GoMol: A 3D Protein Analysis & Visualization Tool

Tyler Katz, Darin Boyes, Minhyek Jeon, Shivank Sadasivan
Carnegie Mellon University
02-601: Programming for Scientists
Dr. Phillip Compeau
15 December 2023

GoMol: A 3D Protein Analysis & Visualization tool

Introduction

In 1957, Francis Crick described the central dogma of biology as a framework in which biological information flows from DNA to RNA to proteins. Proteins are large, complex molecules that are responsible for much of the complexity of life (*Central Dogma of Molecular Biology*, 2023). 20 types of amino acids, encoded in DNA and RNA, serve as the building blocks of proteins, connecting in long chains and folding in a specific manner that gives rise to its function. The primary structure of a protein includes the sequence of amino acids that comprise the protein and the secondary structure confers the local folding within a protein due to backbone atom interactions, with the most common example being α-helices and β-sheets. The tertiary structure thus confers the overall three-dimensional structure of the protein and the quaternary structure involves the conglomeration of multiple protein subunits (*What Are Proteins and What Do They Do*, 2021).

Proteins play important roles in the structure, function, and regulation of biological processes. Disease can thus be thought of as a disruption in the structure or function of these biological processes on a molecular or cellular level (*Biology of Disease*, 2023). Understanding the mechanisms behind protein assembly and function is a crucial step in controlling biological processes and developing treatments for disease. With an ever increasing amount of biological data being generated from advancing experimental technologies, the field of computational biology has aimed to apply computational techniques to extract insight from these vast data sets. The ability to predict protein structures and efficiently analyze and visualize them is therefore, an important area of research, as the time consuming and expensive process of designing small molecule drugs for a protein target can be made more efficient with computational tools. The comparison of proteins in meaningful ways is an interesting problem facing this field, as insight can be extracted and applied from one protein to another similar protein. Additionally, recent applications of artificial intelligence and generative technologies have created a need to compare computationally predicted protein structures to experimentally determined structural data.

The RCSB Protein Data Bank includes structural data of proteins for over 200,000 experimentally determined structures and over 1,000,000 computed structure models from Alpha Fold DB and Model Archive (*Protein Data Bank*, 2023). To address this need for computational tools to compare and visualize proteins, our group has developed GoMol, a software tool, written

in Go, that can aid scientists in aligning primary structures of proteins, aligning three-dimensional structures of similar proteins, and visualizing the three-dimensional structure of proteins from publicly available PDB data.

Project Components

1. Sequence Alignment

Sequence alignment is a fundamental technique in bioinformatics and molecular biology, crucial for understanding the relationships and functional characteristics of proteins. When dealing with proteins that are evolutionarily related and share a common ancestor, it's essential to analyze their similarities and differences at the sequence level. This process provides valuable insights into the conserved regions, which often play critical roles in biological functions and can be targets for therapeutic interventions. The Needleman-Wunsch algorithm, implemented in Go for this project, is an instrumental tool for this purpose.

The Needleman-Wunsch algorithm, developed by Saul B. Needleman and Christian D. Wunsch, is a classic example of a global sequence alignment method. Unlike local alignment algorithms that find the best matching subsegments, the Needleman-Wunsch algorithm aligns entire sequences from beginning to end, making it ideal for sequences of similar length and overall similarity. The algorithm employs dynamic programming, a method that solves complex problems by breaking them down into simpler subproblems. In the context of sequence alignment, dynamic programming is used to systematically build an alignment matrix, where each cell represents an optimal score for aligning subsequences up to that point.

Dynamic programming is integral to the Needleman-Wunsch algorithm. It ensures efficiency and accuracy, enabling the algorithm to run with a time complexity of O(nm), where m and n are the lengths of the two sequences being aligned. The Needleman-Wunsch algorithm begins by creating a scoring matrix. If we have two sequences to align, one along the rows and the other along the columns, the first step is to initialize the first row and column with gap penalties. This initialization facilitates the alignment of the first few characters of each sequence with gaps.

The score of any cell $C(i,j)$ is the maximum of:

$$\begin{aligned} q_{diag} &= C(i-1,j-1) + S(i,j) \\ q_{up} &= C(i-1,j) + g \\ q_{left} &= C(i,j-1) + g \end{aligned}$$

where $S(i,j)$ is the substitution score for letters i and j, and g is the gap penalty.

