Interactive Protein Manipulation

Oliver Kreylos* Nelson L. Max* Bernd Hamann* Silvia N. Crivelli† E. Wes Bethel†

Abstract

We describe an interactive visualization and modeling program for the creation of protein structures "from scratch." The input to our program is an amino acid sequence – decoded from a gene – and a sequence of predicted secondary structure types for each amino acid – provided by external structure prediction programs. Our program can be used in the set-up phase of a protein structure prediction process; the structures created with it serve as input for a subsequent global internal energy minimization, or another method of protein structure prediction. Our program supports basic visualization methods for protein structures, interactive manipulation based on inverse kinematics, and visualization guides to aid a user in creating "good" initial structures.

CR Categories: J.3 [Life and Medical Sciences]: Biology and Genetics; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques; I.3.8 [Computer Graphics]: Applications

Keywords: Protein Structure Prediction, Protein Manipulation, Inverse Kinematics, Molecular Modeling, Molecular Visualization, Interactive Visualization

1 Introduction

One of the grand challenges in computational biology is the prediction of the three-dimensional structure of a protein from its chemical composition alone. A protein's *primary structure*, i. e., its amino acid sequence, is directly translated from its messenger RNA (mRNA) sequence, which is transcribed from a DNA sequence in a gene (but perhaps subsequently modified by splicing). The protein's primary structure, however, is purely one-dimensional and does not directly encode a three-dimensional shape. It is commonly believed that the *native* shape of a protein is the one corresponding to the global minimum of its internal energy. Thus, the *protein folding problem* has been treated as an optimization problem in recent years. As with any high-dimensional optimization problem, it is important to start solving the folding problem from a set of "good" initial configurations that allow the optimization code to search the complete parameter space for a minimum. Our work focuses on providing an interactive, visual tool assisting a user to rapidly create many three-dimensional protein structures for a given amino acid sequence. These structures are then used as initial configurations for an optimization algorithm.

1.1 Protein Structure Hierarchy

In biochemistry, protein structure is described at four different levels [1]:

Primary Structure A protein's primary structure is its amino acid sequence. It is directly encoded in a protein's mRNA, with each group of three bases defining one amino acid. The chemical structure of proteins handled by our program is a single chain of amino acid residues connected by peptide bonds, see Figure 1.

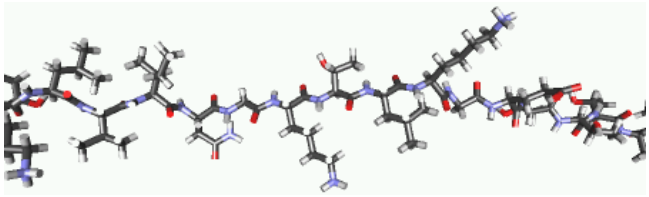

Figure 1: Part of the primary structure of a protein.

As shown in Figure 2, the chemical structure of proteins is highly regular since all amino acids have the same N–C–C *backbone* and differ only in their side chains, denoted in the figure by R. In each amino acid residue, the two interior backbone bonds (N–C and C–C) are single covalent bonds that can rotate around their respective bond axes[1]. Thus, each residue has two rotational degrees of freedom in its backbone – the *dihedral angles* ϕ and ψ. The peptide bond connecting two adjacent residues, though usually drawn as a single covalent bond, has partial double bond character, with the effect that the six atoms C–C=O and H–N–C are always coplanar and arranged in a *trans* configuration.

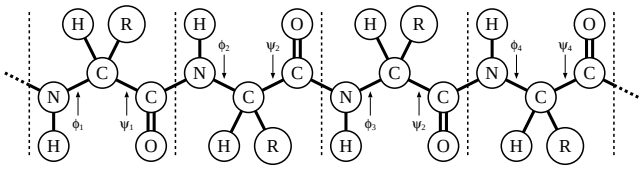

Figure 2: Primary protein structure and rotational degrees of freedom along a residue chain. Adjacent residues are separated by dashed lines. Amino acid side chains are denoted by R.

Secondary Structure Adjacent amino acid residues inside a protein can interact with each other, forming regular substructures: α-helices, see Figure 3, and β-strands. Inside an α-helix, each residue forms hydrogen bonds with two other residues, accounting for the rigidity of the helix. For each amino acid type, the probabilities of forming either one of these structures are known, and neural networks have been used succesfully to predict secondary structure occurrences from amino acid sequences [2]. A third type of secondary structure, a *coil region*, is defined by the absence of any other structure. Since they are highly flexible, we use coil regions as basis for interactive manipulation.

Tertiary Structure A protein's overall three-dimensional structure is the result of interactions between amino acid residues

*Center for Image Processing and Integrated Computing (CIPIC), Department of Computer Science, University of California, One Shields Avenue, Davis, California, {kreylos,max,hamann}@cs.ucdavis.edu

†Lawrence Berkeley National Laboratory, One Cyclotron Road, Berkeley, California, {sncrivelli,ewbethel}@lbl.gov

[1]This degree of freedom does not exist in proline, which forms a ring structure and cannot rotate around the N–C bond.

