ball-stick representation, ribbon representation, continuous rotation about the X,Y or Z axis, viewing orientation, color, etc.).

1.7 Outline of this dissertation

This thesis consists of 5 chapters. The first chapter provides the background and context to the work and results described in the next 4 chapters. Chapter 2 describes a protein tertiary structure determination/prediction program that uses a combination of distance geometry and chain mapping to generate viable protein structures from known or predicted secondary structure components. Chapter 3 describes a protein structure superposition web server that can be used for detailed structure comparisons between two or more proteins having as little as 10% sequence identity or only fragmentary structural similarity. In Chapter 4 another web server is described which uses a simple, but rapid approach to predicting and rendering molecular motions. The fifth chapter provides a general discussion, a brief description of future directions and a conclusion.

1.8 My contribution to this dissertation

The program described in Chapter 2 was developed almost entirely by me, with occasional advice and periodic assistance provided by my supervisor, Dr.

Wishart. Some portions of the code were obtained from the web and modified while some portions (work of summer students, Philip Luk and Robert Yang) were rewritten and refined. With regard to the work described in Chapter 3 (SuperPose), it is important to acknowledge the work done by Dr. Gary Van Domselaar and Haiyan Zhang in Dr. Wishart's laboratory. Dr. Van Domselaar developed the web interface using perl and cgi scripting. The code for the sequence and structural alignments which are needed prior to superposition was also written by Dr. Van Domselaar. However, the code for the superposition operation, RMSD calculations, difference distance matrix and average structure calculations were developed entirely by me. The superposition program was translated in C from FORTRAN to improve platform compatibility and to permit faster execution. Haiyan Zhang helped with developing the graphs for the RMSD and difference distance matrix plots.

With regard to the work described in Chapter 4, the cgi and perl programs from the SuperPose server were reused, but with additional modifications. Again, this was done by Dr. Gary Van Domselaar. The programs for calculating or interpolating the motions and generating the images were developed, tested and written in C by me.

1.9 References

Alexandrov, N.N., Nussinov, R. and Zimmer, R.M. (1995) "Fast protein fold recognition via sequence to structure alignment and contact capacity potentials", *Pacific Symposium on Biocomputing '96 (Lawrence Hunter & Teri E. Klein, Eds), World Scientific Publishing Co., Singapore,* pp. 53-72.

Altschul, S.F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W. and Lipman, D. J. (1997) "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs", *Nucleic Acids Res.* **25**: 3389-3402.

Anfinsen, C. B. (1959) "The Molecular Basis of Evolution", *John Wiley & Sons, New York.*

Anfinsen, C. B.(1973) "Principles that govern the folding of protein chains", *Science.* **181**: 223-230.

Anfinsen, C. B., Haber, E., Sela, M., and White, F. H. (1961) "The Kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain", *Proc. Natl. Acad. Sci. U.S.A.* **47:**1309-1314.

Bonneau, R., Tsai, J., Ruczinski, I., Baker, D. (2001) "Functional inferences from blind ab initio protein structure predictions", *J Struct Biol.* **134**:186-190.

Brooks, B. R., Bruccoleri, R. E., Olafson, B. D., States, D. J., Swaminathan, S. and Karplus, M. (1983) "CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations", *J. Comp. Chem.* **4**: 187-217.

Brünger, A. T. (1988) "X-PLOR Reference Manual 2.1", *Yale University, New Haven.*

Bryant, S.H., and Lawrence, C.E. (1993) "An empirical energy function for threading a protein sequence through a folding motif", *Proteins.* **5**: 92-112.

Chan, H. S. and Dill, K.A. (1993) "The protein Folding Problem", *Physics Today.* Feb.24-32.

Crippen, G.M. (1981) "Distance Geometry and Conformational Calculations", *Chemometrics Research Studies Series 1. New York, Wiley.*

Crippen, G. M. and Havel, T. F. (1988) "Distance Geometry and Molecular Conformation", *Chemometrics Research Studies Series 15. New York, Wiley.*

Diamond, R. (1992) "On the multiple simultaneous superposition of molecular structures by rigid body transformations", *Protein Sci.* **1**:1279-1287.

Faerman, C., et al. (1996) "Site-directed mutants designed to test back-door hypotheses of acetylcholinesterase function", *FEBS Lett.* **386**: 65-71.

Flower, D.R. (1999) "Rotational Superposition: A Review of Methods", *J.Mol.Graph Model.* **17**: 238-244.

Gerstein, M., Jansen, R., Johnson, T., Tsai, J., and Krebs, W. (1999) "Studying Macromolecular Motions in a Database Framework: from Structure to Sequence", *Rigidity theory and applications. Kluwer Academic/Plenum Publishers.* pp. 401-442.

Gilson, M. K., et al. (1994) "Open "Back Door" in a Molecular Dynamics Simulation of Acetylcholinesterase", *Science.* **263**:1276-1278.

Harrison, P.M., Bamborough, P., Daggett, V., Prusiner, S. B., and Cohen, F.E. (1997) "The prion folding problem", *Curr. Opin. Struct. Biol.*7:53-59.

Huang, E.S., Samudrala, R., Ponder, J.W. (1998) "Distance geometry generates native-like folds for small helical proteins using the consensus distances of predicted protein structures", *Protein Science.* **7**:1998-2003.

Hung, L.H., Samudrala R.(2003) "PROTINFO: Secondary and tertiary protein structure prediction", *Nucleic Acids Research.* **31**: 3296-3299.

Jones, D.T., Taylor, W.R., and Thornton, J.M. (1992) "A new approach to protein fold recognition", *Nature.* **358:**86-89.

Kabsch, W.A. (1978) "Discussion of solution for best rotation of two vectors", *Acta. Crystallogr. A.* **34**:827-828.

Kearsley, S.K. (1990) "An algorithm for the simultaneous superposition of a structural series", *J. Comput. Chem.* **11**:1187-1192.

Klebe, G. (2000) "Recent developments in structure-based drug design", *J. Mol. Med.* **78**: 269-281.

Krebs, W.G. and Gerstein, M. (2000) "The morph server: a standardized system for analyzing and visualizing macromolecular motions in a database framework", *Nucleic Acids Res.* **28**:1665-1675.

Landry, S.J., Zeilstra-Ryalls, J., Fayet, O., Georgopoulos, C., and Gierasch, L.M. (1993) "Characterization of a functionally important mobile domain of GroES", *Nature.* **364**:255-258.

Madej, T., Boguski, M.S., and Bryant, S.H. (1995) "Threading analysis suggests that the obese gene product may be a helical cytokine", *FEBS Lett.* **373**: 13-18.

McDonald, N.A., Carlson, H.A., and Jorgensen, W.L. (1997) "Free energies of solvation in chloroform and water from a linear response approach", *J. Phys. Org. Chem.***10**:563-576.

McLachlan, A.D. (1982) "Rapid comparison of protein structures", *Acta. Crystallogr. A.* **38**: 871-873.

Monge, A., Friesner, R.A. and Honig, B. (1994) "An algorithm to generate low-resolution protein tertiary structures from knowledge of secondary structure", *Proc. Natl. Acad. Sc. U.S.A.* **91**: 5027-5029.

Novotny, .J, Bruccoleri, R., and Karplus, M. (1984) "An analysis of incorrectly folded protein models. Implications for structure predictions", *J. Mol. Biol.* **177**: 787-818.

Orengo, C.A., Pearl, F.M. and Thornton, J.M. (2003) "The CATH domain structure database", *Methods Biochem. Anal.* **44**:249-271.

Pandey, A., and Mann, M. (2000) "Proteomics to study genes and genomes", *Nature.* **405**: 837-846.

Pedersen, J.T. and Moult, J. (1996) "Genetic algorithms for protein structure prediction", *Curr. Opin. Str. Biol.* **6**: 227-231.

Ringe, D. (1995) "Structure-aided drug design: crystallography and computational approaches", *J. Nucl. Med.* **36**: 28-30.

Rohl, C. A. and Baker, D. (2002) "De novo determination of protein backbone structure from residual dipolar couplings using Rosetta", *J. Am. Chem. Soc.* **124**: 2723-2729.

Rohl, C. A., Strauss, C. E., Misura, K. M., Baker, D. (2004) "Protein structure prediction using Rosetta", *Methods Enzymol.* **383**: 66-93.

Rost, B., Schneider, R., and Sander, C. (1997) "Protein fold recognition by prediction-based threading", J. Mol. Biol. **270**: 471-480.

Samudrala, R., Xia, Y., Huang, E., Levitt, M. (1999) "Ab initio protein structure prediction using a combined hierarchical approach", *Proteins.* **37**:194-198.

Sawaya, M.R., Prasad, R., Wilson, S. H., Kraut, J. and Pelletier, H. (1997) "Crystal structures of human DNA polymerase beta complexed with gapped and nicked DNA: evidence for an induced fit mechanism", *Biochemistry.* **36**:11205-11215.

Sheridan, R.P., Dixon, J.S., and Venkataraghavan, R. (1985) "Generating plausible protein folds by secondary structure similarity", *Int. J. Pept. Protein Res.* **25**: 132-143.

Simons, K.T., Kooperberg, C., Huang, E., Baker, D. (1997) "Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions", *J Mol Biol.* **268**: 209-225.

Spahn, C.M.T., Beckman, R., Eswar, N., Penczek, P., Sali, A., Blobel, G., and Frank, J. (2001) "Structure of the 80S ribosome from Sacharomyces cerevisiae - tRNA-ribosome and subunit-subunit interactions", *Cell.* **107**:373-386.

Thiel, K. A. (2004) "Structure-aided drug design's next generation",*Nat. Biotechnol.* **22**: 513-519.

Torda, A.E. (1997) "Perspectives on Protein Fold Recognition", *Curr. Opin. Struct. Biol.* **7**: 200-205.

Veselovsky, A.V., and Ivanov, A.S. (2003) "Strategy of computer-aided drug design", *Curr. Drug Targets Infect Disord.* **3**: 33-40.

Vonrhein, C., G. Schlauderer and G. Schulz. (1995) "Movie of the structural changes during a catalytic cycle of nucleoside monophosphate kinases", *Structure.* **3**:483-90.

Wade, N. (1997) "Scientists find a key weapon used by H.I.V., in *NewYork Times: New York.p.A1.*

Wang, W., Donini, O., Reyes, C.M., and Kollman, P.A. (2001) "Biomolecular simulations: recent developments in force fields, simulations of enzyme catalysis, protein-ligand, protein-protein, and protein-nucleic acid non covalent interactions", *Annu. Rev. Biophys. Biomolec. Struct.* **30**:211-243.

Wedemeyer, W.J. and Baker, D. A. (2003) "Efficient Minimization of Angle-Dependent Potentials for Polypeptides in Internal Coordinates", *Proteins: Struct. Funct. Genet.* **53**:262-272.

Wishart, D.S. (2004) in "Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins, Third Edition", Baxevanis, A.D. and Ouellette, B.F.F. (eds.) *Wiley, New York,* pp. 215-247.

Xu, Z., Horwich, A.L. and Sigler, P.B. (1997) "The crystal structure of the asymmetric GroEL-GroES-(ADP)7 chaperonin complex", *Nature.* **388**: 741-750.

Chapter 2: A Distance Geometry Approach for 3D Protein Structure Prediction

2.1 Introduction

Ever since the experiments of Anfinsen and co-workers (Anfinsen, 1973) proved that proteins can be denatured and refolded without the loss of activity, the study of how proteins fold into their 3D shape has been one of the most important and challenging problems in biological science. Indeed the opportunity to design and construct new proteins, better vaccines (Mutter, 1985), more effective drugs, improved herbicides, and safer pesticides (Blundell et al., 1987), all await a better understanding of the protein folding problem.

One might guess that a protein folds into its conformation (3D shape) by randomly exploring all the conformations available to it until it stumbles onto the correct one. A simple calculation made by Cyrus Levinthal (Levinthal, 1968), however, demonstrated that this is impossible. For example if we consider a 100 residue protein we know that each of its amino acids can have potentially dozens of different conformations, based on its main chain and side chain dihedral angles. If we take a very simplistic view and consider that each residue has only 2 possible conformations, then the whole chain will have 2^{100} conformations. If the protein tries each possible conformation during folding at a rate of 1 pico second, the time required for the folding process would be 10^{18} seconds (this is equal to 10^{10} years), which is greater than the age of the universe. Rather than taking such a long time, proteins appear to fold into their native 3D conformations within milliseconds or microseconds (Eaton et al., 1998; Mayor et al., 2000). This has led researches to believe that proteins fold to their native conformations via directed pathways rather than stumbling on them through random conformational searches (Dill and Chan, 1997).

Experimental observations indicate that the folding process begins with the formation of secondary structures (helices and beta sheets). This early stage is very rapid, with much of the secondary structure appearing within 5ms of the initiation of folding. Since native proteins contain compact hydrophobic cores, it is likely that the driving force in folding is a *hydrophobic collapse* (hydrophobic residues arranging themselves in the core). This collapsed state is called the *molten globule*, a form that has much of the secondary structure but little tertiary structure (completely folded 3D shape - Figure 2.1). Over the next few milliseconds, the secondary structure becomes stabilized and the final tertiary structure begins to form. In the final stage of folding, the protein undergoes a series of complex motions in which it attains its relatively rigid internal side chain packing and hydrogen bonding while it expels the remaining water molecules from its hydrophobic core (Matthews, 1993; Stickle and Rose, 1992).

Over the past two decades, a variety of computational approaches have been attempted to simulate protein folding (Cohen et al., 1979; Sternberg et al., 1982; Feldman and Hogue, 2000; Baker and Sali, 2001). Since it is evident that proteins do not sample each and every available conformer before reaching their lowest energy structure, various conformational search strategies have been tried to obtain the native or near- native protein structure from a number of unfolded conformations. Most of these conformational search procedures are highly CPU intensive and rely on extensive networks of distributed computers. These efforts have only been modestly successful and have typically required hundreds of CPU days to generate an approximate structure of small (~100 residue) protein. It has been estimated that it would take a Petaflop (FLOP = floating point operations per second, Peta = 10^{15}) approximately 1 CPU year to crudely fold a protein of 300 residues (Larson et al., 2002). In this chapter we describe a novel approach to address the conformational search problem which could aid in protein structure prediction. Our algorithm uses a reduced representation of protein secondary structure in combination with distance geometry, "atomic remapping" and heuristic pseudo energy evaluation that allows near-native protein structures to be generated in a matter of hours on almost any desktop computer. In the following pages, a brief description of various conformational search methods is provided followed by a detailed description of our structure generation algorithm.

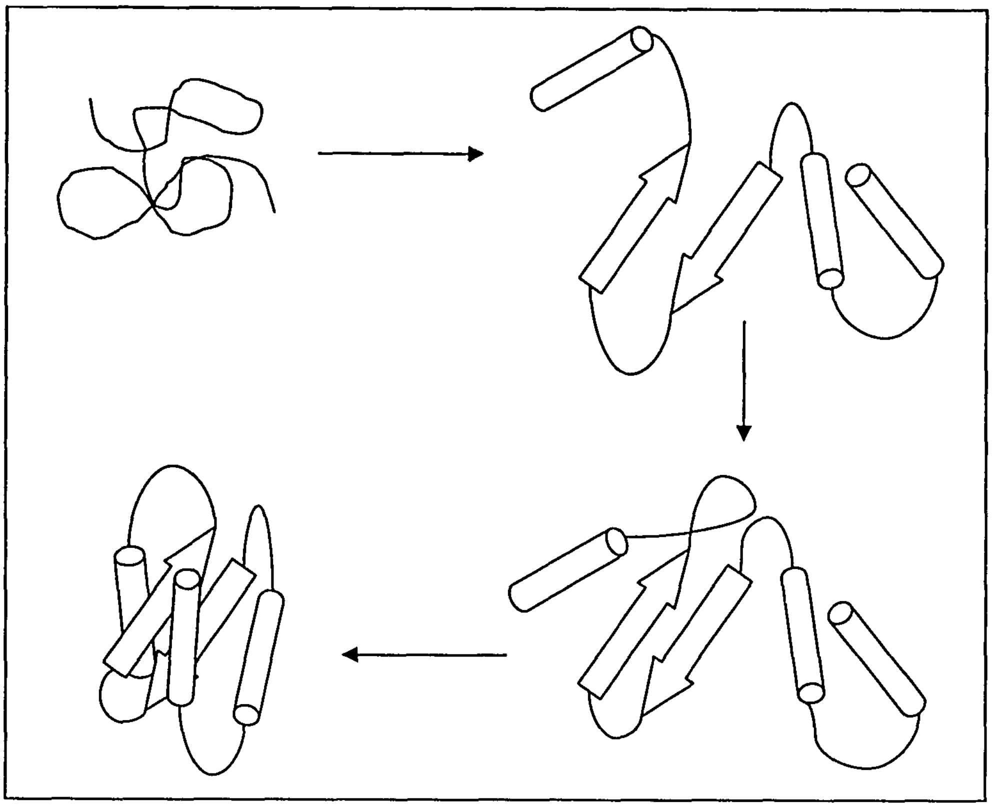


Figure 2.1 *A protein folding pathway.*

2.2 Conformational Search Strategies

Levinthal's paradox (Karplus, 1997) clearly demonstrated that proteins cannot sample all available conformations before reaching their lowest energy native structure. This implies that the search space is both finite and manageable. This realization has provided some hope to computational chemists trying to simulate protein folding on computers. Over the past three decades a number of different approaches aimed at overcoming the sampling problem have been attempted. These approaches can be grouped into four different categories: 1) systematic search algorithms; 2) Monte Carlo sampling and/or molecular dynamics; 3) genetic algorithms and 4) distance geometry (Zhou and Abagyan, 1999).

2.2.1 Systematic Search Algorithms

In systematic searching, conformational space is explored by making regular and predictable changes to the polypeptide conformation. In the simplest type of systematic search (a grid search), all rotatable bonds in the molecule are identified and these bond lengths and angles are kept unchanged throughout the process. Each of these bonds is then systematically rotated through 360° using a fixed increment and each generated conformation is subjected to energy minimization to derive the lowest energy conformer (Bruccoleri and Karplus, 1987). The search stops when all possible combinations of torsion angles have been generated and minimized. A major drawback of the grid search is that the number of generated structures increases rapidly with the number of rotatable bonds. The number of conformations equals $\prod_{i=1}^{N}(360/\theta_i)$ where N is the number of bonds and θ_i is the dihedral increment chosen for the ith bond. For example, if there are five bonds and an increment of 30° is used for each bond, then 248832 structures will be generated. If the number of bonds is increased to seven, then the number of structures increases to almost 36 million. If each structure takes just one second to minimize, the five bond problem will require 69 days of CPU time and the seven bond problem will take 415 days. As proteins contain hundreds of bonds, this approach is clearly not feasible with larger polypeptide fragments and has largely been abandoned.

2.2.2 Monte Carlo Sampling and Molecular Dynamics

To reduce the combinatorial explosion of systematic search, a random search of the conformational space can be performed. Random Monte Carlo (Metropolis et al., 1953) sampling techniques have been used with some success (Hunt et al., 1994). These techniques start with an initial conformation and make a change either in the atomic Cartesian coordinates or to the torsion angles of the rotatable bonds. The energy of the molecule before and after the change is evaluated and the move is accepted if the energy is reduced. Sometimes, a move may also be accepted if it increases the energy as this allows the protein to escape from local minimum energy traps. This process continues until some predetermined condition occurs or until a certain number of moves have been made. David Baker's ROSETTA program (Simons et al., 1999) employs such a Monte Carlo (MC) approach. ROSETTA employs a fragment-based Monte Carlo approach wherein the program uses a large number of pre-built peptide fragments to build an initial structure, and then swaps fragments for each Monte Carlo move. ROSETTA is generally recognized as one of the best MC sampling methods currently available.

Molecular dynamics (MD) on the other hand, makes use of Newton's laws of motion to fold an arbitrary starting conformation towards the native state. Theoretically, if run to completion, MD should be able to fold a protein into its correct shape. In practice, however, a number of problems are faced. In order to simulate movement on a computer, the process must be broken down into discrete time steps. From Newton's second law of motion, we know that the rate of change of momentum is equal to the force applied, or $dp = F \cdot dt$, or $\Delta p \approx F \cdot \Delta t$ for sufficiently small Δt (where p = momentum and F is the net force acting on the body). Also, $F = -\vec{\nabla} E$ where E is the energy of the protein, as a function of atomic coordinates. This energy function consists of several components: a) covalent bonds, b)van der Waals energies to ensure that atoms do not have steric clashes, and 3) an electrostatic term accounting for Coulomb's law, the long-range force between charged and partially charged atoms. All these terms need to be parameterized and these parameters are collectively referred to as the force field. Some popular molecular force fields developed for proteins include CHARMM (Brooks et al., 1983), AMBER (Weiner et al., 1984), and GROMOS (Gunsteren et al., 1987) (Figure 2.2).

Energy Function E = $K_r(r_i - r_j)^2$ +	(Bond length component)
$K_\theta(\theta_i - \theta_j)^2$ +	(Bond bending component)
$K_\varphi(1 - \cos(\phi_j))^2$ +	(Bond torsion component)
$q_i q_j / 4\pi\varepsilon r_{ij}$ +	(Coulomb forces component)
$A_{ij}/r^6 - B_{ij}/r^{12}$ +	(Van der Waals component)
$C_{ij}/r^{10} - D_{ij}/r^{12}$	(Hydrogen bond component)

$K_r = Bond\ stretching\ force\ constant$ (~ 600 $Kcal/\ mol/A^2$)

$K_\theta = Bond\ bending\ force\ constant$ (~ 80 $Kcal/\ mol/radian^2$)

$K_\varphi = Torsional\ angle\ force\ constant$ (~ 1 $Kcal/mol$)

$q = Charge\ on\ an\ atom,\ r = separation\ between\ two\ atoms$

$A_{ij}, B_{ij} = Van\ der\ Waals\ repulsive\ and\ attractive\ force\ constants$

$C_{ij}, D_{ij} = Hydrogen\ bond\ force\ constants$

Figure 2.2 *Different parameters of a standard energy function.*

Δt is typically chosen to be around 1 femto second, which corresponds roughly with the time for a covalent bond to vibrate. Since a real protein takes milliseconds to fold, we will require 10^{12}-10^{15} energy and gradient calculations to fold a protein. This is simply not feasible with current computing power. In fact, the longest single MD simulation of a protein with solvent molecules so far conducted was 1 μsec. This extended simulation required several months on a Cray supercomputer (Duan and Kollman, 1998). As supercomputing power continues to progress, the "tardiness" of MD simulations will continue to be less significant. Recently, IBM invested $100 million into their Blue Gene project, a 1 million CPU, 1 petaflop shared memory machine that is intended to have enough compute power to use MD simulation to fold a protein (Pool, 1999). Distributed computing approaches have also been successfully used to fold very small peptides (Zagrovic, et al., 2001). However, improvements in force fields are still necessary for folding larger peptides, as there is no guarantee that, even if run to completion, MD will lead to the correct folded state. For instance, with current force fields it has been found that running MD to refine a structure prediction does not, in general, improve the prediction (Lee et al., 2001).

2.2.3 Genetic Algorithms

Another approach to addressing the conformational search problem borrows from the field of artificial intelligence (and from evolutionary biology). This is called the genetic algorithm (Forrest, 1993). Briefly, a genetic algorithm (GA) requires a population of individuals represented by their 'genes', or properties, and a fitness function which takes an individual as input and provides a score as output. An initial population of genes (in this case protein conformations defined by differing backbone torsion angles) is randomly generated to begin the simulation or conformational search. The subsequent search steps are carried out iteratively in "phases". Each phase typically consists of a mutation, reproduction and a crossover event (Figure 2.3). Mutation randomly changes genes (phi/psi angles) according to some specified mutation rate while reproduction tends to replicate more copies of individuals having a high fitness score. Crossover randomly selects pairs of individuals and swaps their genes (phi/psi angles) with each other. As this process repeats, the population becomes progressively more fit over time and is ultimately optimized by the fitness function (the population becomes more fit as the energy of the system gets lower). Thus the behavior mimics nature's survival of the fittest.

When GAs are applied to protein folding, each protein conformation represents an individual "organism" and the torsion angles resemble the genes. The fitness function is typically an energy scoring function, so that the algorithm will converge towards structures of optimal (minimum) energy. A number of attempts to use GAs to predict protein folds have been made with promising results. Pedersen and Moult (1997) were able to generate structures with an RMSD of 1-2Å for small 10-15 residue protein fragments. Reasonable structure predictions have also been made on small 50-70 residue domains (Cui et al., 1998). GAs are quite efficient at optimizing an ensemble of structures and are generally found to be superior to Monte Carlo optimization, but again, the choice of energy function to use is all important and no single energy function seems to work well for all proteins.

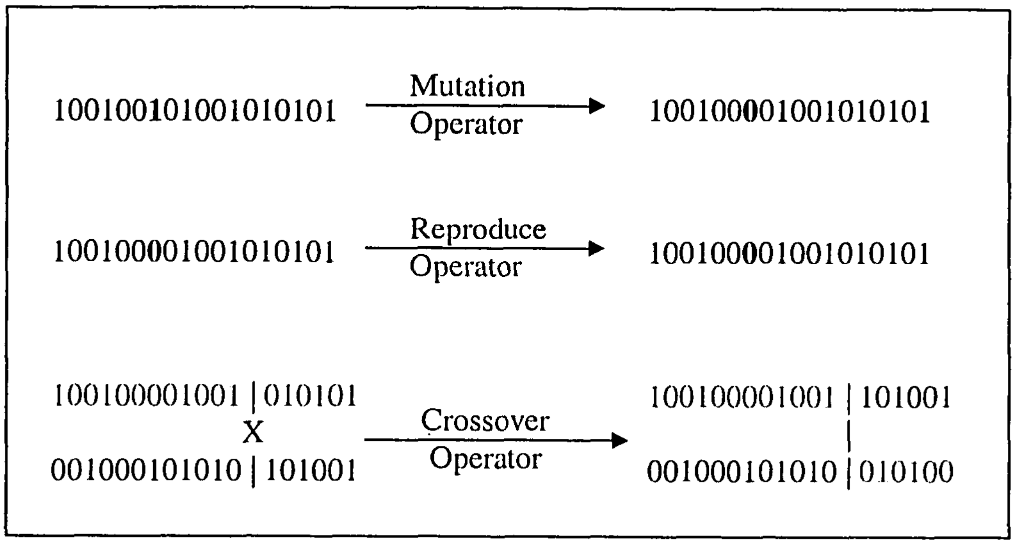


Figure2.3 *A simple diagram of a Genetic Algorithm showing the mutation, reproduction and the crossover operators. The ones and zeroes can be thought of as specimens or genes in a population.*

2.2.4 Distance Geometry

As we have already seen, MC sampling procedures, GA searches and MD simulations can take a very long time to generate viable protein structures. This begs the question: "Are there more efficient sampling procedures?" We believe there are. Instead of describing protein conformation at the scale of individual atoms, considerable savings in compute time could be achieved if the protein was described in terms of secondary structure elements or blocks. By treating protein folding as a packing problem defined by distance restraints, it is possible to frame the problem as a distance geometry problem (Kuntz and Oshiro, 1989).

Distance based methods are particularly useful when experimental information is available and inter-residue or inter-atomic distances can be used to generate a single structure or a cluster of structures. Indeed distance geometry is frequently used in generating structures via NMR, where thousands of distance restraints are measured. In some cases even less distance information is required. For example, Kuntz et al. (1979) used distance information from just three disulfide bridges to generate structures as close as 5 Å root mean square deviations (RMSD) from the native bovine pancreatic trypsin inhibitor (BPTI) fold. In addition to its application in experimental structure determination, distance geometry has also been used in theoretical studies of protein folding. One method utilized distances as a term in a force field or as a scoring function parameter. Specifically, Hanggi and Braun (1994) used inter-residue distance information as a scoring function to generate the structure of a 4 helical protein. Distance information has also been used as a component term in a molecular force field (Skolnick et al., 1997).

Most kinds of distance geometry techniques are based on a method called metric matrix distance geometry. This approach utilizes a mathematical projection from distance space to 3D space known as *embedding* (Crippen 1981; Havel et al., 1983; Crippen and Havel 1988). This method describes a molecule in terms of it's inter atomic or inter residue distances. Thus, in a molecule with N atoms, there are $N*(N-1)/2$ inter atomic distances which can be represented using an N x N symmetric matrix. Initially, a matrix of upper and lower inter atomic/residual distance bounds is calculated. This matrix contains the maximum and minimum possible values permitted for each distance in the molecule. Random values are then assigned to each distance between the upper and lower bounds and the distance space is converted into a set of Cartesian coordinates via the *embedding* process. The process of creating a distance matrix and the embedding protocol is described in more detail in Appendix A.

Using this type of algorithm, which will henceforth be referred to as "distance geometry" (DG), Taylor and co-workers determined that a set of native inter alpha carbon distances combined with local geometry information can correctly generate native-like tertiary structures (Aszodi et al., 1995). Huang et al. (1998) also used inter alpha carbon distances to generate native like folds for small helical proteins.

Encouraged by these results, we attempted to develop a DG algorithm that could rapidly generate a library of topologically feasible protein structures. The near-native structure can then be selected from the library by a heuristic scoring function such as one used by ROSETTA, PROTINFO or other protein folders (Simons, 1999; Hung and Samudrala, 2003).

Historically, inter atom/residue distances have been used in the distance matrix calculation. Typically, proteins have thousands of atoms and hundreds of residues and all these distances results in a fairly large distance matrix. Instead of using all distances between atoms and residues, we use a reduced representation of the protein's secondary structure. In this representation, the secondary structure elements are regarded as rigid blocks. The start and end coordinates of these blocks are regarded as points and approximate distances separating these points are then used to create a distance matrix. The length of each block and the distances separating them are far less than all inter-atom or inter-residue distances. This significantly reduces the size of the distance matrix which leads to a significant improvement in the time taken for the conformational search.

