Masaryk University
Faculty of Informatics

Comparative Visualization of Protein Sequences

Master's Thesis

Pavol Ulbrich

Brno, Spring 2018

Masaryk University
Faculty of Informatics

Comparative Visualization of Protein Sequences

Master's Thesis

Pavol Ulbrich

Brno, Spring 2018

This is where a copy of the official signed thesis assignment and a copy of the Statement of an Author is located in the printed version of the document.

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Pavol Ulbrich

Advisor: doc. RNDr. Barbora Kozlíková, Ph.D.

Acknowledgements

I would like to thank my supervisor, Bára Kozlíková, for an excellent mentoring and guidance through the last stages of my master studies.

Then my thanks go to two of my colleagues, Víťa Matela and Vojta Frodl, for countless nights in sixth floor of the faculty building. Writing our theses.

Abstract

To better understand the constitution and spatial arrangement of protein sequences, L. Kocincová et al. [1] proposed a novel method of comparative visualization, which combines traditionally used 1D and 3D representations. Its main contribution is the ability to observe the spatial differences between the proteins without any occlusion problems, commonly present in 3D view. However, the practical implementation of the innovative method has remained unfinished. This thesis aims to create a web application for comparative visualization of protein secondary structures, which will benefit from the qualities of the method proposed by Kocincová et al. In addition, the tool will provide both sequential and structural protein alignment based on modern methods, which rely on evolutionarily-related residues and ensure a precise alignment of secondary structures.

Keywords

molecular visualization, molecular analysis, protein, alignment, structure, sequence, comparison, Protein Data Bank, d3.js

Contents

1 Introduction **1**

2 Visualization of Protein Structures **3**
2.1 *Protein Structure Definition and Visualization* 3
2.1.1 Comparative Protein Visualization 5
2.1.2 Comparative Visualization of Protein Secondary Structures – Summary of Publication 6
2.1.3 Visualization Techniques Used in the Original Solution 8
2.2 *Related Work* 9

3 Methods of Protein Alignment **13**
3.1 *Sequence and Structure Alignment* 13
3.1.1 Alignment Methods 15
3.2 *Alignment Tools* 15
3.2.1 Gap Insertion Algorithm 16
3.2.2 DeepAlign and 3DCOMB Alignment Methods . 17
3.3 *Estimation of Protein Secondary Structures* 19
3.3.1 DSSP 20
3.3.2 Reproduction of the Secondary Structures . . . 20

4 Implementation of Former Application **23**
4.1 *Functional Description* 23
4.2 *Drawbacks of the Former Application* 26
4.2.1 Application Design Drawbacks 28
4.2.2 Summary of Implementation Drawbacks 30

5 CVASS – Application Design and Implementation **31**
5.1 *Design of the Client-Server Application* 31
5.2 *Server Pipeline* 34

6 CVASS – Visualization Design and Implementation **37**
6.1 *Front-end Pipeline* 37
6.1.1 Data Structures 39
6.1.2 Layout Design 39
6.2 *Visualization Methods* 42

6.2.1 PV Viewer for Structure Alignment Visualization 42
6.2.2 Sequence Alignment Visualization Methods . . 43
6.2.3 Interaction with the Comparative Visualizations 45

7 Conclusion **49**

Bibliography **51**

List of Tables

5.1 Time performance of the server pipeline *36*
6.1 Time performance of the visualization pipeline *48*

List of Figures

2.1 One-letter visual representation of protein sequence *3*
2.2 Tertiary structure of protein *4*
2.3 Topology cartoon representation *10*
2.4 Comparative visualization by Schäfer et al. *10*

3.1 Sequence alignment of protein structures *14*
3.2 Aligned and Not-aligned pair of proteins *14*
3.3 Gap Insertion Algorithm example *16*
3.4 Protein sequence without secondary structures *19*

4.1 Former application layout *24*
4.2 Defects of secondary structure positions and gaps *26*
4.3 Defects of the PV viewer *27*
4.4 Former application front-end pipeline *29*
4.5 Dependencies of former JavaScript classes *29*

5.1 Client-Server design *32*
5.2 Express.js example *33*
5.3 Server pipeline *34*

6.1 Architecture design *38*
6.2 Visualization pipeline *38*
6.3 Data structures *40*
6.4 Web layout design of the CVASS *41*
6.5 Sequence alignment view *43*
6.6 Sequence alignment highlighted elements *44*
6.7 Juxtaposition view *45*
6.8 Superimposition view *45*
6.9 Interaction example *46*

1 Introduction

Visualization of molecular structures has been one of the oldest branches of data visualization and serves as an exceptional way to demonstrate and explore the molecular features. Past two decades witnessed the development of *interactive* visualization of biomolecular structures – macromolecules such as proteins, polysaccharides, lipids, DNA, and RNA. Its purpose is to comprehensibly represent their complex structures, properties, and interactions.

