

Case Study: An Environment for Understanding Protein Simulations Using Game Graphics

Donna Gresh and Frank Suits*
IBM T.J. Watson Research Center

Abstract

We describe a visualization system designed for interactive study of proteins in the field of computational biology. Our system incorporates multiple, custom, three-dimensional and two-dimensional linked views of the protein. We take advantage of modern commodity graphics cards, which are typically designed for games rather than scientific visualization applications, to provide instantaneous brushing and three-dimensional interactivity on standard personal computers. Furthermore, we anticipate the usefulness of game techniques such as bump maps and skinning for scientific applications.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques

Keywords: visualization, proteins, computational chemistry, game graphics, DirectX

1 Introduction

Proteins are the machinery that makes life happen. They are, in simplest terms, a polypeptide chain; that is, a linear sequence of amino acid residues. (When the amino acids join to form a protein, their ends are modified, thus they are referred to as amino acid *residues*). Proteins have the interesting property of self-organization, on a very short time scale, into a particular “native,” or folded configuration, and, in fact, a protein’s functional behavior ultimately depends on this folded configuration, with its particular loops, cavities, and other aspects of its externally presented surface. The consequences of subtle differences in the shape of the protein due to, for example, protein misfolds, can be severe. For example, the deletion of a single amino acid is implicated in the disease cystic fibrosis [5]. Other diseases linked to protein misfolds are Creutzfeldt-Jakob (or mad cow) and Alzheimer’s.

To date, most protein configurations have been determined by experimental measurements, such as x-ray crystallography, electron crystallography, or NMR techniques. However, these processes require that the protein first be synthesized and isolated in measurable quantities. The visualization environment we have developed is intended to facilitate the study of proteins for researchers in the field of computational biology, where the motion and behavior of proteins and other molecules are studied in the computer rather than in the test tube. As computational power and algorithmic sophistication increases, computational biology and molecular dynamics are active areas of research. The goal of our system is to provide an environment where the results of these computations can be interactively studied using a variety of linked two-dimensional and three-dimensional views. Key to the value of the visualization environment is fast, interactive brushing enabled by our use of modern commodity graphics technology. Though the computational demands of molecular dynamics are often extreme, the visualization

requirements are relatively modest, and using modern commodity graphics cards, the system we have developed runs at interactive speed on standard desktop and laptop personal computers.

2 Current Trends in Molecular Dynamics Visualization

Molecular visualization has been a focus of the graphics community for many years, and now a large number of high performance viewing environments are available commercially or freely downloadable from the web[11, 14, 10, 12]. These environments comprise two main types, each with its own design requirements: standalone viewers of fixed molecular structures, and visualization front-ends integrated with a molecular dynamics simulation program. In addition, many viewers allow output of the molecular structure in a format compatible with ray-tracing packages such as POV-Ray[13], allowing non-interactive but photorealistic renderings suitable for publication. Since each viewer has its own requirements for appearance and speed, it must strike a balance between rendering quality and interaction performance based on the needs of the application.

Another factor that drives the design of a molecular visualization system is the intended viewing platform, which may range from a high-end graphics workstation to a web browser running on a laptop. Commercial packages intended for the high-end platforms can assume interactive performance with large amounts of geometry, while simple viewers intended for web browsers must be lightweight and able to create compelling results without too much computation or geometry. Furthermore, each platform may have a variety of application programming interfaces (API’s) for three-dimensional programming, such as OpenGL, Java3D, and DirectX3D. The key factors that determine the API choice are portability among the target platforms and the ability to make optimal use of the graphics capabilities on the target platforms. Viewers intended for the web make additional demands based on functionality within a number of different browsers, and the need for a plug-in.

Molecular visualization is not limited to three-dimensional views; there are a variety of two-dimensional representations that provide insight complementing the spatial view of a molecule. Many viewers provide such plots in addition to the three-dimensional image, along with some degree of interactive coupling between the representations.