2.3 Systems and Methods

2.3.1 Protein Secondary Structures

Before the modified distance geometry approach is described in detail, a short description about the secondary structure elements of a protein is given below. The first X-ray structures revealed that proteins did not adopt regular or symmetrical structures, but appeared to be highly irregular. Certain structural motifs were observed to appear frequently. The most common motifs are the α-helix and β-strand (Pauling et al., 1951). The β-strands often form extended structures called β-sheets in which the strands are hydrogen bonded to each other and the strands can run in either parallel or anti-parallel directions. These secondary structural elements are connected by variable regions often referred to as "loops" and they adopt less regular structures. These elements fold compactly to form the tertiary structure of the protein (Figure 2.4).

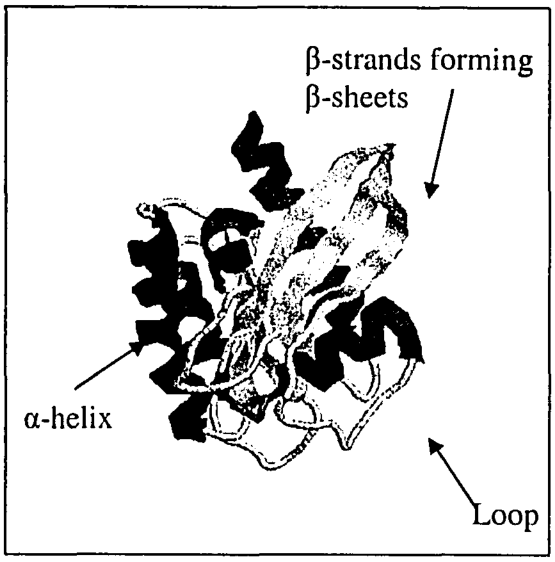


Figure 2.4 *Secondary structural elements of a protein.*

Our algorithm uses the target protein's sequence and predicted (or observed) secondary structure information as its initial input. From this information, the secondary structural elements are identified and the number of residues found in each secondary structure is noted. The secondary structure is then converted into an equivalent set of cylinders (helices), blocks (beta strands) and strings (loop regions). The lengths of each block and the allowable distances separating them are then used to create a distance matrix. Additionally, the start and end Cartesian coordinates for each block are obtained by the distance geometry process (Figure 2.5).

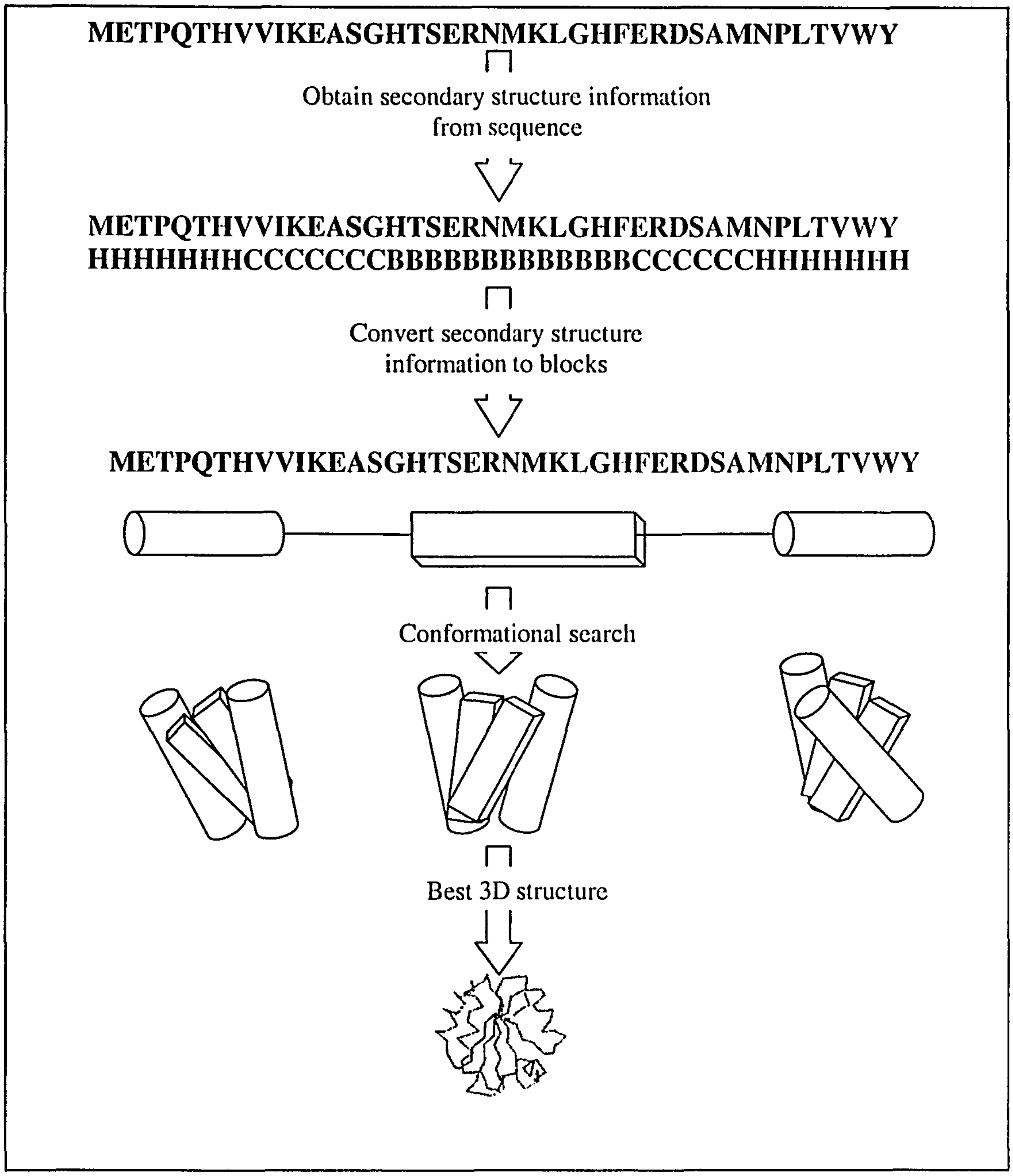

METPQTHVVIKEASGHTSERNMKLGHFERDSAMNPLTVWY
Obtain secondary structure information from sequence
METPQTHVVIKEASGHTSERNMKLGHFERDSAMNPLTVWY
HHHHHHHCCCCCCCBBBBBBBBBBBBBCCCCCCHHHHHHH
Convert secondary structure information to blocks
METPQTHVVIKEASGHTSERNMKLGHFERDSAMNPLTVWY
Conformational search
Best 3D structure

Figure2.5 *Representation of the secondary structure elements as blocks.*

The process of obtaining the distances and setting up of the distance matrix is described below. From the number of residues found in each secondary structure (helix or beta strand), the approximate lengths of each of these secondary structures are calculated using a heuristically derived formula. This was done by analyzing the lengths of helices and beta sheets for a set of all helical proteins and proteins with mixed secondary structures. The distance between the start and end alpha carbon atoms of these secondary structures were found and these were divided by the number of residues making up the secondary structure to give a rough per residue length of each secondary structure (Table 2.1).

Table 2.1 *Common lengths of helices and beta sheets per residue.*

PDB ID	Structural Class	Per residue length in Angstroms (Å)		
		H1/No. res.	H2/No.res.	H3/No.res.
1ROP	2α	39.76/27 = 1.47Å	34.79/24 = 1.45Å	
2SPZ	3α	16.71/12 = 1.39Å	16.37/12 = 1.36Å	20.39/15 = 1.36Å
1BW5	3α	18.65/13 = 1.43Å	15.39/11 = 1.40Å	20.77/14 = 1.48Å
		H1/No.res.	B1/No.res.	B2/No. res.
1AHO	1α,2β	14.33/10 = 1.43Å	16.11/6 = 2.68Å	16.11/6 = 2.68Å

α stands for alpha helix and β for beta strands.

Per residue length is obtained by dividing the length of secondary structure elements by the number of residues in each secondary structure

H1, H2 and H3 represent the first, second and third helices, while B1 and B2 represent the beta strands.

PDB stands for the Protein Data Bank.

From these commonly occurring lengths, the following formulae were developed to generalize the relationship between the number of residues and the lengths of helices, beta sheets and loops in Angstroms (equations 2.1, 2.2, 2.3).

$$Helix\ length = \left(1.04 + \left(\frac{Number\ of\ residues\ in\ helix}{100}\right)\right) * Number\ of\ residues\ in\ helix \quad (2.1)$$

$$Beta\ Sheet\ length = 2.7 * Number\ of\ residues\ in\ betasheet \quad (2.2)$$

Next, various loops in different proteins connecting the helices and beta sheets were analyzed to get a rough correlation between the loop lengths and the number of residues making up each loop (Table 2.2).

Table 2.2 *Table showing the lengths in Angstroms and the number of residues for 10 different loops of 8 different residue lengths taken from the Protein Data Bank.*

No. of res.		Lengths of different loops in (Å)									Avg. Len.(Å)
1	4.2	3.6	3.3	4.6	3.8	4.1	3.1	3.7	3.2	4.7	3.83
2	6.3	4.2	6.4	. 4.6	5.9	3.3	4.3	3.2	3.3	6.8	4.83
3	9.1	6.1	8.3	4.2	4.6	8.2	4.8	6.4	5.7	3.8	6.12
4	9.1	8.4	4.3	8.2	4.1	9.8	4.3	5.7	7.6	9.1	7.06
5	6.4	8.8	4.2	3.4	12.8	14.4	10.8	11.2	14.6	17.1	10.37
6	15	14.6	10.8	8.8	8.6	10.4	10.1	7.4	9.2	4.6	9.95
7	19.1	13.3	4.8	9.1	7.5	3.2	8.1	8.4	4.5	5.2	8.32
8	17.3	10.4	12.3	13.2	6.1	8.1	14.9	12.1	8.9	8.9	11.22

From this table, it is evident that there is no simple way to correlate loop lengths to the number of residues present in each loop. As a rough estimate the following formula was chosen to get an approximate loop length based from the residue numbers.

$$Loop\ length = 2.50 * Number\ of\ residues\ in\ loop \quad (2.3)$$

From the set of proteins given in Table 2.1, the minimum distance between the outer sides of helices and beta sheets was also calculated (Table 2.3).

Table 2.3 *Minimum separation between helices and beta sheets in Angstroms for four small proteins.*

PDB ID	Separation between helices			Separation between β strands and helices		
	H1-H2	H2-H3	H1-H3	H1-B1	B1-B2	B2-H1
1ROP	4.8Å					
2SPZ	7.1Å	6.4Å	5.9Å			
1BW5	5.9Å	6.0Å	8.7Å			
1AHO				4.3Å	4.1Å	5.3Å

H1, H2 and H3 represent the first, second and third helices, while B1 and B2 represent the beta strands.

This separation also depends on the length of the loop connecting the consecutive secondary structure elements. Since loops are highly flexible structures, there is no simple way to derive a formula to get the length of the loop (and the distance separating the consecutive secondary structure elements). From the distances in Table 2.3, a consensus was reached to set the separation distance between consecutive secondary structure elements (helices and beta sheets). The separation distance between secondary structure elements was determined by drawing imaginary axes along the centre of helices and beta sheets and the minimum separation was set to this axial separation. The internal structures of helices and beta sheets were also analyzed to obtain this minimum separation. The neighboring alpha carbons of a helix are separated by 3.8 Å while the closest separation between the alpha carbons of two neighboring helices is 4.8 Å. If the helix is modeled as a cylinder, then the diameter of the helix will be less than 3.8 Å. Approximating the diameter as 3.0 Å, a radius of 1.5 Å is obtained. Hence, as seen from Figure 2.6, the axial separation will be the radius + 4.8Å + radius (ie. 1.5 + 4.8 + 1.5 = 7.8 Å). Since these distances are not absolute and can vary from protein to protein, an approximate distance of 7.5 Å was chosen (Figure 2.6).

Similar analyses were done for a pair of parallel beta sheet proteins. Since beta strands can be simplified as planar structures, the axial separation of these strands is chosen as the minimum separation between each strand (Figure 2.7).

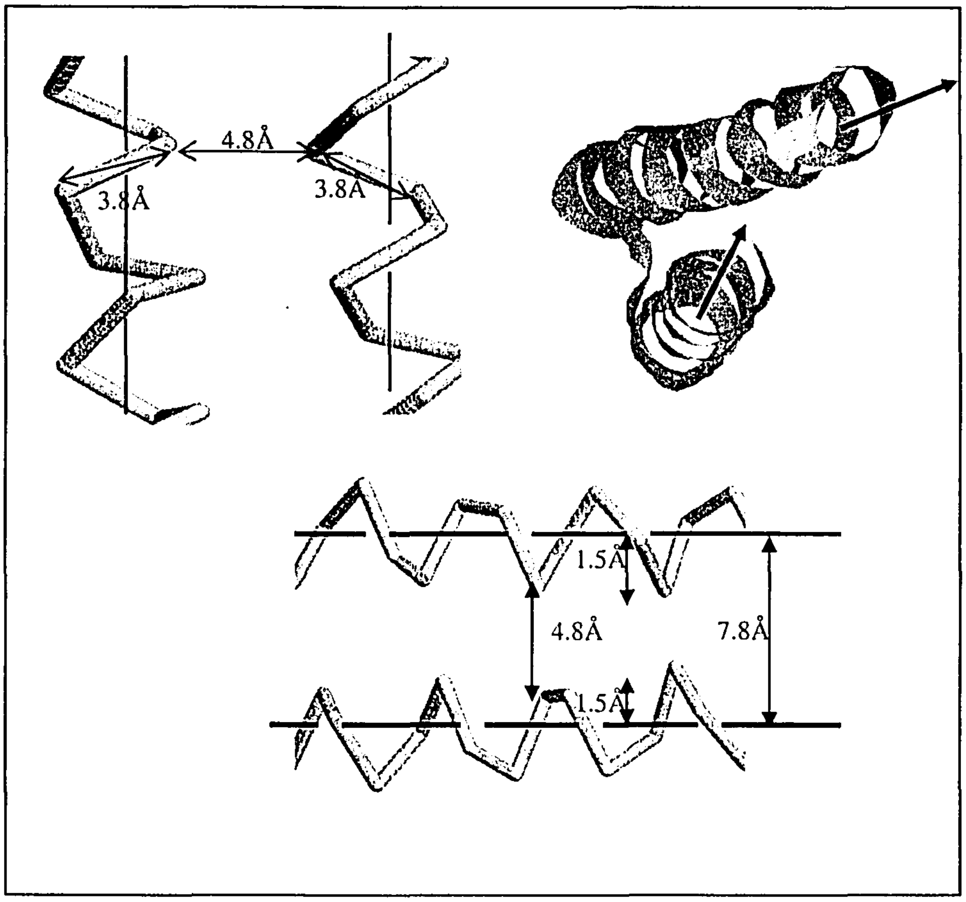


Figure 2.6 *Helix secondary structure and separation between the axes of helices.*

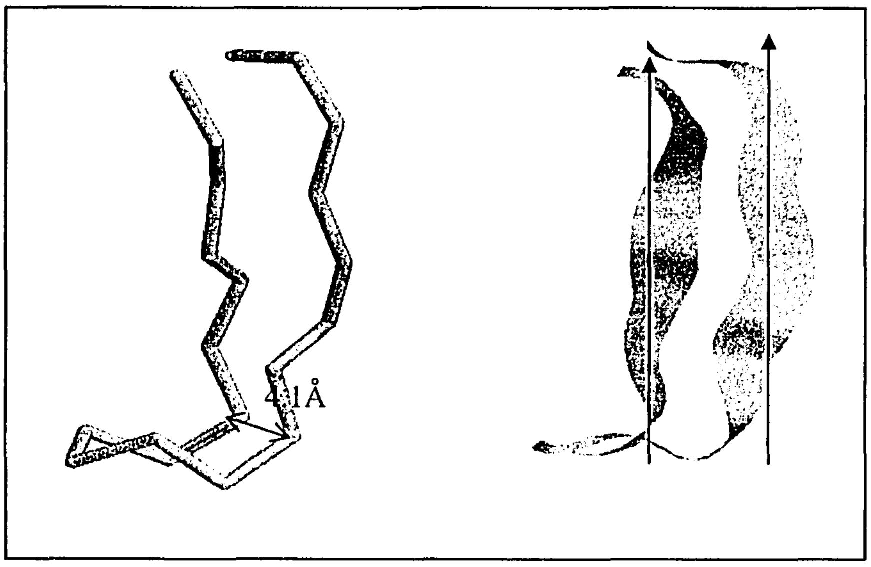


Figure 2.7 *Beta Sheet secondary structure and axial separation.*

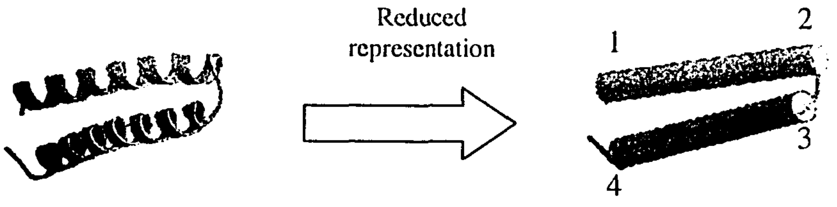
Since proteins exist as compact entities a rough measure of its compactness can be calculated by a radius of gyration. The complete protein can then be imagined to be enclosed in a sphere with a radius equal to the radius of gyration. Equation 2.4 was used to calculate this radius of gyration of the predicted protein (Huang and Powers, 2001).

$$\textit{Radius of gyration} = 3.875 * \left(\sqrt[3]{\textit{Number of residues in the protein}}\right) \qquad (2.4)$$

The maximum separation of non-consecutive structural elements was restrained to be within twice the calculated radius of gyration to limit the placement of all secondary structure segments to within a reasonably compact volume. Using these separation values and the lengths of the helices and beta sheets, an initial distance matrix can be created. These values and a distance matrix for a two-helix protein (PDB ID. 1ROP) is shown below (Figure 2.8).

```
Sequence:   MTKQEKTALN MARFIRSQTL TLLEKLNELD ADEQADICES LHDHADELYR SCLARFGDDG ENL
Sec Struct:   HHHHHHHH HHHHHHHHHH HHHHHHHHTT  HHHHHHHHH HHHHHHHHHH HHHHH
```

The beginning and end residues constituting loop fragments are removed and an input file is created with the remaining secondary structure elements. Thus, the secondary structure reduces to H27 L4 H24.



$D_{12} = 35.37$

$D_{23(\max)} = 10.00,\ D_{23(\min)} = 4.50$

$D_{34} = 30.72$

$D_{14(\max)} = 35.37,\ D_{14(\min)} = 7.50$

$$\xrightarrow[\text{Setup}]{\text{Distance Geometry}}\quad \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0.00 & 35.37 & 35.37 & 35.37 \\ 2 & 35.37 & 0.00 & 10.00 & 35.37 \\ 3 & 29.37 & 4.50 & 0.00 & 30.72 \\ 4 & 7.50 & 24.72 & 30.72 & 0.00 \end{array}$$

Figure 2.8 *Distance matrix setup of the 1ROP protein*

For a protein with N atoms, there are N*(N-1)/2 intra atomic pair-wise distances between all the atoms. If each residue is thought to have 7 atoms, then for this 53 residual protein, there will be 371 x 371 matrix entries if all pair-wise distances are taken into account. Using the reduced representation, we obtain a 4 x 4 matrix with reasonably well determined upper and lower bounds. The problem is therefore simplified by a factor of $(371*371)/(4*4) \cong 8000$ using this segmental approach.

Once the Distance Matrix is obtained, the second step, embedding is then used to derive the start and end Cartesian coordinates of these blocks. The embedding process creates the start and end coordinates of these rigid blocks, but it does not check whether the rigid blocks pass through each other. An excluded volume screening method is then employed to sift out these structures. This excluded volume creates lines from the start and end coordinates and calculates the closest distance between them. This algorithm weeds out lines that are within the minimum separation of 7.5Å (Figure 2.9).

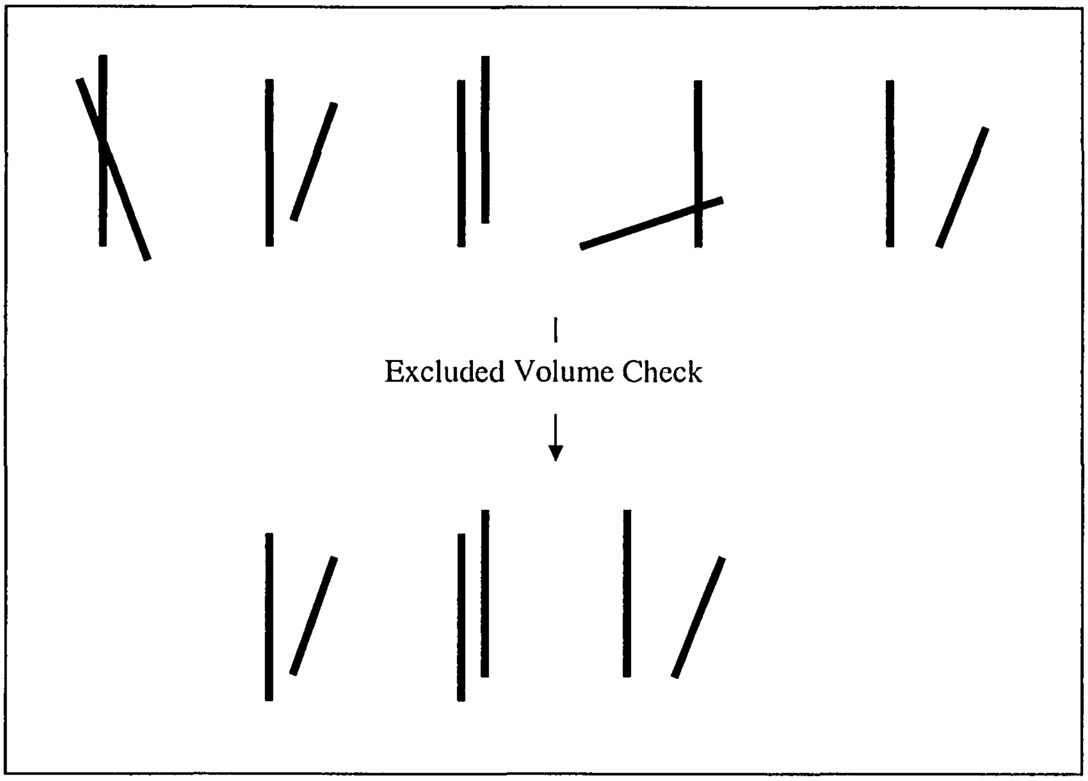


Figure 2.9 *Excluded Volume check removing non-viable structures.*

The remaining structures are then checked to see whether they satisfy a radius of gyration constraint. Atomic coordinates are then mapped (see below) to the cylinders, blocks and strings from existing templates using superposition methods to generate an "atomic level" 3D structure of the target protein (Figure 2.10).

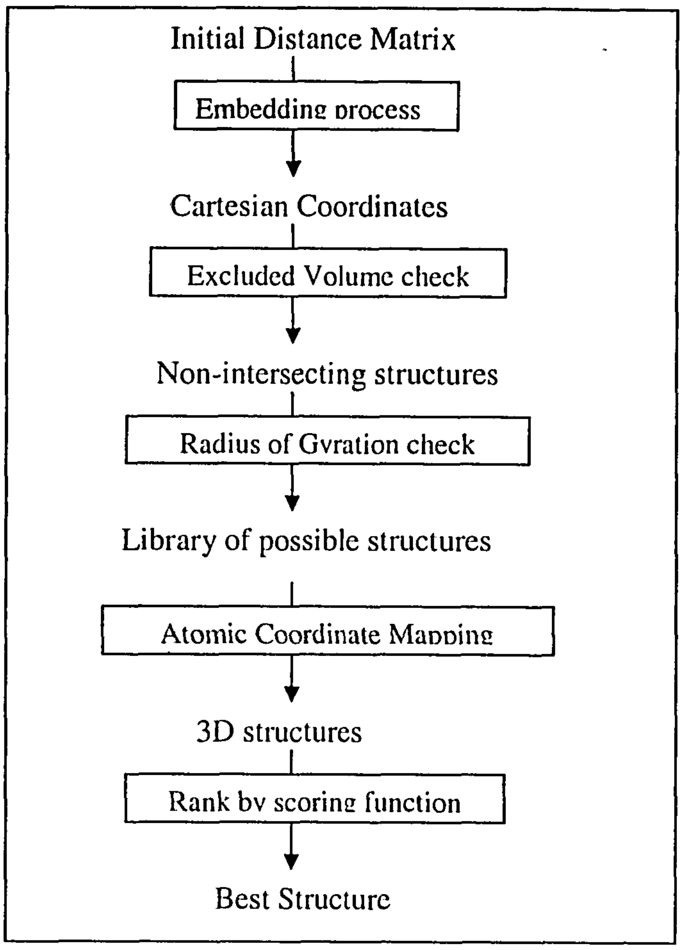


Figure 2.10 *Flow Chart of the structure generation algorithm*

2.3.2 Mapping of atomic coordinates

Once the start and end coordinates of the blocks have been deduced, coordinates taken from well-resolved helical, sheet and loop templates are mapped onto the blocks by standard superposition methods (see Chapter 3 for more details). Specifically the templates consist of a single long helix, various combinations of parallel and anti-parallel beta sheets and a library of loops. The helical template is a 27 residue helical fragment which was cut from the structure of the rop protein (PDB id. 1ROP). Various parallel and anti-parallel beta sheets templates were produced by arranging pairs and triplets of individual beta strands (Figure 2.11). The loop library was built by extracting the loop portions of well resolved proteins from a 1999 version of the PDB database (Berman et al., 2000) (Figure 2.12).

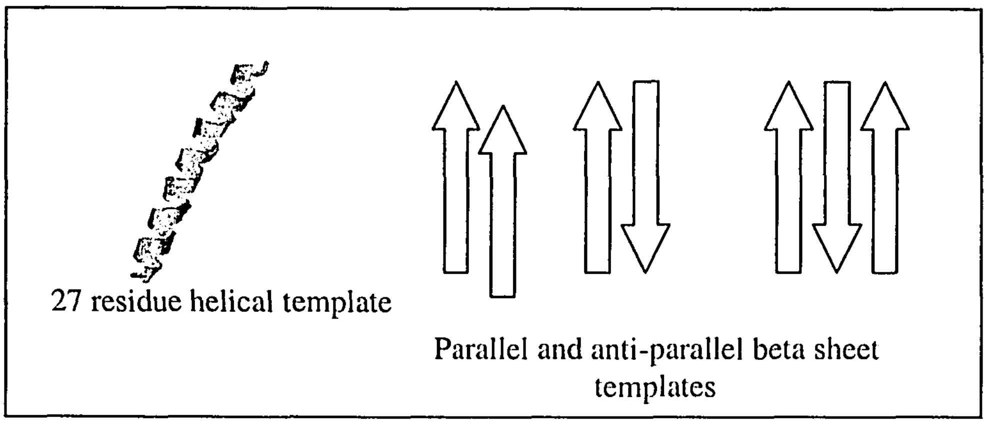

27 residue helical template
Parallel and anti-parallel beta sheet templates

Figure 2.11 *Helical and beta sheet templates.*

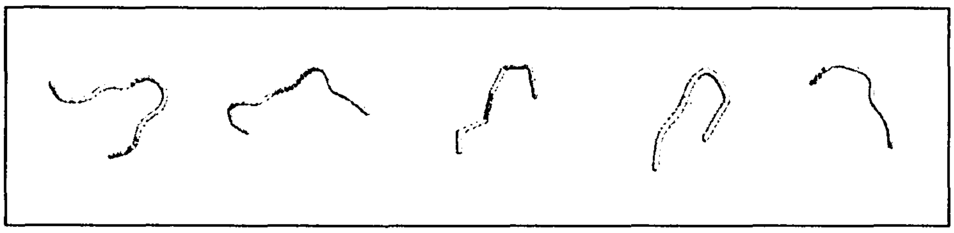


Figure 2.12 *Examples of various loops in the loop library.*

Coordinates for the helices and beta sheets are mapped first and these are then connected by inserting loops in between them (Figure 2.13).