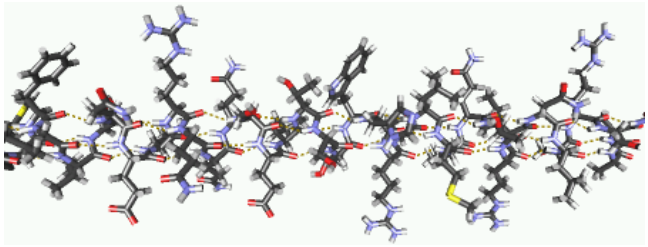

Figure 3: An α-helix. Hydrogen bonds stabilizing the helix are shown as dashed yellow lines.

from distant parts of the chain with each other and the surrounding medium: β-strands, not very rigid by themselves, align to each other to form stable β-sheets, see Figure 4, whereas α-helices cluster with other structures to "hide" hydrophobic amino acid side chains from the surrounding watery solution. It is important to note that β-sheets are stabilized by adjacent β-strands forming hydrogen bonds along the backbone, while the clustering of α-helices is based on interactions between side chains. Prediction of tertiary structure is still an unsolved problem.

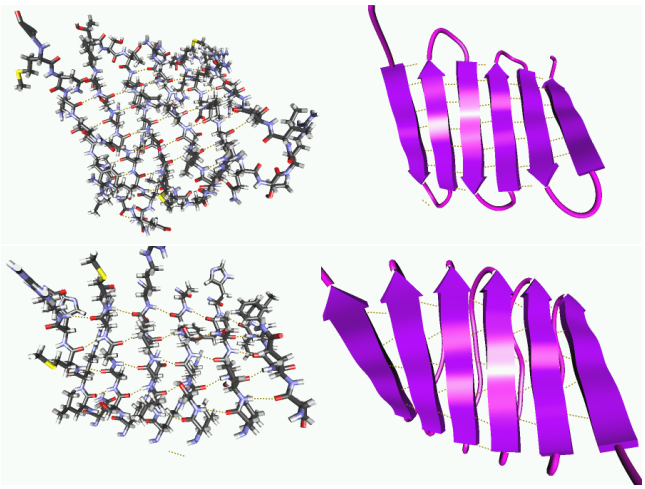

Figure 4: Two β-sheets. Hydrogen bonds stabilizing the sheets are shown as dashed yellow lines. Top row: anti-parallel sheet (left) and cartoon rendering (right). Bottom row: parallel sheet (left) and cartoon rendering (right).

Quaternary Structure Many proteins, e. g., hemoglobin, contain more than one amino acid chain. For those, quaternary structure describes how separate chains interact with each other to define an overall shape. Our interactive manipulation tool, and the internal energy optimization code it interacts with, do not yet consider proteins with more than one chain.

2 Related Work

Existing programs for molecular visualization/interaction fall into two classes: visualization programs that additionally allow rigid body transformations of molecules or molecule parts, and full-fledged molecular dynamics (MD) simulations based on physical and chemical principles. The most commonly used program in the first class is VMD [3], and a popular program in the second class is NAMD [4], developed by the same group. The rendering and visualization techniques used in our program are modelled after those supported by VMD, but the interaction paradigm of our program is entirely different. VMD supports transformations of protein parts, but those parts have to be "cut out" from the rest of the protein before transformation, and VMD does not assist in reconnecting parts afterwards. VMD contains a steering part that allows it to be coupled with MD codes, especially NAMD, but even in combination it does not support real-time manipulation of protein parts due to the complexity of full MD simulations. One such coupled system, Interactive Molecular Dynamics (IMD) [5], combines VMD and NAMD and supports manipulation of molecules by applying forces to single atoms with real-time force feedback. The major advantage of our program over other systems is that ours supports interactive real-time motion of protein parts with respect to each other without breaking the manipulated protein's chemical structure in the process.

3 Protein Structure Visualization

Though the development of our program was driven more by interactive manipulation capability than rendering, it offers several methods for protein visualization that aid a user in creating protein structures. We describe these methods briefly.

3.1 Van-der-Waals Spheres

In this visualization mode, each atom inside a protein is rendered as a sphere, see Figure 5. The radius of a sphere is the van-der-Waals radius of its atom's element [1]. Spheres are colored according to standard usage in chemical visualization programs and textbooks. Van-der-Waals sphere rendering is most useful for judging the compactness of a protein structure. In their native shapes, proteins typically fold tightly, with hardly any empty space between atoms. Van-der-Waals spheres are a good representation of the overall volume of a protein structure, and the amount of empty space inside it.

3.2 Bond Sticks

The relatively large sizes of van-der-Waals spheres leads to high degrees of occlusion, which can make it difficult to understand the chemical structure of a protein in close-up views. To deal with this, our program can render covalent bonds between atoms as "bond sticks," two-color cylinders of fixed radius, where the halves of a bond stick are colored according to the elements of the atoms at the ends, see Figure 5. Bond stick rendering can lead to high visual clutter in overview renderings, but enables a user to judge the exact alignment and chemical makeup of protein structures in close-up views.

3.3 Structure Cartoons

Neither van-der-Waals spheres nor bond sticks are appropriate for modeling purposes since the important visualization of structure alignment is lost due to too much rendered detail, occlusion, and visual clutter. *Structure cartoons* offer a solution by ignoring amino acid residue side chains, and rendering only a protein's backbone. Types of secondary structures are visualized by using different glyphs for each of the three basic structure types: α-helices are rendered as helices of thick ribbons; β-strands are rendered as thick arrows; and coil regions are rendered as cylindrical tubes, see Figure 5. All three glyph types are modeled by non-uniform B-splines [6], with control points located at the central C_α carbon atoms in each amino acid residue. By using B-splines, structure cartoons naturally follow the shape of the structures they visualize.

β-strand arrows intuitively visualize a β-strand's position, orientation, direction, and the four shape parameters described in Section 4.3, while coil region tubes visualize the position, orientation, and "desirability" of coil region conformations.