The standard three- and two-dimensional views of molecules map well to the functionality provided by graphics API’s since the geometry consists of well-structured triangle strips (for ribbons), repeated glyphs (for ball and stick), or triangle manifolds (for molecular surfaces). However, there are many new features available on graphics cards that are either not supported in OpenGL, or are available only as vendor-specific hardware extensions. As a result, molecular visualization software tends to use only a subset of the capabilities of the graphics hardware, or it has specific code to deal with a variety of hardware possibilities.

*gresh, suits@us.ibm.com, IBM T.J. Watson Research Center, P.O. Box 704 Yorktown Heights, NY 10598

3 The Molecular Visualization System

Our system provides a variety of both two-dimensional and three-dimensional views of proteins, all linked via dynamic, instantaneous brushing. It is programmed in C++, and uses the DirectX®¹ API to gain direct access to the graphics hardware without the need to write hardware-dependent code. We use splitter windows to build the application from its individual views to maximize screen use and to allow the user to easily expand a view at the expense of other views to focus on a particular area of interest.

At IEEE Visualization 2000[8], there was a growing consensus that games are driving graphics card features, and the way to access those features is via DirectX. We were interested in experimenting with novel ways to use these game features in a scientific visualization application. The rationale and some of the implications of this design decision are discussed in Section 3.5.

Figure 2 shows the application with a particular protein, human guanylate binding protein-1, loaded. While our visualization application was developed specifically to study proteins, the viewing infrastructure is generic, and could be easily applied to a variety of visualization applications.

Below we will discuss the various views our system provides; however first it is helpful to define a few of the terms common in protein studies. A protein is, fundamentally, a large molecule, consisting of on the order of a few thousand to many thousand atoms, and a few hundred to a few thousand amino acid residues, joined by peptide bonds. While there are approximately 20 different amino acids, each has in common a central carbon atom (C_α), to which is attached a hydrogen atom, an amino group (NH_2), and a carboxyl group ($COOH$). Each peptide in the chain is essentially a planar unit, that has two degrees of rotational freedom about the *backbone* of the protein, defined by the sequence of C_α atoms. These two degrees of freedom are described by the angles ϕ and ψ .

A protein is also often described in terms of its primary, secondary and tertiary structure. The primary structure is simply the sequence of amino acid residues. The secondary structure is the description of particular motifs into which the protein locally arranges itself. Examples of secondary structure are α -helices and β -sheets, which are coils and aligned runs of the backbone, respectively. Tertiary structure is the folding of distant parts of the sequence into associated structures. (In addition, quaternary structure describes how different peptide chains arrange themselves together.)

3.1 Three-Dimensional Views

A protein is a three-dimensional object, and its three-dimensional shape is critical to its functional behavior. Accordingly, an interactive three-dimensional view of the protein is critical to any visualization environment. Since a protein is, in simple terms, just a large molecule, one might expect a ball-and-stick representation to be typical. While we provide such a view as an option, it is more common to display proteins as ribbons[7]. α -helices and β -sheets are usually shown as coiled and flat ribbons, respectively, which follow the backbone of the protein. We use a variation on an algorithm described by [3] to create the three-dimensional geometry of the ribbon model from the atom positions. This algorithm computes the backbone of the protein, following C_α atoms, and orients the ribbon such that the ribbon normal is always perpendicular to the plane of the peptides. Helices are slightly displaced outward relative to the backbone from their true location for ease of visualization. We represent the ribbon as either a flat sheet or as a thickened slab. (For performance reasons discussed later, we draw the ribbons with rectangular rather than elliptical cross-section.) In addition, we add arrows indicating the ends of the structures with respect to the beginning of the protein chain.

The ribbon can be colored using a variety of color maps, including a structure based map, in which helices, loops, and sheets are each colored differently, and a residue-based map in which each residue is assigned a color using the Shapely scheme (see Figure 3). The left portion of Figure 2 shows our ribbon view, colored by secondary structure type.

3.2 Two-Dimensional Views

In addition to the three-dimensional views of the protein described above, we also present three complementary two-dimensional views.