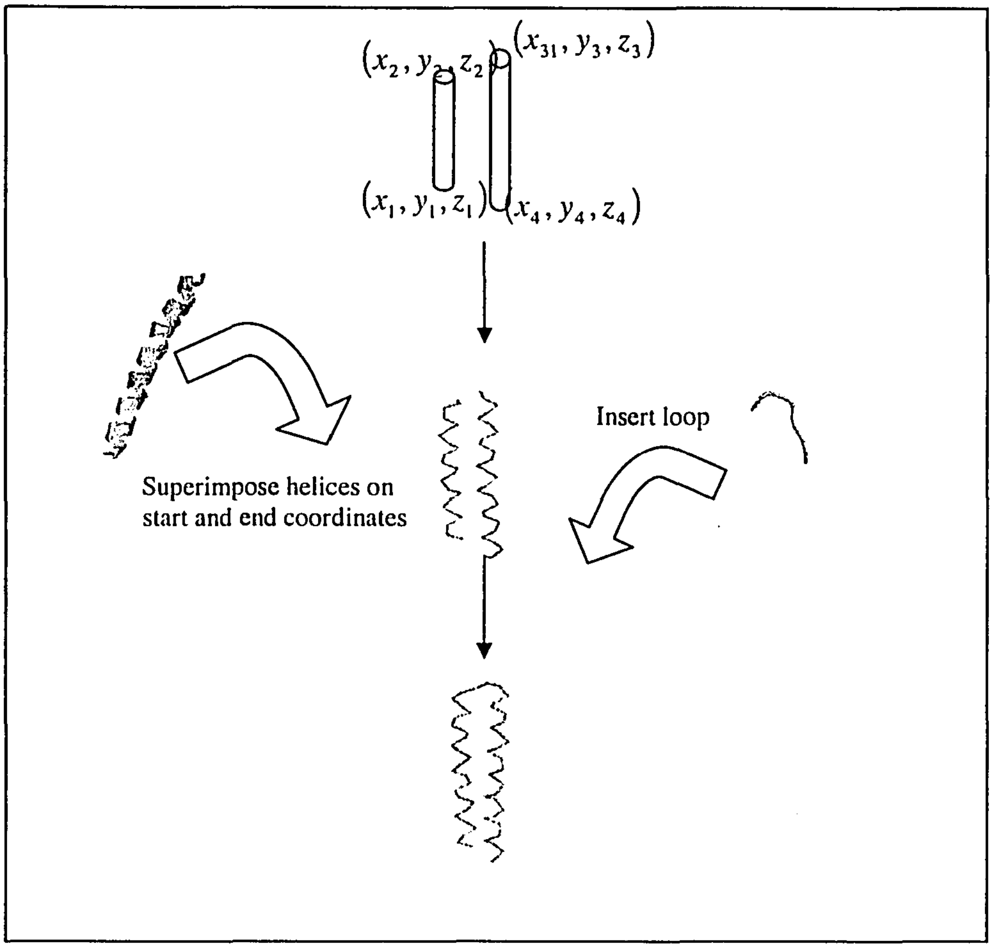


Figure 2.13 *Coordinate mapping of secondary structure fragments.*

The loop insertion component is generally the most difficult or time consuming part of this molecular reassembly process. In the past, various methods have been employed to determine the optimal conformations of loop regions. Go and Sheraga developed an algorithm that could calculate loop geometries based on end-to-end distance of the loops (Go and Sheraga, 1970). Methods based on random search algorithms have also been devised for modeling protein loops (Shenkin et al., 1987). Such algorithms randomly change the backbone torsion angles to enable the loop to satisfy a set of distance constraints. Here we employed a similar method in calculating the correct loop conformation to connect a pair of secondary structure elements. Each loop in the loop library consists of the loop residues and two additional residues: one each from the preceding and following secondary structures to which the loop was connected. These preceding and following residues serve as anchor residues when a loop is inserted between two secondary structure elements (Figure 2.14).

HHHHHHH HHHHHHHHH

2 helical segments that has to be connected by a loop. The last residue of the first helix and the first residue of the last helix serve as anchor residues and are highlighted in blue.

ACCCCB

A 4 residue loop fragment that will connect the two helices. The anchor residues are highlighted in red.

ACCCCB

HHHHHHH HHHHHHHHH

Superimpose the red residues over the blue ones.

HHHHHHHCCCCHHHHHHHHH

Remove the red residues after superposition to get the connected structure.

Figure 2.14 *The process of loop insertion.*

After a loop has been inserted, a check is done to ensure whether the peptide bond has been formed between the loop residues and the helical or beta sheet residues. If peptide bonds are not formed, then the backbone torsion angles of the loop are tweaked via a standard Simplex minimization algorithm (Nelder and Mead,

1965) to allow the loop to take a different conformation and the bond lengths are checked again. If peptide bonds are still not formed, then the next loop from the loop library is selected and the process is repeated (Figure 2.15).

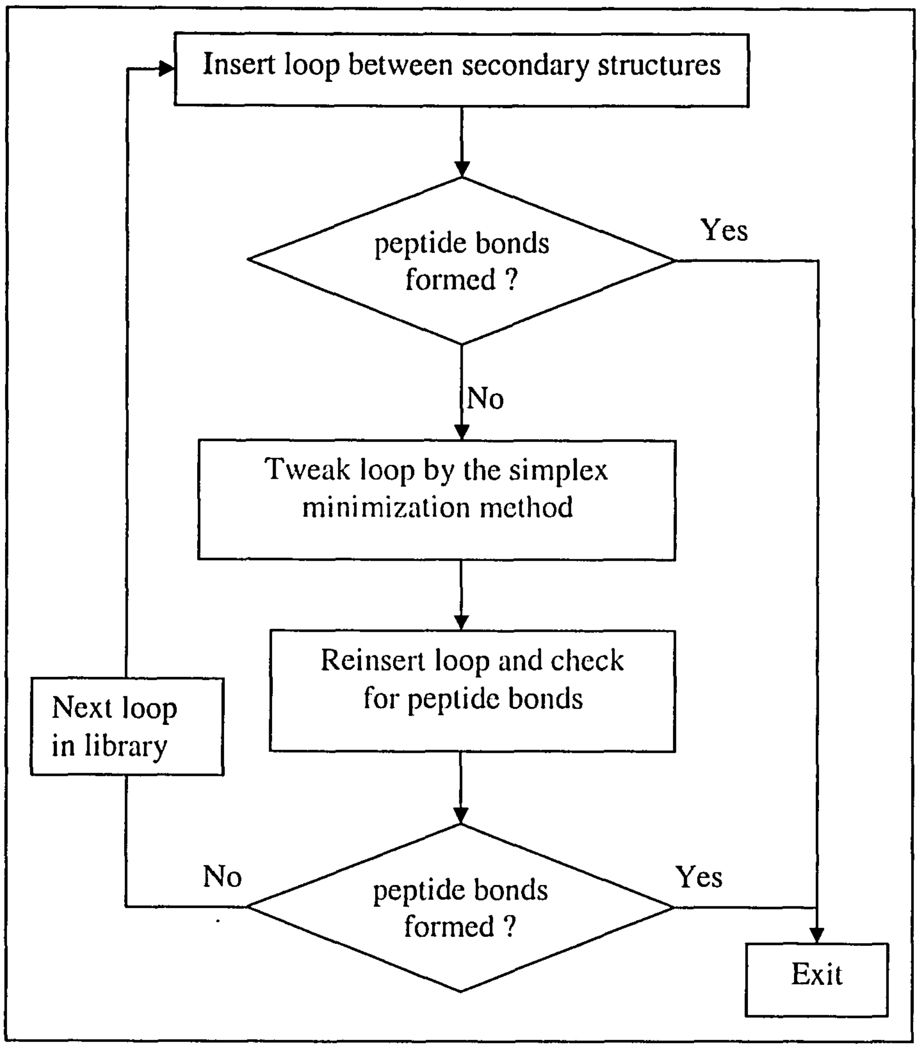

Insert loop between secondary structures
peptide bonds formed ?
Yes
No
Tweak loop by the simplex minimization method
Reinsert loop and check for peptide bonds
Next loop in library
No
peptide bonds formed ?
Yes
Exit

Figure 2.15 *The process of selecting the best fitting loop from the loop library.*

When the coordinates of all the secondary structure elements have been mapped and a complete 3D structure has been generated, the structure is superposed on the native structure and the backbone RMSD is noted. This is done for all the generated structures.

2.4 Results and Discussion

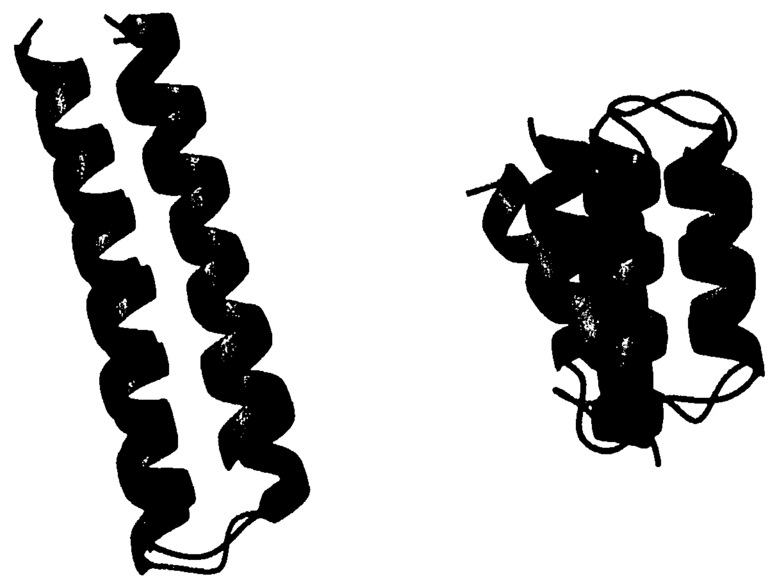
By reducing the most time-consuming steps of the protein folding problem (i.e. the conformational search) to a rapidly solvable distance geometry calculation, it is possible to greatly accelerate the search speed while at the same time reducing the "effective" search space. As seen with the results presented in Table 2.4, a search for 10,000 conformations approximately takes only 17 minutes on a 2.0 GHz Pentium IV processor with 512 MB RAM, as opposed to procedures that rely on clusters of computers (greater than 50 hours) (Feldman and Hogue, 2000).

Table 2.4 *Total number of conformations searched and the time taken for the test set on a 2.0 GHz Pentium IV processor.*

PDB ID	Structure Class	Search space (No. of Conformations)	No. of viable conformations found	Time Taken (minutes)
1ROP	2α	10,000	2,218	11.6
2SPZ	3α	10,000	994	17.2
1BW5	3α	10,000	229	16.9
1VII	3α	5,000	1,653	5.4
1ENH	3α	10,000	253	16.9
1BDC	3α	10,000	1526	17.7
1AHO	1α,2β	10,000	440	18.9
256B	4α	50,000	0	428mins = 7.1 hrs

α stands for alpha helix and β for beta strands.

Under the current implementation of the algorithm, the largest search space that was attempted was 50,000 conformations. With greater computing power, this number can be extended into the millions. On an average, searching for 10,000 conformations and generation of non-overlapping structures and their radii of gyration calculations takes up to 17 minutes on a 2.0 GHz Pentium IV processor with 512 MB RAM. This time is highly dependent (~N^2) on the number of secondary structure elements in the protein. Since the atomic mapping from the templates is the most time consuming process, this step is reserved for structures that pass the topology check and the radius of gyration check. On an average, atomic mapping for each structure takes ~ 15 seconds. The following figures show the native and predicted 3D structures of several small proteins along with the backbone RMSD difference between them. The native structure is always colored blue while the predicted structure is colored red.



1ROP
BB rmsd = 2.32Å

2SPZ
BB rmsd = 3.81Å

Figure 2.16 *Comparison of predicted and generated structures of 1ROP and 2SPZ (native = blue and predicted = red).*

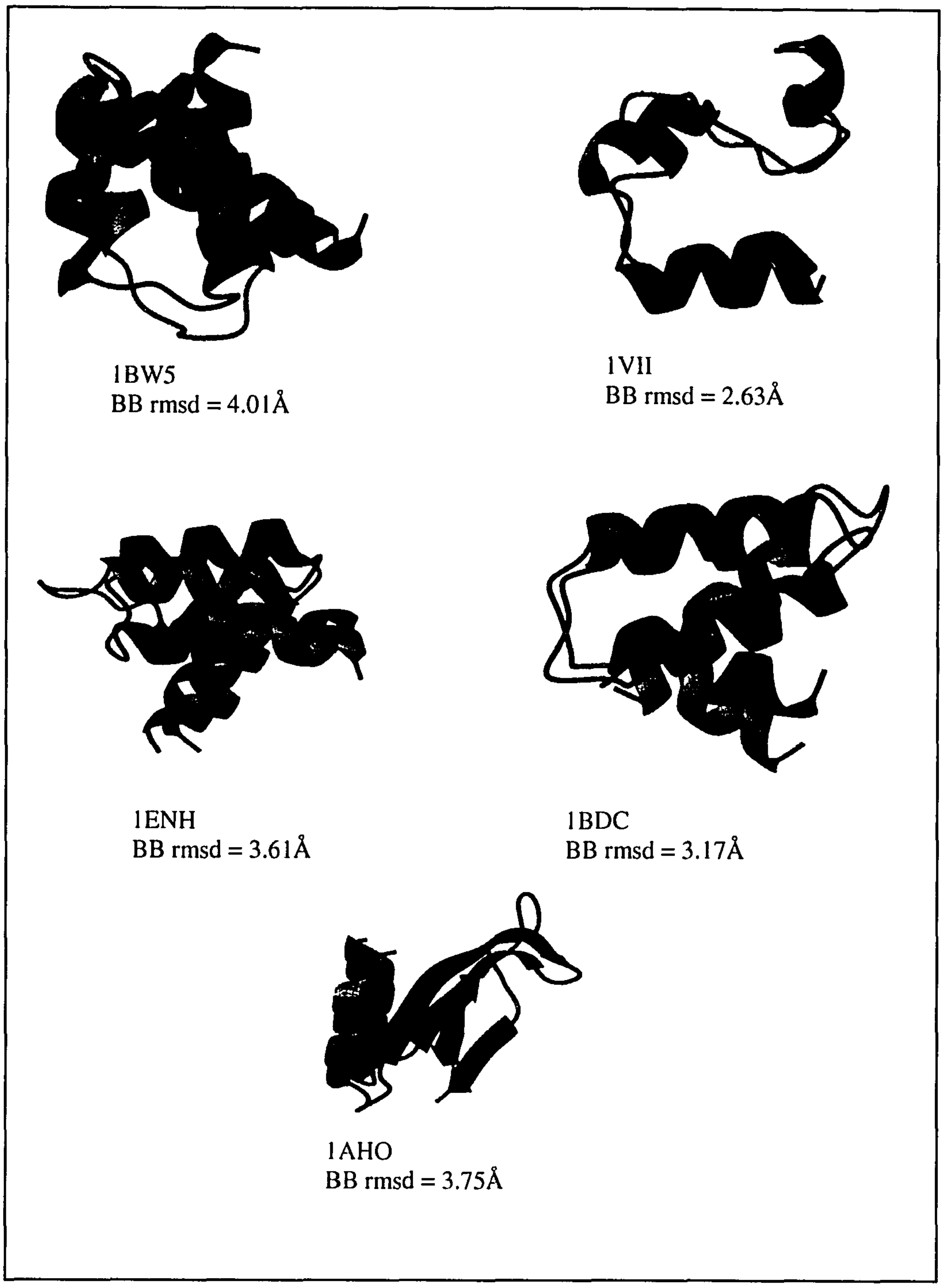


Figure 2.17 *Comparison of predicted and generated structures of 1BW5, 1VII, 1ENH, 1BDC, and 1AHO (native = blue and predicted = red).*

Table 2.5 and 2.6 show the time taken and the RMSDs for the test set of proteins. We also compared the performance of the DG algorithm against an ab-initio method called FOLDTRAJ (Feldman and Hogue, 2000). FOLDTRAJ runs on a cluster of 216 Pentium III 450 MHz CPUs and can generate backbone conformations of small proteins. The alpha carbon RMSDs of the structures generated by FOLDTRAJ compared to their actual native states along with the time taken are shown in Table 2.5.

Table 2.5 *RMSD and time taken for some small proteins generated by FOLDTRAJ.*

FOLDTRAJ			Distance Geometry		
PDB ID	RMSD	CPU hrs	RMSD	CPU hrs	
1VII	3.95Å	70*216 = 15120	2.63Å	12	
1ENH	5.12Å	123*216 = 26568	3.61Å	2	

In contrast to these results, our DG algorithm could generate the approximate structures of 2 and 3 helical bundle proteins in as short as 2 CPU hours (CPU time is the time taken by the processor when it is actively doing the calculations) on a single Pentium 2.0 GHz machine. The backbone RMSDs and the total time taken (sampling + coordinate mapping + evaluating each structure with the scoring potential) for our test set of proteins are shown in Table 2.6.

Table 2.6 *RMSD and time taken for the test set of proteins generated by our DG algorithm.*

PDB ID	RMSD	CPU hrs
1ROP (55 residues)	2.32Å	16
2SPZ (49 residues)	3.81Å	7
1BW5 (46 residues)	4.01Å	2
1VII (29 residues)	2.63Å	12
1ENH(45 residues)	3.61Å	2
1BDC(44 residues)	3.17Å	12
1AHO(31 residues)	3.75Å	7

As seen from the above table, 1ROP, 1VII and 1BDC take the longest time. This is because these two proteins had the largest ensembles of structures (2218, 1653 and 1526 respectively). The most time consuming steps are the coordinate mapping and structure evaluating by the scoring potential. Since sampling is the faster process, increasing the search space manifold times will not slow down the process. Limiting the number of structures that are being sent to the coordinate mapping process and hence get evaluated will definitely speed up the process. This can be done by applying a stricter radius of gyration check or by clustering the generated structures and choosing the most representative ones from the ensemble.

Ideally, the near native structure (structure with the best RMSD to the native fold) should be picked up by a scoring function from the ensemble of structures. This brings us to the second part of the protein folding problem: "*identifying a scoring function to select the closest near native structure from the ensemble of generated structures*". Although proteins have large absolute energies, the difference in energies between any two conformations is as small as a few kcal/mol. Thus, a good scoring function has to be sensitive enough to discriminate these small differences in large quantities. To account for this, heuristic scoring functions must incorporate terms which are believed to assist in the folding process. These include 1) the hydrophobic effect, 2) formation of hydrogen bonds, 3) avoidance of steric clashes, 4) compact radius of gyration and 5) dihedral angles lying in the favorable Ramachandran space.

To date, the majority of scoring functions can be grouped into two categories: 1) empirical or database-derived potentials and 2) true energy or molecular force field potentials (Moult, 1997). Empirical potentials or 'contact potentials' are derived statistically from observing known structures in the protein database, while true energy-based potentials employ aforementioned force fields like AMBER and CHARMM. Molecular force field potentials suffer from slow computing and inclusion of solvent terms. This compounds the complexity and further increases the total computing time. To avoid the long computing times taken by molecular force field potentials we decided to use an empirical potential. This potential is called the Bryant-Lawrence Threading potential (Bryant and Lawrence, 1992). This is a residue contact potential which was derived by statistical analysis of protein crystal structures and gives mean hydrophobic and pairwise contact energies as a function of residue types and distance intervals. Our scoring function includes this potential as well as parameters which check for steric clashes, favorable dihedral angle distributions and compact radius of gyrations. Scores are assigned for each of these parameters. The score assigned for steric clashes is done by a bump-checking function. Inter atom to atom distances between the alpha carbons, beta carbons, alpha carbon and carbonyl carbon, nitrogen and oxygen atoms are checked and if any of these bond lengths are less than their ideal values, the score is incremented by a default penalty value (5 units). If these inter atom lengths have an improbable value (too short) then a penalty value of 50 is assigned. A small bump score signifies that the structure is relatively free of atomic collisions while a large score (in the thousands) points to atomic collisions in the structure. The third parameter is a score which is related to how well the backbone torsion angles (phi and psi) of the residues lie in the Ramachandran space. A correctly folded protein will have its back bone torsion angles lying in the favorable Ramachandran space. The backbone torsion angles are checked and proper phi and psi angles which conform to the peptide geometry are not penalized while any misbehaving angles increase the score. A radius of gyration score is assigned by first calculating the centre of mass of the protein. This is done by averaging the x, y, and z coordinates of all the alpha carbons. Distances from this centre of mass to each alpha carbon is noted and an average distance is calculated. This distance acts as a pseudo radius of gyration. The square of the difference between the real radius of gyration and this pseudo radius is then assigned as the score. Hence, the smaller the score, the closer is the pseudo radius of gyration to the real radius of gyration. Presently, we are trying to create a scoring function that can select near native structures from our ensemble, and this ensemble can serve as a good decoy set for the scoring function. The scores obtained by the different parameters for the closest top ten near native structures for our test set of proteins are shown in Tables 2.7.

Table 2.7.1 *Scores assigned by the different parameters for the top ten structures of 1ROP out of an ensemble of 2218 structures.*

	Scores assigned by				
	Bryant - Lawrence Potential	Steric clash	Ramachandran space	Radius of Gyration	Back Bone rmsd to native
Native scores	-59.7	0	1.05	8.27	0
Structure No.					
256.pdb	-42.65	241.45	5.07	8.5	2.32
82.pdb	-38.44	421.47	5.07	7.03	2.43
182.pdb	-41.21	346.53	5.07	7.97	2.46
189.pdb	-41.87	342.7	5.07	8.05	2.51
397.pdb	-38.77	230.79	5.07	9.7	2.55
395.pdb	-42.71	170.81	10.07	9.85	2.56
169.pdb	-41.61	189.14	10.07	7.67	2.58
512.pdb	-41.92	179.29	5.07	8.49	2.61
785.pdb	-42.3	17.06	16.12	9.64	2.68
116.pdb	-43.73	247.91	10.07	7.43	2.69

Table 2.7.2 *Scores assigned by the different parameters for the top ten structures of 2SPZ out of an ensemble of 994 structures.*

	Scores assigned by				
	Bryant - Lawrence Potential	Steric clash	Ramachandran space	Radius of Gyration	Back Bone rmsd to native
Native scores	-22.85	0	8.14	0.14	0
Structure No.					
98.pdb	33.43	71.21	24.24	0.38	3.81
97.pdb	18.97	1098.82	29.22	0.17	4.03
283.pdb	11.1	295.16	37.26	0.83	4.18
165.pdb	18.64	198.31	32.24	0.76	4.34
8.pdb	8.72	123.43	31.18	0.19	4.37
504.pdb	1.21	185.23	26.24	1.78	4.4
347.pdb	13.62	301.71	23.22	0.54	4.42
164.pdb	29.69	436.34	32.22	0.92	4.48
185.pdb	9.62	10.33	20.22	1.65	4.56
167.pdb	1.83	201.46	17.22	0.38	4.57

Table 2.7.3 *Scores assigned by the different parameters for the top ten structures of 1BW5 out of an ensemble of 229 structures.*

		Scores assigned by			
	Bryant - Lawrence Potential	Steric clash	Ramachandran space	Radius of Gyration	Back Bone rmsd to native
Native scores	-32.64	0.1	6.19	1.24	0
Structure No.					
27.pdb	-56.27	427.8	16.17	0.46	4.01
17.pdb	-53.79	502.44	31.26	0.01	4.07
65.pdb	-28.01	370.58	27.23	0.78	4.37
82.pdb	-7.92	240.75	27.28	0.53	4.51
62.pdb	-49.07	238.74	26.21	0.97	4.56
98.pdb	-14.97	375.09	33.23	0.73	4.58
124.pdb	-12.67	70.56	40.28	1.26	4.64
140.pdb	-18.19	655.93	27.28	2.86	4.68
119.pdb	-24.18	250.6	34.28	1.16	4.72
91.pdb	-27.75	69.27	25.23	1.77	4.74

Table 2.7.4 *Scores assigned by the different parameters for the top ten structures of 1VII out of an ensemble of 1653 structures.*

			Scores assigned by		
	Bryant - Lawrence Potential	Steric clash	Ramachandran space	Radius of Gyration	Back Bone rmsd to native
Native scores	-13.05	0.24	12.41	1.19	0
Structure No.					
1039.pdb	-11.96	72.83	26.34	2.55	2.63
1268.pdb	1.69	8.01	22.37	1.63	2.81
1367.pdb	-6.15	482.7	19.37	5.12	2.89
1377.pdb	2.78	72.87	35.37	1.14	3.03
985.pdb	-6.52	299.5	8.27	2.35	3.03
1501.pdb	8.63	13.73	30.45	2.92	3.08
1378.pdb	-6.13	255.32	14.31	2.81	3.09
602.pdb	5.27	6.95	18.34	0.46	3.12
252.pdb	11.8	468	15.24	0.49	3.22
805.pdb	-9.05	258.93	21.34	0.54	3.25

Table 2.7.5 *Scores assigned by the different parameters for the top ten structures of 1ENH out of an ensemble of 253 structures.*

			Scores assigned by		
	Bryant - Lawrence Potential	Steric clash	Ramachandran space	Radius of Gyration	Back Bone rmsd to native
Native scores	-21.49	0	0.06	0.38	0
Structure No.					
13.pdb	-2.52	473.75	27.24	0.24	3.61
26.pdb	0.62	360.54	34.22	0.1	3.69
133.pdb	-1.97	81.38	29.22	1.8	3.84
75.pdb	8.06	125.49	25.24	0.7	4.49
9.pdb	-7.94	477.62	16.22	0.03	4.54
94.pdb	15.09	145.78	40.24	1.93	4.6
86.pdb	11.4	268.89	27.22	1.46	4.73
59.pdb	1.06	128.74	28.2	1.15	4.85
54.pdb	19.35	79.59	33.26	2.14	4.93
14.pdb	27.49	1110.13	46.28	0.03	5.03

Table 2.7.6 *Scores assigned by the different parameters for the top ten structures of 1BDC out of an ensemble of 1526 structures.*

			Scores assigned by		
	Bryant - Lawrence Potential	Steric clash	Ramachandran space	Radius of Gyration	Back Bone rmsd to native
Native scores	-43.3	0.58	7.22	0.28	0
Structure No.					
222.pdb	3.09	78.29	32.29	0.26	3.17
48.pdb	-8.58	320.97	17.25	0.89	3.23
1049.pdb	-5.69	8.09	26.27	0.9	3.31
8.pdb	-20.02	335.59	33.31	0.01	3.39
380.pdb	-15.28	1.39	26.29	2.51	3.48
332.pdb	-12.19	8.73	30.31	2.09	3.64
466.pdb	-9.11	12.4	50.27	0.86	3.68
126.pdb	-6.62	10.36	21.27	0.84	3.73
297.pdb	-0.15	15.31	15.2	1.67	3.91
331.pdb	16.84	195.33	25.31	2.98	3.94

Table 2.7.7 *Scores assigned by the different parameters for the top ten structures of 1AHO out of an ensemble of 440 structures.*

			Scores assigned by		
	Bryant - Lawrence Potential	Steric clash	Ramachandran space	Radius of Gyration	Back Bone rmsd to native
Native scores	-42.49	0	2.22	0.51	0
Structure No.					
15.pdb	-17.64	75.32	37.38	1.19	3.75
56.pdb	-19.02	0.53	29.38	2.54	3.94
30.pdb	-12.42	5.45	29.38	3.67	3.95
335.pdb	-4.52	5.02	37.45	4.53	3.97
178.pdb	-12.02	3.47	28.35	1.85	4.01
46.pdb	-16.69	4.14	30.38	3.06	4.08
397.pdb	-7.22	2.24	28.35	3.07	4.14
248.pdb	-10.78	2.74	32.38	5.09	4.18
194.pdb	-6.36	1.94	28.35	2.27	4.19
425.pdb	-0.97	61.71	38.41	4.34	4.21

2.5 Conclusion

In summary, we have described a fast conformational search algorithm that can predict the 3D structures of small proteins that are within 4Å RMSD of the native structure. Rather than requiring expensive clusters of processors or hundreds to thousands of CPU hours, our method allows high quality structures to be generated on a single CPU within a few hours. Obviously if cluster computing could be used, the time taken would be reduced linearly with the number of CPUs. This method can be applied to *ab initio* structure prediction, if the secondary structure information can be accurately predicted. Further improvements can clearly be made to our program, both in the structure screening process (i.e. clustering similar structures) and in the form of a more accurate scoring function to select the best structure from the ensemble.

Selecting a single native-like fold from the pool of plausible, low-energy folds is an important hurdle that still remains to be cleared. Although a solution to the much researched protein folding problem is not offered here, we hope our program can serve as a basis for further improvement in this field.

2.6 References

Anfinsen, C. B. (1973) "Principles that govern the folding of protein chains", Science. **181**: 223-230.

Aszodi, A., Gradwell, M.J. and Taylor, W.R. (1995) "Global fold determination from a small number of distance restraints", *J. Mol. Biol.*, **251**: 308-326.

Baker, D. and Sali, A. (2001) "Protein Structure prediction and structural genomics" *Science*, **294:** 93-96.

Berman, H.M., Westbrook, J. Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N. and Bourne, P.E. (2000) "The Protein Data Bank", *Nucleic Acids Research*, **28**: 235-242.

Blundell, T.L., Sibanda, B.L., Sternberg, M.J.E., and Thornton, J.M. (1987) "Knowledge-based prediction of protein structures and the design of novel molecules, *Nature*, **326**: 347-352.

Brooks, B.R., Bruccoleri, R.E., Olafson, B.D., States, D.J., Swaminathan, S. and Karplus, M. (1983) "CHARMM: A program for macromolecular energy, minimization, and dynamics calculations", *J. Comp. Chem.* **4**: 187-217.

Bruccoleri, R.E. and Karplus, M. (1987) "Prediction of the folding of short polypeptide segments by uniform conformational sampling", *Biopolymers*, **26**: 137-168.

Bryant, S.H. and Lawrence, C.E. (1993) "An empirical energy function for threading protein sequence through the folding motif", *Proteins*, **16**: 92-112.

Cohen, F.E., Richmond, J.T., and Richards,F.M.J., (1979) "Protein Folding: Evaluation of some simple rules for the assembly of helices into tertiary structure with myoglobin as an example", *J. Mol. Biol.* **132**: 275-288.

Cui, Y., Chen, R.S., and Wong, W.H. (1998) "Protein folding simulation with genetic algorithm and super secondary structure constraints", *Proteins*, **31**: 247-257.

Dill, K.A. and Chan, H.S. (1997) "From Levinthal pathways to funnels", *Nature Struct. Biol.* **4**: 10-19.

Duan, Y. and Kollman, P.A. (1998) "Pathways to a protein folding intermediate observed in a 1-microsecond simulation in aqueous solution", *Science*, **282**: 740-744.

Eaton, W.A., Munoz, V., Thompson, P. A., Henry, E. R. and Hofrichter, J. (1998) "Kinetics and dynamics of loops, α-helices, β-hairpins, and fast-folding proteins", *Acc. Chem. Res.* **31**: 745-753.