The core of this thesis is closely related to the research done by L. Kocincová et al. [1], which studies the benefits of comparative visualization of molecular structures, principally proteins. The scientific paper presented by her team illustrates an innovative method of visual comparison of two and more protein sequences, focusing mainly on representation and alignment of protein secondary structures without the occlusion commonly presented in traditional approaches. The concept was well received by the community of biochemists, as the paper, presented at the BioVis 2016 in Baltimore, received the Best Paper Award. However, a functional tool, that practically demonstrates the method, was never completed and publicly released.

In order to interpret the newly proposed method as an interactive web application, one needs to understand the fundamental concepts of molecular structures and how to illustrate them. In addition, the quality of a coherent visualization relies on proper alignment of the compared structures. The theoretical part of this thesis analyses the comparative method presented by L. Kocincová, as well as recent methods of biomolecular visualization, and examine the possibilities of an appropriate protein alignment.

In the practical part of this thesis, we discuss the implementation of selected visualization and alignment methods into the newly proposed Comparative Visualization and Analysis of Secondary Structures (CVASS) application, which is inspired by the formerly presented tool. The web application will demonstrate the possibilities of interactive interpretation of protein sequences, as well as the opportunity to upload, align, and visualize new protein structures.

Protein Data Bank

The Protein Data Bank (PDB) [2] is a database containing the three-dimensional structural data of large biological molecules, such as proteins in our case. The archive is freely accessible via the websites of its member organisations and serves as the primary input source for our CVASS tool. The same abbreviation also stands for the file format used by the database for these structures, which provides a standard representation for macromolecular structure data [3]. The data contained in the database include among other records mainly the name of the molecule, sequence database references, the information about its primary and possibly also secondary structure, and individual atomic coordinates and other properties.

Another representation of the protein is the text-based FASTA format, in which sequences are represented using single-letter codes for every amino acid or gap after alignment process. Thanks to its simplicity, it is easy to parse. Both PDB and FASTA format will be used in the CVASS application.

Thesis Structure

In Chapter 2 we define the structure of a protein sequence, discuss possible techniques suitable for comparative visualization of multiple protein sequences, and analyze the work related to this topic. Chapter 3 contains the definitions and overview of methods of structural and sequence alignment, reviews the formerly used gap insertion algorithm for sequence alignment, and introduces the alternate pair-wise and multiple alignment methods, including the estimation of secondary structures. Subsequently, Chapter 4 thoroughly evaluates the former tool. In Chapter 5, we describe the architecture design of the newly proposed client-server application and the implementation of the alignment methods. Finally, Chapter 6 presents the implementation of the visualization methods used in the CVASS application.

2 Visualization of Protein Structures

In this chapter, we will define the basic biochemical terminology necessary for understanding the content of this thesis. Then we explain the role and importance of visualization in biochemical research. Furthermore, we will discuss the concepts of the original work by Kocincová et al. [1], analyze its drawbacks, and compare it with current research trends in this field. Finally, we introduce the main principles and functions of our proposed improved solution, the CVASS visualization tool.

2.1 Protein Structure Definition and Visualization

Proteins are biomolecules consisting of one or more polypeptide chains of amino acid residues. These residues, taken as an ordered sequence, are forming the *primary* structure of the protein. It is usually represented by one-letter or three-letter abbreviations for the amino acid residues. The one-letter abbreviations are commonly used in the visual representation of the chain (see Figure 2.1).The polypeptide chain is also known as the backbone.

Example: ....Try.Gly.Lys.Pro.Val.Gly.Lys.Lys.Arg.Arg.Pro.Val.Lys....

Within the long protein chains, there are regions where the amino acids are organized into regular structures, known as alpha-helices and beta-pleated sheets. These are called the *secondary structures* of proteins, which are held together by hydrogen bonds. In an alpha-helix, the protein chain is coiled in the clock-wise direction, resembling a spiral (depicted in blue in Figure 2.2). In a beta-pleated sheet, the amino acids are folded to a formation, where they are positioned alongside each other (depicted in orange in Figure 2.2). Amino acids

Figure 2.1: One-letter visual representation of protein sequence.

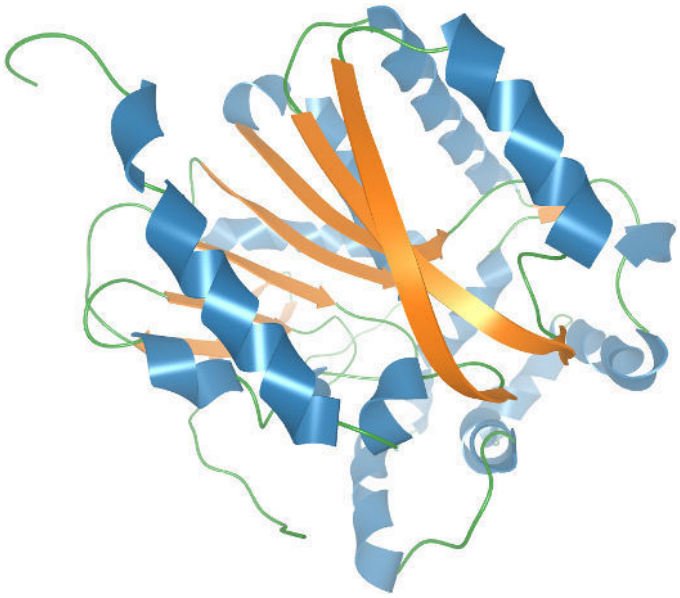

Figure 2.2: Tertiary structure of protein with PDB ID 1CQW and its secondary structures.

which are not in any of the above conformations are considered as coils (green in Figure 2.2).