Fig1a: Scoring system

Source: The needleman-wunsch algorithm for sequence alignment - SJSU. (n.d.). https://www.cs.sjsu.edu/~aid/cs152/NeedlemanWunsch.pdf

The algorithm relies on a scoring system, typically using a substitution matrix like BLOSUM62 for proteins. This matrix provides scores for matching and mismatching each pair of amino acids. There are also penalties for introducing gaps (insertions or deletions), which are subtracted from the score.

BLOSUM62 substitution matrix

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Fig 1b.: BLOSUM 62 Matrix

Source: The needleman-wunsch algorithm for sequence alignment - SJSU. (n.d.). https://www.cs.sjsu.edu/~aid/cs152/NeedlemanWunsch.pdf

Each cell in the matrix (beyond the initialized first row and column) is calculated based on the highest score derived from three possibilities:

Diagonal move (Match/Mismatch): Adding the score of matching or mismatching the current pair of amino acids to the score of the top-left diagonal cell.

Up move (Gap in one sequence): Adding the gap penalty to the score directly above.

Left move (Gap in the other sequence): Adding the gap penalty to the score directly to the left.

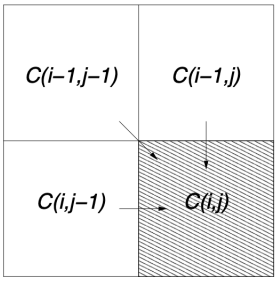

Fig 1c.: Filling cells in the matrix

Source: The needleman-wunsch algorithm for sequence alignment - SJSU. (n.d.). https://www.cs.sjsu.edu/~aid/cs152/NeedlemanWunsch.pdf

Once the matrix is fully populated, the alignment is determined by tracing back from the bottom-right cell to the top-left cell. The path chosen at each step depends on which of the three possible moves (diagonal, up, left) was used to compute the cell's score. This process constructs the optimal alignment.

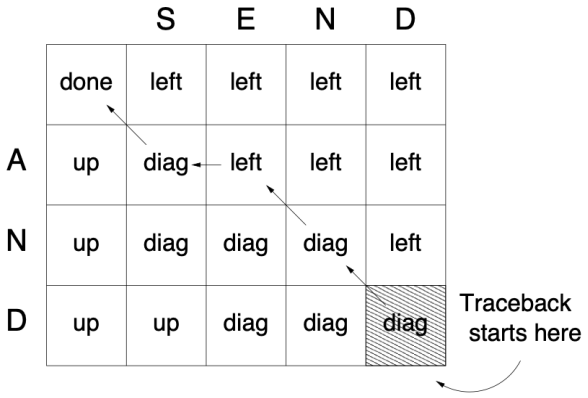

Fig1d.:Tracing back

Source: The needleman-wunsch algorithm for sequence alignment - SJSU. (n.d.). https://www.cs.sjsu.edu/~aid/cs152/NeedlemanWunsch.pdf

The alignment generated by the Needleman-Wunsch algorithm is not just a string of sequences; it's a map that reveals the evolutionary story of the proteins. The aligned sequences highlight conserved regions, crucial for understanding molecular function and evolutionary relationships. In the context of this project, the aligned sequences serve as a basis for further analysis, including 3D structural alignment and visualization. By integrating sequence alignment with structural analysis, we gain a comprehensive view of the protein's characteristics - not only how they are similar or different at the sequence level but also how these differences manifest in their three-dimensional structures. This holistic approach is particularly valuable in drug discovery and development, where understanding both the sequence and the structure of proteins is key to identifying effective therapeutic targets. The Needleman-Wunsch algorithm thus plays a pivotal role in this integrative analysis, laying the groundwork for deeper insights into the functional and structural aspects of proteins.

2. 3-D Structural Alignment

To accomplish the goal of comparing the three-dimensional structures of two similar proteins, our team incorporated the Kabsch and Qres algorithms into GoMol. The Kabsch algorithm leverages principles of linear algebra to calculate an optimal rotation matrix. This matrix minimizes the root mean squared deviation (RMSD) between two sets of point coordinates (*Kabsch Algorithm*, 2023). This algorithm gives GoMol the ability to rotate the coordinates of one protein, aligning them as closely as possible with the second protein's structure, based on data extracted from their respective PDB files. To implement this algorithm in Go, we utilized the gonum/mat package, a Go library useful for matrix computation capabilities. The algorithm initiates with the computation of the covariance matrix, H, by multiplying one coordinate set with the transpose of the other. Following this, singular value decomposition is performed on the covariance matrix, represented as

$$H = U\Sigma V^T$$

This enables the calculation of the optimal rotation matrix as the product of matrices U and V^T (Vertrees, 2019).