The first such view is a distance matrix showing the distance between the C_α atoms for all pairs of residues. Points near the diagonal thus represent distances between adjacent residues along the protein backbone. We color the distance matrix using a gray scale color map, which is particularly suited to seeing both the overall distance pattern and high frequency fluctuations in distance. High frequency patterns are often associated with α -helices due to their coiled shape. Since the distance matrix is symmetric about the diagonal line from top left to bottom right, we use the upper portion of the matrix to show, additionally, information on “contacts” of the protein; that is, pairs of residues for which the C_α to C_α distance is less than 10 angstroms, disregarding directly adjacent residues. Pairs of α -helices are marked by blue points and pairs of β -sheets by reddish brown points (these colors match those of the secondary structure colors of the three-dimensional plot). Pairs of mixed α -helices and β -sheets are shown by green points, while pairs that include at least one residue that is part of neither a helix nor a sheet are colored dark gray.

The contact points indicate secondary and tertiary structure. Blue points close to the diagonal represent an α -helix, while red points parallel to the diagonal represent a *parallel* β -sheet (in which the strands of the β sheet are pointing in the same direction), while red points orthogonal to the diagonal represent an *antiparallel* β -sheet (in which the strands are pointing in opposite directions). Colored points further away from the diagonal point to tertiary structure (regions of the protein separated in sequence but close in distance). In the case of proteins for which the secondary structure is not well characterized, one can use the contact plot to directly infer the secondary structure.

We chose the colors in the matrix view to be, first, natural, in the sense of matching the use of color in the three-dimensional plot, and second, easily discriminable. Gray is used for the pairs that include at least one non-helix or non-sheet residue, to downplay their prominence, since most tertiary structure arises from helices and sheets. We are able to use dark gray even though it is superimposed on a gray-scale background because the paired residues highlighted are, by definition, close, thus appearing as very light gray in the background. We present a larger version of this matrix view in Figure 4 so that the colors, patterns, and high frequency behavior can be seen more easily.

The second two-dimensional view is a Ramachandran plot[6], which displays a two-dimensional histogram of the ϕ and ψ angles of the protein. It is standard to plot the angles from -180 to 180° in each dimension, with the result that regions of the two-dimensional Ramachandran plot are associated with particular types of secondary structure: the upper left region with β -sheets, and the central left region with α -helices. We use a blue to yellow color map to indicate the number of residues whose (ϕ, ψ) values fall within each bin of the histogram. Black represents bins into which no residues fall. This view can be seen in the central right portion of Figure 2.

The third two-dimensional view is a plot of ϕ and ψ as a function of residue number. Runs of similar ϕ and ψ are strongly indicative of particular secondary structure motifs. This view is shown on the bottom of Figure 2. We use a special presentation of the ϕ and ψ an-

¹DirectX is a registered trademark of Microsoft Corporation

gles to avoid breaking the β -sheet region across the angle $\psi=180^\circ$ to -180° [2]. Thus the plots on the bottom of Figure 2 start at 0° , continue to 180° , then continue from -180° to 0° . Colors are consistent with the secondary structure colors of the three-dimensional view.

3.3 Interaction and Brushing

The three-dimensional view of the protein is completely interactive, with mouse-driven rotation, panning, and zooming. In addition, the plot of ϕ and ψ vs. residue allows a magnifying “lens” to be dragged over the plot to highlight any desired region. Cursor keys allow the user to step in the ϕ and ψ plots residue by residue. Most importantly, however, all of the views of the protein are linked via interactive brushing, which occurs whenever the mouse simply pauses at a particular point in the frame; that is, no mouse clicks are required. All of the brushing is contextually appropriate, depending on the plot the mouse is residing in and its location in the plot. Thus, for example, if one dwells in the three-dimensional ribbon view, an indication of the residue name and number is drawn at the cursor position, and at the same time, the residue is indicated by a vertical line at the appropriate residue in the ϕ and ψ line plot, by a square around the appropriate histogram bin in the Ramachandran plot, and by an indicator along the diagonal in the distance matrix. Similar behavior results from a dwell in the line plot, or a dwell on the diagonal of the distance matrix. However, since selecting an off-diagonal point in the distance matrix is equivalent to selecting *two* residues, two vertical lines are drawn in the ϕ and ψ plot, the bin locations of both residues are shown in the Ramachandran plot, and both residues are indicated in the three-dimensional plot, along with a line connecting them and a display of the length of the line, representing the distance between the residues. A dwell in the two-dimensional histogram selects all residues within the selected bin and each residue selected is shown in each of the other plots.