Feldman, H. J. and Hogue, C. W. (2000) "A fast method to sample real protein conformational space", *Proteins*, **39**: 112-131.

Forrest, S. (1993) "Genetic algorithms: principles of natural selection applied to computation", *Science*, **261**: 872-878.

Go, N. and Scheraga, H.A. (1970) "Ring closure and local conformational deformations of chain molecules", *Macromolecules*, **3**:178-187.

Gunsteren, Van, W.F., and Berendsen, H.J.C. (1987) "Groningen Molecular Simulation (GROMOS) Library Manual", (Biomos). Nijenborgh 4, 9747 AG Groningen, The Netherlands.

Hanggi, G. and Braun, W. (1994) "Pattern recognition and self-correcting distance geometry calculations applied to myohemerythrin", *FEBS Lett.*, **344**: 147-153.

Havel, T.F., Kuntz, I.D. and Crippen, G.M. (1983) "The theory and practice of distance geometry", *Bull. Math Biol.*, **45**: 665-720.

Huang, E.S., Samudrala, R. and Ponder, J.W. (1998) "Distance geometry generates native-like folds for small helical proteins using the consensus distances of predicted protein structures", *Protein Science*, **7**: 1998-2003.

Hung L.H. and Samudrala, R. (2003) "PROTINFO: Secondary and tertiary protein structure prediction", *Nucleic Acids Research*, **31**: 3296-3299.

Huang, X. and Powers, R. (2001) "Validity of using the radius of gyration as a restraint in NMR protein structure determination", *J. Am. Chem. Soc.*, **123**: 3834-3835.

Hunt, N.G., Gregoret, L.M., and Cohen, F.E. (1994) "The origins of protein secondary structure. Effects of packing density and hydrogen bonding studied by a fast conformational search", *J.Mol. Biol.*, **241**: 214-225.

Karplus, M. (1997) "The Levinthal paradox: yesterday and today", *Fold Des.* **2**: S69 - S75.

Kuntz, I.D., Crippen, G.M., and Kollman, P.A. (1979) "Application of distance geometry to protein tertiary structure calculations", *Biopolymers*, **18**: 939-957.

Kuntz, I.D., Thomason, J.F., and Oshiro,C.M. (1989) "Distance Geometry", *Meth. Enzymol.*, **177**: 159-204.

Larson, S.M., Snow, C.D., Shirts, M.R. and Pande, V.S. "Folding@Home and Genome@Home: Using Distributed Computing to Tackle Previously Intractable Problems in Computational Biology", (Grant, R. ed.), Horizon Press, 2002.

Lee, M.R., Tsai, J., Baker, D. and Kollman, P.A. (2001) "Molecular dynamics in the endgame of protein structure prediction", *J. Mol. Biol.* **313**: 417-430.

Levinthal, C. (1968) "Are there pathways for protein folding?", *J. Chem.* Phys. **65:** 44-45.

Matthews, B.W. (1993) "Structural and genetic analysis of protein stability", *Annu. Rev. Biochem.* **62**: 139-160.

Mayor, U., Johnson, C. M., Daggert, V. and Fersht, A. R. (2000) "Protein folding and unfolding in microseconds to nanoseconds by experiment and simulation", *Proc. Natl. Acad. Sci. USA* **97**: 13518-13522.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. and Teller, E. (1953) "Equations of state calculations by fast computing machines", *J. Chem. Phys.*, **21**: 1087-1092.

Moult, J. (1997) "Comparison of database potentials and molecular mechanics force fields", *Curr. Opin. Struct. Biol.*, **7**: 194-199.

Mutter, M. (1985) "The construction of new proteins and enzymes. A prospect for the future", *Angew. Chem. [Engl.]* **24**: 639-653.

Nelder, J.A. and Mead, R. (1965) *Computer Journal*, **7**: 308.

Pauling, L., Corey, R.B. and Bronson, H.R. (1951) "The Structure of Proteins: Two hydrogen bonded helical configurations of the polypeptide chain", *Proceedings of the National Academy of Sciences, USA.* **37**:205-211.

Pedersen, J.T. and Moult, J. (1997) "Protein folding simulations with genetic algorithms and a detailed molecular description", *J.Mol. Biol.* **269**: 240-259.

Pool, R. (1999) "Assembling life's building blocks", *Think Research Magazine*, **4.**

Ramachandran, G.N. and Sasiskharan, V. (1968) *Adv. Protein Chem.*, **23**: 283-437.

Shenkin, P.S., Yarmusch, D.L., Fine, R.M., Wang, H. and Levinthal, C. (1987) "Predicting antibody hypervariable loop conformation. I. Ensembles of random conformations for ringlink structures", *Biopolymers*, **26**:2053-2085.

Simons, K.T., Bonneau, R., Ruczinski, I. and Baker, D. (1999) "Ab initio protein structure prediction of CASPIII targets using ROSETTA", *Proteins*, **Suppl 3**: 171-176.

Skolnick, J., Kolinski, A. and Ortiz, A.R. (1997) "MONSSTER: A method for folding globular proteins with a small number of distance restraints", *J. Mol. Biol.*, **265**: 217-241.

Sternberg, M.J.E., Cohen, F.E., and Taylor, W.R. (1982) "A combinatorial approach to the prediction of the tertiary fold of globular proteins", *Biochem. J.* **10**: 299-301.

Stickle, D.F., Presta, L.G., Dill, K.A., and Rose, G. D. (1992) "Hydrogen bonding in globular proteins", *J. Mol. Biol.* **226**: 1143-1159.

Weiner, S.J., Kollman, P.A., Case, D.A., Singh, U.C., Ghio, C., Alagona, G., Profeta, S., and Weiner, Jr. P. (1984) "A new force field for molecular mechanical simulation of nucleic acids and proteins", *J. Amer. Chem. Soc.*, **106**: 765-784.

Zagrovic, B., Sorin, E.J. and Pande, V. (2001) "Beta-hairpin folding simulations in atomistic detail using an implicit solvent model", *J. Mol. Biol.* **313**: 151-169.

Zhou, Y. and Abagyan, R. (1999) "Efficient stochastic global optimization for protein structure prediction", in Rigidity theory and Applications, Thorpe, M.F. and Duxbury. P.M. (eds.) Plenum Publishing, New York, pp. 345-356.

Chapter 3: SuperPose[*] and Protein Structural Superposition

3.1 Introduction

Structure comparison, like sequence comparison, lies at the heart of structural biology. As structures are much more conserved than sequence, structure comparisons allow us to look much further back into Earth's prehistory to track the origins and evolution of many key enzymes and proteins. Unfortunately, structure comparison is a much more computationally difficult process than sequence comparison. In sequence comparison, character string matching or dynamic programming methods can be used to generate alignments and identify regions of sequence similarity (Needleman and Wunsch, 1970). However, structural comparison requires a completely different scheme as one is comparing or aligning complex 3D shapes. While computers are very adept at handling strings, they are not particularly good at identifying or comparing 3D objects. Indeed humans still outperform even the fastest computer in identifying or comparing modestly dissimilar 3D objects.

A very common method for structural comparisons is called structural superposition. Superposition or superimposition is simply the process of rotating or orienting an object until it can be overlaid on top of a similar object. The simplest route to 3D superposition is to identify a minimum of two sets of three common reference points, one set for the object to be superimposed and another set on the reference object that is being overlaid.

*A portion of this chapter appeared in Rajarshi Maiti, Gary H. Van Domselaar, Haiyan Zhang, and David S. Wishart "SuperPose: a simple server for sophisticated structural superposition" Nucleic Acids Res. 2004 July 1; 32 (Web Server issue): W590-W594.

Once these points are identified, the object to be superimposed can be translated and rotated until the two sets of reference points are almost matching. The difficult problem is identifying which three points are most suitable. This problem increases with proteins as we typically want to superimpose not just three points but hundreds of points (or atoms) at the same time.

Fortunately, mathematical approaches have been developed which allow this superposition to be performed – as long as the reference points are identified and the two objects have the same number of identified points. These approaches include rotation by Euler angles (ϕ, θ, ψ), Lagrangian multipliers, least squares minimization, and quaternion methods (MacLachlan, 1982; Kabasch, 1976; Kearsley, 1989; Hamilton, 1853). Throughout the 1970's and 1980's, a number of stand-alone computer programs were described that employed these methods. More recently, structure superposition programs have also found their way into commercial visualization packages such as those offered by Tripos and Accelrys. Additionally, several freeware modeling programs including DeepView (Kaplan and Littlejohn, 2001) and MolMol (Koradi et al., 1996) are also capable of performing and visualizing certain kinds of molecular superpositions. However, in order to perform most molecular superpositions with either commercial or freeware products, users must become quite familiar with some rather complex interfaces. Furthermore, not all stand-alone packages are compatible with common operating systems or compilers used in many molecular biology or teaching labs.

Unfortunately relatively few commercial or freeware products seem to superimpose two or more molecules in a consistent or reliable way. Beyond the continuing problems of overly complex graphical user interface (GUI) design and operating system incompatibility we have frequently found the reported RMSD (root mean square deviation) values differ substantially between packages or that the RMSD values are incompletely described. Furthermore, packages like DeepView and MolMol are quite restrictive in what can be superimposed (structures must be of identical length), how many molecules can be superimposed (most permit just two molecules to be superimposed), how the molecules are superimposed and how different two structures can be when superimposed.

To overcome these ongoing problems we have gone back to the drawing board and written a robust macromolecular superposition web server – called SuperPose -- that appears to overcome the underlying problems among stand-alone programs regarding GUI complexity, platform incompatibility, RMSD inconsistency, sequence length compatibility and limited superposition capability. SuperPose uses advanced and very rapid/robust methods for comparing or filtering structures and in performing the necessary superpositions. In the following pages I will give a brief description of the different rigid body rotation and superposition methods that have been used in protein structure superposition (along with their shortcomings), followed by a more complete description of the SuperPose program and web server.

3.2 Structural superposition methods

3.2.1 Rigid body rotation

The superposition of any 3 dimensional objects requires an orthogonal transformation comprising of two rotations and translations. The human eye (and brain) is able to do this quite easily and this process of structure comparison and analogizing is vital to our capacity to recognize shapes and patterns. This intuitive visual process is also describable in terms of a mathematical framework that was originally derived by Leonhard Euler in 1765. In fact, the Euler angle approach has been the staple method for performing rigid body rotations for hundreds of years. According to Euler's rotation theorem, any rotation about an axis may be described using three angles—now called the Euler angles. The Euler angles (ϕ,θ,ψ) relate two orthogonal coordinate systems having a common origin. The transition from one coordinate system to the other is achieved by a series of two-dimensional rotations. The rotations are performed about coordinate system axes generated by the previous rotation step. There are several conventions for Euler angles, depending on the axes about which the rotations are performed (Goldstein, 1980; Landau and Lifschitz, 1976; Tuma, 1974). The so-called "x-convention" is the most common definition and in this convention, the rotation is given by the angles (ϕ,θ,ψ), where the first rotation is by an angle ϕ about the z axis, the second is by an angle θ about the transformed x axis (x'), and the third is by an angle ψ about the new z axis (z") (Figure 3.1).

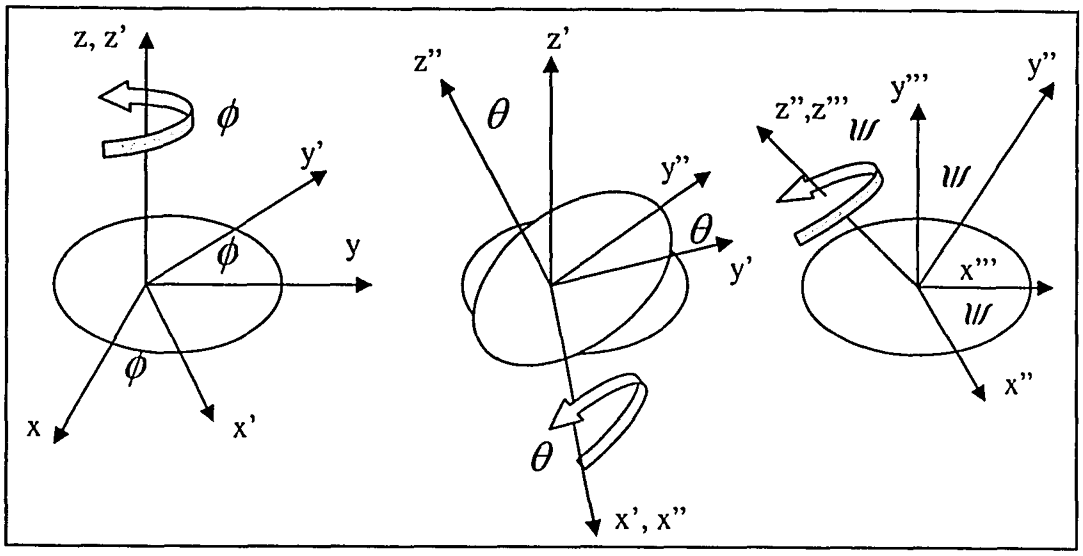


Figure 3.1 *The three Euler angles* (ϕ,θ,ψ) *with rotation about the axes.*

If the rotations are written in terms of rotation matrices **B, C,** and **D,** then a general rotation **A** can be written as **A** = **BCD**, where the rotation matrices are given by

$$D_Z(\phi)=\begin{bmatrix}\cos\phi & \sin\phi & 0\\ -\sin\phi & \cos\phi & 0\\ 0 & 0 & 1\end{bmatrix}, \tag{3.1}$$

$$C_X(\theta)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta & \sin\theta\\ 0 & -\sin\theta & \cos\theta\end{bmatrix}, \tag{3.2}$$

$$\text{and } B_Z(\psi)=\begin{bmatrix}\cos\psi & \sin\psi & 0\\ -\sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{bmatrix}. \tag{3.3}$$

So multiplying the three matrices we get the rotation matrix **A**:

$$\mathbf{A}=\begin{bmatrix}\cos\phi\cos\psi-\sin\phi\cos\theta\sin\psi & \sin\phi\cos\psi+\cos\psi\cos\theta\sin\psi & \sin\theta\sin\psi\\ -\cos\phi\sin\psi-\sin\phi\cos\theta\cos\psi & -\sin\phi\sin\psi+\cos\phi\cos\theta\cos\psi & \sin\theta\cos\psi\\ \sin\phi\sin\theta & -\cos\phi\sin\theta & \cos\theta\end{bmatrix} \tag{3.4}$$

Two different rigid bodies in space can be described by two orthogonal systems of axes and once their centres of mass have been superimposed by translation, then the rotation matrix **A** (equation 3.4) can be used to rotate the two sets of axes and superimpose the two rigid bodies (Figure 3.2).

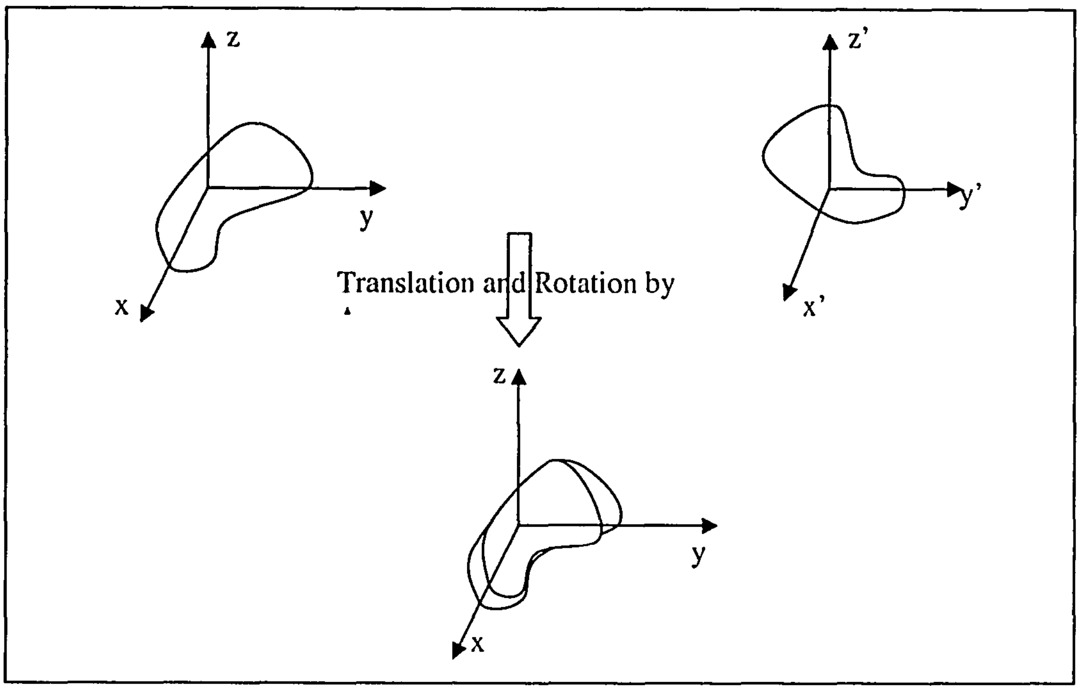


Figure 3.2 *Superposition of two different rigid bodies.*

3.2.2 3 point superposition

In 3 point superposition, Euler angle rotations are used to rotate and superimpose one object (B) over the other (A). If a set of three points (called a *triplet* of points) on A are labeled as **a**, **b**, and **c**, and those on B as **r**, s, and **t**, then B can be superimposed on A by the following steps.

i) Translating point **r** onto **a**: the second triplet now becomes **a**, s', **t**'.

ii) Superimposing the planes formed by **a**, **b**, **c** and **a**, **s'**, **t'** by rotating by θ around the axis given by the cross product of their normal vectors (**n** x **n'**) (Normal vectors are vectors that are perpendicular to a plane and can be computed by the cross product of two vectors lying in that plane). This superimposes the normal **n** to the plane formed by **a**, **b**, **c** and the normal **n'** to the plane formed by **a**, **s'** and **t'**. The triplet **a**, **s'**, **t'** now becomes **a**, **s''**, **t''**.

iii) Superimposing vectors **as''** and **ab** by rotation along **n**.

Figure 3.3 shows the set of superimposed points chosen for two objects. This set of rotations and translations must be performed on all points in B to superimpose B onto A. This technique requires some assumptions as to the choice of the triplet sets for superposition and it privileges the first point **a**(**r**), then **b**(s) over **c**(**t**).

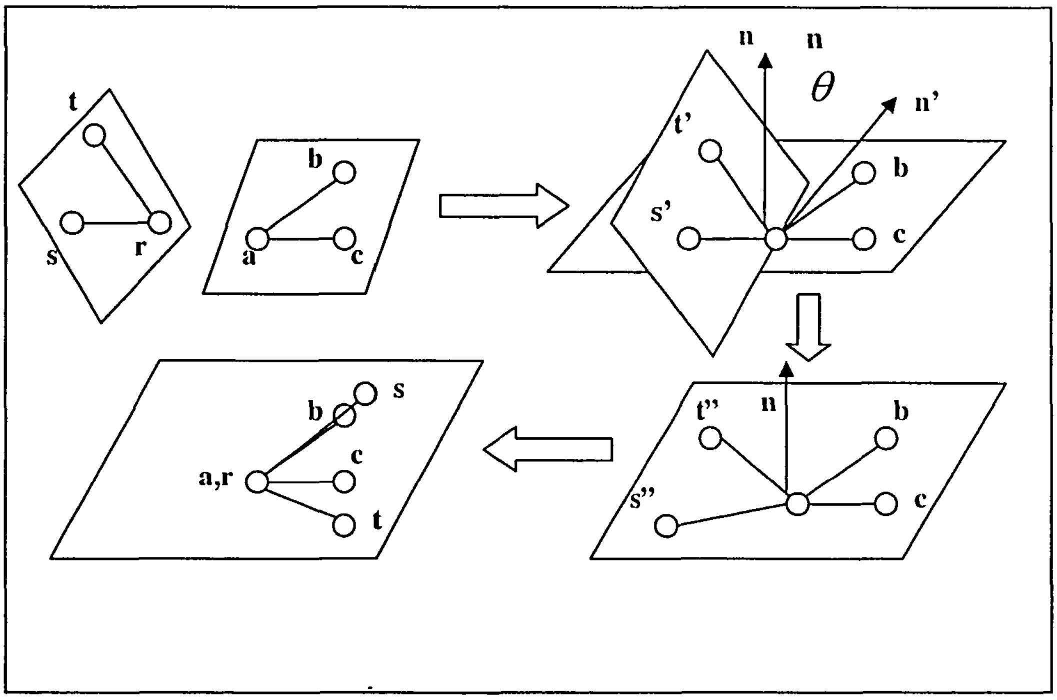


Figure 3.3 *Three point superposition using (a, b, c) and (r, s, t).*

For simple objects containing a few tens of points, the 3 point superposition performs well, provided a judicious choice of the reference points has been made. However, proteins contain hundreds of atoms and superposing them based on a triplet of points often leads to very crude or intuitively incorrect superpositions. For optimal superpositions of proteins, it is necessary to overlay many corresponding atom pairs and this can only be done by finding a set of rigid body translations and rotations that minimize the RMS distance between all corresponding atoms. Methods like least squares fitting, Lagrangian multipliers and quaternion approaches all try to minimize this RMS distance and they are described below.

3.2.3 Least Squares Fitting approach to superposition

Least squares fitting involves rotating and translating one structure (B) over a reference structure (A) until the coordinate differences between the two are minimal. Specifically, one tries to minimize a function:

$$E = \sum_{i,n} (b'_{in} - a_{in})^2 \qquad (n = \textit{Number of atoms}), (i = 1,2,3) \qquad (3.5)$$

where a_n, b_n are the position vectors of the coordinates of A and B. b'_{in} are the position vectors of B when B has been superimposed on A.

Simply stated, least squares superposition involves four steps:

1) Determining the centre of mass of molecule A.
2) Determining the centre of mass of molecule B.
3) Translating molecule B to molecule A's centre of mass position.

4) Rotating B by small amounts using the Euler rotation matrix until E is a minimum.

This rotation process (step 4) could be done randomly using a simple grid search or a slightly more sophisticated Monte Carlo approach (Metropolis and Ulam, 1949). However this is very inefficient and would inevitably take a long time (even with today's fast computers) to calculate the optimal superposition for a 1000 atom protein. To speed up this naïve search approach it is always possible to use more efficient optimization procedures that use derivatives and iterative least squares fitting to find the optimal Euler angles. To see how this works, let us consider a situation where the two molecules have been initially (and incorrectly) superimposed on their centre of masses(prior to rotation). Therefore, the difference between the coordinates of structure A and structure B can be represented as

$$\Delta_{in} = b'_{in} - a_{in} \tag{3.6}$$

If **p** is a column vector of length 3 corresponding to the three Euler rotation angles and **D** is an $n \times 3$ matrix containing the derivatives of b'_{in} with respect to these three Euler angles, then multiplying **p** with **D** will result in a small change ($\delta b'_{in}$) in the b'_{in} coordinates. Thus, $\delta b'_{in} = \mathbf{Dp}$. If $\delta b'_{in}$ is the correct amount of rotation to bring B onto A then $\delta b'_{in}$ will equal Δ_{in}. We assume $\delta b'_{in}$ will be different from Δ_{in} (since the structures were initially superimposed incorrectly) and so there will be an error E with most of the incremental rotations. In light of these calculations and assumptions, E can be written as $E = \Delta_{in} - \delta b'_{in} = \Delta_{in} - \mathbf{Dp}$. (3.7)

If the above equation is multiplied by the transpose of **D**, which is denominated as $\mathbf{D}^{\mathbf{T}}$, then we get

$$\mathbf{D}^{\mathbf{T}}\mathrm{E} = \mathbf{D}^{\mathbf{T}}\,\Delta_{in} - \mathbf{D}^{\mathbf{T}}\,\mathbf{Dp}. \tag{3.8}$$

The least squares condition that $\sum_{i,n}(b'_{in} - a_{in})^2$ become a minimum requires that

$\mathbf{D}^{\mathbf{T}}\,\mathrm{E} = 0$ (i.e. the error between rotations approaches 0). Hence we get

$$\mathbf{D}^{\mathbf{T}}\,\Delta_{in} = \mathbf{D}^{\mathbf{T}}\,\mathbf{Dp} \quad \text{or} \qquad \mathbf{p} = (\mathbf{D}^{\mathbf{T}}\mathbf{D})^{-1}\,\mathbf{D}^{\mathbf{T}}\Delta_{in}. \tag{3.9}$$

Solving for **p**, we can obtain the Euler angles that will optimally superimpose molecule B on A. Because the process is a little imprecise, it is often necessary to iterate this process a few times before the best set of Euler angles are obtained.

3.2.4 Lagrangian Multipliers

While the least squares method uses an iterative process, a direct solution can be obtained by Lagrangian multipliers. This can be achieved by minimizing the RMSD difference between the superposed molecules in the presence of some constraint functions. We will use the same notations as described above, i.e. coordinates of A and B will be denoted by position vectors $a_n, b_n (n = Number\ of\ atoms)$ and the orthogonal transformation that converts the coordinates $b_{in} (i = 1,2,3)$ into b'_{in}, where b'_{in} are the superimposed coordinates that can be related by

$$b'_{in} = \sum_{j}^{3} r_{ij} b_{jn} + t_i \tag{3.10}$$

Here r_{ij} are the elements of the rotation matrix A and t_i is a translation operation. The rotation matrix and a vector of points can be related by

$$\begin{bmatrix} b_1' \\ b_2' \\ b_3' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{3.11}$$

For an optimal superposition, the RMS difference (E) between the superposed coordinates (b_{in}') and the reference coordinates (a_{in}) must be minimized:

$$E = \sum_{i,n} (b_{in}' - a_{in})^2 \tag{3.12}$$

Note how similar this equation is to eq. 3.5. Since rotation is an orthogonal transformation, the rotation matrix should be an orthogonal matrix. As vectors preserve their length, the square of their lengths before and after rotation will be equal. Hence,

$$b_1'^2 + b_2'^2 + b_3'^2 = b_1^2 + b_2^2 + b_3^2 \tag{3.13}$$

Expanding equation 3.11 we get

$$b_1' = r_{11}b_1 + r_{12}b_2 + r_{13}b_3 \text{ and so on for } b_2' \text{ and } b_3'. \tag{3.14}$$

Expanding $b_1'^2 + b_2'^2 + b_3'^2$ we get

$$\begin{aligned} b_1'^2 + b_2'^2 + b_3'^2 = {} & (r_{11}^2 + r_{21}^2 + r_{31}^2)b_1^2 + (r_{12}^2 + r_{22}^2 + r_{32}^2)b_2^2 + (r_{13}^2 + r_{23}^2 + r_{33}^2)b_3^2 + \\ & 2(r_{11}r_{12} + r_{21}r_{22} + r_{31}r_{32})b_1b_2 + \\ & 2(r_{11}r_{13} + r_{21}r_{23} + r_{31}r_{33})b_1b_3 + \\ & 2(r_{12}r_{13} + r_{22}r_{23} + r_{32}r_{33})b_2b_3. \end{aligned} \tag{3.15}$$

Comparing the coefficients of the right hand side of 3.15 with the coefficients of 3.13, we obtain the constraints

$(r_{11}^2 + r_{21}^2 + r_{31}^2) = 1$, $(r_{12}^2 + r_{22}^2 + r_{32}^2) = 1$, and $(r_{13}^2 + r_{23}^2 + r_{33}^2) = 1$ and

$2(r_{11}r_{12} + r_{21}r_{22} + r_{31}r_{32}) = 0$,

$2(r_{11}r_{13} + r_{21}r_{23} + r_{31}r_{33}) = 0$, and

$2(r_{12}r_{13} + r_{22}r_{23} + r_{32}r_{33}) = 0$.

Thus, our problem reduces to minimizing the multivariate function E in the presence of the above constraints. Minimizing multivariate functions in the presence of multivariate constraint equations can be solved by Lagrangian multipliers and this method was used by Kabsch (1976) to obtain the best rotation matrix.

3.2.5 Quaternion method

Recall from section 3.2.1 that the rotation matrix for the three Euler angles is given by:

$$\mathbf{A} = \begin{bmatrix} \cos\phi\cos\psi - \sin\phi\cos\theta\sin\psi & \sin\phi\cos\psi + \cos\psi\cos\theta\sin\psi & \sin\theta\sin\psi \\ -\cos\phi\sin\psi - \sin\phi\cos\theta\cos\psi & -\sin\phi\sin\psi + \cos\phi\cos\theta\cos\psi & \sin\theta\cos\psi \\ \sin\phi\sin\theta & -\cos\phi\sin\theta & \cos\theta \end{bmatrix}.$$

This particular matrix contains six trigonometric terms. While these are not particularly difficult to handle mathematically, the calculation of trigonometric terms by computers is a relatively slow and time-consuming process. This is because the architecture of computers requires that trigonometric functions be expressed as long series of Taylor functions (Taylor, 1715; Arfken, 1985). Obviously if trigonometric expressions could be expressed algebraically, then computers would be able to handle them much more efficiently. Fortunately, the idea of re-formulating trigonometric expressions in an algebraic form was developed by Sir William R. Hamilton in 1843 (Hamilton, 1853) who created the theory of quaternion algebra. Since quaternions involve algebra and no trigonometric terms, first and second derivatives can be easily calculated and quaternion based rotations are much faster as computers work faster with algebraic rather than trigonometric functions. Quaternions are also known as hypercomplex numbers, Rodriques parameters, and somewhat confusingly, Euler parameters. They are also called Euler parameters as the Euler rotation angles can be represented by them. In simple terms, quaternions can be thought of as quadruplets of (sets of 4) real numbers: one real and three imaginary components. Hence, a quaternion q can be represented as

$$q = q_0 + q_1\hat{i} + q_2\hat{j} + q_3\hat{k}. \tag{3.16}$$

where q_0 is the real number and $q_1, q_2, and\ q_3$ are the imaginary components.