The *tertiary structure* of a protein is the description of the way the whole backbone (including the secondary structures) folds itself into its three-dimensional shape. Finally, the *quaternary structure* is formed by several polypeptide chains, which function as a single protein complex.

The size of the protein chain, measured by the number of amino acids it contains, is usually in hundreds of units [4]. However, one protein chain can contain up to 27,000 amino acids [5]. Each residue, one of 20 amino acids, consists of several atoms. For large molecules, it is impossible to recognizably display all atoms and bonds using any common representation. So the visual abbreviations and glyphs are used instead. Typically, the most commonly used form of visual representation of a protein is either 1D chain of sequence of its amino acids (Figure 2.1), or a 3D model displaying the spatial orientation of secondary structures (Figure 2.2).

The 1D representation is easily comprehensible and widely used for basic understanding of sequence arrangement and configuration. It is suitable for comparison of two or more protein chains, as the differences between them are easily visible. Its simplicity results in the lack of any additional information, e.g., completely omitting the spatial orientation and position on the chain, which can be essential for correct evaluation of protein function and behavior.

Contrarily, in the 3D model, the spatial information is very well recognizable. We are able to observe alpha-helices and beta-sheets as well as the whole tertiary structure of the protein. With the increasing complexity of the structure or the number of displayed chains, the whole representation becomes very cluttered. Its readability can be enhanced by various techniques, such as interaction with the model, application of LOD (Level of Detail) and DOF (Depth of Field) view modifications, and using abstract visual representations, such as cartoons or ribbons[6]. However, these enhancements improve the problem of visual clutter only to some extent. For many structures, the comprehensibility of the spatial representation decreases significantly.

2.1.1 Comparative Protein Visualization

In order to understand the behavior and function of proteins, biochemists and biologists study their spatial arrangement, constitution, and dynamic behavior. These factors can be understood by comparing their structures, hence to reveal similarities and differences in their polypeptide chains. The chains are often aligned for better recognition of these comparisons, using either pairwise or multiple structure alignment tools. We will discuss these algorithms in the following chapter.

The objective of the comparative protein visualization is to ease the process of recognition of similarities and differences and to reduce it into simple visual observation. As mentioned above, neither 3D [7] nor 1D representation is fully capable of adequate distinction of all important aspects. A hybrid representation, which combines the strengths of both approaches mentioned above, is the technique proposed in the paper by Kocincová et al. [1]. The improvement of this technique forms the content of this thesis.

The scientific paper [1] was published in fall of 2016 as a collaborative work of scientists from the Masaryk University in Brno, Czech Republic and the University of Bergen, Norway. Its main aim was to present a new visualization technique that takes into consideration the mutual positions of protein chains represented by their secondary structures, so it is possible to observe the spatial differences between proteins. This approach has never been used in applied biochemistry and its highly innovative with respect to commonly used 1D and 3D representations.

This thesis is meant to be a resumption of the paper and the method that is introduced in it. Its main outcome is to rebuild the web-driven visualization tool into the fully functioning state, as well as to discuss possible improvements and reactions to the innovation in last two years, since the publication of the original paper. Let us start with a brief summary of the main principles of the proposed method and related issues discussed in the paper.

2.1.2 Comparative Visualization of Protein Secondary Structures – Summary of Publication

During the exploration of protein sequences, biochemists can benefit from the information about the mutual position of proteins' secondary structures. Even though their arrangement can be observed in a 3D view, the clarity of the resulting interpretation rapidly decreases with raising the number of sequences. To be compared with only a few structures, it is very hard to perceive the differences in the secondary structure positions. Therefore, a new proposed solution combines requirements, such as constitution and comparison of protein chains based on their secondary structures, as well as traditionally shown primary structures' chain. The solution adds tools for easy manipulation and interaction with them through a web interface.

The application combines the benefits of the 1D sequential representation and 3D view. Protein chains are shown in a form of superposed and juxtaposed sequential representations of the secondary structures, while the linked 3D visualization is supported by the PV Viewer reference tool [8]. This setup demonstrates the alignment of two or more protein sequences, where one sequence is considered as a reference and its chain of glyphs representing secondary structures in

the superposed or juxtaposed views is completely straightened. Other proteins are rendered as sequences of secondary structure glyphs with changed position and rotation, according to the actual angle and shift with respect to the reference chain. Arrows represent beta-sheets, spirals stand for alpha-helices, and lines represent coils. The layout should enable the navigation through the chain, selection of specific parts, and zooming. Selection of the protein structure in the superposed or juxtaposed view is linked with the 3D representation and allows intuitive navigation.