Conceptually, the Kabsch algorithm interprets the covariance matrix as a quantifiable difference between the point sets, deconstructing this difference through singular value decomposition to

yield the rotation matrix. With this mathematical alignment achieved, minimizing RMSD, GoMol can then visually render the proteins in a comparative format. This visualization highlights structural variations, which may not be intuitive when examining the proteins from a different perspective.

To enhance our comparative analysis of protein structures in GoMol, we next implemented the Qres algorithm for a more localized approach, shifting from the global approach of the Kabsch algorithm. The Qres algorithm synthesizes spatial information from contact maps to produce a similarity score at each amino acid position (Compeau, 2022). Contact maps represent the proximity of all alpha carbon pairs within their three-dimensional structure as a matrix. The premise of the Qres metric is to encapsulate both the spatial and sequential aspects of protein structure in a single numerical score. This approach offers a more detailed perspective, enabling us to discern and quantify structural similarities and differences at the amino acid level. The metric ranges from 0, or no structural similarity, to 1, or high structural similarity. The equation for the Qres metric is as follows:

$$Q^{(i)}_{res} = \frac{1}{N-k} \sum_{j \neq i-1,i,i+1}^{N} exp[-\frac{[d(s_i, s_j) - d(t_i, t_j)]^2}{2\,\sigma^2_{ij}}]$$

3. Visualization

Our group has opted for a simple, but robust approach to visualizing proteins in three dimensions. Understanding how to go about modeling objects in three dimensions has been a long-standing issue in computer graphics, and here we make use of some of the most common strategies that have been used in the rendering of 3D objects for decades.

Basics of Ray-tracing

The main rendering technique that we have implemented in GoMol is ray tracing, where we try to simulate the way that light interacts with objects in a virtual environment. It works by first specifying a camera, which will serve as the point of origin which all rays will be cast from. The direction of each ray is specified by spaces representing individual pixels in a viewport, where all of the area within the viewport is visible to the user. We can imagine this as a camera being located behind a viewport, and being pointed at the center of mass of our protein of interest, as shown in Figure 3a.

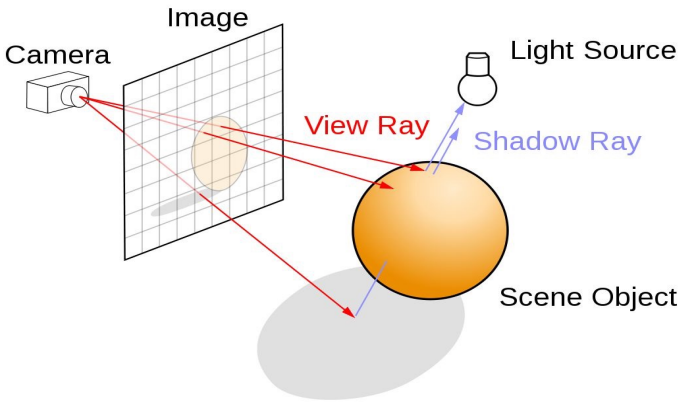

Figure 3a. Ray-tracing Diagram.

We can also think of each pixel on our screen as having a ray casted through it from the camera and interacting with objects in the scene. Then, we define a light source as a point light, which emits light uniformly in all directions. This piece will be important when rendering the proteins, because the color of individual atoms in the protein will be affected by factors such as distance from the light source, the light intensity, and various attenuation coefficients.

Reading in information from PDB

Finally, we specify our object, which will be all atoms of a protein loaded from a PDB file. Each atom in a PDB file has a significant amount of information associated with it including its index, element, amino acid, amino acid index, and x, y, and z coordinates. We make use of this information, in addition to information about the Pauling radii of individual elements to produce an accurate representation of the protein in 3-dimensional space. An example PDB file can be found in Figure 3b.

```
ATOM      1  N   VAL A   1      10.720  19.523   6.163
ATOM      2  CA  VAL A   1      10.228  20.761   6.807
ATOM      3  C   VAL A   1       8.705  20.714   6.878
ATOM      4  O   VAL A   1       8.164  20.005   6.015
ATOM      5  CB  VAL A   1      10.602  22.000   5.966
ATOM      6  CG1 VAL A   1      10.307  23.296   6.700
ATOM      7  CG2 VAL A   1      12.065  21.951   5.544
```

Figure 3b. Example PDB file.