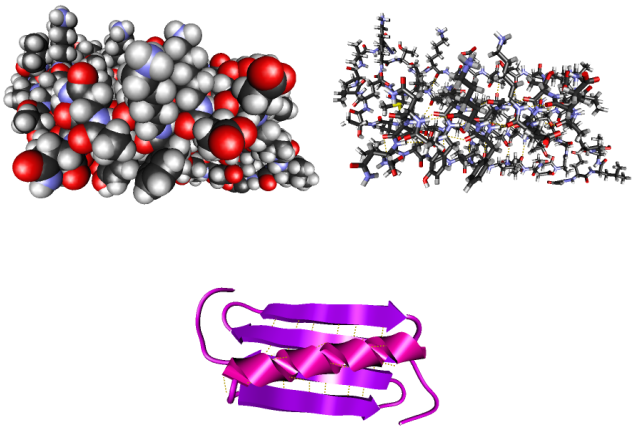

Figure 5: One configuration of protein 1pgx [7] visualized in three different modes, using identical view parameters. Top left: van-der-Waals spheres; top right: bond sticks; bottom: structure cartoons.

4 Protein Structure Creation and Manipulation

Our interactive tool supports the creation of protein structures "from scratch," i.e., solely from a sequence of amino acid residues decoded from a gene, and from a sequence of secondary structure types for each residue, supplied by one or several structure prediction servers [2]. Interactively created structures are intended to be fine-tuned by subsequent optimization. Consequently, our program focuses on creating overall layouts, without much regard for details like the extra torsion angle degrees of freedom within the amino acid side chains, and the interactions between different side chains. More specifically, we concentrate on the creation of backbone-to-backbone hydrogen bonds and β-sheets and let the optimization code take care of α-helix clustering, determination of side chain configurations, and clean-up of side chain interferences.

The process of protein structure creation is done in two major steps. First, the program creates a *pre-configuration* by assembling amino acid residue templates. Second, a user manipulates the created pre-configuration by manually – and semi-automatically – aligning protein substructures to each other. Our program supports several manipulation tools: manual reassignment of structure types to individual residues; global β-strand shape adjustment; interactive manipulation using an inverse kinematics algorithm [8]; semi-automatic β-sheet formation; and β-strand shape optimization. The following sections describe these tools in detail.

4.1 Creating Pre-configurations

Our program creates pre-configurations, i.e., protein structures exhibiting only primary and secondary structure, in a fully automatic process inspired by the working of a ribosome. A ribosome creates a protein guided by messenger RNA (mRNA) transcribed from a gene by successively adding amino acids to a growing chain. In each step, an amino acid of the type prescribed by the mRNA is selected, and concatenated to the (partial) protein by forming a peptide bond with the last amino acid in the chain. It is believed that the protein under construction starts folding into its native shape even before it is completely assembled, but the specifics of this mechanism are still unknown.

The approach underlying our program uses only secondary structure information during protein creation. Given a specific amino acid sequence, a secondary structure prediction server classifies each amino acid as being part of either an α-helix, a β-strand, or a coil region [2]. The server also provides confidence values for each prediction (ranging from 0 – weakest to 9 – strongest), but these confidence values are not yet used by our method. The structures assigned to each residue can be changed interactively during manipulation.

To create a protein, the program assembles amino acid residues one at a time – simulating the action of a ribosome. The position of all atoms making up a single residue, and their connectivity, are defined by a set of template files, one for each residue type. A template file models a residue in "standard configuration" in its own local coordinate system. To concatenate residues, we keep track of an "end-of-chain" transformation describing how to translate local template coordinates to the end of the partial protein.

While adding a new residue, its dihedral angles ϕ and ψ are set to the standard values for α-helix, β-strand, or coil region, respectively, and the end-of-chain transformation is updated accordingly. Thus, proteins are created with secondary structures already fully formed and intact, as opposed to previous approaches that constructed completely unfolded proteins first, and then used constrained energy optimization to form secondary structures [9]. Such optimization typically took several hours, whereas our approach is instantaneous even for large proteins containing hundreds of amino acid residues. A pre-configuration created by our method is shown in Figure 6.

Figure 6: Pre-configuration for protein 1pgx created by assembling amino acid residues and using standard dihedral angles to form secondary structures.

4.2 Structure Type Reassignment

As described above, structure types are predicted for each amino acid residue by one or several available structure prediction servers. One can not always rely on these predictions for the following reasons: different servers often disagree; predicted structure types can be flagged with a low confidence value; interactive manipulation can show that predicted structures do not fit into an otherwise desirable alignment; and global energy optimization sometimes "unravels" predicted structures or attempts to transform one structure type into another.

To address these concerns, our tool enables a user to re-assign structure types during manipulation. Each amino acid residue can individually be set to any of the three basic structure types. Changing a residue's structure type does not immediately set that residue's dihedral angles to the standard angles for the new type; a user has to explicitly request any dihedral angle changes.

Changing structure types dynamically is especially useful for enlarging short coil regions for more freedom during manipulation, or for inserting temporary single-residue structures as additional manipulation "handles" into long coil regions for finer control over a

coil region's behaviour during interaction. This feature can also be used to "touch up" already optimized protein structures by resetting partially unraveled structures to their default shapes.

4.3 β-strand Adjustment

β-strands have a certain amount of flexibility to adjust to their environment in sheets. Coherent shape changes result from changing all ϕ angles by the same increment, or all ψ angles by the same increment. Richardson and Richardson [10] suggest a different basis for this two-dimensional space of coherent shape changes, which has a more comprehensible geometric meaning. *Twist* increments both ϕ and ψ by the same amount, while *pleat* increments all ϕ angles by the same amount, and decrements all ψ angles by this amount.