The implementation part was originally based on web technologies, namely JavaScript and its graphics library D3.js [9]. The paper also describes a method for sequence alignment built upon a gap insertion algorithm. It translates the sections of sequence containing secondary structures by inserting gaps between them. The original solution enables to process a set of time steps representing molecular dynamics of a single protein in a similar manner. In this case, the solution helps to determine the most stable and unstable parts of the protein over time. However, the used approach lacked the desired quality when working with more complex input proteins with significant differences in their sequences. The team decided to use the solution included in the CAVER project [10] instead. This step was necessary to make as this solution was aligning the structures according to their spatial representation and not according to the constitution of their sequences. This issue will be discussed in Chapter 3 of this thesis.

In the original paper, the capabilities of the application are demonstrated on two examples. In the first scenario, the spatial orientation of the secondary structures is substantial. The second one focuses on the exploration of protein dynamics. For both scenarios, the proposed solution behaved sufficiently and presented an innovative way to visualize similar protein sequences. In the very end of the paper, the limitations are discussed, mentioning problems with performance of the gap insertion algorithm and many occlusions. Possible extensions were just briefly suggested. Therefore, our goal within this thesis was to thoroughly address the limitations of the previously designed solution and propose a working tool for visual comparison of protein sequences, based on the idea of the original paper. The description of our improvements forms the core of this thesis.

2.1.3 Visualization Techniques Used in the Original Solution

As mentioned above, the protein chains are displayed in the web application layout in three ways: as the 3D model, the Superposition model of the overlayed sequences of glyphs (Figure) and the Juxtaposition model of sequences placed below each other (Figure). In these abstracted views, every sequence is represented by a chain of glyphs. This chain is composed of *narrow gray line* indicating protein coil, *spiral* glyph in colour signifying alpha-helices, and simple one-side *arrow* in colour in the direction of amino acid chain, that represents beta-sheets and *thick gray lines* forming the gaps in protein sequences. Every protein chain has a different colour for spirals and arrows, which remain constant through the whole propagation of the chain.

After the alignment process, similar secondary structures are situated in the same horizontal position in the portrayed schemes. The position and orientation of the glyph abbreviates the relative geometry of the compared secondary structures. Below each glyph chain there is a very simple 1D line visualization displaying gaps in the protein sequence. Gaps are filled with black colour, while the rest is coloured in the same manner as the protein's glyphs. The line serves as a navigational panel, since one can select the section of protein sequence and the juxtaposition and the superimposition models will display only the selected part. The latter permits the selection over all sequences, while the former allows the zoom of each glyph chain individually.

For the 3D visualization, that folds the secondary structures' glyphs into the tertiary structure, the authors used the PV viewer [7]. It is an open source JavaScript viewer for visualizing protein structures directly in the browser and it is available within the SWISS-MODEL tool [8]. The 3D viewer uses WebGL [11] standard for 3D rendering and is able to display large biochemical structures at interactive frame rates. The tool provides similar ribbon rendering as described above with helix-sheet-coil cartoon glyphs and same colours. The PDB files can be imported into the PV viewer and it is capable to assign the secondary structures into the model, which marginally simplifies the process of its integration into the web application. The 3D model of protein is interactive and can be zoomed and rotated.

All three visualization models are directly linked together. This connection allows us to highlight structures in one model by clicking

on it with the cursor and they will be highlighted also in other two models, changing their colour to bright green. It significantly improves the interaction with the 3D viewer and orientation in molecule's structure. However, the structure can be unmarked only after highlighting a large set of other structures, since there is a limitation on the number of selected glyphs.

2.2 Related Work

In this section, we will mention several existing approaches to problems tightly related to our comparative visualization of protein sequences. These are namely protein unfolding to 2D representation and visual comparison of more proteins. However, to the best of our knowledge, the current solutions, except for the original paper behind this thesis, lack of combining the unfolding representation with the spatial comparison of protein secondary structures.

The main benefit of the unfolded 2D representation of the molecule is that it aims to avoid the occlusion problems present inherently in 3D. There are several traditionally used techniques for flattened representation of molecular chains. A comprehensible overview of them was published by Zhou and Shang [12]. They present numerous approaches to generating and rendering of schematic layouts of molecules. Many of them are integrated in the existing tools, such as HERA [13], TOPS [14, 15], or the topology diagrams in the PDBsum database [16]. The protein topology cartoons were first defined by Jane Richardson [17]. Stivala et al. [18] presented Pro-origami, a tool for interaction and editing of the topology cartoons. Figure 2.3 shows an example of topology cartoon representation for two protein structures, which are almost identical when aligned in 3D. However, their flattened representation in Pro-origami is significantly different. Therefore, such a representation is not suitable for comparative visualization of proteins.

Schäfer et al. [19] presented folding graphs which are more appropriate for comparative visualization of protein structures (see Figure 2.4). The encoding of the secondary structures and the protein sequence chain is highly abstracted and therefore hard to comprehend.