The categories in a PDB file can be read from left to right as follows: label, atom number, atom element, associated amino acid, chain number, amino acid sequence number, then x, y, and z coordinates.

Ray-Sphere Collision Math

To accomplish the rendering of the protein, we use the previously described ray tracing technique to cast rays into the scene and check for collisions with any of the atoms in the protein. This can be done using very simple vector math. To accomplish this, we first have to define the notion of a "ray". A ray can be defined as a function P(t), where P(t) = A + tB. A is the origin point of the ray represented in x, y, and z coordinates, and B is a direction vector, represented using the same coordinate system. t is a real number, and increasing or decreasing t moves a point along this ray. Positive t values indicate points in front of A, whereas negative t values indicate points behind A.

Once we have a notion of a ray, we then need to define a sphere, which can be done with the equation in Figure 3c.

$$x^2 + y^2 + z^2 = r^2$$

Figure 3c. Equation of a sphere in 3-dimensional space with center at the origin.

We can then extend this equation to a sphere that contains a center at an arbitrary point that is not the origin using the equation found in Figure 3d.

$$(x - C_x)^2 + (y - C_y)^2 + (z - C_z)^2 = r^2$$

Figure 3d. Equation of a sphere in 3-dimensional space with center at arbitrary location.

After this, we can define r to be an arbitrary point on the sphere P minus the center of the sphere C. Using this definition, we get the following equation in Figure 3e, then we can derive the equation in Figure 3f.

$$(\mathbf{P} - \mathbf{C}) \cdot (\mathbf{P} - \mathbf{C}) = (x - C_x)^2 + (y - C_y)^2 + (z - C_z)^2$$

Figure 3e. Modifying equation to represent radius as point on sphere minus center.

$$(\mathbf{P} - \mathbf{C}) \cdot (\mathbf{P} - \mathbf{C}) = r^2$$

Figure 3f. Setting point minus center equation equal to radius.

Then, each point on the sphere can be defined as a point of collision between the ray and the sphere. This, in conjunction with the equation for a ray P(t) = A + tB, creates the following equation in Figure 3g.

$$((\mathbf{A} + t\mathbf{b}) - \mathbf{C}) \cdot ((\mathbf{A} + t\mathbf{b}) - \mathbf{C}) = r^2$$

Figure 3g. Substituting point for ray equation.

Some simple arithmetic leads to the following equation in Figure 3h.

$$t^2\mathbf{b} \cdot \mathbf{b} + 2t\mathbf{b} \cdot (\mathbf{A} - \mathbf{C}) + (\mathbf{A} - \mathbf{C}) \cdot (\mathbf{A} - \mathbf{C}) - r^2 = 0$$

Figure 3h. Arithmetic to transform to quadratic form.

Because we are left with a simple quadratic equation, we now have the coefficients shown in Figure 3i.

$$a = \mathbf{b} \cdot \mathbf{b}$$

$$b = 2\mathbf{b} \cdot (\mathbf{A} - \mathbf{C})$$

$$c = (\mathbf{A} - \mathbf{C}) \cdot (\mathbf{A} - \mathbf{C}) - r^2$$

Figure 3i. Coefficients for quadratic formula.

Using these coefficients, we can then solve the quadratic equation in Figure 3j, in which all solutions to the quadratic equation will indicate that the ray has an intersection point with the sphere.

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Figure 3j. Quadratic formula.

To sum up the ray-sphere intersection computations, we first define formulas for a ray and a sphere, and we try to show that there is some point on this ray that is equal to some point on this sphere. We then plug in the formula for a ray (P(t) = A + tB) into the equation, then perform simple arithmetic operations to get it into quadratic form. Now, we simply solve the quadratic equation because the coefficients consist of vectors and other values that have already been defined, and any solution to this equation means that there is some intersection point between the ray and the sphere (Shirley, 2023).

Color Computation

Once we determine that there has been a collision, we take the point of intersection between the ray and the atom, and compute a color for that point based on a well-defined shading algorithm. The shading algorithm that we had opted for is called Phong shading, in which each atom is shaded based on some initial color, and its diffuse, ambient, and specular properties. The equation for Phong shading is shown in Figure A.

$$f(\vec{p}) = \underbrace{I_a K_a}_{\text{ambient component}} + \underbrace{\sum_{i}^{\text{nb\_lights}} (\vec{n}(\vec{p}).\vec{l}_i) K_d I_i}_{\text{diffuse component}} + \underbrace{\sum_{i}^{\text{nb\_lights}} f_{spec}(\vec{l}_i(\vec{p}), \vec{v}(\vec{p})) K_s I_i}_{\text{specular component}}$$

Figure 3k. Phong shading equations.