Since the dihedral angles of a β-strand are close to $180°$, the backbone forms a zig-zag pattern, with the residues on alternating sides. Thus changes in ϕ and ψ of period two also have coherent effects on the strand shape. *Curl*, a period two change also suggested in [10], decreases the ϕ and increases the ψ of odd numbered residues, counting from the beginning of the strand, and does the opposite for even numbered residues. To fill out the four-dimensional space of period two changes to ϕ and ψ, we added a fourth basis element, not described in [10], which we called *braid*. It increases both ϕ and ψ for odd numbered residues, and decreases them for even numbered residues.

These changes were implemented in the form of four dials, which can coherently change the shape of a selected β-strand in the interactive user interface. These dials are useful in adjusting a strand's shape in order to form more hydrogen bonds with adjacent strands in a β-sheet.

4.4 Structure Manipulation

Our program uses inverse kinematics (IK) to transform pieces of a protein with respect to each other, without breaking chemical bonds in the protein backbone between those pieces [8]. As mentioned in Section 1.1, coil regions are flexible because they do not pose tight constraints on their residues' dihedral angles. Thus, they can serve as "buffers" for transformations of protein parts.

To begin manipulation, a user selects a single secondary structure, typically an α-helix or a β-strand. The program then renders the "3D interaction widget," a translucent green box surrounding the selected structure, see Figures 7 and 8. The widget can be translated or rotated by dragging it with the mouse. Additionally, a user activates one or more coil regions that will serve as buffers for subsequent manipulation. In Figures 7 and 8, active coil regions are highlighted in yellow.

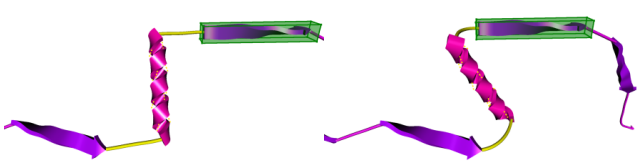

Figure 7: A β-strand has been selected and is surrounded by the 3D interaction widget. The two coil regions surrounding the central α-helix have been activated for manipulation. Left: before dragging the widget; right: after dragging the widget.

In the simpler case, all active coil regions are on the same side of the selected structure (either before it or after it according to chain order). When, for example, all coil regions are before the selected structure, then dragging the interaction widget will transform the selected structure, and the rest of the protein after it, with respect to the part of the protein before the first active coil region. All active coil regions will change shape, and all structures between active coil regions will undergo rigid body transformations, as dictated by the IK algorithm. This mode of interaction allows a user to align protein parts to each other, especially to form β-sheets by manually aligning β-strands.

In the more complex case, where active coil regions exist on both sides of the selected structure, dragging the widget will move the selected structure with respect to the two parts of the protein before and after any active coil regions. Those two parts, even though unrelated according to chain order, will not move with respect to each other. As in the simpler case, shape changes of active coil regions and transformations of intermediate structures are guided by the IK algorithm. This second interaction mode can be used to fine-tune the placement of intermediate parts in an already assembled structure, see Figure 8.

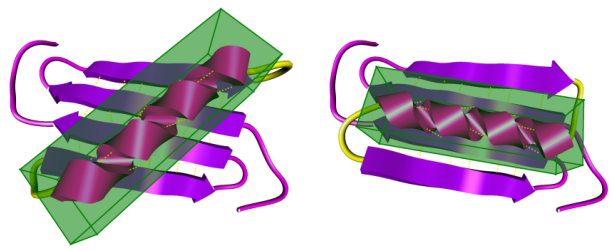

Figure 8: An α-helix has been selected and is surrounded by the 3D interaction widget. Both surrounding coil regions are activated for manipulation. Left: before dragging the widget; right: after dragging the widget.

4.5 Inverse Kinematics

Every rotatable single covalent bond along a protein's backbone can be interpreted, in an IK sense, as a joint with a single axis of unconstrained rotation. After a user selects a structure and activates coil regions, and before manipulation begins, the program constructs a linked assembly by creating two rotational joints for each amino acid residue[2] inside each active coil region. Let us assume that all active coil regions are before the selected structure according to chain order[3]. In this simpler manipulation case, the linked assembly is rooted at the rear end of the last structure before the first active coil region, and the "leaf" joint is connected to the front end of the selected structure, see Figure 9.

Creating a linked assembly in the described way results in intuitive behaviour during manipulation. The selected structure, and the rest of the manipulated protein behind it, are treated as a rigid body and move together following the motion of the interaction widget. If the interaction widget is moved into a position/orientation that cannot be realized by setting dihedral angles for the currently active coil regions, the IK algorithm will automatically approximate the requested position/orientation in a least-squares sense.

A more complex manipulation case occurs when active coil regions are located on both sides of the selected structure. The current version of our manipulation code handles this case by creating two independent linked assemblies: one for all active coil regions before the selected structure, and one for all active coil regions after

[2] In the case of proline, the IK algorithm only creates a single joint since proline has a rigid N–C bond.

[3] The case where all active coil regions are behind the selected structure is symmetrical.