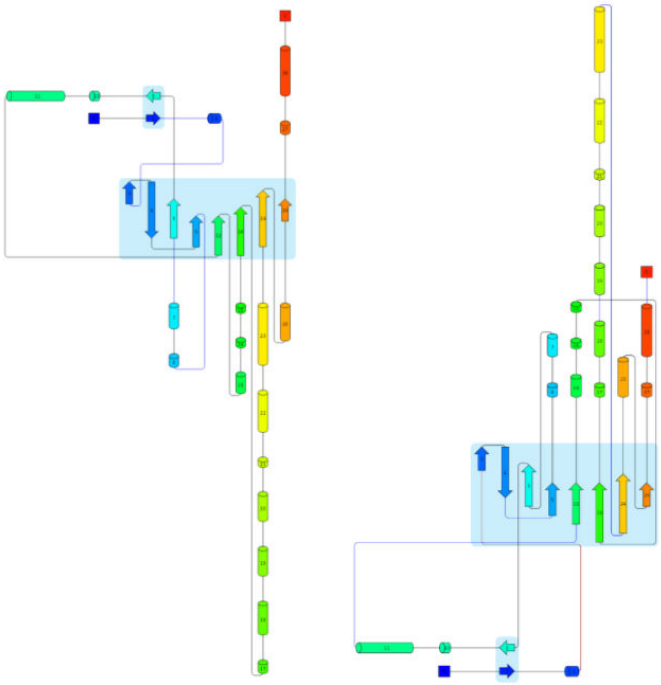

Figure 2.3: Topology cartoon representation for proteins with PDB ID 1EDD (left) and 1EDB (right), generated using Pro-origami [18].

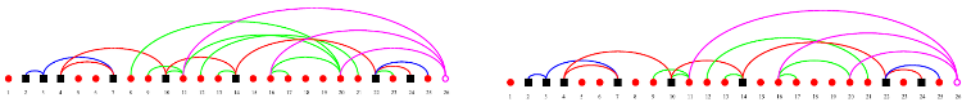

Figure 2.4: Comparative visualization of proteins with PDB ID 1EDD (left) and 1EDB (right), generated by the method by Schäfer et al. [19]. Red circles represent helices, black squares represent sheets, and magenta circle represents ligand interacting with the protein.

Stolte et al. [20] introduced the Aquaria tool for visual exploration of protein structures, combining the 3D representation with sequential information which encodes the information about secondary structures. This idea of encoding the additional information to the sequential representation was further extended in our proposed visualization.

3 Methods of Protein Alignment

In this chapter, we will introduce the sequence and structure alignment problem and describe the existing methods and recent achievements in this domain. Then we will discuss the existing tools suitable for protein alignment, including the gap insertion algorithm, which was used in the previous solution by Kocincová et al. [1]. Then we will focus on the description of DeepAlign and 3DCOMB methods, which were chosen as substitutes of the gap insertion algorithm in our new CVASS solution. Finally, we will define the problem of estimation of protein's secondary structures and we will describe two possible solutions.

3.1 Sequence and Structure Alignment

When aiming to align two or more proteins, we distinguish between two possible solutions – the sequence and structure alignment. The former takes into account the succession of amino acids in the protein chain and builds a trace for every sequence, consisting of elements, which are either matches or gaps. The match is inserted into a position, where the residues in two or more chains are identical and it defines that these amino acids are vertically aligned in these positions (Figure 3.1). In contrast, a gap is inserted into the position of divergent amino acids, indicating either deletion, insertion, or completely different region of the protein sequence. The number, length, and location of inserted gaps from the resulting traces express the degree of similarity between the compared sequences and serve as a comprehensive tool for defining functional, structural, or evolutionary relationships between protein regions or whole sequences [21].

In the structural alignment, the three-dimensional conformation of protein, which corresponds to the tertiary structures of compared proteins, are taken into consideration. The output is represented as superposition of the atomic coordinate sets and the minimal root mean square deviation (RMSD) between the aligned structures (Figure 3.2), indicating the level of divergence between the sequences. The optimal solution to the multiple sequence alignment has been proven to be NP-complete [23], therefore the approximate methods are used. At

```
1aboA   NL-FVALYDFVASGDNTLSIT--------KGEK-------LRVLGYNHN-------GEWCEA--QTKNGQGWVPS
1ycsB   KGVIYALWDYEPQNDDELPMK--------EGDC-------MTIIHREDE-----DEIEWWWA--RLNDKEGYVPR
1pht    GYQYRALYDYKKEREEDIDLH--------LGDILTVNKGSLVALGFSDGQEARPEEIGWLNGYNETTGERGDFPG
1vie    --------DRVRKKSGA--AW--------QGQIVGWYCTNLTPEGYAVESEAHPGSVQIY------------PV
1ihvA   NFRVYY----RDSRD---PVWKGPAKLLWKGEG------AVVI-QDNSD-------IK------------VVPR
```

Figure 3.1: An example of the sequence alignment, generated by the T-COFFEE MLA method [22]. The colour represents the score of the alignment, from the darker shades of red (high) to the bright yellow (low).