Figure 2 shows the state after a dwell in a region indicating a close distance between two α -helix residues which are relatively far apart in sequence, that is, a blue region significantly off of the diagonal. Notice that the three-dimensional view confirms that we are seeing the close juxtaposition of two helices; the region of the two red lines in the ϕ and ψ plots also indicates that we are in the region of helices, with runs of relatively constant ϕ and ψ . Finally, the Ramachandran plot shows both residues have (ϕ, ψ) values in the central left, or helical region.

Picking is also persistent, in that one can select an interesting region in one of the two-dimensional plots, and then rotate the three-dimensional object to get a better view of the selected residue. The pick markers remain until another pick is made.

3.4 Essential Dynamics

Above we have described how our visualization system allows multiple views of a protein. An additional aspect of studying proteins in detail is understanding their motion. Simply displaying multiple time steps of the protein atom positions is not typically conducive to an intuitive understanding of the mechanisms, as the atoms display a large amount of thermal vibration motion. We have incorporated an essential motion implementation [1, 4] to “smooth” the trajectory of the protein in a physically meaningful way.

The method computes the $3N$ by $3N$ covariance matrix of the N atoms (or residues) in the protein. The matrix is diagonalized to obtain the eigenvalues and eigenvectors. One can then select some small number of the largest eigenvalues, and their corresponding eigenvectors. One can use the eigenvectors to displace each atom from its mean position. The result is a smoothed trajectory which captures the essence of the correlated protein motion. In addition, the use of the essential motion eigenvectors can enable significant compression of the information necessary to reconstruct the protein

motion, for example, in web applications. Thus for a 1000 atom system with 2000 time steps, even the use of the fairly large number of 10 eigenvectors allows a 200-fold decrease in information that needs to be transferred.

We have implemented the animation of the motion of the protein such that it is possible to rotate and zoom the object while it is animating. This allows the user to gain a better understanding of a particular region in motion by orienting it in an optimal way. The user can also control the speed of the animation using keyboard keys while rotating the object. This in itself is of great value in contrast to the typical way in which moving proteins are viewed: a simple animation from a particular viewpoint at fixed speed.

3.5 Use of Game Technology

We chose to use DirectX for our three-dimensional views to explore the capabilities of modern commodity graphics cards and their applicability to scientific visualization. There is much debate on the merits of DirectX vs. OpenGL, and our choice in this application was motivated by a combination of research and utilitarian interests rather than superior performance or preferred API design. The immediate users of this application are scientists who tend to use Windows laptops and desktops (even though their simulations are running on large parallel computers), and writing in DirectX allowed one code-base to work on all their personal computers and take advantage of the available hardware without, for example, having to code to vendor-specific OpenGL extensions required for different graphics cards.

Protein simulations are ideally suited to commodity graphics cards due to the relatively small amount of geometry required and the well-defined structure of the triangles and glyphs. For ball-and-stick views we can store single instances of the needed glyphs (a sphere and a cylinder) in memory and use transforms to create the full molecule; we assign colors on the glyphs with small, one-dimensional texture maps corresponding to the possible atom pairs in each bond. For protein ribbons with under 1000 residues we can use compact one-dimensional textures to color the geometry based on residue-dependent values. For both visualization modes, we create vertex buffers corresponding to the ribbons and glyphs with fixed texture coordinates corresponding to the atom or residue of the ball-and-stick or ribbon geometry, respectively. We can thus create the geometry and dynamically change color maps without rewriting to the vertex buffers; we write the geometry and textures into memory once, and apply different color maps by changing the applied texture, not the texture coordinates. DirectX not only allows us to create these texture maps and vertex buffers, but we can assign their location in memory based on size and access needs: Since many of the textures are small and do not have to be read by the CPU after creation, we can place them directly into local video memory on the graphics card.