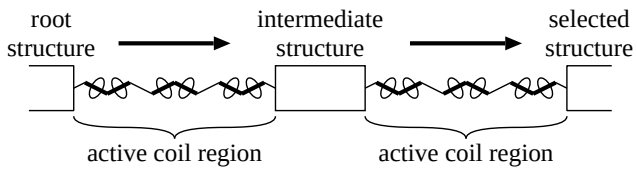

Figure 9: Linked assembly created by IK algorithm from two active coil regions on one side of the selected structure. Each coil region contains three residues, leading to six rotational joints per coil region. The bold arrows denote assembly direction from root to leaf.

the selected structure. As opposed to the simpler case, the direction of assemblies is reversed. The selected coil region serves as root structure for both assemblies, and each of the two leaf joints is connected to the first structure after the last active coil region on either side of the selected structure, see Figure 10.

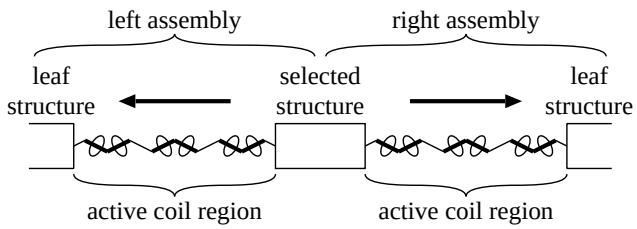

Figure 10: Double linked assembly created by IK algorithm from one active coil region on both sides of the selected structure. Each coil region contains three residues, leading to six rotational joints per coil region. The bold arrows denote assembly direction from root to leaf.

The benefit of creating two linked assemblies in the more complex manipulation case is that it is possible to use the same IK algorithm in both cases. The major drawback is reversed behaviour at the dragging limits. The selected structure will always follow the interaction widget, and if the requested position/orientation can not be realized, the two protein parts behind the assemblies' leaf structures will move relatively to each other to compensate.

The only technical difference between the two manipulation cases is the number of linked assemblies, and their direction according to chain order. The IK algorithm works the same way in both cases. While dragging the interaction widget, the IK algorithm computes the difference between the widget's current requested position and orientation and the position and orientation of the selected structure, computed from the link assembly's current rotation angles. The algorithm updates all joint angles to minimize that difference. Our IK algorithm uses a transposed Jacobian method with force integration [8]. This method has two major benefits: first, it is based on an, albeit simplified, physical model causing intuitive link assembly behaviour during interaction; and, second, its computational efficiency leads to high update rates even with large assemblies.

The major problem with an IK approach to protein manipulation is one of scale. In robotics or animation, typical assemblies have up to a dozen joints, whereas assemblies created by our program can have 80 or more joints for larger proteins. To achieve stable behaviour and high update rates for large proteins, we improved the basic IK algorithm by using a second-order Runge-Kutta method with adaptive step size control for force integration [11].

4.6 Semi-automatic β-sheet Formation

The manipulation process described above is sufficient to create arbitrarily complex protein structures, but the major task of creating β-sheets by forming hydrogen bonds between β-strands can be tedious and time-consuming. To address this problem, our system supports the automatic formation of hydrogen bonds between two user-selected amino acid residues. In the context of β-sheet formation, hydrogen bonds appear in two shapes: parallel and anti-parallel, see Figure 11. In the anti-parallel case, two residues from different β-strands form a double hydrogen bond: one residue's N–H group bonds with the other residue's C=O bond, and vice versa. The parallel case involves three residues: one residue R_i in one strand, and two residues R_{j-1} and R_{j+1} in another strand (the latter two residues being separated by one central residue in the chain). R_i's C=O group bonds with R_{j-1}'s N–H group, and R_i's N–H group bonds with R_{j+1}'s C=O group.

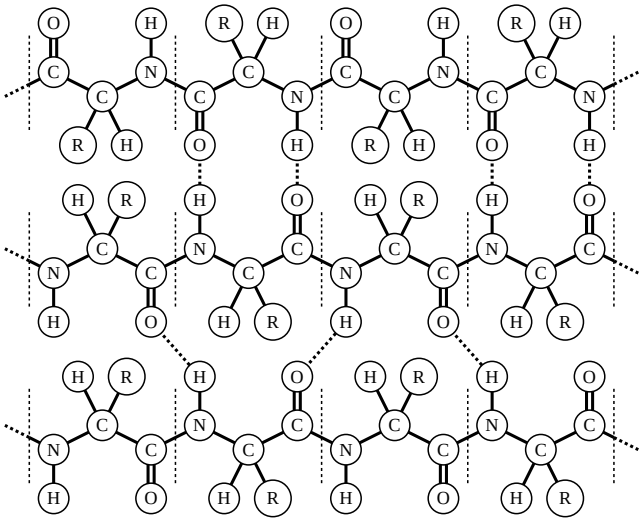

Figure 11: The two shapes of hydrogen bonds between β-strands in a β-sheet. The upper two strands are aligned in an anti-parallel way; the lower two strands are aligned in a parallel way. Note that the direction of the topmost strand is right-to-left in chain order.

Our automatic bonding feature supports both shapes. To invoke it, a user selects a β-strand and activates coil regions as usual, and then selects two residues – the first being inside the selected strand – and a bonding shape. In the anti-parallel case, the program calculates a transformation that moves the selected structure to a position and orientation that will form a double hydrogen bond between the selected residues. In the parallel case, it calculates a transformation to bond the first selected residue with both neighbours of the second one. In either case, the IK algorithm updates dihedral angles in all active coil regions to realize the calculated transformation. This process typically requires a few seconds, and is animated in real time. Once two β-strands have been aligned semi-automatically, it is a matter of seconds to manually fine-tune the alignment to involve more hydrogen bonds.