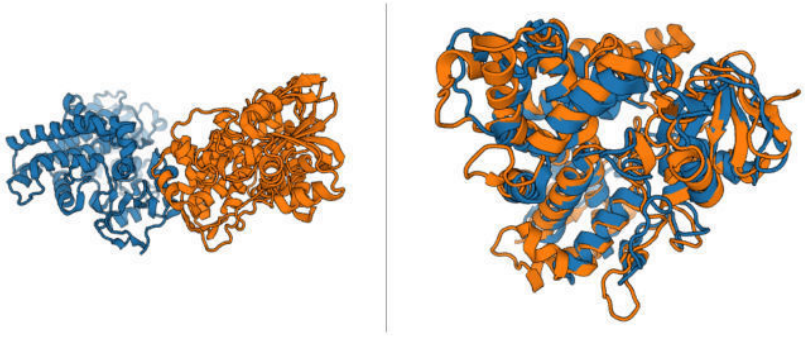

Figure 3.2: An example of structurally non-aligned (on the left) and aligned (on the right) 3D visualizations of the pair of proteins: 1NR6 (orange) and 1IZO (blue).

the residue level, the relation between the sequence, structure, and function is not statistically expected [24], which amplifies the necessity of calculating and demonstrating both sequence and structural alignments.

The proposed visualization model utilizes both alignment methods and the presented approach combines them for the best possible presentation of selected proteins. The structure alignment is used in 3D model. It is able to demonstrate the similarities in even very distinct proteins, where the standard sequence alignment techniques fail to easily depict the spatial correspondence. On the other hand, the juxtaposition and superimposition visualization models benefit from the sequence alignment, where it clearly demonstrates the similar regions and the parts of the sequences filled with gaps.

3.1.1 Alignment Methods

Within the scope of this thesis, we distinguish between the pairwise and the multiple alignment method. The former is used to find the best-matching piecewise (local or global) alignments of two protein sequences. The latter extends the pairwise alignment to incorporate more than two sequences at once.

"A protein structure alignment method consists of two major components: a scoring function measuring protein similarity and a search algorithm optimizing the scoring function. It is very challenging to design a scoring function to exactly capture all the (implicit and explicit) rules used by human experts, who align and classify protein structures using not only geometric similarity, but also evolutionary and functional information." [25]

A variety of the currently available tools for the protein sequence alignment can be found in the OMICtools [26] database of omic (next-generation sequencing, microarray, mass spectrometry, and nuclear magnetic resonance) technologies. The majority of the tools today use only 3D geometric similarity, such as DALI (Distance alignment matrix method) [27], CE (Combinatorial extension) [28] and TMalign [29]. Nevertheless, these methods may produce bad alignments even if there exists the existence of shared ancestry between a pair of structures [30].

3.2 Alignment Tools

The previous solution introduced a heuristic method for sequence alignment, which took into consideration the secondary structures of proteins. However, it relied on already structurally aligned proteins, computed manually using the CAVER Analyst tool. This solution was used only for demonstration purposes but it was never replaced by correct solution, enabling to load and automatically align arbitrary structures from the PDB database. Additionally, this solution suffers from several other drawbacks so we decided to completely replace it. In this section, we will describe in detail the original gap insertion algorithm and introduce our proposed solution which utilizes tools for both pairwise and multiple alignment: DeepAlign and 3DCOMB.

Figure 3.3: An example of the sequence alignment calculated by the gap insertion algorithm, indicating gaps with the black colour.

3.2.1 Gap Insertion Algorithm

Kocincová et al. [1] presented a basic greedy approach for sequence alignment, based on the comparison of secondary structures of two proteins. The main benefit is its simplicity and, analogically, the execution speed. On the other hand, thanks to the nature of the algorithm, the output sequences may not be aligned optimally. Especially for very different input protein sequences, the method performs insufficiently and produces incomprehensible results.

The idea behind the gap insertion algorithm is to sequentially read the secondary structures of two aligned protein chains and calculate the length of gap that is necessary to insert for the correct alignment of the corresponding secondary structures. The length of the gap also influences the value of the scoring function. The correspondence between the secondary structures is determined from their spatial distance and type (i.e., helix, sheet, coil). The gap value corresponds to the number of residues which ought to be skipped. For any two structures, two gaps are calculated (from the first structure to the second one and vice versa) and the smaller one is inserted into the corresponding chain. If two compared parts of sequences are perfectly aligned, no gap is inserted. The tail of the shorter chain after the alignment process is also filled with gaps, if the longer one still contains any remaining secondary structures. The example of an alignment computed by this approach can be seen in Figure 3.3.

This method serves also as the basis of the algorithm for processing molecular dynamics, which is described in the paper as well. In this case, the algorithm uses the fact that the structure of the protein chain does not change between individual snapshots of the molecular dynamics. The only possible change is in the constitution of the secondary structures. However, only small changes of secondary structures occur over the molecular dynamics. On top of that, the amino

acids at the ends of the secondary structures can change their membership. For shorter secondary structures, it is also possible that they disappear or new ones appear. Therefore, each time snapshot is not compared to the other snapshots, but to the aggregated chain instead. The aggregated chain consists of all secondary structures that appear at least in one time snapshot and it serves as the reference structure.

The alignment process is designed to run as a web service on client's side and cannot demand the processing and memory requirements of global, optimal solutions. Thanks to its heuristic nature, it does not take into consideration the history of processed secondary structures and does not look ahead further in the sequence than just one secondary structure. This may result in the insertion of unnecessary, redundant gaps. Continuously, the process is tightly related to the definition of the correspondence between the compared secondary structures. This is a complex problem, considered to be one of the fundamental problems in computational structure biology and often approached and discussed by the domain experts [31, 32].