Diffuse lighting essentially colors an object based on its orientation to the light, meaning that certain points that are more distant from the light source will be shaded darker, while closer points will be shaded lighter. Diffuse lighting requires information about surface normals at a point, a vector to the light source, a specified diffuse color, and the light intensity. Ambient light is just a uniform light factor that is assigned to an entire object. It is done by multiplying the ambient light intensity by an ambient RGB color. This does not change with respect to the object's orientation or surface normals. Finally, specular lighting serves to provide specular highlights, which highlight points on the object with respect to the direction of the camera. Specular lighting requires use of a vector to the light source, vector to the camera, light intensity, specular color, and a factor that indicates how shiny the specular lighting should be. All three of these factors combined serve to provide a realistic model of how light interacts with objects in the real world (Phong, 1975). Finally, we have implemented some quality of life improvements, including the ability to rotate the protein by clicking and dragging, zooming the camera in and out using the scroll wheel, moving the camera using the WASD keys, and options for coloring the protein based on the atom elements, side chains, and similar regions determined from sequence alignment or 3D structure alignment.

4. Graphical User Interface

For the graphical user interface, upon entering the PDB ID of interest, various coloring options are available to facilitate visual comparisons between different alignments. Some of these

options include side chain coloring, different atom coloring, highlighting differing regions from sequence alignment, default rendering of the protein, rendering of protein 1, rendering of protein 2, and coloring based on the Kabsch algorithm.

Within the visualized GUI, users can modify the camera location, enabling them to customize the perspective. Additionally, zoom-in and out functions are incorporated to facilitate a closer examination of specific regions. To enhance the visualization of atoms and enable a comprehensive analysis, a rotation function is provided. This feature allows users to view the protein structure from various angles, including the rear aspect, ensuring a thorough exploration of the molecular details.

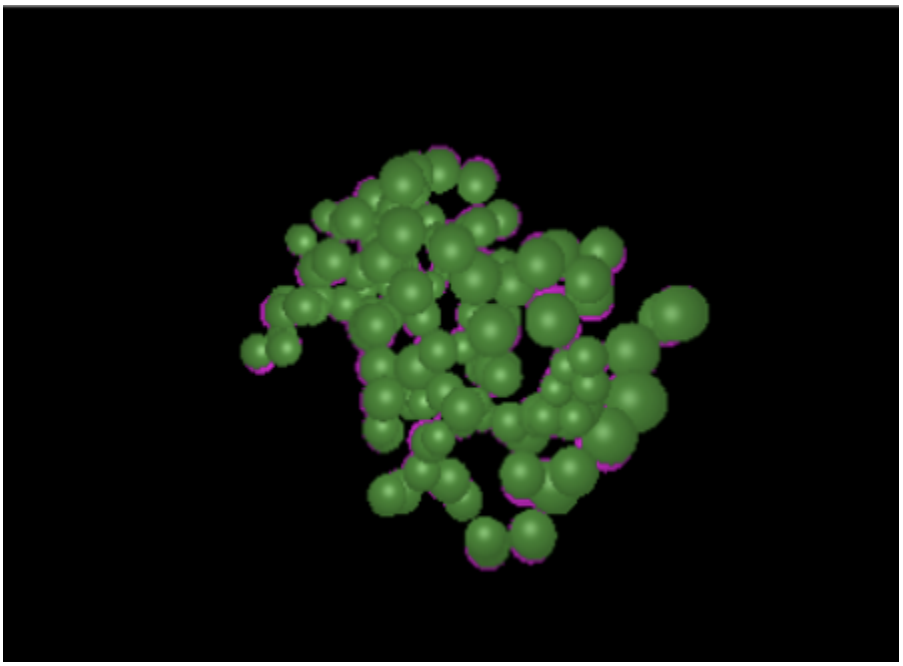

Fig.4a.: 3D visualization output

Upon closing the GUI, users can observe a one-to-one comparison of the 1D sequence alignment between two PDB files. In this representation, empty boxes denote gaps, while "|" signifies identical alignments between the sequences.