In Appendix B, the basic rules of quaternion mathematics is explained. It is also shown how rotations can be performed by unit quaternions. From Appendix B we see that the rotation matrix **A** can be represented via quaternions as

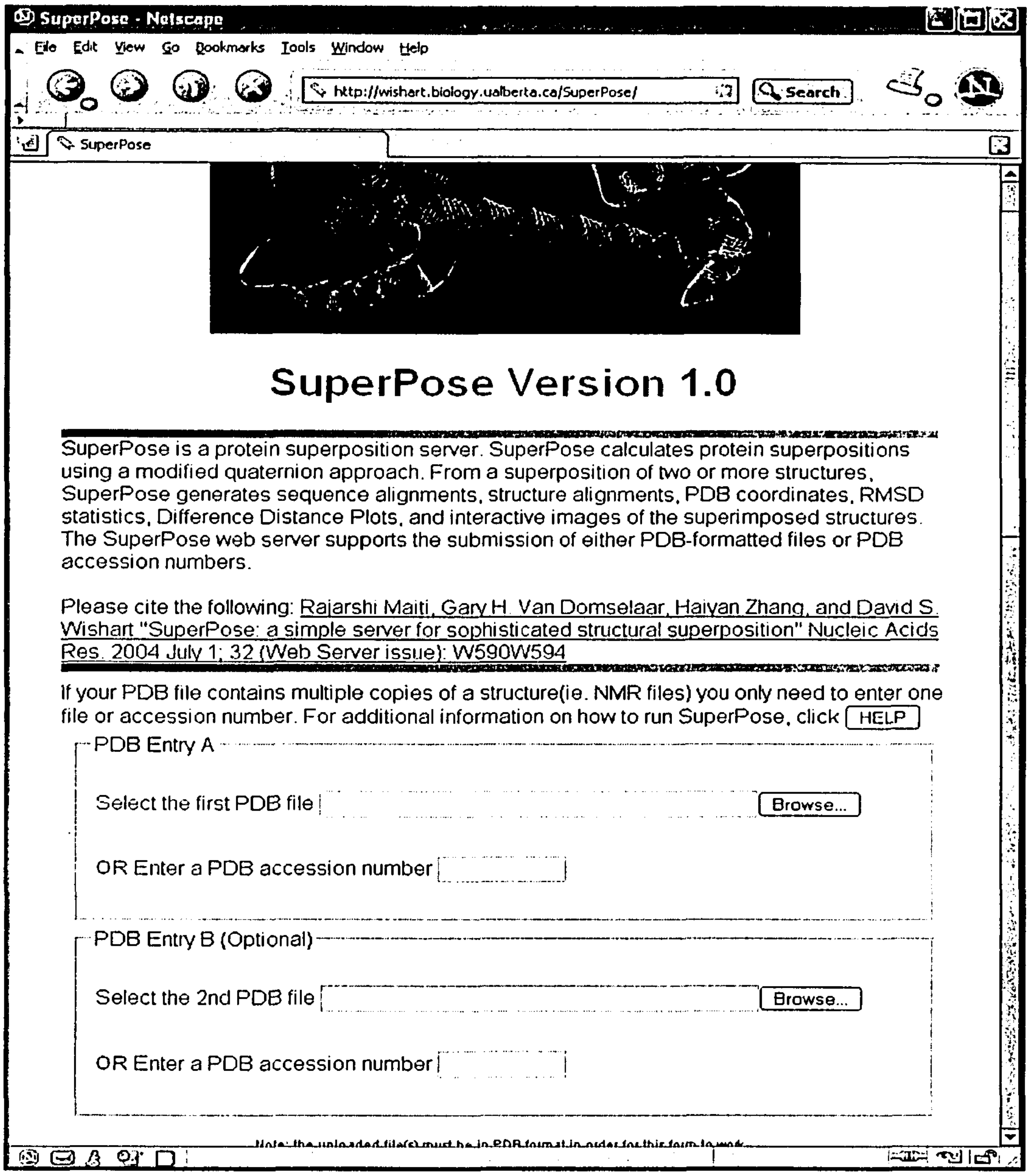
$$\mathbf{A} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{3.17}$$

If rotations are represented by 3x3 orthogonal matrices, then it is necessary to store all 9 terms. Two successive rotations can be performed by multiplication by two rotation matrices. For each product matrix element, we need to do 3 multiplications and 2 additions for a total of 27 multiplications and 18 additions. If quaternions are used to represent rotations, then only 4 elements need to be stored. Composing of rotations corresponds to quaternion multiplication and for each product element we require 4 multiplications and 3 additions for a total of 16 multiplications and 12 additions. Thus, quaternions require fewer computations and greatly speed up the process. The quaternion method for superposition was first implemented in a program called PDBSUP developed by Rupp and Parkin (Rupp and Parkin, 1996).

3.3 System and Methods

3.3.1 Platform, Software and Availability

The SuperPose web server is composed of two parts, a front-end web interface and a back-end for performing the superposition and generating the results. The web interface was written in HTML (HyperText Markup Language), Dynamic HTML and Javascript. The forms for user input were handled using PERL (Wall et al., 1996) and CGI (Common Gateway Interface). While the front end handles the user inputs, the back-end deals with sequence/structural alignments, superposition and RMSD calculations. These portions were written in C and PERL. The complete package was ported to a LINUX web server and can be accessed at http://wishart.biology.ualberta.ca/SuperPose (Figure 3.4).


SuperPose - Netscape
File Edit View Go Bookmarks Tools Window Help
http://wishart.biology.ualberta.ca/SuperPose/
Search
SuperPose

SuperPose Version 1.0

SuperPose is a protein superposition server. SuperPose calculates protein superpositions using a modified quaternion approach. From a superposition of two or more structures, SuperPose generates sequence alignments, structure alignments, PDB coordinates, RMSD statistics, Difference Distance Plots, and interactive images of the superimposed structures. The SuperPose web server supports the submission of either PDB-formatted files or PDB accession numbers.

Please cite the following: Rajarshi Maiti, Gary H. Van Domselaar, Haiyan Zhang, and David S. Wishart "SuperPose: a simple server for sophisticated structural superposition" Nucleic Acids Res. 2004 July 1; 32 (Web Server issue): W590W594

If your PDB file contains multiple copies of a structure(ie. NMR files) you only need to enter one file or accession number. For additional information on how to run SuperPose, click HELP

PDB Entry A

Select the first PDB file [ ] Browse...

OR Enter a PDB accession number [ ]

PDB Entry B (Optional)

Select the 2nd PDB file [ ] Browse...

OR Enter a PDB accession number [ ]

Note: the uploaded file(s) must be in PDB format in order for this form to work.

Figure 3.4 *The SuperPose Homepage located at http://wishart.biology.ualberta.ca/SuperPose*

3.3.2 Program Description

Currently, SuperPose can handle five different kinds of macromolecular superpositions. These include:

i) Superposition of two or more molecules of identical sequences but slightly different structure.

ii) Superposition of two molecules of identical sequence but profoundly different structure.

iii) Superposition of two or more molecules of modestly dissimilar sequence, length and structure.

iv) Superposition of two or more molecules with profoundly different lengths but similar structure or sequence.

v) Superposition of two or more molecules that are profoundly different in sequence but similar in structure.

The most common scenario, and the one supported by most superposition packages is case (i). This type of superposition is frequently done in generating NMR (Nuclear Magnetic Resonance) structure ensembles, in comparing ligand-bound and ligand-free molecules and in comparing two different crystal isoforms. For case (i), sequence and sequence length differences are irrelevant and the problem can be framed as a pure geometrical optimization problem. However, for the other four cases, sequence and length information are relevant – as is the information about local structural similarity. Unfortunately, most available superposition packages do not account for this kind of information and so they frequently perform poorly or require considerable user knowledge or manipulation to get them to perform well. To deal with all five cases, SuperPose combines sequence/structural alignments along with difference distance matrix calculations to identify the common set of points and then uses quaternion based rotation and least squares minimization to overlay these points.

As mentioned earlier, finding the set of points that should be overlaid is one of the main problems in structural superpositions. For proteins, this amounts to finding the common set of points that are structurally similar. Beginning with an input PDB file or a pair of files, SuperPose first extracts the sequences of all chains in the file(s). Each sequence pair is then aligned using a Needleman-Wunsch pairwise alignment algorithm (Needleman and Wunsch, 1970) employing a BLOSUM62 scoring matrix. If the pairwise sequence identity falls below the default threshold (25%), SuperPose determines the secondary structure using VADAR (volume, area, dihedral angle reporter) (Willard et al., 2003) and performs a secondary structure alignment using a modified Needleman-Wunsch algorithm. After the sequence or secondary structure alignment is complete, SuperPose then generates a difference distance (DD) matrix (Richards and Kundrot, 1988) between the aligned alpha carbon atoms. A difference distance matrix can be generated by first calculating the distances between all pairs of alpha carbon atoms in one molecule to generate an initial distance matrix. A second pairwise distance matrix is generated for the second molecule and, for equivalent/aligned alpha carbon atoms, the two matrices are subtracted from one another, yielding the DD matrix. From the DD matrix it is possible to quantitatively assess the structural similarity/dissimilarity between two structures. In fact, the distance difference matrix is particularly good at detecting domain or hinge motions in proteins (case ii). A color coded plot of the elements of the difference distance matrix with progressively brighter colors representing larger distance differences can be used to visually infer the hinge regions of the proteins (Figure 3.5).

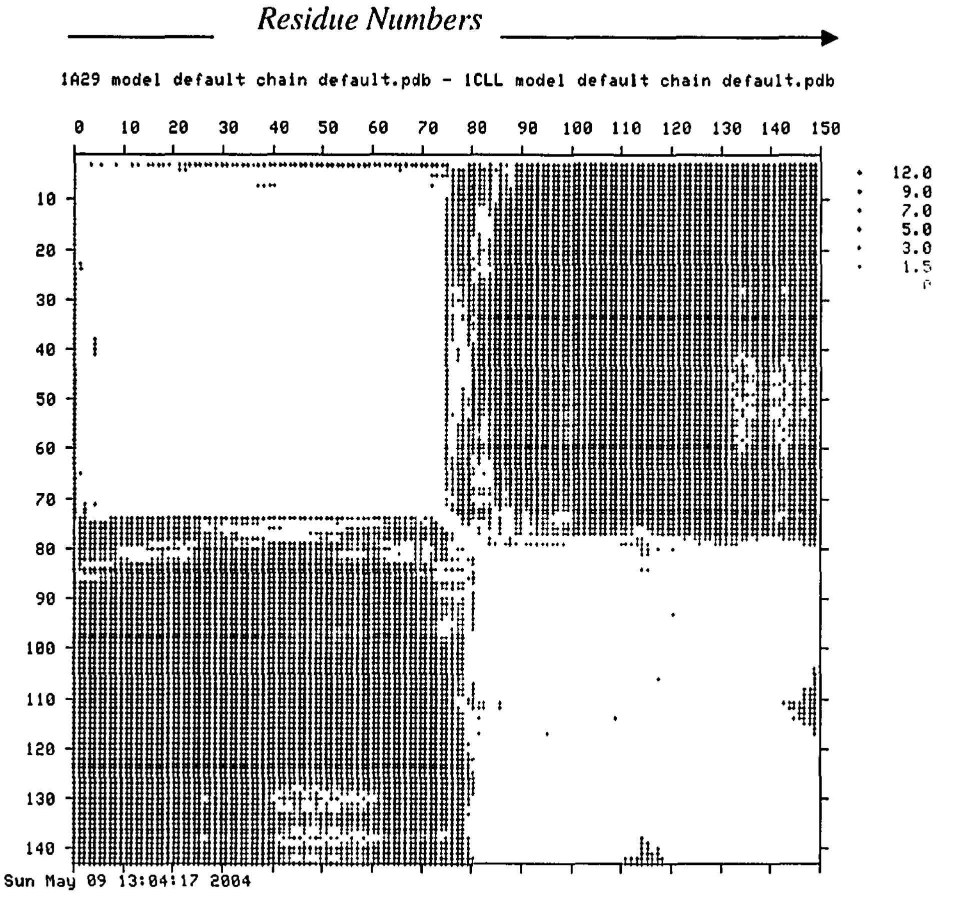


Figure 3.5 *A DD matrix plot between peptide-bound and free calmodulin (1A29 and 1CLL). The lighter/darker colors represent structurally similar/dissimilar regions.*

SuperPose analyzes the DD matrices and identifies the largest contiguous domain between the two molecules that exhibits < 2.0 Å difference. From the information derived from the sequence alignment and DD matrix analysis, the program then makes a decision regarding which regions should be superimposed and which atoms should be counted in calculating the RMSD. This information is then fed into the quaternion superposition algorithm and the RMSD calculation subroutine (Figure 3.6).

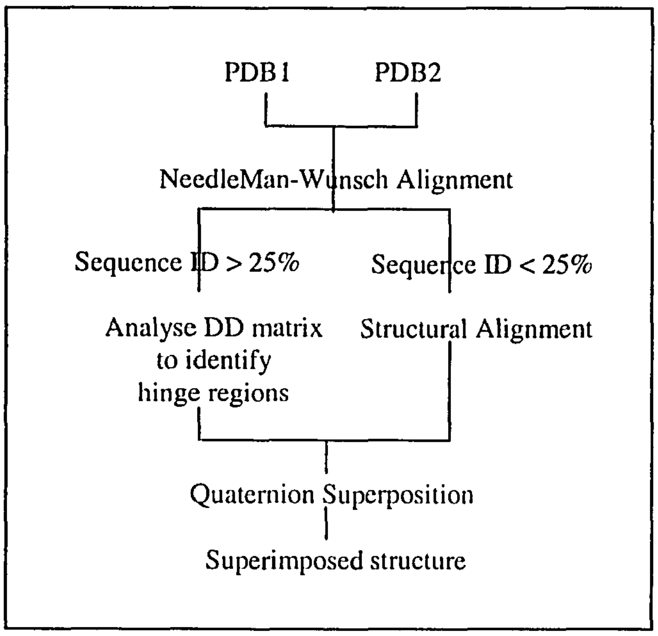


Figure 3.6 *Flow chart for conducting pair-wise superposition with SuperPose.*

SuperPose also calculates multiple structure superpositions (Diamond, 1992). Multiple structure superpositions often need to be performed for an NMR ensemble or between multiple chains of different proteins. If the sequences share 100% similarity (NMR ensembles) then superposition is done by calculating an average structure. First, an all-against-all difference distance matrix calculation is done for all the sequences to arrange the sequences in a hierarchical manner. The two most structurally similar sequences are superposed and an average structure is calculated. The third most similar structure then gets superposed onto the average structure and the process is iterated until all sequences/structures are superposed. For dissimilar sequences, an average structure cannot be calculated and hence multiple superposition is done by arranging the sequences in a hierarchy of structural similarity and superposing the two most similar structures. The third structure gets added onto this 'pileup' and the process is iterated (Figure 3.7).

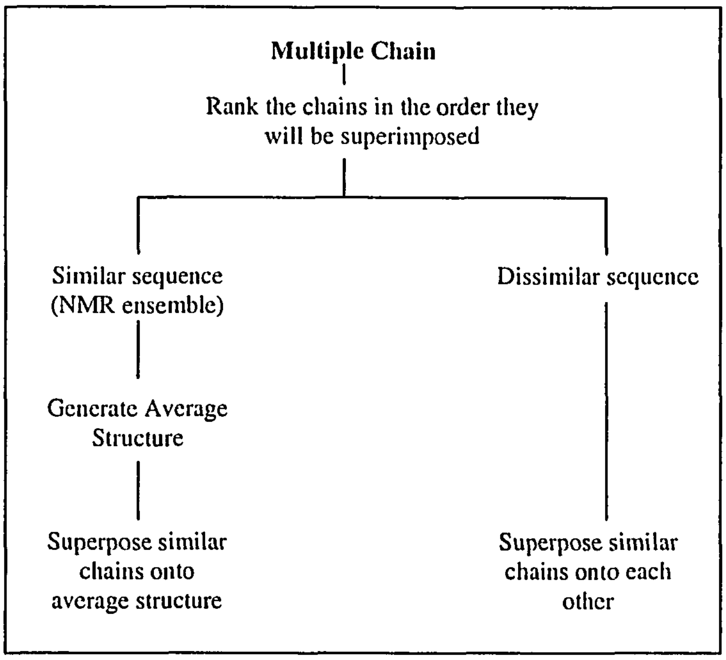

Multiple Chain
Rank the chains in the order they will be superimposed
Similar sequence (NMR ensemble)
Dissimilar sequence
Generate Average Structure
Superpose similar chains onto average structure
Superpose similar chains onto each other

Figure 3.7 *Flow chart for multiple chain superposition*

SuperPose uses a quaternion approach to handle its superposition processes. This algorithm was written in C and has been modified from the PDBSUP Fortran program developed by Rupp and Parkin (Rupp and Parkin, 1996). SuperPose generates both local and global RMSD values for alpha carbons, backbone atoms, heavy atoms and all atoms. When identical sequences are compared, SuperPose also generates 'per residue' RMSD tables and plots to allow users to identify, assess and view individual residue displacements (Figure 3.8).

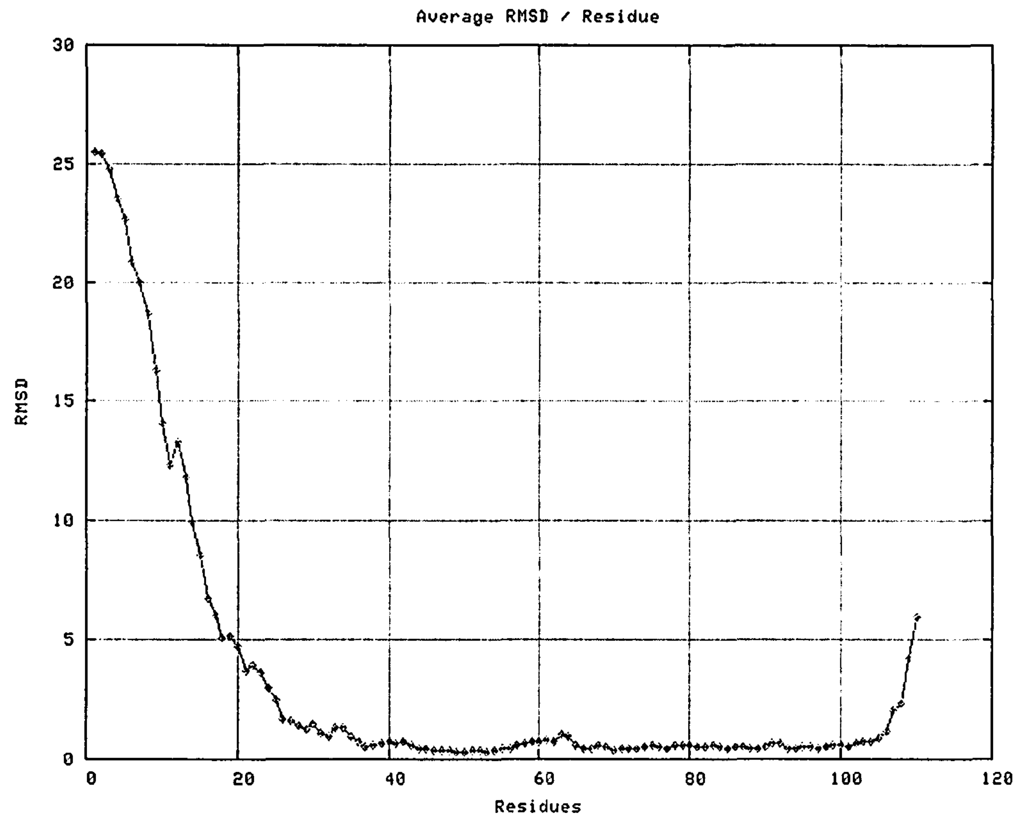


Figure 3.8 *Average RMSD/residue plot for the 28 chains of 1BQV. Each residue of each chain is compared against the corresponding residue of the averaged chain to obtain the RMSD for each residue. These RMSDs are then added up and divided by the number of chains to obtain the average RMSD/residue.*

3.4 Results and Discussion

We designed SuperPose to provide outputs which will be intuitive to the user and easily understood. SuperPose produces up to seven kinds of output and they clearly and succinctly describe the superposition process. The outputs are:

(i) a PDB file of the coordinates of the superimposed molecules.

(ii) a PDB file of the backbone coordinates of an average structure.

(iii) a sequence or structural alignment of the sequences.

(iv) a difference distance matrix plot (for a pair of molecules).

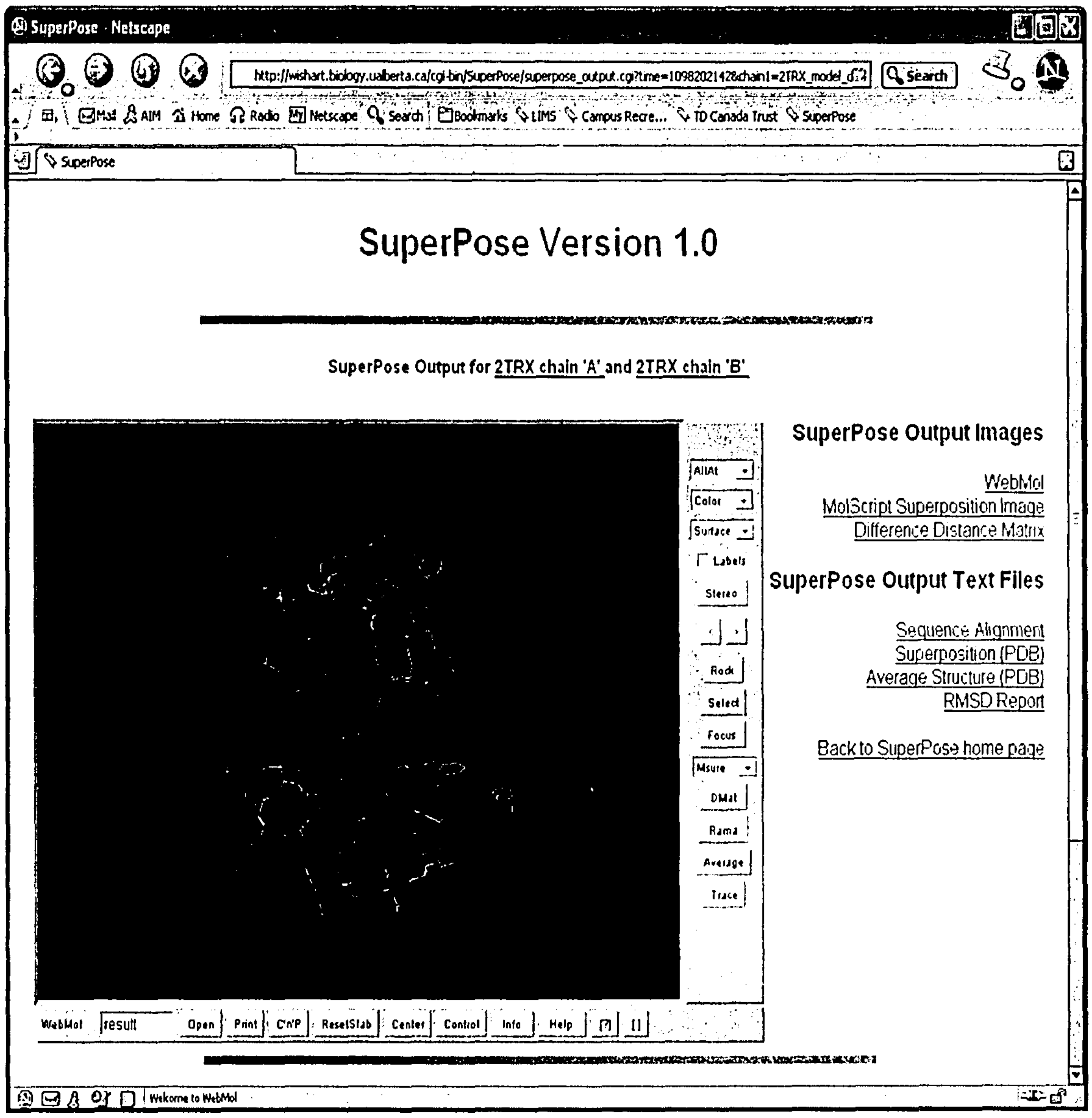
(v) a RMSD report containing both local and global RMSD values.

(vi) a still image (PNG) of the superimposed molecules generated by Molscript.

(vii) a WebMol (Walther, 1997) applet view of the superimposed molecules.

By default, SuperPose produces the WebMol image of the superimposed structure and a set of hyperlinks located on the right side of the screen (Figure 3.9). All other data (RMSD values, PDB files, images, etc.) can be accessed, saved or viewed via the hyperlinks on the SuperPose output page. Generated images and PDB files can also be saved or copied directly to the user's hard disk or loaded into standard presentation or molecular visualization programs.

As far as we are aware, there are only two other operational structure superposition servers. These are the Combinatorial Extension or CE server (Shindyalov and Bourne, 1998) and ProSup (Lackner et al., 2000). ProSup is designed only to identify and superimpose small regions of proteins that are structurally very similar (<1.1 Å RMSD). Further, ProSup performs only pairwise structural comparisons, not multiple structure superpositions. On the other hand, the CE server performs both pairwise and multiple structural alignments but does not provide many of the outputs that SuperPose provides. Only a single RMSD value is provided by CE, compared to global and local RMSDs provided by SuperPose. SuperPose also provides RMSDs based on alpha carbons, all atoms and heavy and backbone atoms. CE only provides an alignment of the two structures and links to the superposed structure on its main output page. No image is produced and although an applet to view the structure is provided, it does not work in Internet Explorer (but does work in Netscape). In contrast, SuperPose provides an image in a variety of formats (grayscale/color, mono/stereo, backbone/ribbon) and the WebMol applet works on all browsers. CE is not user friendly (users have to specify the chains/models of a protein) and provides very little control over the superposition process. SuperPose however, parses a protein and allows the user to choose the chains/models for superposition. SuperPose also provides advanced structural alignment options for more advanced users.



SuperPose Version 1.0

SuperPose Output for 2TRX chain 'A' and 2TRX chain 'B'

SuperPose Output Images

WebMol
MolScript Superposition Image
Difference Distance Matrix

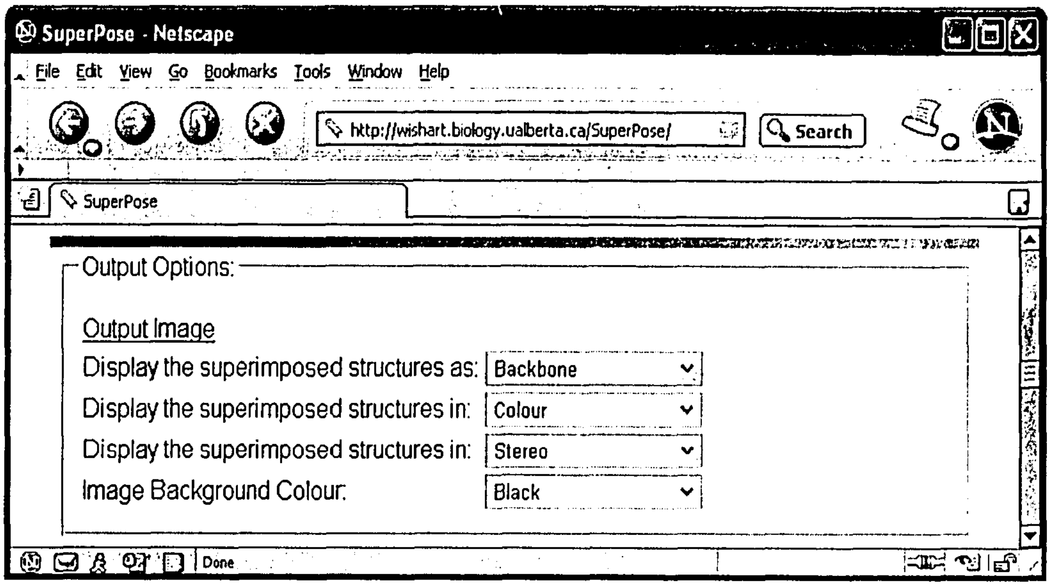
SuperPose Output Text Files

Sequence Alignment
Superposition (PDB)
Average Structure (PDB)
RMSD Report

Back to SuperPose home page

Figure 3.9 *WebMol applet for structure visualization with additional SuperPose hyperlinks.*

SuperPose appears to be unique as both a general superposition server and in its ability to handle difficult superposition tasks. Additionally, superpose provides a wide range of interactive viewing options (color, black and white, stereo, mono, ribbon, backbone) (Figure 3.10), file (text/image) outputs and RMSD outputs not found on other servers or in other standalone packages.