4.7 β-strand Shape Optimization

In an effort to automate the adjustment of β-strand shape in a β-sheet environment, we tried to define optimal twist, pleat, curl, and braid parameters for β-strands of length four through twelve, in combination with rotated and translated copies of an identically configured adjacent strand at either side in the β-sheet. Since these two adjacent strands can be either in the parallel or anti-parallel

direction, and the two sides of a strand are not equivalent, this gives a total of four cases, each of which extend to a global twisted helicoid-like β-sheet of spiral symmetry, either all parallel, all anti-parallel, or alternating parallel and anti-parallel, with two strands in one direction, followed by two strands in the other.

In forming the two adjacent copies of a central strand, there are three translational degrees of freedom for each, and three rotational degrees of freedom, for a total of twelve. There are also four degrees of freedom for the twist, pleat, curl, and braid parameters. In addition, account for the difference in geometry near the edges of the helicoid, compared to the center, we allowed a quadratic variation in the increments used in these four parameters, which was proportional to the square of the difference between the residue index and the center of the strand. Adding these four quadratic coefficients gives a total of twenty degrees of freedom.

We optimized the Amber energy of the strand [12], counting interactions of atoms within the strand, and with its two adjacent copies in the sheet, using the version of the BFGS non-linear local optimization in [13]. Since only the ϕ and ψ angles were varied in this optimization, and our standard initial configurations of the larger residues were not optimal for β-sheets, we got unfavorable results for the larger residues. Therefore we performed the optimization only for polyalanine, and adjust all the atomic positions for the side chains later in other optimizations once the initial backbone configuration is specified interactively. The optimization results for each length and sheet environment of polyalanine are saved in tables of dihedral angles, which are loaded when requested by the user.

5 Visual Manipulation Guides

To be most useful for protein structure manipulation, our program offers several visualization methods that guide a user towards creating good structures, and decrease the interaction time required to create a structure that is fit for subsequent optimization.

5.1 Hydrogen Bond Visualization

The main task in creating protein structures is the alignment of individual β-strands to form β-sheets – mostly because the optimization algorithm used subsequently has problems creating them automatically. As β-sheets are stabilized by hydrogen bonds between their strands, protein manipulation is mostly used to align β-strands to each other in such a way that the number of hydrogen bonds between them is maximized.

To assist a user in this task, the program offers several alignment guides. The first guide is real-time detection and visualization of backbone hydrogen bonds. As a protein changes shape, the program constantly monitors position and orientation of hydrogen bonding sites along the backbone, and renders a dashed yellow line between all pairs of negatively charged C=O and positively charged N–H groups that satisfy certain constraints. This visualization provides immediate feedback about the quality of the current alignment during interaction, but it does not advise how to change an alignment to improve it.

To guide interaction, the program provides two different ways of rendering potential hydrogen bonds, see Figure 12. First, it can render a *bond site*, i.e., the midpoint of a hypothetical hydrogen bond, for each charged backbone group, showing a potential bond's midpoint position and orientation. Forming bonds therefore is reduced to aligning the midpoints and orientations of two differently charged backbone groups. Alternatively, the program can render a *hydrogen cage* around each backbone N–H group. Hydrogen cages enclose the space a C=O group's oxygen atom must fall into to form a hydrogen bond. Hydrogen cages are more accurate than hydrogen bond sites, but the latter are more appropriate for rapid coarse alignments, and lead to less visual clutter.

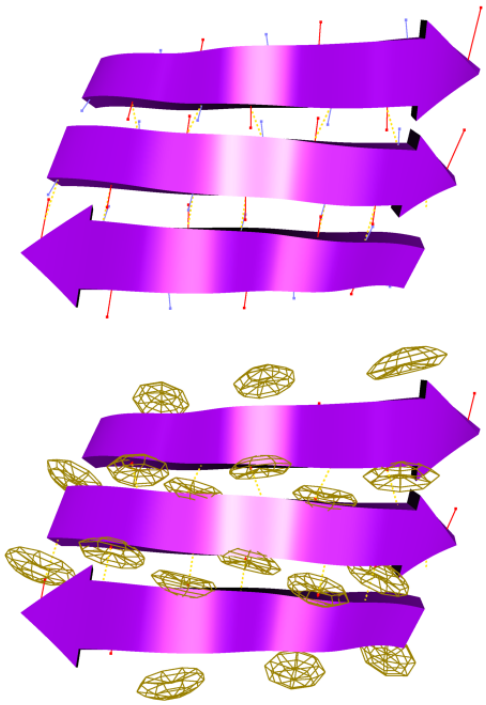

Figure 12: A mixed parallel and anti-parallel β-sheet. Formed hydrogen bonds are shown as dotted yellow lines. Left: hydrogen bond sites for positively charged N–H groups are rendered in blue; those for negatively charged C=O groups are rendered in red. Right: Hydrogen cages are rendered as yellow wireframe cages around N–H groups; C=O groups are rendered in red.

5.2 Ramachandran Plots

The current IK algorithm assumes that a residue's backbone bonds can freely rotate, and that dihedral angles can take any value in the interval $[0, 2\pi)$. In reality, however, not all dihedral angle value combinations are equal. Due to interference of a residue's side chain with its backbone, some values of ϕ and ψ are more favourable than others. The internal energy of a single residue can be treated as a bivariate function of ϕ and ψ, and graphs of this function are called *Ramachandran plots* [1]. Incidentally, some especially low-energy regions in Ramachandran plots correspond to the particular angle combinations that form α-helices and standard β-strands. Our program visualizes the dihedral angles of active coil regions in the style of Ramachandran plots to help a user evaluating the "naturalness" of a configuration created by the IK algorithm.