Results

To elucidate the functionality of GoMol as a tool for scientific analysis of proteins, we first analyzed two lysozyme protein PDB entries, 1LYZ and 2LYZ.

Figure 4a: This image shows the visualization of the Kabsch algorithm run on the GoMol GUI for 1LYZ (green) and 2LYZ (pink).

The 1LYZ and 2LYZ proteins from the PDB represent the same hen egg-white lysozyme for which the three-dimensional atomic coordinates were experimentally determined and refined following two distinct protocols. Given they share identical primary sequences, the GoMol output from the Needleman-Wunsch implementation showed 100% identity and therefore the visualization is trivial and not shown here. The RMSD determined for these two structures was essentially zero, which was expected given that vastly different structures would indicate invalidity in one or more of the coordinate determination protocols. In Figure 4a, both structures are overlaid such that it is clear to the user that the structures are very similar. This seemingly simple execution of GoMol analysis serves as a proof of concept and highlights various important applications of this software. The GoMol three-dimensional structure analysis allows for the validation of novel protocols for the experimental determination of atomic coordinates. Similarly, regarding research into the application of machine learning and artificial intelligence for protein structure prediction, it is clear how this functionality of GoMol could be applied for the validation of computationally determined models against experimentally determined structures.

Figure 4b: Sequence alignment of 1cfd and 1cll shows structural similarity by residue index

1cfd and 1cll are the same calmodulin protein with and without calcium bound. The binding of calcium to this protein induces a significant structural change, so when the Qres algorithm is run to elucidate structural differences the values per residue should reflect these structural changes. The Qres metric ranges from 0 to 1 and is colored from red to blue with more blue corresponding to higher Qres score and more red corresponding to lower score. Unaligned loci are colored red. This result is interesting because it allows a researcher to focus on different portions of the primary sequences based on structural differences based on what they are trying to analyze or study. For example, if we were trying to develop a drug to target this protein regardless of

calcium binding, we would want to focus on regions of the protein that remain similar in both conformations.

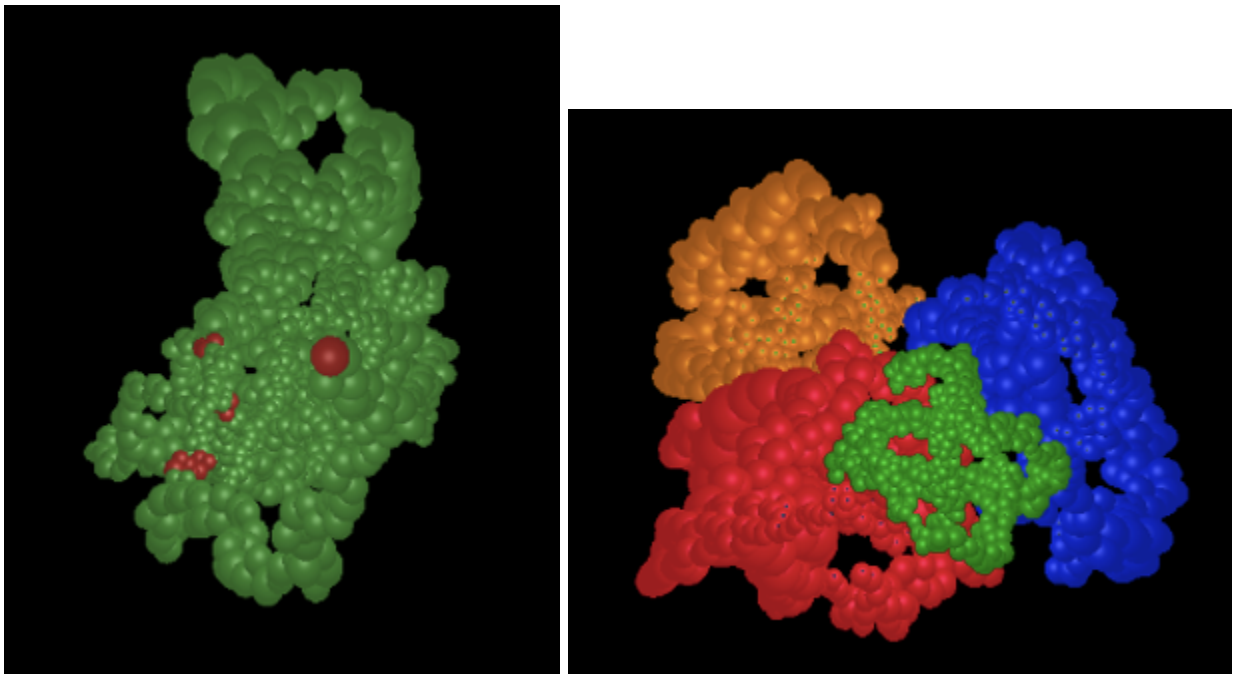

Figure 4c, 4d: Protein structure rendering can highlight unaligned residues from Needleman-Wunsch or render by side chain.