One technique common to games is vertex blending or “skinning,” which we have used to create a visualization of the molecular surface, shown in Figure 1. Computation of a molecular surface [9] is relatively expensive, and one efficient visual approximation is to draw large spheres around each atom based on its van der Waals radius. This technique is easy to implement but loses much of the surface shape due to the sphere approximation and lack of coupling between separated atoms. We have created the first, to our knowledge, approximation of a molecular surface based on hardware vertex blending, a technique commonly used in games to create realistic motion of a smooth surface covering multiple articulated segments. In our implementation, we first find all atom pairs that are near each other (within two Angstroms) and we use vertex blending to create a stretched surface corresponding to the molecular surface of that pair. Figure 1 shows all such pair surfaces rendered together, and the resulting molecular surface is evident. Although we never

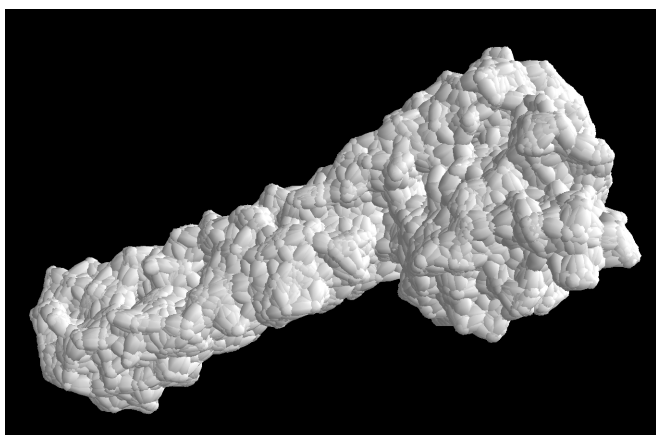


Figure 1: Approximate solvent-accessible surface for protein shown in Figure 2, using DirectX vertex blending.

calculate the actual surface geometry, the image is a good visual compromise between a space-filling spheres and a full surface calculation.

The interactive brushing so crucial to our merged 3D and 2D visualization environment requires fast queries of the geometry under the mouse pointer, and since the protein is already colored using a texture map, we are able to do the query by rendering to a background buffer using a sequentially colored texture that encodes the residue number along the protein. With lighting and shading disabled, the pixel color at the pointer location on the unseen background buffer uniquely identifies the residue without any geometric hit-detection. This relies on direct readback of the unadulterated RGB value from memory, which is provided by DirectX since it allows direct reads from buffers in video memory. Although the bandwidth of such reads may not be optimized for reading large areas, the readback expense is minimal for the one pixel, and we experience no noticeable latency.

4 Conclusions and Future Directions

Simple animations of moving ball-and-stick or ribbon representations are limited in being able to provide insight and intuition about proteins. We have developed an environment where multiple, linked views of the protein allow a user to interactively probe and query the protein to see patterns and relationships. Key to the value of the system we have developed is the ability to generate immediate results to queries that result simply from letting the mouse dwell for a fraction of a second, even in the three-dimensional views. The results of these picks can then be further investigated by zooming or rotating to obtain a more convenient viewpoint. This leads to a very natural environment for investigating relationships.

In today's economy, commodity graphics hardware cards are designed for game applications, not for scientific visualization. While the requirements of scientific visualization don't drive the technology, this doesn't mean that the features of the technology are not useful for scientific applications. While we have taken advantage of a number of the features of the DirectX API in our application, such as vertex buffers and vertex blending, there are a number of features for which we can see potential value. For example, bump maps or procedural textures on generated geometry can indicate other information; as a simple example, the many local properties calculated during a molecular dynamics simulation could be indicated by a bump map whose frequency profile depended on the value of the property. As another example, a large system of atoms could be

represented by point sprites, which are implemented in DirectX to simulate particle systems but also might work well for molecular dynamics simulations.

In summary, there are a number of worthwhile technologies available in the commodity graphics arena that can be of immediate value in scientific visualization applications. The value of interaction in developing intuition and understanding has long been appreciated by the visualization community; the advent of a market for fast graphics (albeit not one driven by scientific needs) has brought the capability of such interactive applications into reach on standard desktop and laptop machines.