5.3 Visualization of Atom Collisions

Another aspect not considered by the IK algorithm is global interference of atoms inside a protein. Ignoring atom collisions during manipulation gives a user more freedom to rapidly create "good" structures and to move from one structure to another one. However, before a protein structure can be used as initial configuration for optimization, atom intersections that are too "deep" to be automatically cleaned up by the optimization code must be resolved manually.

To assist a user in this task, the program calculates and visualizes atom intersections in real-time during manipulation. We use a simple grid-based algorithm to quickly find all pairs of atoms whose van-der-Waals-spheres intersect deeper than some threshold value. This algorithm is efficient enough for real-time collision detection, even for large proteins consisting of thousands of atoms. To visualize intersections, the program renders red spheres of radii proportional to the depth of an intersection at the midpoint between two intersecting atoms, see Figure 13.

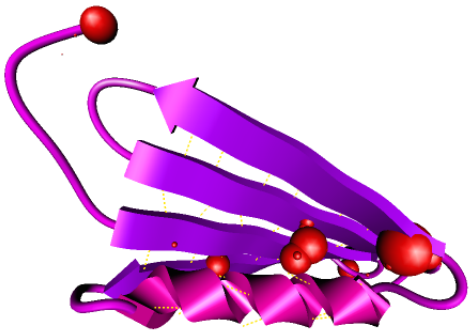

Figure 13: Protein structure where the central α-helix is too close to the β-sheet; parts of the helix' side chains interfere with the sheet. The collision spheres are updated in real-time during manipulation.

Figure 14: Protein structure consisting of several α-helices grouped around a central β-barrel, i.e., a closed β-sheet. The structure was created from scratch using an artificial amino acid sequence and a custom secondary structure type sequence.

6 Conclusions and Future Work

We have described a program that allows a user to interactively create protein structures from "scratch," i.e., from an amino acid sequence, by using simple geometric constructions and inverse Kinematics, see Figures 14 and 15. Structures created with this program are meant to be refined by subsequent global energy optimization, or another method of protein structure prediction, see Figure 16. Our collaborators from the Lawrence Berkeley National Laboratory, the University of California at Berkeley, and the University of Colorado at Boulder have used the program to create initial configurations for several large proteins during preparation for the Fifth Meeting on the Critical Assessment of Techniques for Protein Structure Prediction (CASP5) in Summer 2002 [14], see Figures 15 and 16.

Our current research focuses on improving the IK algorithm to better handle bidirectional manipulation, see Section 4.5, and to optionally consider the "desirability" of (ϕ, ψ) angle pairs to find better coil region conformations, see Section 5.2. Furthermore, we are adding a steering component that enables to use our program as a monitoring and steering front-end for protein structure prediction.

7 Acknowledgments

This work was supported by the U.S. Department of Energy under contract DE–AC03–76SF00098, the National Science Foundation under contract ACI 9624034 (CAREER Award), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, through the National Partnership for Advanced Computational Infrastructure (NPACI) and a large Information Technology Research (ITR) grant; and the National Institutes of Health under contract P20 MH60975–06A2, funded by the National Institute of Mental Health and the National Science Foundation. We thank the members of the Visualization and Graphics Research Group at the Center for Image Processing and Integrated

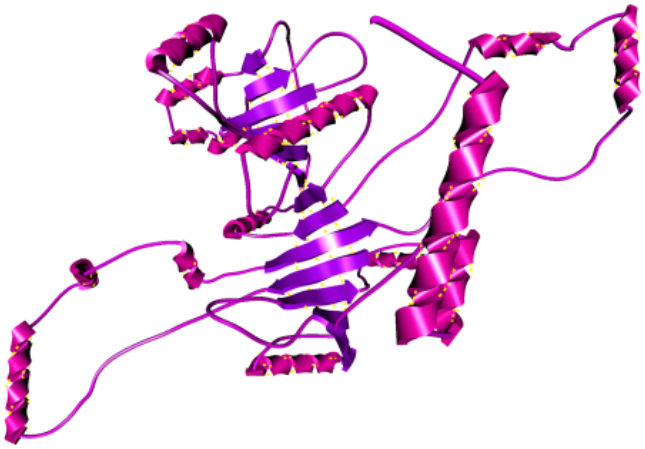

Figure 15: One initial structure for protein T187, one of the targets from the CASP5 protein structure prediction competition. We focused on the central anti-parallel β-sheet and did not attempt to cluster the surrounding α-helices into a compact shape.