The alternation of the sequence alignment algorithm is one of the main changes of our proposed solutions. Instead of relying on the spatial structure of the compared proteins, we are utilizing their sequence alignment which gives us the information about inserted gaps automatically.

3.2.2 DeepAlign and 3DCOMB Alignment Methods

Due to the drawbacks of the original gap insertion algorithm, one of the main changes of our proposed solution is to substitute this approach and present an alternative for both sequence and structure alignment methods. The design of the client-server application allows us to abandon the limitations of the processing power, since the alignment computation can be executed on the server's side. Still, for the sake of interactive user experience, we need to provide the results of the alignment process with no or very small delay. Finally, the main element in the CVASS visualization tools is a secondary structure, so the required alignment method should take into consideration its positions in the chain. Considering these objectives, the DeepAlign pairwise alignment method and 3DCOMB multiple alignment method were selected and implemented into the CVASS visualization tool.

DeepAlign and 3DCOMB were developed under the supervision of Sheng Wang of Toyota Technological Institute at Chicago, USA, and were published in 2012 and 2011 [33, 25], respectively. Their main contribution was to accurately align distantly related protein structures. The 3DCOMB is a multiple structure alignment (MSA) method, that uses a probabilistic model to combine both local and global structure information. It applies statistical learning method to accurately identify highly similar fragment blocks among all proteins to be aligned. The length of a fragment is set in such a way that it is likely to cover at least one secondary structure segment. Then the marginal probability is defined as the similarity score of two structure fragments, using the forward–backward algorithm [34]. The RMSD is set as a score.

The method also introduces a novel scoring function to generate the MSA with a large number of cores. A core is a fully aligned column, consisting of one residue from each input protein. The distance between two aligned residues is used as denominator in the scoring function, so it favors the aligned residue pairs within small distance and disfavors or even ignores those with large deviation, enabling a detection of even a small conserved region among proteins of very different structures. It takes into consideration the distance deviation of aligned residue pairs and is almost independent to the protein length [35] since the distance at each position is normalized by a length-dependent factor. However, in case of the pairwise protein alignment, the method is outperformed by the DeepAlign.

The DeepAlign is a method for automatic pairwise protein structure alignment. It aligns two protein structures using geometric similarity after rigid-body superposition, evolutionary relationship, and hydrogen-bonding similarity, generating alignments highly consistent with manually-curated alignments. The method favors the alignment of evolutionarily-related residues and provides a precise alignment of secondary structures. Its scoring function is composed of amino acid mutation score, local substructure substitution potential, hydrogen-bonding similarity, and geometric similarity. The alignment is improved by an iterative dynamic programming refinement procedure in order to maximize the scoring function. At the end, heuristics are used to merge very short aligned fragment pairs to reduce the number of gap openings.

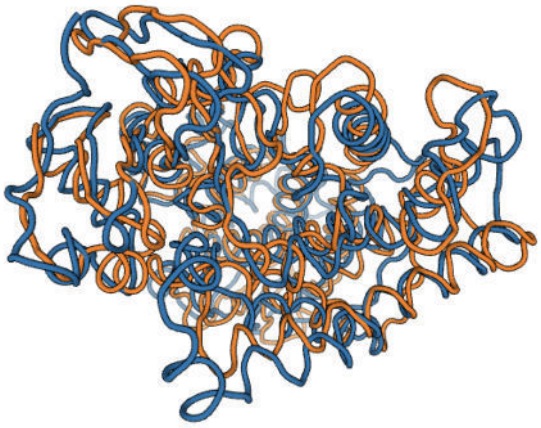

Figure 3.4: The visualization of the pair of proteins 1NR6 (orange) and 1IZO (blue) without records about their secondary structures.

For a small number of compared protein structures with the length in hundreds of amino acids, both methods yield excellent alignment results computed in units of seconds. They are available as an open source software with no restrictions to their use. Their implementation into the working application will be discussed in Chapter 5.2.

3.3 Estimation of Protein Secondary Structures

Once the alignment process is completed, its output files rise the question about the correct estimation of the secondary structures. The resulting PDB file contains just the information about a plane positions of atoms in 3D space, while it does not mention any information about the secondary structures, nor does the .fasta or any other output file. However, the presence of the records about the alpha-helices and beta-sheets in the output PDB file are essential for the visualization tools. Without the knowledge of the correspondence between the amino acids and secondary structures, these tools are able to display the whole backbone just as a single coil without any additional structures (Figure 3.4). Our solution provided two suitable options: the estimation of the secondary structures, using the well-known DSSP algorithm 3.3.1 and reproduction of the information from the original protein files.