Acknowledgements

We would like to acknowledge Julia Valuyeva for her help with the eigenvector computations described in Section 3.4. We also thank James Klosowski for his suggestion of the three-dimensional picking technique. We thank the entire IBM Blue Gene team for their input and advice, and especially thank Mike Pittman for providing trajectory data used for our essential motion computations.

References

- [1] A. Amadei, A. Linssen, and H. Berendsen. Essential dynamics on proteins. *Proteins: Structure, Function, and Genetics*, 17:412–425, 1993.
- [2] O. M. Becker. Representing protein and peptide structures with parallel-coordinates. *Journal of Computational Chemistry*, 18:1893–1902, 1997.
- [3] M. Carson and C. E. Bugg. Algorithm for ribbon models of proteins. *J. Mol. Graphics*, 4:121–122, 1986.
- [4] H. Huitema and R. van Liere. Interactive visualization of protein dynamics. In *Proceedings of IEEE Visualization*, pages 465–468, 2000.
- [5] E. S. B.-H. Qu and P. J. Thomas. Cystic fibrosis: A disease of altered protein folding. *Journal of Bioenergetics and Biomembranes*, 29:483–490, 1997.
- [6] G. N. Ramachandran and V. Sasisekharan. Conformation of polypeptides and proteins. *Adv. Protein Chem.*, 23:283–438, 1968.
- [7] J. S. Richardson. The anatomy and taxonomy of protein structure. *Adv. Protein Chem.*, 34:167, 1981.
- [8] The impact of computer games on scientific and information visualization: If you can't beat them, join them. Panel at IEEE Visualization, 2000. Theresa-Marie Rhyne, Chair.
- [9] A. Varshney, J. Frederick P. Brooks, D. C. Richardson, W. V. Wright, and D. Manocha. Defining, computing, and visualizing molecular interfaces. In *Proceedings of IEEE Visualization*, pages 36–43, 1995.
- [10] kinemage.biochem.duke.edu/website/kinhome.htm.
- [11] www.ks.uiuc.edu/Research/vmd.
- [12] www.mdlchime.com/chime.
- [13] www.povray.org.
- [14] www.proteinexplorer.org.