SuperPose - Netscape
File Edit View Go Bookmarks Tools Window Help
http://wishart.biology.ualberta.ca/SuperPose/
Search
SuperPose
Done

Output Options:

Output Image

Display the superimposed structures as: Backbone

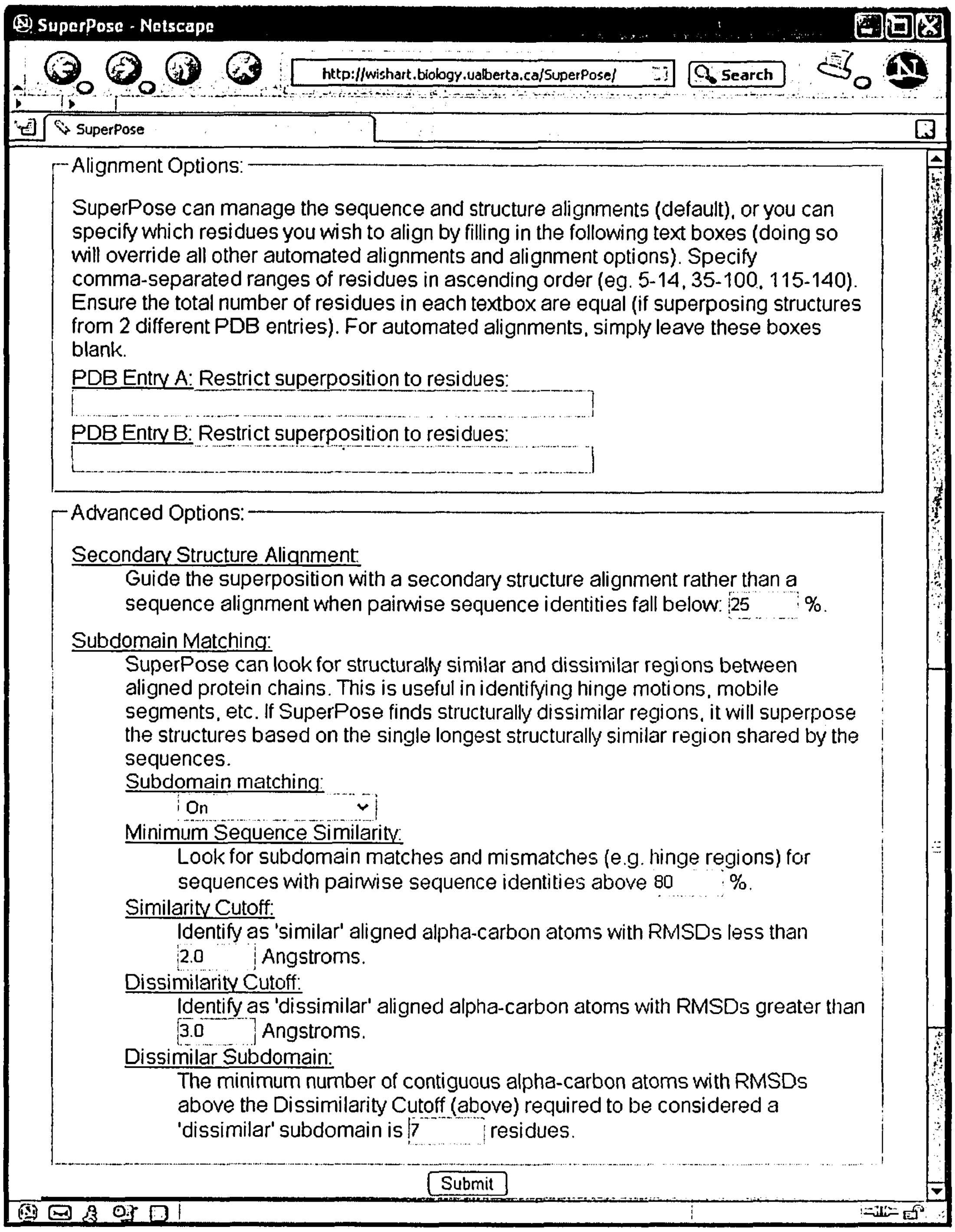
Display the superimposed structures in: Colour

Display the superimposed structures in: Stereo

Image Background Colour: Black

Figure 3.10 *Example of the output options offered by SuperPose.*

SuperPose also offers advanced options for sequence or structural alignments. For instance, if users do not like the automated alignment initially generated by SuperPose, they can select the residues by which alignment should be done (forced alignment). Other advanced options allow the user to set the minimum pairwise sequence identity for secondary structural alignment, and various options for subdomain matching. SuperPose looks for structurally similar and dissimilar regions between protein chains and if it finds structurally dissimilar regions, it will superimpose the structures based on the single longest structurally similar region shared by the sequences. Users can also toggle subdomain matching and can guide the superposition with their choice of minimum sequence similarity between a pair of chains. RMSD thresholds for the similarity and dissimilarity cutoffs as well as the minimum number of residues that constitute a dissimilar subdomain can also be set by the user (Figure 3.11).



Alignment Options:

SuperPose can manage the sequence and structure alignments (default), or you can specify which residues you wish to align by filling in the following text boxes (doing so will override all other automated alignments and alignment options). Specify comma-separated ranges of residues in ascending order (eg. 5-14, 35-100, 115-140). Ensure the total number of residues in each textbox are equal (if superposing structures from 2 different PDB entries). For automated alignments, simply leave these boxes blank.

PDB Entry A: Restrict superposition to residues:

PDB Entry B: Restrict superposition to residues:

Advanced Options:

Secondary Structure Alignment:
Guide the superposition with a secondary structure alignment rather than a sequence alignment when pairwise sequence identities fall below: 25 %.

Subdomain Matching:
SuperPose can look for structurally similar and dissimilar regions between aligned protein chains. This is useful in identifying hinge motions, mobile segments, etc. If SuperPose finds structurally dissimilar regions, it will superpose the structures based on the single longest structurally similar region shared by the sequences.

Subdomain matching:
On

Minimum Sequence Similarity:
Look for subdomain matches and mismatches (e.g. hinge regions) for sequences with pairwise sequence identities above 80 %.

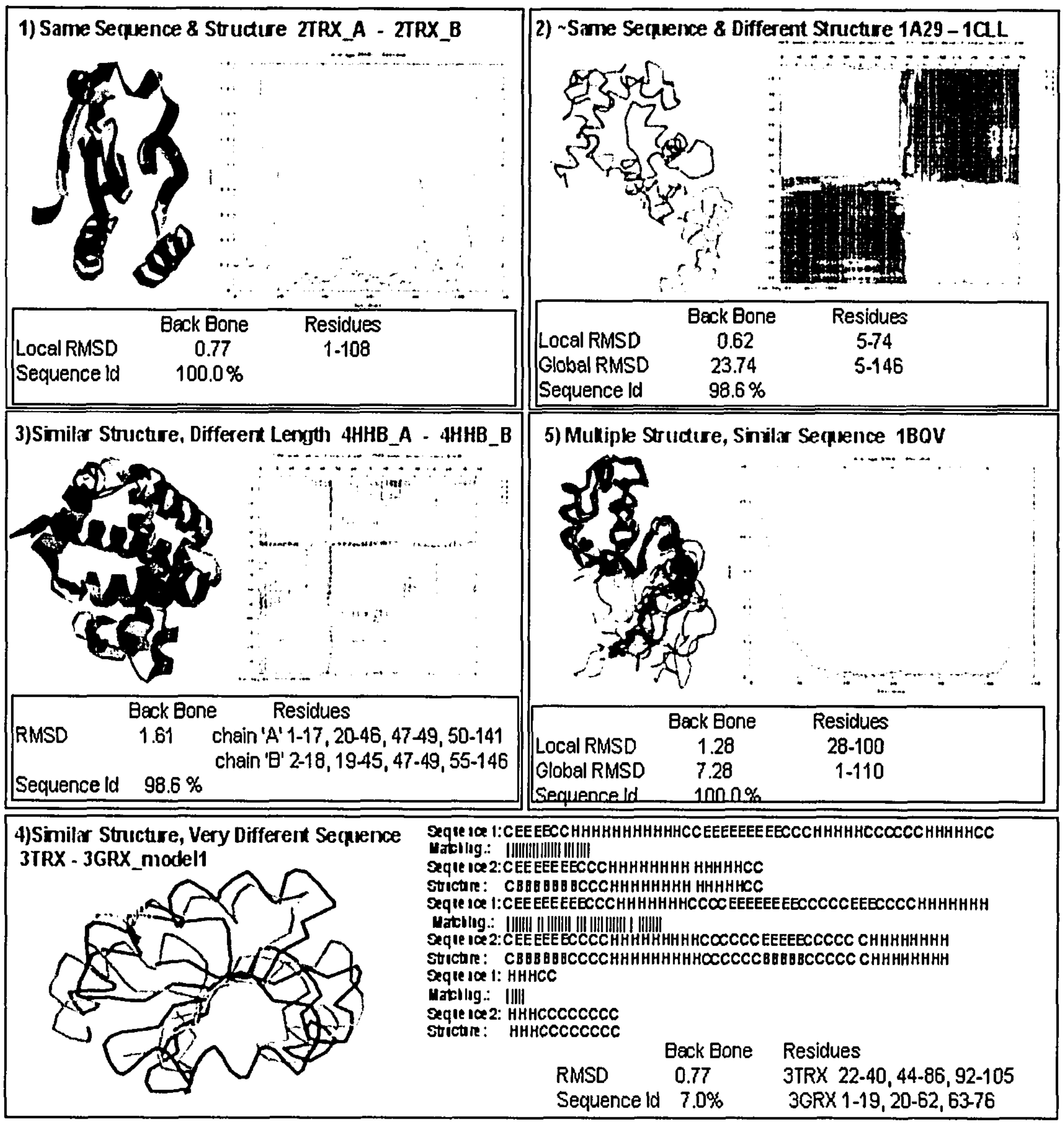
Similarity Cutoff:
Identify as 'similar' aligned alpha-carbon atoms with RMSDs less than 2.0 Angstroms.

Dissimilarity Cutoff:
Identify as 'dissimilar' aligned alpha-carbon atoms with RMSDs greater than 3.0 Angstroms.

Dissimilar Subdomain:
The minimum number of contiguous alpha-carbon atoms with RMSDs above the Dissimilarity Cutoff (above) required to be considered a 'dissimilar' subdomain is 7 residues.

Figure 3.11 *Advanced secondary structure and alignment options*

As seen in Table 3.1, tests performed on a number of difficult superposition tasks (1A29 on 1CLL; 5TNC on 1CLL; 2TRX on 3TRX; ensemble superpositions of 28 ETS pointed domains – 1BQV) indicate that SuperPose is able to automatically identify hinge motions, perform superpositions with structures of very different lengths or atom numbers and correctly superimpose very remotely related structures (Figure 3.12). The same tests performed on two popular superposition packages DeepView and MolMol revealed the shortcomings of these packages. Many of the tests were not possible with MolMol as it requires exact matches of atom/residue numbers and DeepView aligns to the longest contiguous matching segment – regardless of length. However, when the same residues are matched and assessed (a 'forced' or manual superposition), SuperPose was able to reproduce RMSDs for both pairwise and structure superpositions that agree well with those values reported by DeepView or MolMol (Table 3.2). The one difference appears to lie in the fact that MolMol ignores carbonyl oxygen atoms in its evaluation of backbone RMSD values.



1) Same Sequence & Structure 2TRX_A - 2TRX_B

```
              Back Bone      Residues
Local RMSD      0.77          1-108
Sequence Id    100.0%
```

2) ~Same Sequence & Different Structure 1A29 - 1CLL

```
              Back Bone      Residues
Local RMSD      0.62          5-74
Global RMSD     23.74         5-146
Sequence Id     98.6%
```

3) Similar Structure, Different Length 4HHB_A - 4HHB_B

```
              Back Bone      Residues
RMSD            1.61     chain 'A' 1-17, 20-46, 47-49, 50-141
                         chain 'B' 2-18, 19-45, 47-49, 55-146
Sequence Id    98.6 %
```

5) Multiple Structure, Similar Sequence 1BOV

```
              Back Bone      Residues
Local RMSD      1.28          28-100
Global RMSD     7.28          1-110
Sequence Id    100.0%
```

4) Similar Structure, Very Different Sequence 3TRX - 3GRX_model1

```
Sequence 1:CEEEECCHHHHHHHHHHHCCEEEEEEEEECCCHHHHHCCCCCCHHHHHCC
Matching.:  ||||||||||||||| ||||||
Sequence 2:CEEEEEEECCCHHHHHHHHH HHHHHCC
Structure:  CBBBBBBBCCCHHHHHHHHH HHHHHCC
Sequence 1:CEEEEEEEECCCHHHHHHHCCCCEEEEEEEECCCCCEEECCCCHHHHHHH
Matching.:  |||||| || |||||| ||| |||||||| | ||||||
Sequence 2:CEEEEEEECCCCHHHHHHHHHCCCCCCEEEEECCCCC CHHHHHHHH
Structure:  CBBBBBBBCCCCHHHHHHHHHHCCCCCCBBBBBCCCCC CHHHHHHHH
Sequence 1: HHHCC
Matching.:  |||||
Sequence 2: HHHCCCCCCCC
Structure:   HHHCCCCCCCC
                          Back Bone    Residues
              RMSD          0.77       3TRX 22-40, 44-86, 92-105
              Sequence Id   7.0%       3GRX 1-19, 20-62, 63-76
```

Figure 3.12 *Superposition outputs for the five different superposition cases treated by SuperPose.*

Table 3.1 *Comparison of superposition statistics for SuperPose, DeepView and MolMol using superposition tasks of varying difficulty done without user intervention.*

Structure(s), % Sequence ID & Category	**SuperPose** Backbone RMSD (Å) & residue matches	**DeepView** Backbone RMSD (Å) & residue matches	**MolMol** Backbone RMSD (Å) & residue matches
Same Sequence + Similar Structure (pair)			
Thioredoxin (2TRX_A vs. 2TRX_B) - 100% ID	0.77 (1-108)	0.77 (1-108)	0.66 (1-108)
Hemoglobin (4HHB_A vs. 1DKE_A) - 100% ID	0.37 (1-141)	0.37 (1-141)	Fail - atom mismatch
P21 Oncogene(6Q21_A vs. 6Q21_B) -100% ID	1.27 (1-171)	1.27 (1-171)	1.16 (1-171)
~Same Sequence + Different Structure (pair)			
Calmodulin (1A29 vs. 1CLL) - 98.6% ID	0.82 (5-75) 23.83 (5-146)	15.02 (4-146) Could not detect hinge region	Failed - atom mismatch
Maltose Bind Prot. (1OMP vs. 1ANF) - 100% ID	0.83 (1-112) 8.87 (2 -370)	3.76 (1-370) Could not detect hinge region	3.76 (1-370) Could not detect hinge region
Similar Structure + Different Length (pair)			
Hemoglobin (4HHB_A vs. 4HHB_B) - 43% ID	1.61 (98.6% aligned)	1.21 (65%aligned)	Failed - atom mismatch
Thioredoxin (3TRX vs. 2TRX_A) - 29% ID	4.23 (85.7% aligned)	1.70 (25.7% aligned)	Failed - atom mismatch
Lysozyme/Lactalbumin(1DPX vs.1A4V) - 36% ID	1.63 (99% aligned)	2.05 (55% aligned)	Failed - atom mismatch
Calmodulin/TnC (1CLL vs. 5TNC) - 47% ID	6.83 (100% aligned)	5.24 (52% aligned)	Failed - atom mismatch
Similar Structure + Very Different Sequence			
Ubiquitin/Elongin (1UBI vs. 1VCB_A) - 26% ID	3.22 (96% aligned)	2.19 (72.3% aligned)	Failed - atom mismatch
Thio/Glutaredoxin(3TRX vs. 3GRX_A) - 7% ID	4.64 (92.7% aligned)	1.76 (14.6% aligned)	Failed - atom mismatch
Hemoglobins (1ASH vs. 2LHB) - 17% ID	4.11 (93.8 % aligned)	1.90 (19.7% aligned)	Failed - atom mismatch
Thioredoxins (1NHO_A vs. 1DE2_A) - 22% ID	7.77 (77.6 % aligned)	4.04 (21.2% aligned)	Failed - atom mismatch
Multiple Structures + Same Sequence			
Pointed Domain (1BQV, 28 chains) - 100% ID	6.28 (100% aligned)	Failed	5.88 (100% aligned)
Trypsin Inhibitor (1PIT, 20 chains) - 100% ID	1.32 (100% aligned)	Failed	1.30 (100% aligned)
Oligomerization domain (1OLG, 4 chains) 100% ID	0.57 (100% aligned)	Failed	0.58 (100% aligned)
Oxidoreductase (1NHO, 20 chains) - 100% ID	0.96 (100% aligned)	Failed	0.96 (100% aligned)

Table 3.2 *Comparison of superposition RMSD values for SuperPose, DeepView and MolMol using a "forced" superposition of matching residues identified by DeepView's Magic Fit.*

Structure(s), % Sequence ID & Category	**SuperPose** Backbone RMSD (Å) & match residues	**DeepView** Backbone RMSD (Å) & match residues	**MolMol** Backbone RMSD (Å) & match residues
Same Sequence + Similar Structure (pair)			
Thioredoxin (2TRX_A vs. 2TRX_B) – 100% ID	0.77 (1-108)	0.77 (1-108)	0.66 (1-108)
Hemoglobin (4HHB_A vs. 1DKE_A) – 100% ID	0.37 (1-141)	0.37 (1-141)	0.36 (1 -141)
P21 Oncogene(6Q21_A vs. 6Q21_B) –100% ID	1.27 (1-171)	1.27 (1-171)	1.16 (1-171)
~Same Sequence + Different Structure (pair)			
Calmodulin (1A29 vs. 1CLL) – 98.6% ID	14.97 (4 -146)	15.02 (4-146)	14.94 (4-146)
Maltose Bind Prot. (1OMP vs. 1ANF) – 100% ID	3.76 (1 -370)	3.76 (1-370)	3.76 (1-370)
Similar Structure + Different Length (pair)			
Hemoglobin (4HHB_A vs. 4HHB_B) – 43% ID	1.21 4HHB_A (51-141) 4HHB_B (56-146)	1.21 4HHB_A (51-141) 4HHB_B (56-146)	1.16 4HHB_A (51-141) 4HHB_B (56-146)
Thioredoxin (3TRX vs. 2TRX_A) – 29% ID	1.70 3TRX (23-49) 2TRX_A (23-49)	1.70 3TRX (23-49) 2TRX_A (23-49)	1.63 3TRX (23-49) 2TRX_A (23-49)
Lysozyme/Lactalbumin(1DPX vs.1A4V) – 36% ID	2.05 1DPX (32-99) 1A4V (29-96)	2.05 1DPX (32-99) 1A4V (29-96)	1.91 1DPX (32-99) 1A4V (29-96)
Calmodulin/TnC (1CLL vs. 5TNC) – 47% ID	5.24 1CLL (4-78) 5TNC (14-88)	5.24 1CLL (4-78) 5TNC (14-88)	5.21 1CLL (4-78) 5TNC (14-88)
Similar Structure + Very Different Sequence			
Ubiquitin/Elongin (1UBI vs. 1VCB_A) – 26% ID	2.19 1UBI (12-66) 1VCB_A(13-67)	2.19 1UBI (12-66) 1VCB_A(13-67)	2.11 1UBI (12-66) 1VCB_A(13-67)
Thio/Glutaredoxin(3TRX vs. 3GRX_A) – 7% ID	1.75 3TRX (78-89) 3GRX_A(54-65)	1.76 3TRX (78-89) 3GRX_A(54-65)	1.51 3TRX (78-89) 3GRX_A(54-65)
Hemoglobins (1ASH vs. 2LHB) – 17% ID	1.90 1ASH (26-54) 2LHB (34-62)	1.90 1ASH (26-54) 2LHB (34-62)	1.76 1ASH (26-54) 2LHB (34-62)

Thioredoxins (1NHO_A vs. 1DE2_A) – 22% ID	4.04
	1NHO_A(13-30)
	1DE2_A(14-31)
		4.04
		1NHO_A(13-30)
		1DE2_A(14-31)
			3.85
			1NHO_A(13-30)
			1DE2_A(14-31)

Multiple Structures + Same Sequence			
Pointed Domain (1BQV, 28 chains) – 100% ID	6.28 (100% aligned)	Failed	5.88 (100% aligned)
Trypsin Inhibitor (1PIT, 20 chains) – 100% ID	1.32 (100% aligned)	Failed	1.30 (100% aligned)
Oligomerization domain (1OLG, 4 chains) 100% ID	0.57 (100% aligned)	Failed	0.58 (100% aligned)
Oxidoreductase (1NHO, 20 chains) – 100% ID	0.96 (100% aligned)	Failed	0.96 (100% aligned)

3.5 Conclusions and Future Directions

In summary, SuperPose provides a simple-to-use, web-accessible approach to performing a wide range of sophisticated structural superpositions. It is unique in that it combines sequence alignment and difference distance matrix calculations to constrain its quaternion eigenvalue superposition calculations. SuperPose has been designed to provide an abundance of useful textual and visual outputs that allow both structural 'novices' and experienced structural biologists to explore and compare complex protein structures. SuperPose's appeal lies in its simple web interface and its ability to report a wide range of information (local and global RMSDs, sequence/structural alignments) with minimum user interventions. In this regard, we have received positive feedback from a number of researches worldwide who have used SuperPose and have suggested improvements to it. For the future, we hope to expand SuperPose's ability to superimpose more than two proteins and also be able to superimpose DNA and RNA molecules. We hope that addition of these functionalities will make SuperPose more attractive to researches and will make structural superposition far more accessible and far simpler than it currently is.

3.6 References

Arfken, G. (1985) "Taylor's Expansion." §5.6 in *Mathematical Methods for Physicists*, 3rd ed. Orlando, FL: Academic Press, pp. 303-313.

Arvo, J. (1994) *Graphics Gems II.* New York: Academic Press, pp. 351-354 and 377-380.

Diamond, R.(1992) "On the multiple simultaneous superposition of molecular structures by rigid body transformations", *Protein Sci.*, **1**: 1279-1287.

Goldstein, H. (1980) "The Euler Angles" and "Euler Angles in Alternate Conventions." §4-4 and Appendix B in *Classical Mechanics, 2nd ed.* Reading, MA: Addison-Wesley, pp. 143-148 and 606-610.

Hamilton, W. R. (1853) "Lectures on Quaternions: Containing a Systematic Statement of a New Mathematical Method", Dublin: Hodges and Smith.

Hearn, D. and Baker, M. P. (1996) *Computer Graphics: C Version, 2nd ed.* Englewood Cliffs, NJ: Prentice-Hall, pp. 419-420 and 617-618.

Kabsch, W.A. (1976) "A solution for the best rotation to relate two sets of vectors", *Acta Crystallogr. A.* **32**:922-923.

Kaplan, W. and Littlejohn, T.G. (2001) "Swiss-PDB Viewer (Deep View)", *Brief Bioinform.* **2**: 195-197.

Kearsley, S.K.(1989) "On the orthogonal transformation used for structural comparisons", *Acta Crystallogr. A.* **45**:208-210.

Koradi, R., Billeter, M. and Wuthrich, K. (1996) "MOLMOL: a program for display and analysis of macromolecular structures", *J. Mol. Graph.* **14**:29-32.

Lackner. P., Koppensteiner, W.A., Sippl, M.J. and Domingues, F.S. (2000) "ProSup: a refined tool for protein structure alignment", *Protein Eng.* **13**: 745-752.

Landau, L. D. and Lifschitz, E. M. (1976) *Mechanics, 3rd ed.* Oxford, England: Pergamon Press.

MacLachlan, A.D. (1972) "A mathematical procedure for superimposing atomic coordinates of proteins", *Acta Crystallogr. A.* **28**: 656-657.

Metropolis, N. and Ulam, S. (1994) "The Monte Carlo Method." *J. Amer. Stat. Assoc.* **44**: 335-341.

Needleman, S.B. and Wunsch, C.D. (1970) "A general method applicable to the search for similarities in the amino acid sequence of two proteins", *J. Mol. Biol.* **48**: 443-453.

Richards, F.M. and Kundrot, C.E. (1988) "Identification of structural motifs from protein coordinate data: secondary structure and first-level supersecondary structure", *Proteins.* **3**:71-84.

Rupp, B. and Parkin. S. (1996) "PDBSUP – A FORTRAN program that determines the rotation matrix and translation vector for best fit superposition of two pdb files by solving the quaternion eigenvalue problem", Lawrence Livermore National Laboratory.

Shindyalov, I.N. and Bourne, P.E. (1998) "Protein structure alignment by incremental combinatorial extension (CE) of the optimal path." *Protein Engineering,* **11**(9) 739-747.

Taylor, B. (1715) *Methodus incrementorum directa et inverse.*

Tuma, J. J. (1974) *Dynamics.* New York: Quantum Publishers.

Wall, L., Christiansen, T. and Schwartz, R. L. (1996) *Programming Perl, 2nd edn.* O'Reilly & Associates, Inc., Sebastopol, CA. US.

Walther, D. (1997) "WebMol – a Java based PDB viewer", *Trends Biochem. Sci.* **22**: 274-275.

Willard, L., Ranjan, A., Zhang, H., Monzavi, H., Boyko, R.F., Sykes, B.D. and Wishart, D.S. (2003) "VADAR: a web server for quantitative evaluation of protein structure quality", *Nucleic Acids Res.* **31**: 3316-3319.

Chapter 4: MovieMaker: A Web Server For Rapid Rendering of Protein Motions and Interactions

4.1 Introduction

Protein structures are not static. Indeed proteins vibrate, twist, bend, open, close, assemble and disassemble in a variety of ways over many different time scales. Protein motions and conformational accommodation actually lie at the heart of many important protein-ligand interactions including protein-DNA binding (Gehring et al., 1994), enzyme-substrate interactions (Versées et al., 2002), muscle contraction (Cooke, 1986) and oligomerization (Chong et al., 2005). Thanks largely to the continuing developments of X-ray crystallography, NMR spectroscopy and computational molecular dynamics (MD), the temporal and spatial scales involved in protein motions are now being better understood (Berneche and Roux, 2000). Small-scale (<0.3 Å) motions over short periods of time (picoseconds) can be modeled or measured using either X-ray thermal B factors (Oka et al., 2000), NMR order parameters (Petrache et al., 1999) or shorter (<1 ns) molecular dynamics simulations. Mid-scale motions (0.5 – 3 Å) tend to take place over longer periods of time (100's of picoseconds to nanoseconds) and can be discerned through comparing NMR structure ensembles, looking at the X-ray structures of different crystal isomorphs or running long (10-100 ns) molecular dynamics simulations. Large-scale motions (5 – 30 Å), which may take microseconds to complete are typically evident only through comparing two different states or experimentally determined structures of the same molecule (say bound and unbound). These motions cannot normally be modeled via molecular dynamics.

The fact that molecular motions play such an important role in protein function underlies the growing need to be able to illustrate or visualize these motions in an informative manner. Several commercial MD packages allow molecular "movies" to be screen-captured and displayed via standard computer presentations. However, relatively few biologists are familiar with nor do they have the expertise to use these relatively sophisticated and expensive software tools. Likewise, not all motions (especially some of the more interesting or larger-scale ones) can be captured through off-the-shelf molecular dynamics simulations.

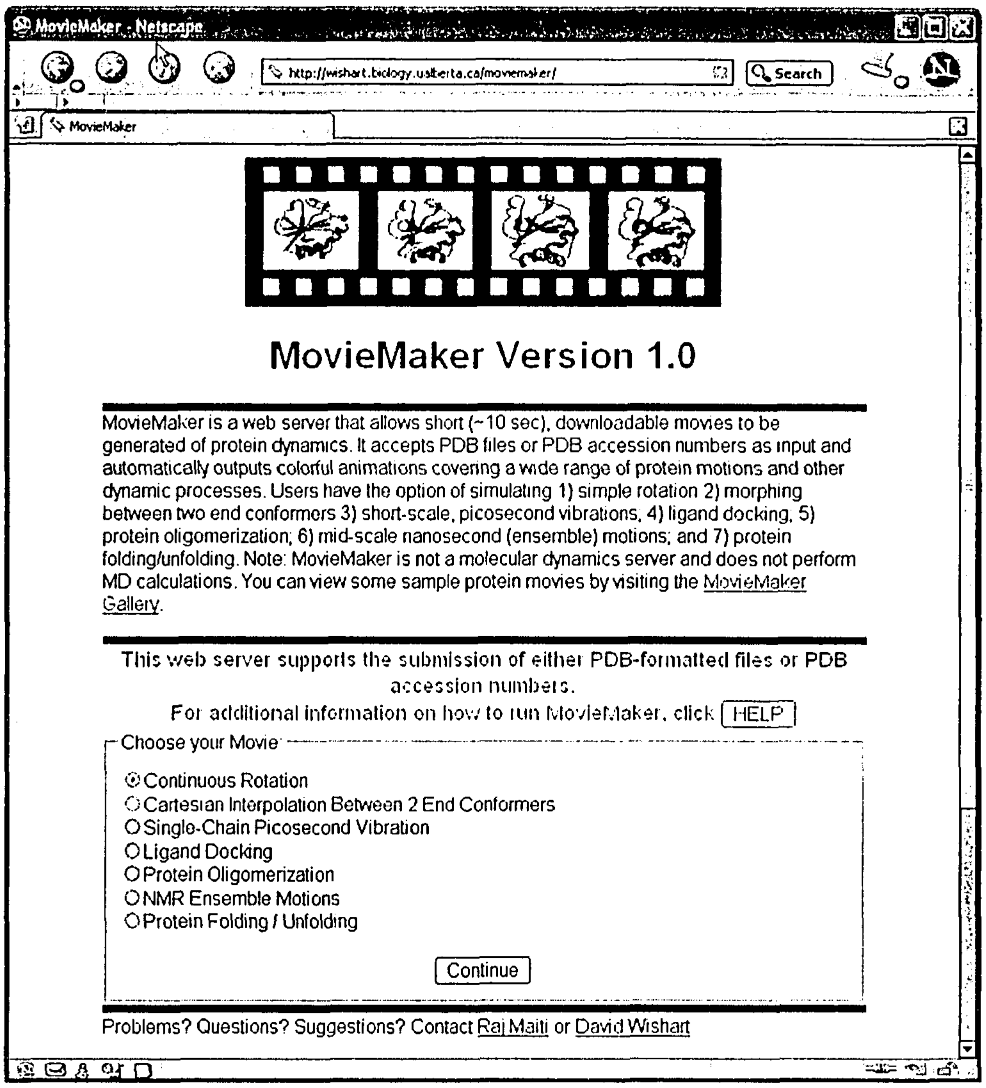
More recently, Mark Gerstein at Yale University has developed an excellent and easy-to-use web server (The Morph server) which allows non-expert users to simulate and visualize certain types of protein motions through the generation of short movies (Krebs and Gerstein, 2000). This tool specifically models larger scale motions or "morphs" by interpolating the structural changes between two different protein conformers and generating a set of plausible intermediate structures. A hyperlink pointing to the morph results is then emailed to the user. The primary focus of the Morph server has been to facilitate research, analysis and classification of different kinds of large-scale molecular motions of monomeric proteins. As such it is not intended to be a general molecular animation tool capable of simulating all aspects of macromolecular motion such as folding/unfolding, docking, oligomerization, multimeric protein motions, vibrational motions or structural ensemble motions. For instance, the Morph server offers only limited user-control over rendering, animation parameters, color or point-of-view. Likewise, the methods used to generate the movies are computationally intensive and can require up to an hour of CPU time before completion. Furthermore, the Morph server does not allow modeling of motions from a single input structure or from more than two input structures. This is somewhat limiting if one is interested in modeling motions from NMR structure ensembles or if one only has a single X-ray structure of a given protein. Additionally, the Morph server does not support the visualization of other kinds of protein motions such as folding/unfolding or of docking and self-assembly events involving two or more structures.