The GoMol GUI allows for different coloring options to elucidate the insight gained from the algorithms implemented and the PDB input files. These options are useful when used in conjugation with the results shown above, as the user can switch back and forth between rendering options to make connections between the results.

Discussion

In conclusion, GoMol, a 3-D analysis and visualization tool for proteins, combines multiple respected algorithms in computational biology into an easy-to-use platform to explore scientific questions about protein structure and sequence. As shown in the results, there are many relevant applications of the GoMol technology across various facets of the field of computational biology. Essentially, GoMol has the potential to serve as an educational tool for students, especially visual learners, to explore the various levels of protein structure from real data frequently used in research. Additionally, the structural comparisons in three-dimensional space provide a simple

means of validating experimental protocols and predictive models against known structures. Within the subfields of drug discovery and molecular dynamics, GoMol provides a mechanism for analyzing conformational states of proteins and can be used to further understanding of sequence and structural differences and provide insight for drug target identification and validation, and lead discovery and hit optimization. Lastly, from the perspective of evolutionary biology, the GoMol analysis pipeline can aid in garnering insight into the conservation of protein sequence and structure or other evolutionary processes via an analysis of homologous or analogous structures across distinct species.

References

1. Biology of Disease. (2023). IU Biology. Retrieved December 14, 2023, from https://biology.indiana.edu/undergraduate/biology/biology-bs/areas-of-concentration/disease.html
2. Central dogma of molecular biology. (2023, August 26). Wikipedia. Retrieved December 14, 2023, from https://en.wikipedia.org/wiki/Central_dogma_of_molecular_biology
3. Compeau, P. (2022). Protein Structure Comparison. Biological Modeling. Retrieved December 3, 2023, from https://biologicalmodeling.org/coronavirus/accuracy
4. Kabsch algorithm. (2023). Wikipedia. Retrieved December 3, 2023, from https://en.wikipedia.org/wiki/Kabsch_algorithm
5. Protein Data Bank. (2023). RCSB. Retrieved December 14, 2023, from https://www.rcsb.org/
6. Vertrees, J. (2019, April 18). Kabsch. PyMOLWiki. Retrieved December 3, 2023, from https://pymolwiki.org/index.php/Kabsch
7. What are proteins and what do they do. (2021, March 26). MedlinePlus. Retrieved December 14, 2023, from https://medlineplus.gov/genetics/understanding/howgeneswork/protein/
8. Phong, B.. (1975, June 1). Illumination for computer generated pictures. Communications of the ACM. https://dl.acm.org/doi/10.1145/360825.360839
9. Shirley, P., Black, T. D. (2023, August 6), ; Hollasch, S.. Ray Tracing in One Weekend. Retrieved December 14, 2023, from https://raytracing.github.io/books/RayTracingInOneWeekend.html
10. The needleman-wunsch algorithm for sequence alignment - SJSU. (n.d.). https://www.cs.sjsu.edu/~aid/cs152/NeedlemanWunsch.pdf

Team Contributions

Tyler

Essay: Introduction, 3-D Structural Alignment, Results, Discussion

Code: Implementation of Kabsch and Qres algorithms

Presentation: 3-D Alignment

Shivank

Essay: Sequence Alignment

Code: Implementation of Needleman-Wunsch algorithm

Presentation: Introduction, Model Architecture, Sequence Alignment

Darin

Essay: Visualization

Code: Connection to PDB/Parsing, Ray Tracing & Visualization, Integration of Modules, Version Control, Readme, Testing

Presentation: Input Data, Ray Tracing, Preliminary Results/Output

Minhyek

Essay: Graphical User Interface

Code: Graphical User Interface, Sequence alignment, visualization, Integration of Modules, Version Control, Demo

Presentation: GUI Demo, Limitation & Future Direction, Design

Group Effort: Literature review, Planning, Debugging, Presenting,