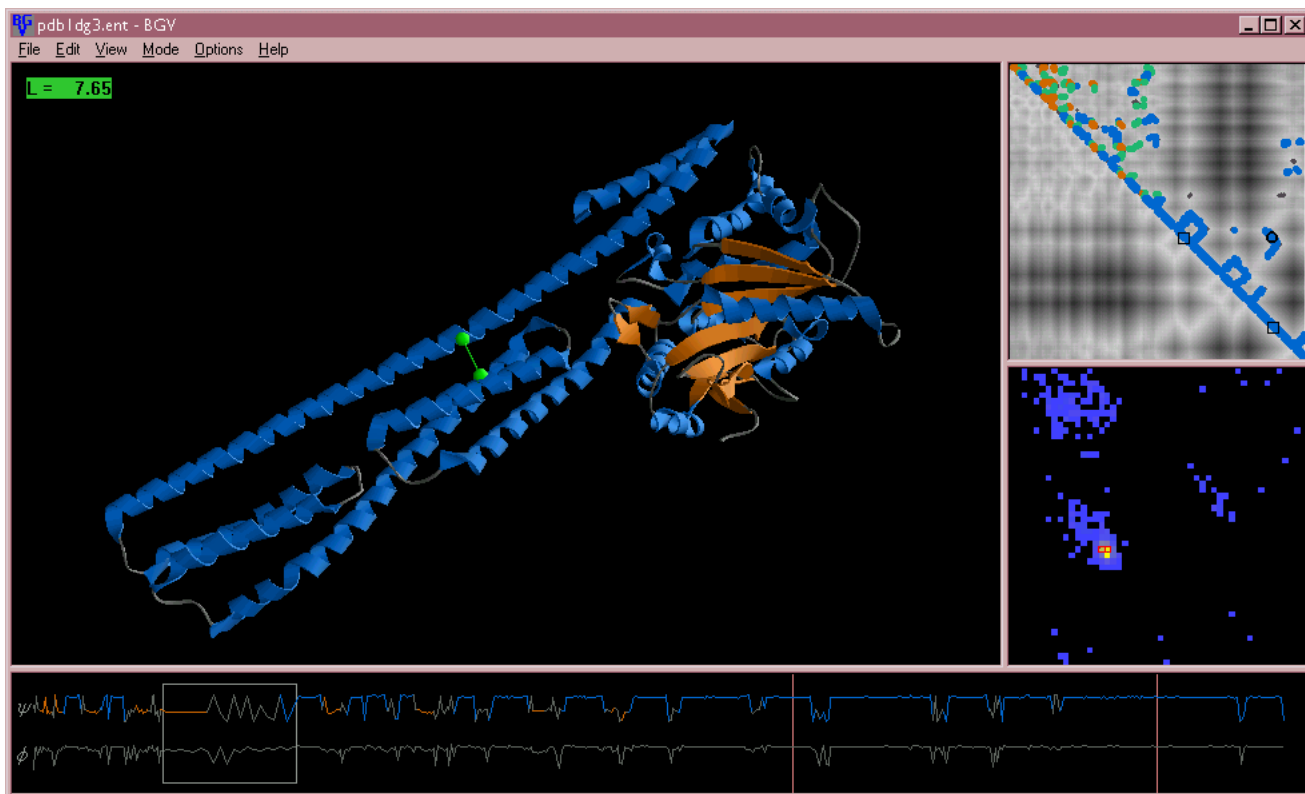


Figure 2: The protein visualization application we have developed. The three-dimensional view displays the protein as a ribbon model, colored by secondary structure type. Top right view shows a distance matrix (described more fully in the caption of Figure 4). The bottom right view shows a two-dimensional histogram of ϕ and ψ , while the plot along the bottom displays ϕ (below) and ψ (above), with a “lens” expanding the view of a particular stretch of residues. This figure captures a pick in the distance correlation plot, which has selected two residues in each view.

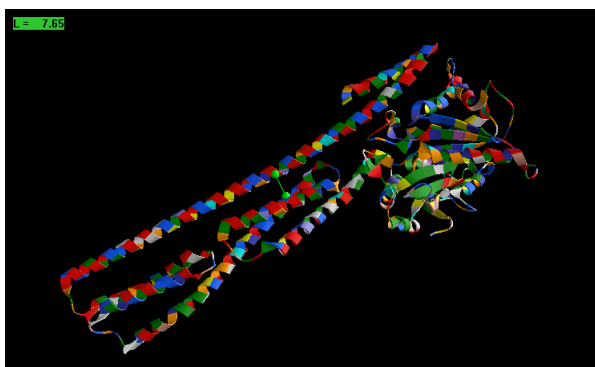


Figure 3: Ribbon diagram of Figure 2, coloring residues individually using the Shapely color scheme. This (as well as the structure-based coloring) is implemented using a one-dimensional texture map.

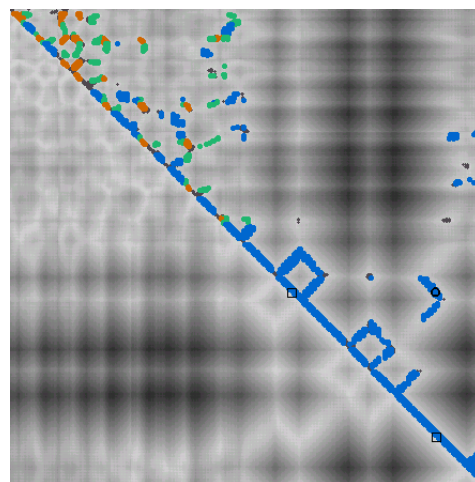


Figure 4: An enlarged view of the distance matrix plot shown in Figure 2. We use a gray scale color map throughout to highlight high-frequency differences in distances, typically indicating α -helical areas. Above the diagonal we additionally encode the secondary structure type whenever two residues lie between 4 and 10 angstroms of one another. Blue indicates a pair of α -helices, reddish-brown a pair of β -sheets, and green a pair of mixed α -helix and β -sheet. Dark gray is used to mark a pair that includes at least one residue which is part of neither a helix nor a sheet.