Here we wish to describe a general molecular animation server that allows a wide range of motions and dynamic events to be simulated and offers a much greater range of user-control over rendering and animation parameters. This server, called MovieMaker, allows small, medium and large-scale motions to be rendered using as little as one and as many as 50 input structures. It allows users full control over the rendering style, background, refresh rate, point of view, rotation rate, transition style and animation quality. It also employs a simplified Cartesian coordinate interpolation approach coupled with an intelligent superposition algorithm that allows most kinds of molecular movies to be rendered automatically in less than a minute. Unlike any other simulation tool that we are aware of, MovieMaker also allows users the option of creating movies of protein folding/unfolding as well as molecular docking or self-assembly (oligomerization) of two or more molecules. Rather than being a specialized analytical tool, the main purpose of MovieMaker is to quickly and conveniently generate realistic, downloadable animations of protein motions that can be used by non-specialists for a variety of educational or instructive purposes.

4.2 Program Description

MovieMaker supports seven kinds of animations; 1) simple rotation 2) morphing between two end-state conformers 3) small-scale vibrations; 4) small molecule docking; 5) self – assembly or oligomerization; 6) mid-scale (structure ensemble) motions and 7) protein folding/unfolding. The type of animation is dependent on both the input data and the type of animation selected by the user from a pull-down menu box. The MovieMaker home page presents the user with the type of movie choices that one can generate (Figure 4.1) along with an extensive gallery illustrating the types of motions that can be modeled. Upon selecting the appropriate movie type, the user is presented with input boxes for file uploads and various display options. The input for all MovieMaker animations is one or more PDB formatted files containing one or more protein structures. These may be directly uploaded to MovieMaker using the file selector boxes or alternately one or more PDB accession numbers may be provided and the program will automatically retrieve the appropriate PDB files from the RCSB website (Berman et al., 2000).

The simplest motion to render in MovieMaker is a basic rotation. The rotation animation is intended to allow all sides of a given structure to be conveniently and continuously viewed. Once the rotation option is selected, only a single PDB file needs to be provided. If multiple structures are found in a single PDB file, the program treats the ensemble as a single oligomeric structure. To generate the rotation animation, MovieMaker takes the input file and applies a series of standard X, Y or Z -axis rotations to the structure(s). Users have the option of changing the speed and extent of the rotation (the default is a rotation of 360° in 10° increments).


MovieMaker - Netscape

http://wishart.biology.ualberta.ca/moviemaker/

MovieMaker Version 1.0

MovieMaker is a web server that allows short (~10 sec), downloadable movies to be generated of protein dynamics. It accepts PDB files or PDB accession numbers as input and automatically outputs colorful animations covering a wide range of protein motions and other dynamic processes. Users have the option of simulating 1) simple rotation 2) morphing between two end conformers 3) short-scale, picosecond vibrations, 4) ligand docking, 5) protein oligomerization; 6) mid-scale nanosecond (ensemble) motions; and 7) protein folding/unfolding. Note: MovieMaker is not a molecular dynamics server and does not perform MD calculations. You can view some sample protein movies by visiting the MovieMaker Gallery.

This web server supports the submission of either PDB-formatted files or PDB accession numbers.
For additional information on how to run MovieMaker, click HELP

Choose your Movie
- [✓ Continuous Rotation]
- Cartesian Interpolation Between 2 End Conformers
- Single-Chain Picosecond Vibration
- Ligand Docking
- Protein Oligomerization
- NMR Ensemble Motions
- Protein Folding / Unfolding

Continue

Problems? Questions? Suggestions? Contact Raj Maiti or David Wishart

Figure 4.1 *MovieMaker homepage.*

Additionally, through the viewing options section located below the data entry section, users may change the orientation of the molecule (by rotating along the X, Y or Z axes), the molecular rendering style (backbone and ribbon), molecular content (backbone only, all atoms), image size (small, medium, large) and molecular color (by secondary structure, sequence or single uniform color). All renderings are performed using the molecule visualization program MOLSCRIPT (Kraulis, 1991). Since MOLSCRIPT generates postscript files, these files must be converted to a GIF (graphic interchange format) image by the Unix utility program "convert". Another freeware program called gifmerge (http://www.lcdf.org/gifsicle/) is then used to string all these GIFs to create an animated GIF. This animated GIF is looped continuously to provide a smooth visualization of the rotation process. The animations are instantly viewable on the user's web browser and may be saved by right-clicking on the animation image and selecting "save file" or "save image". A typical movie file is approximately 500 Kbytes. MovieMaker also generates a downloadable set of PDB text files that user's may use to generate specific images or regenerate animations using their own molecular rendering software.

Small molecule docking requires only a single PDB file containing two or more molecular entities that are already bound. Once the small molecule docking option is selected, the MovieMaker program automatically parses the input file and identifies all molecular entities (small molecules and large). Users must then select one protein entity and one small molecule entity. To generate a pre-docking or two-component state, MovieMaker then calculates the centre of masses for the protein alone, the small molecule alone and the complex together. A vector is then drawn from the complex's centre of mass to the small molecule's centre of mass. This vector defines the direction that the small molecule entity must move to create a pre-docking state. To generate a pre-docking state, the small molecule is translated 15 Å along this vector and randomly rotated (between 15° and 60°) about its individual X, Y and Z axes. MovieMaker then calculates a series of intermediate positions by incrementally rotating and translating the small molecule until it reaches its original bound position. The default increment is 1/20th of the original translation/rotation. As with the rotation option, users have full control over coloring, point of view and rendering styles. Note that the small molecule is always rendered as a ball-and-stick entity.

Oligomerization and self-assembly are handled in a very similar manner to small molecule docking. As with docking, only a single PDB file containing two or more macromolecules is required. However, when the self-assembly option is selected all macromolecular entities within the PDB file will be separated and reassembled. Users do not have the option to select a subset of molecules that are to be assembled or docked. The same centre of mass calculations, direction vector calculations, rotations and translations are repeated for all subunits in the oligomer to create a preliminary disassembled state. The complex is reassembled using the reverse rotation and translation operations. Note that both the docking and oligomerization simulations are inherently "rigid" dockings. No internal motions are currently simulated for the interacting proteins or ligands.

When MovieMaker's small-scale or vibrational motions option is selected only a single PDB file (containing a single chain) is needed. The key trick to most of

MovieMaker's motion generation is to use Cartesian coordinate or torsion angle interpolation between a "perturbed" state and a "ground" state conformation. In interpolation the intermediate positions between two states or positions are calculated in a linear fashion based on the coordinates between the two end points and a chosen increment. The resulting images therefore depict a pathway that two conformers can take when morphing from one to the other. To create a perturbed state, MovieMaker generates random displacements of the x, y and z coordinates of between 0.0 and 2.0 Å for all heavy atoms in the original PDB file according to the magnitude of their corresponding B factors. Specifically, the ad hoc formula $B=100^*\{|\Delta x| + |\Delta y| + |\Delta z|\}$ is used to calculate the x,y and z atomic displacements. If no B factors are present in the file a default value of 60 is used. These perturbed structures are then rendered and infinitely looped to create the illusion of a short term MD simulation.

When the structural ensemble motion option is selected, users must provide a PDB file containing two or more copies of the same protein molecule. This may include an NMR structure ensemble (typically 20-40 structures) or multiple copies (2-10) of the same protein in a single unit cell from a standard X-ray structure. If the molecules are not identical, MovieMaker will provide a warning and cancel the rendering operation. MovieMaker uses a recently developed superpositioning tool called SuperPose (Maiti et al., 2004) to intelligently and automatically compare, rank and superimpose all structures in the ensemble or unit cell. Moving from the most similar pair to the least similar remaining pair of superimposed structures in the ensemble, Cartesian coordinate interpolation is performed to generate a series of intermediate structures. MovieMaker automatically takes into account the number of structures in the ensemble to select an optimal number of intermediate structures for a smooth, fluid transition between states. As with most other simulations in MovieMaker, the ensemble motion option always morphs the structure from a starting state to a perturbed state and back so that the movie can be placed into smoothly running infinite loop.

To depict large-scale motions MovieMaker requires two PDB files, each containing the same protein but in a different conformation. As with the ensemble motion option, MovieMaker employs the SuperPose program to intelligently superimpose the two structures and identify any large scale hinge or domain motions. Displacements between the two states are categorized (< 2 Å over >90% of the protein length or > 2 Å over >10% of the protein length) by calculating a difference distance matrix between the two superimposed structures. After the displacement has been categorized, intermediate structures are created by interpolating between the two end conformers (Figure 4.2). Both small motions and larger hinge motions can be mapped by the Cartesian interpolation method. Minor distortions creep in for very large scale hinge motions, but these distortions can be almost removed by increasing the number of intermediate structures generated between the two conformations.

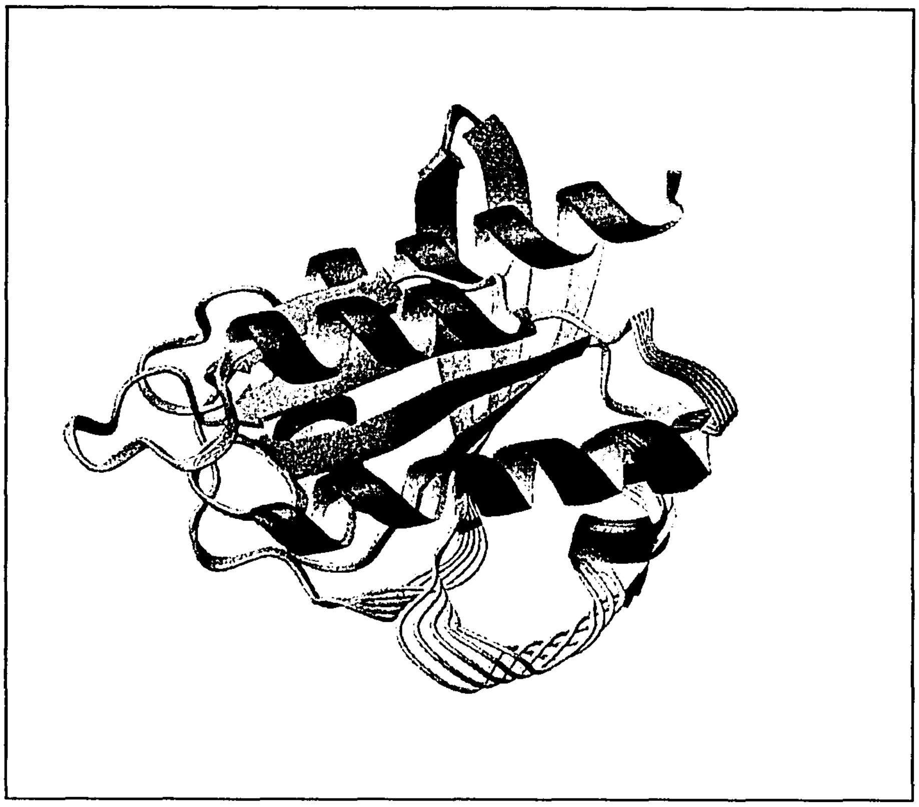


Figure 4.2 *Cartesian interpolation between the oncogene proteins (4Q21 and 6Q21 chain A).*

When the protein folding/unfolding option is selected only a single PDB file (containing a single, folded protein chain) is needed. For this kind of simulation, coordinate interpolation must be done in torsion angle space as the intermediate structures simply get too distorted during the unfolding process. In torsion angle interpolation the native structure is regenerated using phi/psi/omega angles derived from the PDB file. This re-rendering in torsion space requires additional structural optimization and can take several minutes, depending on the size of the structure. Once rendered in torsion angle space, the backbone phi/psi angles are iteratively "relaxed" to an unfolded or extended set of phi/psi torsion angles of $-160^{o} \pm 10^{o}$. All intermediate structures are rendered using the same torsion angle structure generator (called PepMake). As always MovieMaker morphs the structure from the starting state (folded) to the end state (unfolded) and back so that the movie can be placed into a smoothly running infinite loop.

4.3 Results and Discussion

To assess the performance of MovieMaker we chose 30 random protein structures from the PDB consisting of single monomers, complex heteromultimers, NMR ensembles and a variety of proteins with bound ligands. The proteins or protein complexes ranged in size from 56 residues to 1450 residues. We assessed the performance of the program using three criteria: 1) realism; 2) accuracy and 3) speed on as many different types of motions as possible. Assessing realism is somewhat qualitative and highly visual. However, we wanted to make sure that the resulting animations were smooth and did not lead to any obvious "breaches" of the laws of physics such as atoms or chains passing through one another or serious distortions in secondary structure. Of nearly 30 animations studied using the default parameters we found only four animations that exhibited a mildly unrealistic chain distortion or a physically unrealistic event. These were confined primarily to animations with very large hinge movements. Apart from these "breaches", vibrational and ensemble motions, rotations, docking, folding and hinge motions all appeared to perform very well with no obvious problems.

In terms of assessing the accuracy or realism of the small scale vibrational and structural ensemble motions, we used myoglobin (153 residues, PDB 1MYF) and the pointed domain (110 residues, PDB 1BQV) to visually compare MovieMaker's movies with those generated via the MD simulation program, GROMACS (www.gromacs.org). Comparing a short (10 ps) MD simulation for myoglobin calculated by GROMACS with the motion calculated by MovieMaker's small scale vibrational motion generator, one can see very little difference (the two movies are available in the MovieMaker's gallery page). Similarly, a long term (2.5 ns) MD simulation from GROMACS for the pointed domain appears to be qualitatively similar to the ensemble animation generated by MovieMaker (eg. on the gallery page). Hinge motion movements were tested with DNA polymerase beta, cyanovarin N, recoverin and calmodulin proteins.

Table 4.1 lists the approximate CPU time (2.0 GHz processor with 512 MB RAM) taken for each of the six types of motion supported by MovieMaker. Obviously these times will vary with the load on the server and the speed of the user's internet connection. It is clear that the rotation animation is the fastest (10 s) while the motion for the ensemble of structures is the slowest (260 s). Most animations are generated in less than 30 seconds. This underlines one of the key strengths of MovieMaker – its speed. Using conventional MD or non-conventional MD simulations (such as adiabiatic dynamics, activated dynamics or Brownian dynamics) these would typically take many hours or days of CPU time.

4.4 Comparison to Other Servers

The Morph server (Krebs and Gerstein, 2000) was the first web-based "movie making" servers. The Morph server specifically models larger scale motions or "morphs" by interpolating the structural changes between two different protein conformers via adiabatic dynamics and generating a set of plausible intermediate structures. The primary focus of the Morph server has been to facilitate research, analysis and classification of different kinds of large-scale molecular motions of monomeric proteins. In addition to the Morph server, the Gerstein group at Yale has also developed a database of macromolecular motions. This database (Echols et al., 2003) is one of the most complete and comprehensive databases on molecular motions to date. Users can view movies of precompiled motions existing in the database or can create a custom morph of any two molecules using the morph server.

Apart from the Morph server at Yale University, there is only one other (unpublished) web based macromolecular motion server. It is located at NIH and is composed of two smaller servers: a morph server and a movie server. These servers have been named as the Indie morph and Indie movie server respectively. The Indie morph server can produce a movie of two PDB files morphing into each other. It also provides a number of options to the user to control the quality of the movie. These options include initial orientation, display format of the molecule, color, movie format, size and speed of the movie. However, no superpositioning of the molecules is done prior to the morphing and some of the above mentioned options have fixed upper limits. Since no superpositioning is done, users have to do a superposition manually and this severely limits the usability of the Indie morph server. The Indie molecular movie server takes a single molecule and simply rotates or rocks it around one of the Cartesian axes via a user specified angle. No small scale motion is calculated for a single molecule. MovieMaker however, produces a motion map of the molecule and also rotates the molecule continuously around the y axis of the molecule. The Indie morph and movie servers are located at

http://molbio.info.nih.gov/structbio/indie.html and

http://molbio.info.nih.gov/structbio/indie_morph.html respectively.

Table 4.1 *Summary of different simulation or animation scenarios and the CPU time taken to complete the calculation (times will vary according to server load and PDB file size).*

Simulation Example	*PDB IDs*	*Time Taken (seconds)*
Simple rotation about Y axis	4Q21	10
Motion between 2 end-state conformers	1A29 & 1CLL	15
Small scale vibrational motions	1MYF_A	16
Oligomerization (assembly/disassembly)	1C48	25
Ligand docking	1A29 & TFP	20
NMR ensemble simulation	1BQV	260
Protein folding/unfolding	1A29	215

4.5 Conclusion

It is important to emphasize that MovieMaker is a molecular animation server, not a modeling or molecular dynamics server. In animation or simulation one attempts to mimic reality using a variety of ad hoc rules that adhere to the general rules of physics. In modeling or MD, one attempts to precisely regenerate reality by solving Newton's equations of motions using well-calibrated molecular force fields. Simulation or animaiton is frequently employed by video game developers, cartoonists and special effects artists to generate illusions of motion, speed, or impact. Rather than attempting to solve Newton's equations for every motion or event, most simulation specialists employ rapidly calculable interpolations and ad hoc rules to generate the necessary visual effect. This allows them to quickly generate the images needed for interactive game play or tight movie release deadlines. By opting for simulation over modeling (i.e. mimicry over reality) we have been able to create a very fast and flexible molecular animation tool. While the images and files generated by MovieMaker should not be used to calculate or predict key molecular parameters, they certainly could be of considerable use for many educational, instructive or illustrative purposes by non-MD specialists. We believe the animations produced by MovieMaker will potentially allow the facile creation of dynamic web-pages, informative on-line course notes or compelling Powerpoint presentations.

4.6 References

Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I.N., and Bourne, P.E. (2000) "The Protein Data Bank", *Nucleic Acids Research*, **28**:235-242.

Berneche, S. and Roux, B. (2000) "Molecular dynamics of the KesA K(+) channel in a bilayer membrane", *Biophys. J.,* **78**: 2900-2917.

Chong, L.T., Snow, C.D., Rhee, Y.M, and Pande, V.S. (2005) "Dimerization of the p53 oligomerization domain: identification of a folding nucleus by molecular dynamics simulations", *J. Mol. Biol.*, **345**: 869-378.

Cooke, R.(1986) "The Mechanism of Muscle Contraction", *CRC Crit. Rev. Biochem.,***21**: 53-118.

Echols, N., Milburn, D. and Gerstein, M. (2003) "MolMovDB: analysis and visualization of conformational change and structural flexibility", *Nucleic Acids Res.*, **31**: 478-482.

Gehring,W.J., Affolter, M. and Burglin, T.(1994) "Homeodomain Proteins", *Annual Review of Biochemistry*, **63**: 487-526.

Kraulis, P.J. (1991) "MOLSCRIPT: a program to produce both detailed and schematic plots of protein structures", *J. Appl. Crystallogr.*, **24**: 946-950.

Krebs, W.G. and Gerstein, M. (2000) "The morph server: a standardized system for analyzing and visualizing macromolecular motions in a database framework", *Nucleic Acids Res.*, **28**: 1665-1675.

Maiti, R. Domselaar, G.H.V., Zhang, H., and Wishart, D.S. (2004) "SuperPose: a simple server for sophisticated structural superposition", *Nucleic Acids Res.*, **32:** W590-W594.

Oka, T., Yagi, N., Fuhisawa, T., Kamikubo, H., Tokunaga, F. and Kataoka, M. (2000) "Time-resolved x-ray diffraction reveals multiple conformations in the M - N transition of the bacteriorhodopsin photocycle", *Proc. Natl. Acad. Sci. USA.*, **97**: 14278-14282.

Petrache, H.I., Tu, K., and Nagle, J.F. (1999) "Analysis of Simulated NMR Order Parameters for Lipid Bilayer Structure Determination", *Biophys J.*, **76**: 2479-2487.

Versées, W., Decanniere, K., Holsbeke, V.E., Devroede, N. and Steyaert, J. (2002) "Enzyme-Substrate Interactions in the Purine-specific Nucleoside Hydrolase from *Trypanosoma vivax.*", *J. Biol. Chem.*, **277**:15938-15946.

Chapter 5: General Discussion and Conclusion

This thesis has focused on describing a number of novel algorithms and bioinformatics software tools for a variety of applications in structural biology. These computational tools range from simple visualization systems (MovieMaker), to sophisticated comparative systems (SuperPose) to very complex structure prediction tools. Specifically, Chapter 2 describes a novel distance geometry approach that allows one to rapidly generate or predict viable 3D protein structures. The bioinformatics tools described in Chapters 3 and 4 include two freely available and easy to use web servers. The first (SuperPose) enables structural biologists to perform protein structural superpositions while the second (MovieMaker) creates movies of macromolecular motions.

In Chapter 2, an algorithm to predict the 3D structures of proteins from their sequence information was described. Using a process known as distance geometry, the program performs a rapid conformational search to predict viable packing patterns or packing geometries of secondary structure elements and then evaluates the quality or viability of these structures using a heuristic energy function. Structure prediction algorithms are usually computationally intensive, employing large clusters or distributed computing systems (Larson and Pande, 2002). Many of these efforts have required days or weeks of computer time on large super computers (Feldman and Hogue, 2000). The algorithm described in Chapter 2 does not require massive computational power and can produce results within a few hours on a desktop computer. To date, this distance geometry approach has been used to successfully predict the structures of smaller helical proteins (~ 60 residues). Since the complexity of the program scales with the number of secondary structures, predicting the structures of larger proteins with more complicated or varied secondary structure content (i.e. beta sheets) will require searching a much larger conformational space and hence will require more computing power than a single desktop machine.

Chapters 3 and 4 describe two web servers which could aid structural biologists in their understanding and analysis of protein structures. Chapter 3 described an informative, easy to use web server called SuperPose which performs structural superpositions of proteins. SuperPose can handle five different kinds of superpositions: 1) superposition of two or more molecules of identical sequences but slightly different structure; 2) superposition of two molecules of identical sequence but profoundly different structure; 3) superposition of two or more molecules of modestly dissimilar sequence, length and structure; 4) superposition of two or more molecules with profoundly different lengths but similar structure or sequence and 5) superposition of two or more molecules that are profoundly different in sequence but similar in structure. To deal with all five cases, SuperPose combines sequence/structural alignments along with difference distance matrix calculations to identify the common set of points and then uses quaternion based rotation and least squares minimization to overlay these points (Rupp and Parkin, 1996). The output produced by SuperPose includes an image of the superposed molecules, a sequence/structural alignment, a downloadable superposed files in PDB format along with global and local RMSD values. Users have considerable control over the rendering or colour of the image and its background (backbone, ribbon, color, grayscale, mono or stereo). Advanced superposition options such as guiding the superposition with a selected set of residues are also available to the user. In short, SuperPose provides a comprehensive resource wherein complex superpositions can be performed with little or minimal user input.

Chapter 4 describes MovieMaker, a web server which is designed to rapidly generate a wide variety of protein animations or defined motions. Protein motions play an important role in almost all protein or enzyme-based processes and so it is important to be able to visualize or conceptualize these processes. Protein motions can be best visualized through the creation of "molecular movies". Historically such movies were generated by performing Molecular Dynamics (MD) simulations and then superimposing "snapshots" of the structures calculated at different time steps. Since molecular dynamics simulations are computationally intensive, the creation of these movies can take hours or even days. More recently, it has been realized that these kinds of simulations could be generated using a much simpler and computationally faster "morphing" approach (Krebs and Gerstein, 2000). In morphing, multiple intermediate conformations between two (superimposed) end state conformers are created via linear or Cartesian coordinate interpolation. Consequently a movie can be created by rapidly rendering the set of intermediate "snapshot" structures. This is the concept we employed in MovieMaker. Specifically, MovieMaker employs Cartesian and torsional angle interpolation techniques to create its intermediate structures. These structures are then rendered as a series of coloured GIF images which are looped continuously to create a movie. MovieMaker can depict seven kinds of macromolecular motions: 1) simple rotations; 2) morphing between two end states; 3) docking of ligands to proteins; 4) oligomerization process; 5) small scale vibrational motions; 6) structural ensemble motions; and 7) protein folding/unfolding. Since MD calculations are not performed, MovieMaker can produce these motions very rapidly and conveniently display them over a web browser. Users can select from various rendering options, as well as determine the speed of the movie by selecting the number of intermediate structures that are created. Since MovieMaker provides a platform where a variety of protein motions can be visualized easily and rapidly, we believe that this server could be quite useful to both advanced researchers as well as beginning students who want to create informative molecular animations.

While the programs described in Chapters 3 and 4 have been published or await publication, much work needs to be done on the protein folding work described in Chapter 2 in order to have its results published. Although we have demonstrated that the structures of small proteins were predicted very rapidly with this system, it is clear that additional structural validation needs to be performed. Presently, a new heuristic scoring function is being developed (by Mr. Steven Neal) to improve the ranking and scoring of the structures generated by our distance geometry method. We are hopeful that the best structure (lowest RMSD relative to the native structure) will be routinely chosen by this new and "improved" scoring function. It is also evident that further improvement to the algorithm used to pack secondary structural elements – especially beta sheet packing – is clearly needed. Under the present implementation of the program, structures of both small and large proteins can be predicted. However, due to the constraints in computing power, we limited our structure predictions to relatively small proteins. The prediction of larger proteins could be attempted if more computing resources become available.

Although the web servers (SuperPose and MovieMaker) have been released into the public domain, we believe there is definitely room for improvement. For instance, SuperPose can be easily enhanced by including superpositioning capability for hetero-atoms, as well as DNA and RNA molecules. Likewise, the MovieMaker server could be enhanced by providing the user with more rendering abilities. Right now, only backbone and ribbon representations of the molecules are provided. Further additions could include ball-and –stick, wireframe and CPK representations of protein structures. Under the present implementation of MovieMaker, side chains are shown for all residues. It will be attractive if the user was given the option to select specific residues for which he/she wants the side chains to be displayed. Presently, MovieMaker supports only animated GIF displays. Richer movie formats such as MPEG could potentially be generated and made available for download. An option to choose the size (dimensions) of the movie image could also be added. Additionally, the script generated by the MOLSCRIPT rendering program could be made available for download so that users could edit it according to their choices. The ligand docking animation seems to have the most scope for improvement. To simulate (as opposed to animate) the docking process, further tests need to be performed to detect and correct steric clashes during the docking process. We are presently working on these aspects of MovieMaker to make it more attractive and useful to users.

Overall, we believe that the tools and algorithms described in this thesis will aid biologists and increase their understanding of protein structures. We also hope that further additions and improvements to our folding algorithm will eventually provide a novel and rapid computational approach for generating protein structures and that it may serve as an important contribution to this field.

5.1 References

Feldman, H. J. and Hogue, C. W. (2000) "A fast method to sample real protein conformational space", *Proteins*, **39**: 112-131.

Krebs, W.G. and Gerstein, M. (2000) "The morph server: a standardized system for analyzing and visualizing macromolecular motions in a database framework", *Nucleic Acids Res.*, **28**: 1665-1675.

Larson, S.M., Snow, C.D., Shirts, M.R. and Pande, V.S. "Folding@Home and Genome@Home: Using Distributed Computing to Tackle Previously Intractable Problems in Computational Biology", (Grant, R. ed.), Horizon Press, 2002.

Rupp, B. and Parkin. S. (1996) "PDBSUP – A FORTRAN program that determines the rotation matrix and translation vector for best fit superposition of two pdb files by solving the quaternion eigenvalue problem", Lawrence Livermore National Laboratory.

Appendix A: The Distance Geometry Approach

In this Appendix, the metric matrix distance geometry process is described via a five atom molecule. To create a distance matrix, the minimum and maximum possible distances between each pair of atoms are required. These interatomic distance bounds can be obtained from simple chemical principles (atomic number and hybridization of the atoms). The distance between two atoms which are both connected to a third atom can be calculated from the angle of the central atom and the lengths of the two bonds. The distance between two atoms that are separated by three bonds can vary with the torsion angle of the central bond, the minimum distance corresponding to a 0° torsion angle and the maximum distance to a 180° torsion angle. Distance limits for other interatomic distances (between atoms in a 1,n relationship where $n>4$) is difficult to determine but it is usual to require that such atom pairs do not approach closer than the sum of the van der Waals radii of the two atoms. The upper bound for the distance is then usually set to a large value.

A procedure called *triangle smoothing* is then used to refine the initial set of distance bounds. Triangle smoothing uses two simple trigonometric restrictions on groups of three atoms (Figure A.1). The first restriction is that the distance between two atoms A and C cannot be greater than the sum of the distances AB and BC, i.e. $u_{AC} \leq u_{AB} + u_{BC}$. The second restriction says that the minimum distance between A and C cannot be less than the difference between the lower bound on AB and the upper bound on BC, i.e. $l_{AC} \geq l_{AB} - u_{BC}$. These two inequalities are repeatedly applied to the set of distance bounds until the entire set of distance bounds is self consistent and all possible distance triplets satisfy both inequalities.