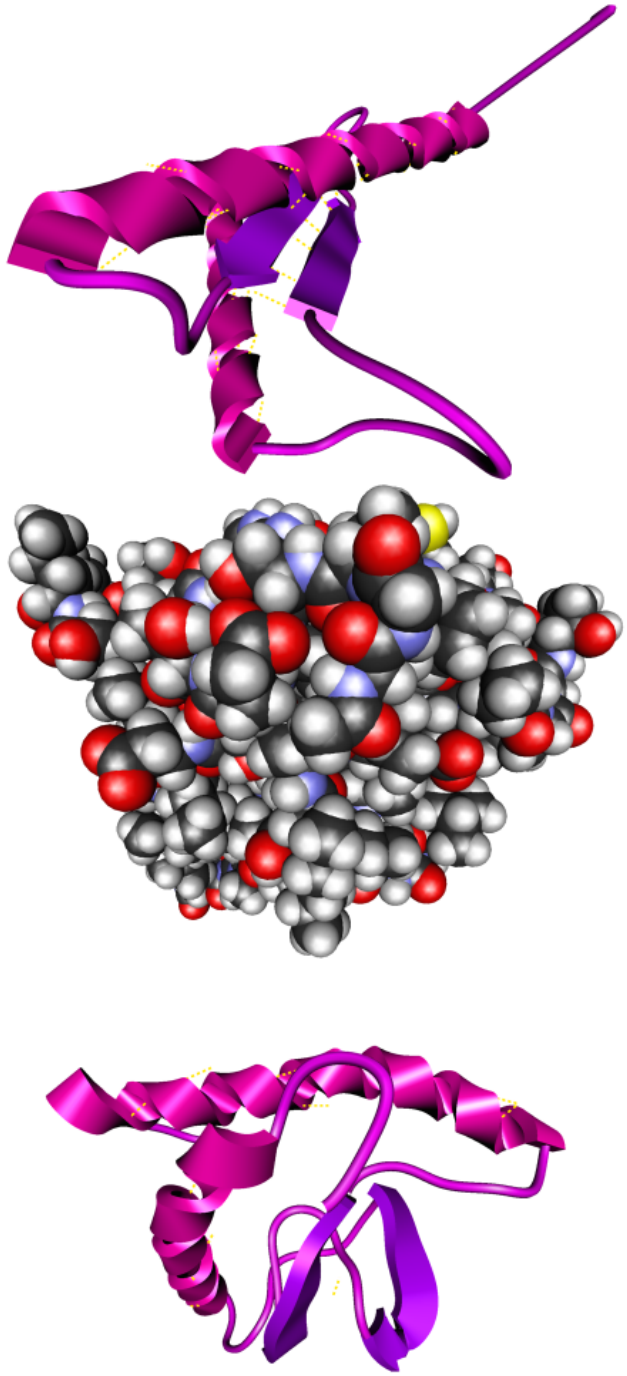

Figure 16: Protein T139, another CASP5 target. Top: an initial structure for T139 created from scratch. Middle and bottom: two renderings of a candidate configuration for T139 returned by the global energy optimization process. Note the compactness of the returned configuration, and how α-helices and β-strands bend and twist to form a compact shape.

Computing (CIPIC) at the University of California, Davis and the Visualization Group at the Lawrence Berkeley National Laboratory. We thank Prof. Head-Gordon, Prof. Byrd, Prof. Schnabel, and Elizabeth Eskow for their helpful comments.

References

[1] Lehninger, A. L., Nelson, D. L. and Cox, M. M., *Principles of Biochemistry*, 2nd ed. (1993), Worth, New York, New York

[2] McGuffin, L. J., et al., *PSIPRED: a Protein Structure Prediction Server*, `http://www.psipred.net`

[3] Humphrey, W., Dalke, A. and Schulten, K., *VMD – Visual Molecular Dynamics*, Journal of Molecular Graphics 14, pp. 33–38

[4] Nelson, M. T., Humphrey, W. F., Gursoy, A., Dalke, A., Laxmikant, V. K., Skeel, R. D. and Schulten, K., *NAMD: A Parallel Object-Oriented Molecular Dynamics Program*, International Journal of Supercomputer Applications and High Performance Computing 10(4) (1996), pp. 251–268

[5] Stone, J. E., Gullingsrud, J. and Schulten, K., *A System for Interactive Molecular Dynamics Simulation*, 2001 ACM Symposium on Interactive 3D Graphics, ACM SIGGRAPH (2001), pp. 191–194

[6] Farin, G., *Curves and Surfaces for Computer Aided Geometric Design*, 3rd ed. (1993), Academic Press, Boston, Massachusetts

[7] *The Protein Data Bank*, `http://www.pdb.org`

[8] Welman, C., *Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation*, Master's Thesis, Simon Fraser University, Vancouver, Canada, 1993

[9] Crivelli, S., Eskow, E., Bader, B., Lamberti, V., Byrd, R., Schnabel, R. and Head-Gordon, T., *A Physical Approach to Protein Structure Prediction*, Biophysical Journal 82 (2002), pp. 36–49

[10] Richardson, J. S. and Richardson, D. C., *Principles and Patterns of Protein Conformation*, in: Fassman, G. D., ed., *Prediction of Protein Structure and the Principles of Protein Conformation*, Plenum Press, New York, New York, 1989

[11] Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. *Numerical Recipes in C*, 2nd ed. (1992), Cambridge University Press, Cambridge, Massachusetts

[12] Cornell, W. D., Cieplak, P., Bayly, I., Gould, I. R., Merz, K. M., Ferguson, D. M., Spellmeyer, D. C., Fox, T., Caldwell, J. W. and Kollman, P. A., *A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids and Organic Molecules*, Journal of the American Chemical Society 117 (1995), pp. 5179–5197

[13] Liu, D. C. and Nocedal, J., *On the Limited Memory BFGS Method for Large Scale Optimization Methods*, Mathematical Programming 45 (1989)

[14] Head-Gordon, T., Crivelli, S., Kreylos, O., Eskow, B., Choi, H., Byrd, R. and Schnabel, R., *A Physical Approach to Protein Structure Prediction*, in: Moult, J., Fidelis, K., Zemla, A. and Hubbard, T., eds., *Proceedings of CASP5 – Fifth Meeting on the Critical Assessment of Techniques for Protein Structure Prediction*, Pacific Grove, California, December 1–5, 2002, pp. A76–A78