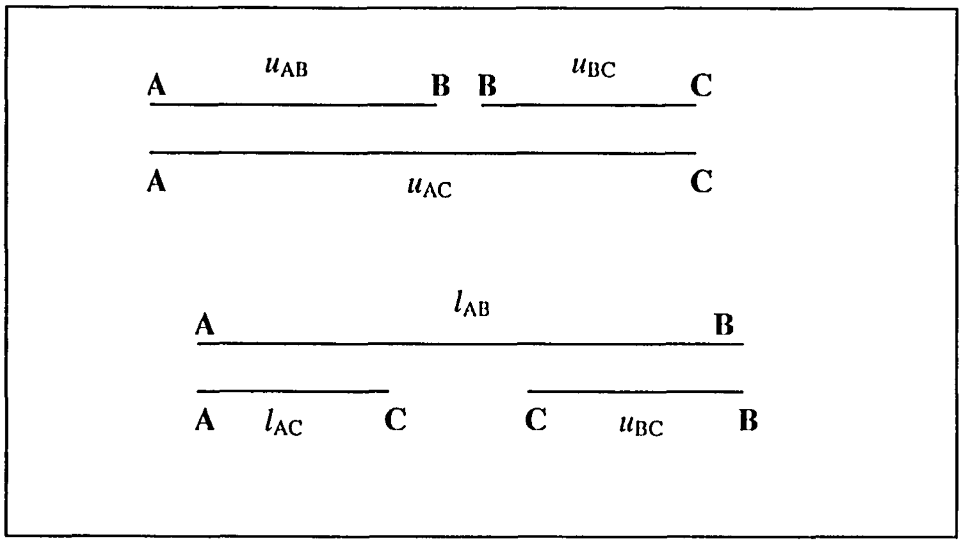


Figure A.1 *The two triangle inequalities used in distance geometry*

Once the distance matrix has been constructed, random values are assigned to all the interatomic distances between the upper and lower bounds to give a starting distance matrix. This trial matrix is then subjected to a process called embedding, in which the 'distance space' representation of the conformation is converted to a set of Cartesian coordinates by performing a series of matrix operations. These matrix operations are described below.

Let us consider a five atom molecule in which all the bonds are assumed to have a length of 1.3 Å and all the internal angles are 120°. If it is further assumed that the van der Waals radius of each atom is 1.4 Å, then the initial trial matrix (T) is as follows:

$$\mathbf{T} = \begin{bmatrix} 0.0 & 1.3 & 2.2517 & 3.4395 & 99.0 \\ 1.3 & 0.0 & 1.3 & 2.2517 & 3.4395 \\ 2.2517 & 1.3 & 0.0 & 1.3 & 2.2517 \\ 2.6 & 2.2517 & 1.3 & 0.0 & 1.3 \\ 2.8 & 2.6 & 2.2517 & 1.3 & 0.0 \end{bmatrix} \qquad \text{A.1}$$

The lower bound for the distance between atoms 1 and 5 equals the sum of their van der Waals radii and the upper bound has been arbitrarily set to 99Å. All the other distances have been set on the basis of geometric arguments. After triangle smoothing, the distance between atoms 1 and 5 is changed to 4.5033 Å. This distance is equal to the sum of the upper bounds between atoms 1 and 3 and 3 and 5. The resulting smoothed bound matrix is

$$\begin{bmatrix} 0.0 & 1.3 & 2.2517 & 3.4395 & 4.5033 \\ 1.3 & 0.0 & 1.3 & 2.2517 & 3.4395 \\ 2.2517 & 1.3 & 0.0 & 1.3 & 2.2517 \\ 2.6 & 2.2517 & 1.3 & 0.0 & 1.3 \\ 2.8 & 2.6 & 2.2517 & 1.3 & 0.0 \end{bmatrix} \qquad \text{A.2}$$

Distances are now randomly assigned between the upper and lower bounds to give the following random matrix **R**.

$$\mathbf{R} = \begin{bmatrix} 0.0 & 1.3 & 2.25 & 3.11 & 3.42 \\ & 0.0 & 1.3 & 2.25 & 2.85 \\ & & 0.0 & 1.3 & 2.25 \\ & & & 0.0 & 1.3 \\ & & & & 0.0 \end{bmatrix} \qquad \text{A.3}$$

From this random matrix **R**, a metric matrix **G**, is calculated each of whose elements (i, j) is equal to the scalar product of the vectors from the origin to atoms i and j:

$$G_{ij} = i \bullet j \qquad \text{A.4}$$

The elements G_{ij} can be calculated from the distance matrix using the cosine rule:

$$G_{ij} = \left(d_{io}^2 + d_{jo}^2 - d_{ij}^2\right)/2 \qquad \text{A.5}$$

where d_{io} is the distance from the origin to atom i and d_{ij} is the distance between atoms i and j. Usually, the centre of the molecule is taken as the origin of the coordinate system. The distance of each atom from the centre can be calculated directly from the interatomic distances using the following expression:

$$d_{io}^2 = \frac{1}{N}\sum_{j=1}^{N} d_{ij}^2 - \frac{1}{N^2}\sum_{j=2}^{N}\sum_{k=1}^{j-1} d_{jk}^2 \qquad \text{A.6}$$

Thus the corresponding metric matrix is

$$\mathbf{G} = \begin{bmatrix} 3.571 & 1.569 & -0.427 & -2.276 & -2.436 \\ 1.569 & 1.256 & 0.105 & -1.122 & -1.808 \\ -0.427 & 0.105 & 0.644 & 0.261 & -0.583 \\ -2.276 & -1.122 & 0.261 & 1.569 & 1.569 \\ -2.436 & -1.808 & -0.583 & 1.569 & 3.259 \end{bmatrix} \qquad \text{A.7}$$

Metric matrix **G** is a square symmetric matrix and it can be decomposed as follows:

$$\mathbf{G} = \mathbf{V}\mathbf{L}^2\mathbf{V}^T \qquad \text{A.8}$$

The diagonal elements of $\mathbf{L}^2$ are the eigenvalues of **G** and the columns of **V** are its eigenvectors. The eigen values of **G** are 8.18, 1.74, 0.26, 0.10 and 0.0 and the eigenvectors matrix is:

$$\mathrm{W} = \begin{bmatrix} 0.621 & 0.455 & -0.425 & 0.164 \\ 0.355 & -0.184 & 0.800 & 0.020 \\ 0.0 & -0.573 & -0.368 & -0.580 \\ -0.408 & -0.287 & -0.153 & 0.727 \\ -0.567 & 0.590 & 0.145 & -0.330 \end{bmatrix} \qquad \text{A.9}$$

The atomic coordinates can be derived from the metric matrix by rewriting equation A.8 as

$$\mathbf{G} = \mathbf{X}\mathbf{X}^{\mathrm{T}} \qquad \text{A.10}$$

where **X** is a matrix containing the atomic coordinates. Equating equations A.8 and A.10 gives

$$\mathbf{X} = \mathbf{V}\mathbf{L} \qquad \text{A.11}$$

As **L** has only diagonal entries, the matrix **L** is identical to its transpose: **L** = $\mathbf{L}^{\mathrm{T}}$. The atomic coordinates are thus obtained by multiplying the square roots of the eigenvalues by the eigenvectors. The best 3D structure is obtained by taking the eigenvectors that correspond to the three largest eigenvalues, provided they are all positive. If these eigenvalues are λ_1, λ_2 and λ_3 with **W** being the eigenvector matrix, then the Cartesian coordinates of each atom i can be calculated as follows:

$$x_i = \sqrt{\lambda_1}W_{i1}\ ,\quad y_i = \sqrt{\lambda_2}W_{i2}\ \text{ and }\ z_i = \sqrt{\lambda_3}W_{i3}\ . \qquad \text{A.12}$$

For the five atom example, the coordinates obtained using the three largest eigenvalues are

Atom	*x*	*y*	*z*
1	1.777	0.601	−0.218
2	1.014	−0.244	0.410
3	−0.001	−0.757	−0.188
4	−1.166	−0.379	−0.079
5	−1.623	0.799	0.075

Reference

Leach, R. Andrew, (1996) "*Molecular Modeling, Principles and Applications*", Addison Wesley Longman Limited.

Appendix B: Quaternion formulae and quaternion based rotation

In this Appendix, I will provide a brief background on quaternion mathematics and describe how rotations can be done by a quaternion.

To understand quaternions, it is necessary to understand complex numbers as quaternions are an extension of complex numbers. A system of complex numbers is defined in terms of $\hat{i}$, the square root of -1 i.e. $\hat{i} * \hat{i} = -1$.

A complex number can be written in terms of a real number and $\hat{i}$. eg: z = a + b$\hat{i}$.

The complement of a complex number is given by z' = a – b$\hat{i}$.

The modulus of a complex number is ||z|| which equals $\sqrt{z * z'} = \sqrt{a^2 + b^2}$.

Using the above properties, we can describe multiplication for complex numbers:

Let $z_1 = a_1 + b_1\hat{i}$ and $z_2 = a_2 + b_2\hat{i}$ be two complex numbers.

Then $z_1 * z_2 = (a_1a_2 - b_1b_2) + (a_1b_2 + b_1a_2)\hat{i}$.

To describe quaternions, we need three different numbers that are all square roots of –1, labeled $\hat{i}$, $\hat{j}$,and $\hat{k}$, and $\hat{i} * \hat{i} = -1$, $\hat{j} * \hat{j} = -1$, and $\hat{k} * \hat{k} = -1$.

When we multiply two of these numbers together, they behave similar to cross products of the unit vectors. $\hat{i} * \hat{j} = -\hat{j} * \hat{i} = \hat{k}$, $\hat{j} * \hat{k} = -\hat{k} * \hat{j} = \hat{i}$ and $\hat{k} * \hat{i} = -\hat{i} * \hat{k} = \hat{j}$.

A quaternion can be represented as $\mathbf{q}_1 = w_1 + x_1\hat{i} + y_1\hat{j} + z_1\hat{k}$ and it's conjugate by $\mathbf{q}_1 = w_1 - x_1\hat{i} - y_1\hat{j} - z_1\hat{k}$.

The magnitude of a quaternion is given by $\|\mathbf{q}\| = \sqrt{\mathbf{q} * \mathbf{q}'} = \sqrt{\left(w^2 + x^2 + y^2 + z^2\right)}$.

An unit quaternion has a magnitude of 1. i.e. $\|\mathbf{q}\| = 1 \Rightarrow \mathbf{q}^{-1} = \mathbf{q}'$.

Quaternions are associative: $(\mathbf{q}_1 * \mathbf{q}_2) * \mathbf{q}_3 = \mathbf{q}_1 * (\mathbf{q}_2 * \mathbf{q}_3)$, but they are not commutative $(\mathbf{q}_1 * \mathbf{q}_2) \neq (\mathbf{q}_2 * \mathbf{q}_1)$.

The inverse of a quaternion refers to the multiplicative inverse or $\frac{1}{\mathbf{q}}$ and can be computed by $\mathbf{q}^{-1} = \frac{\mathbf{q}'}{(\mathbf{q} * \mathbf{q}')}$.

Let $\mathbf{q}_1 = w_1 + x_1\hat{i} + y_1\hat{j} + z_1\hat{k}$ and $\mathbf{q}_2 = w_2 + x_2\hat{i} + y_2\hat{j} + z_2\hat{k}$ be two quaternions.

The product of $\mathbf{q}_1$ and $\mathbf{q}_2$ gives us the following formula:

$$\mathbf{q}_1 * \mathbf{q}_2 = \left(w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2\right) + \left(w_1x_2 + x_1w_2 + y_1z_2 - z_1y_2\right)\hat{i} +$$
$$\left(w_1y_2 + x_1z_2 + y_1w_2 + z_1x_2\right)\hat{j} + \left(w_1z_2 + x_1y_2 - y_1x_2 + z_1w_2\right)\hat{k}.$$

These basic formulae about quaternion operations will enable us to perform rotations with quaternions. In the next section, we will rotate a vector **v** about another vector **u** by an angle θ. We will perform this rotation geometrically and then derive the expression of the rotated vector using quaternions.

Figure B.1 shows a unit vector $\mathbf{u}$, another vector $\mathbf{v}$ and its components. The vector $\mathbf{v}$ will be rotated about $\mathbf{u}$ by an angle θ to get the rotated vector $\mathbf{v}_{rot}$. $\mathbf{u}$ and $\mathbf{v}$ lie in the plane of the paper while the cross product $\mathbf{u} \times \mathbf{v}$ points straight out of the page. In Figure B.2, $\mathbf{v}_{rot}$ and some other vectors are shown. The vector $\mathbf{v}_3$ is orthogonal to both **u** and **v** and has the same length as $\mathbf{v}_2$. This vector can be computed by $\mathbf{v}_3 = \mathbf{u} \times \mathbf{v}_2 = \mathbf{u} \times \mathbf{v}$ since **u** has unit length and $\mathbf{v}_2$ is orthogonal to **u**. It is also important to notice that $(\mathbf{u} \times \mathbf{v}) \times \mathbf{u} = \mathbf{v}_3 \times \mathbf{u} = \mathbf{v}_2$. This is true because **u** is orthogonal to $\mathbf{v}_3$ and $\|\mathbf{v}_2\| = \|\mathbf{v}_3\|$.

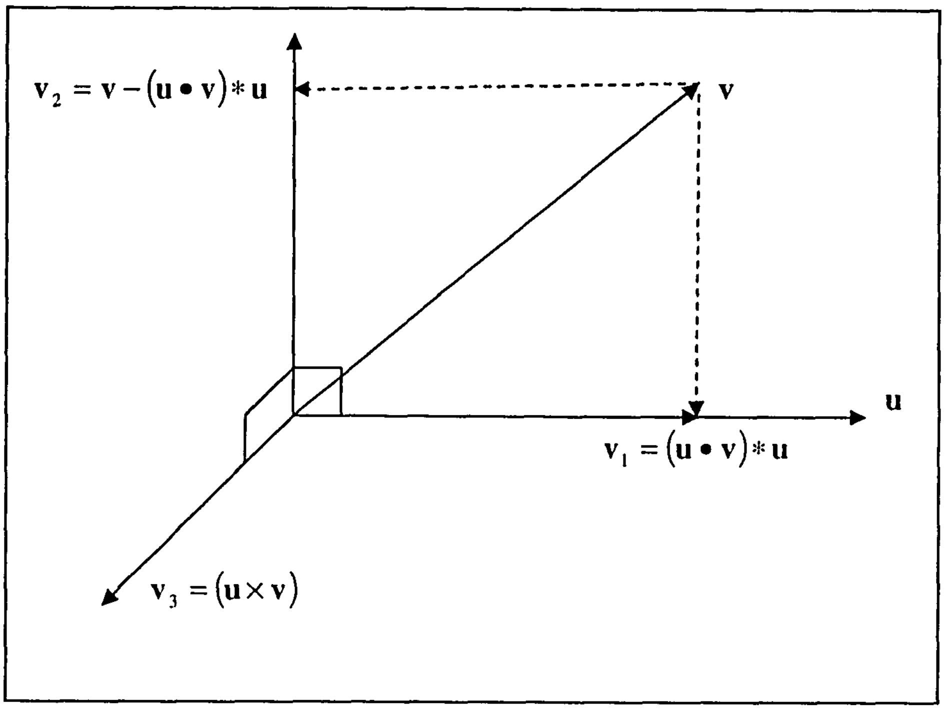


Figure B.1 *The unit vector u, vector v and the components of v.*

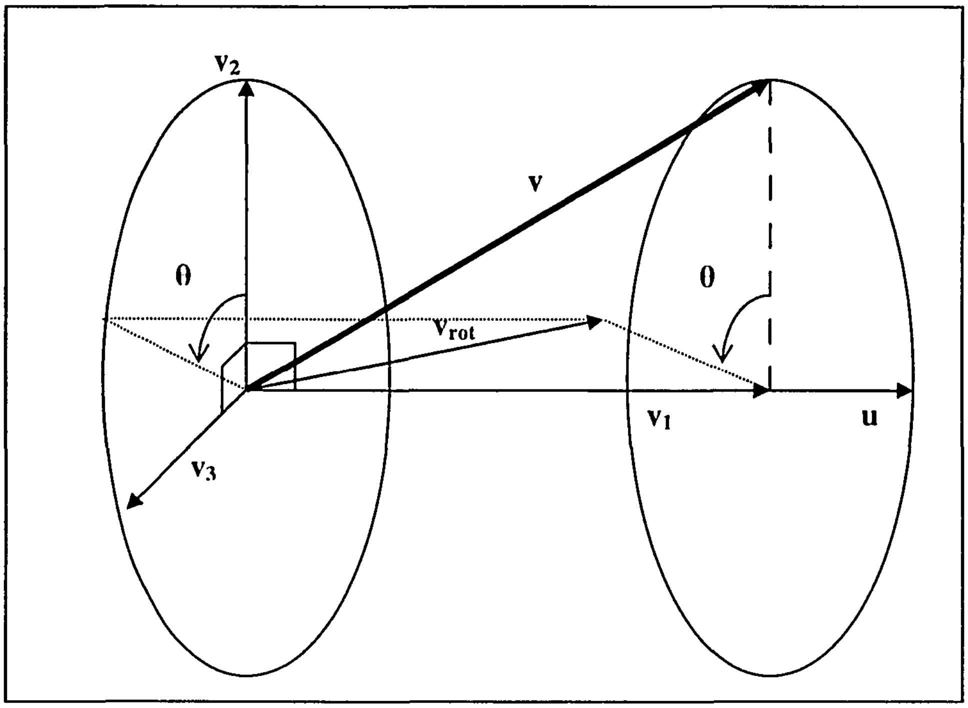


Figure B.2 *Vector v rotated by an angle θ about the vector u.*

This rotation can be represented geometrically as:

$$v_{rot} = v_1 + v_2 \cos\theta + v_3 \sin\theta \qquad \text{or} \tag{B.1}$$

$$v_{rot} = (\mathbf{u} \bullet \mathbf{v}) * \mathbf{u} + (\mathbf{v} - (\mathbf{u} \bullet \mathbf{v}) * \mathbf{u})\cos\theta + (\mathbf{u} \times \mathbf{v})\sin\theta \tag{B.2}$$

Now we derive the same expression for v_{rot} using quaternions. A quaternion can also be represented as follows: $q = q_0 + q_1\hat{i} + q_2\hat{j} + q_3\hat{k} = [s \ \mathrm{v}]$, where $s = q_0$ is a scalar and $\mathrm{v} = [q_1 \ q_2 \ q_3 \ ]$ is a vector. A rotation about a unit vector $\hat{\mathbf{u}}$ by an angle θ can be computed by the quaternion

$$q = \left(\cos\left(\frac{\theta}{2}\right), \hat{\mathbf{u}}\sin\left(\frac{\theta}{2}\right)\right) \qquad \text{B.3}$$

Hence s and v can be represented as

$$s = q_0 = \cos\left(\frac{\theta}{2}\right) \text{ and} \qquad \text{B.4}$$

$$\mathbf{v} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \hat{\mathbf{u}}\sin\frac{\theta}{2} \qquad \text{B.5}$$

After rotation, a point $p = (0, \mathbf{v})$ will be rotated to p_{rot} where $p_{rot} = qpq^{-1}$.

The product of two quaternions $q_1 = (s_1, \mathbf{v}_1)$ and $q_2 = (s_2, \mathbf{v}_2)$ is given by

$$q_1 * q_2 = (s_1 s_2 - \mathbf{v}_1 \bullet \mathbf{v}_2,\ s_1\mathbf{v}_2 + s_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2) \qquad \text{B.6}$$

Expanding $p_{rot} = qpq^{-1}$ we get

$$p_{rot} = \left(\left(\cos\left(\frac{\theta}{2}\right), \hat{\mathbf{u}}\sin\left(\frac{\theta}{2}\right)\right) * (0, \mathbf{v}) * \left(\cos\left(\frac{\theta}{2}\right), -\hat{\mathbf{u}}\sin\left(\frac{\theta}{2}\right)\right)\right) \qquad \text{B.7}$$

$$= \left[\left(\cos\frac{\theta}{2}, \hat{\mathbf{u}}\sin\frac{\theta}{2}\right) * (0, \mathbf{v})\right] * \left(\cos\frac{\theta}{2}, -\hat{\mathbf{u}}\sin\frac{\theta}{2}\right) \qquad \text{B.8}$$

$$= \left[(\hat{\mathbf{u}} \bullet \mathbf{v})(-\sin\frac{\theta}{2}), \mathbf{v}\cos\frac{\theta}{2} + (\hat{\mathbf{u}} \times \mathbf{v})\sin\frac{\theta}{2}\right] * \left(\cos\frac{\theta}{2}, -\hat{\mathbf{u}}\sin\frac{\theta}{2}\right) \qquad \text{B.9}$$

$$= \Big[(\hat{\mathbf{u}} \bullet \mathbf{v})(-\sin\frac{\theta}{2}\cos\frac{\theta}{2}) + (\mathbf{v} \bullet \hat{\mathbf{u}})\sin\frac{\theta}{2}\cos\frac{\theta}{2} - ((\hat{\mathbf{u}} \times \mathbf{v}) \bullet \hat{\mathbf{u}})\sin^2\frac{\theta}{2},$$

$$(\hat{\mathbf{u}} \bullet \mathbf{v})\hat{\mathbf{u}}\sin^2\frac{\theta}{2} + \mathbf{v}\cos^2\frac{\theta}{2} + (\hat{\mathbf{u}} \times \mathbf{v})\sin\frac{\theta}{2}\cos\frac{\theta}{2} - (\mathbf{v} \times \hat{\mathbf{u}})\sin\frac{\theta}{2}\cos\frac{\theta}{2}$$

$$-((\hat{\mathbf{u}} \times \mathbf{v}) \times \mathbf{u})\sin^2\frac{\theta}{2}\Big] \qquad \text{B.10}$$

Now we use some known facts about vectors and trigonometric identities to reduce this equation. Rewriting the above equation we see that the scalar component of p_{rot} reduces to zero. This is shown below.

scalar component:

$$=(\hat{u}\bullet v)(-\sin\frac{\theta}{2}\cos\frac{\theta}{2})+(v\bullet\hat{u})\sin\frac{\theta}{2}\cos\frac{\theta}{2}-((\hat{u}\times v)\bullet\hat{u})\sin^2\frac{\theta}{2} \qquad \text{B.11}$$

$$=(v\bullet\hat{u})(-\sin\frac{\theta}{2}\cos\frac{\theta}{2})+(v\bullet\hat{u})\sin\frac{\theta}{2}\cos\frac{\theta}{2}-((\hat{u}\times v)\bullet\hat{u})\sin^2\frac{\theta}{2} \qquad \text{B.12}$$

$$[\mathit{as}\ \mathbf{u}\bullet v = v\bullet\mathbf{u}]$$

$$=0-((\hat{u}\times v)\bullet\hat{u})\sin^2\frac{\theta}{2} \qquad \text{B.13}$$

$$=0 \quad [(u\times v)\bullet u = 0\ \mathit{as}\ (u\times v)\ \mathit{is\ orthogonal\ to}\ u]$$

Therefore,

$$p_{rot}=\left[0,(\hat{u}\bullet v)\hat{u}\sin^2\frac{\theta}{2}+v\cos^2\frac{\theta}{2}+(\hat{u}\times v)\sin\frac{\theta}{2}\cos\frac{\theta}{2}-(v\times\hat{u})\sin\frac{\theta}{2}\cos\frac{\theta}{2} -(\,(\hat{u}\times v)\times u)\sin^2\frac{\theta}{2}\right] \qquad \text{B.14}$$

$$=\left[0,(\hat{u}\bullet v)\hat{u}\sin^2\frac{\theta}{2}+v\cos^2\frac{\theta}{2}+(\hat{u}\times v)\sin\frac{\theta}{2}\cos\frac{\theta}{2}+(\hat{u}\times v)\sin\frac{\theta}{2}\cos\frac{\theta}{2} -(\,(\hat{u}\times v)\times u)\sin^2\frac{\theta}{2}\right] \quad [\mathit{as}\ v\times\hat{u}=-(\hat{u}\times v)] \qquad \text{B.15}$$

$$=\left[0,(\hat{u}\bullet v)\hat{u}\sin^2\frac{\theta}{2}+v\cos^2\frac{\theta}{2}+(\hat{u}\times v)2\sin\frac{\theta}{2}\cos\frac{\theta}{2}-(\,(\hat{u}\times v)\times u)\sin^2\frac{\theta}{2}\right] \qquad \text{B.16}$$

$$=\left[0,(\hat{\mathbf{u}}\bullet\mathbf{v})\hat{\mathbf{u}}\sin^2\frac{\theta}{2}+\mathbf{v}\cos^2\frac{\theta}{2}+(\hat{\mathbf{u}}\times\mathbf{v})2\sin\frac{\theta}{2}\cos\frac{\theta}{2}-(\mathbf{v}-(\hat{\mathbf{u}}\bullet\mathbf{v})\hat{\mathbf{u}})\sin^2\frac{\theta}{2}\right] \quad \text{B.17}$$

$$\left[as\ (\hat{\mathbf{u}}\times\mathbf{v})\times\hat{\mathbf{u}}=(\mathbf{v}-(\hat{\mathbf{u}}\bullet\mathbf{v})\hat{\mathbf{u}})\right]$$

$$=\left[0,(\hat{\mathbf{u}}\bullet\mathbf{v})\hat{\mathbf{u}}2\sin^2\frac{\theta}{2}+\mathbf{v}(\cos^2\frac{\theta}{2}-\sin^2\frac{\theta}{2})+(\hat{\mathbf{u}}\times\mathbf{v})2\sin\frac{\theta}{2}\cos\frac{\theta}{2}\right] \quad \text{B.18}$$

$$=\left[0,(\hat{\mathbf{u}}\bullet\mathbf{v})\hat{\mathbf{u}}(1-\cos\theta)+\mathbf{v}(\cos\theta)+(\hat{\mathbf{u}}\times\mathbf{v})\sin\theta\right] \quad \text{B.19}$$

$$\left[as\ \cos\theta=1-2\sin^2\frac{\theta}{2},\ \cos\theta=\cos^2\frac{\theta}{2}-\sin^2\frac{\theta}{2}\ \&\ \sin\theta=2\sin\frac{\theta}{2}\cos\frac{\theta}{2}\right]$$

$$=\left[0,(\hat{\mathbf{u}}\bullet\mathbf{v})\hat{\mathbf{u}}+\cos\theta(\mathbf{v}-(\hat{\mathbf{u}}\bullet\mathbf{v})\hat{\mathbf{u}})+(\hat{\mathbf{u}}\times\mathbf{v})\sin\theta\right] \quad \text{B.20}$$

Equation B.20 is the quaternion representation of equation B.2 and therefore $\mathbf{v}_{rot}=p_{rot}$.

From above, we see that the four parameters, q_0, q_1, q_2, q_3 describe a rotation. Since Euler's rotation theorem states that a rotation can be described by only three parameters, a relationship must exist between these four quantities. Therefore,

$$q_0^2+\mathbf{q}\bullet\mathbf{q}=q_0^2+q_1^2+q_2^2+q_3^2=1 \quad \text{B.21}$$

The rotation angle is then related to the Euler parameters by

$$\cos\theta=2q_0^2-1=q_0^2-\mathbf{q}\bullet\mathbf{q}=q_0^2-q_1^2-q_2^2-q_3^2 \quad \text{B.22}$$

$$\hat{\mathbf{u}}\sin\theta=2\mathbf{q}q_0 \quad \text{B.23}$$

The Euler parameters can be written in terms of the Euler angles as

$$q_0 = \cos\left[\frac{1}{2}(\phi + \psi)\right]\cos\left(\frac{\theta}{2}\right) \qquad \text{B.24}$$

$$q_1 = \sin\left[\frac{1}{2}(\phi - \psi)\right]\sin\left(\frac{\theta}{2}\right) \qquad \text{B.25}$$

$$q_2 = \cos\left[\frac{1}{2}(\phi - \psi)\right]\sin\left(\frac{\theta}{2}\right) \qquad \text{B.26}$$

$$q_3 = \sin\left[\frac{1}{2}(\phi + \psi)\right]\cos\left(\frac{\theta}{2}\right) \qquad \text{B.27}$$

According to B.2,

$$\mathbf{v}_{rot} = (\mathbf{u} \bullet \mathbf{v}) * \mathbf{u} \; + (\mathbf{v} - (\mathbf{u} \bullet \mathbf{v}) * \mathbf{u})\cos\theta + (\mathbf{u} \times \mathbf{v})\sin\theta .$$

This can be rewritten as

$$\mathbf{v}_{rot} = \mathbf{v}\cos\theta + \hat{\mathbf{u}}(\hat{\mathbf{u}} \bullet \mathbf{v})(1 - \cos\theta) + (\mathbf{u} \times \mathbf{v})\sin\theta \qquad \text{B.28}$$

In terms of Euler parameters, this can be written as

$$\mathbf{v}_{rot} = \mathbf{v}(q_0^2 - q_1^2 - q_2^2 - q_3^2) + 2\mathbf{q}(\mathbf{q} \bullet \mathbf{v}) + (\mathbf{v} \times \hat{\mathbf{u}})\sin\theta \qquad \text{B.29}$$

and the rotation matrix in terms of the Euler parameters becomes

$$\begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

References

http://www.cs.berkeley.edu/~laura/cs184/quat/quatproof.html.

http://www.cs.berkeley.edu/~laura/cs184/quat/quaternion.html.

Eric W. Weisstein. "Quaternion." From *MathWorld*--A Wolfram Web Resource. http://mathworld.wolfram.com/Quaternion.html.