3.3.1 DSSP

The DSSP (Define Secondary Structure of Proteins) [36, 37] algorithm is the standard method for assigning the secondary structures to the amino acids of a protein, given the atomic-resolution coordinates of the protein. DSSP starts with identifying the intra-backbone hydrogen bonds of the protein, using a purely electrostatic definition to estimate the hydrogen bonds [38]. Basically, the energy of every bond is measured and based on that, eight types of secondary structures are assigned. The 310 helix, alpha helix, and pi helix, represented by symbols G, H, and I, are recognized by having a repetitive sequence of hydrogen bonds. Two types of beta sheet structures exist; a beta bridge, symbolized by B, while longer sets of hydrogen bonds and beta bulges are marked by symbol E. Hydrogen bonds, that are typical for helices and turns, are indicated by T. High curvature regions are distinguished by letter S. A blank space or C is used if no other rule applies, referring to coils. These eight types are grouped into three larger classes: helix (G, H, and I), strand (E and B) and coils (S, T, and C).

DSSP is currently the most widely used algorithm for secondary structure estimation and it is also implemented in the CAVER Analyst visualization tool [39]. The abbreviation DSSP stands also for the database of secondary structure assignments for all protein entries in the Protein Data Bank (PDB) [2]. Its interpretation of the secondary structures annotation became a standard for modern definition of proteins. Slightly more complicated approach is provided by the STRIDE algorithm [40], which also includes dihedral angle potentials in addition to sole hydrogen bond criteria. This method was observed to correct the propensity of DSSP to assign shorter secondary structures.

3.3.2 Reproduction of the Secondary Structures

The process of estimation of the secondary structures is time and resource consuming and, as will be discussed below, redundant. Since the alignment tools require the knowledge of the secondary structures position in the protein chain of amino acids for the correct computation (and as will be mentioned in Chapter 4 for their successful execution as well), the entry PDB files provide the detailed description of all

data structures enclosed in the protein. By the definition, the PDB file carries the input information about alpha-helices and beta-sheets in form of records in its header, formed as lines with initial fields set as HELIX and SHEET. The helix entries contain 14 records, including the information about their intentifier, corresponding sequence number, helix class, its length, etc. The sheet entries are composed of 23 records, containing the strand indentifier, direction of strand with respect to the previous strand in the sheet, various registrations, etc.

Example of the secondary structure record in PDB file format:
HELIX 12 12 LYS A 206 ASP A 209 5 4
SHEET 5 B 7 GLN B 177 VAL B 182 1 O GLN A 177 N LEU A 143

The secondary structure records are generated automatically by the PDB database using the DSSP algorithm, although they may be provided by the depositor instead. Many times they would be assigned by an expert crystallographer, usually due to the minor local variations in structure that are most common near to the endpoints of the secondary structure elements. Therefore, the records already included in PDB the files are at least as good as the ones computed separately with local instance of the DSSP algorithm. This statement was proven by an experiment within this thesis, in which the original and newly calculated estimation of secondary structures before and after the alignment process was compared. There were two proteins used for this experiment, with PDB identifiers 1CQW [41] and 1R6A [42]. In both cases, the results differed only in details, slightly shifting and changing the length of the newly computed secondary structures, that are hardly recognizable.

Moreover, there were differences between the DSSP output before and after applying the DeepAlign alignment of these two proteins. Despite the fact that the structure itself does not change during the alignment process, it can be displaced and bent, thus it can result in the different calculation of energy between the atoms of the protein. Hence, the determination of the position of the secondary structures, even if only moderately, is directly related to a protein the sequence is aligned to.

The experiment showed that the DSSP estimation is approximately the same as the reference estimation from the input PDB files, while

it can vary on the set of input proteins. On top of that, the input files may often contain empirically measured records. Therefore, the use of the original secondary structure settings is the correct approach. The process of preprocessing the input protein data from the implementation point of view, including the alignment and secondary structure assignment, will be briefly discussed in Chapter 5.2.

4 Implementation of Former Application

The innovative way of protein visualization, described in the scientific paper and demonstrated in the demo web application, that was published by Kocincova et al [1]. It presented the visual output and design for the intuitive comparison of protein sequences, represented by their secondary structures. The layout of the visualization methods is suitable for simple interaction, perception of aligned secondary structures, and was meant to evolve into a complex visualization tool with multiple settings and adjustments. Unfortunately, the development of the application ended prematurely and was not improved for the following two years after its publication. The software lacked the fundamental functionality, which prevented it to be publicly available. In this section, we will discuss the advantages and deficiencies of the former solution.

4.1 Functional Description

The demo application was built as a single-page website, displaying all necessary information and navigation in one full-screen layout (Figure 4.1). The website screen space is visually divided into four separated sectors: the PV viewer, the Juxtaposition and Superimposition visualization views, and the control panel. First three parts are responsible for the visualization of compared protein chains and are interactively linked together: if a secondary structure would get selected in one view, it will be highlighted in the remaining views as well. The program also displays the essential information about the selected structure in a small pop-up text element, such as the name of the protein chain, secondary structure's position in this chain, and secondary structures of other chains it is compared to.

The PV viewer [8] panel is rendered on the top-left position (see Figure 4.1, part 1). It is the canvas used for the visualization of 3D protein structures. Every protein chain is represented by a so-called Cartoon representation, consisting of a connected sequence of tubes (coils), arrows (beta-sheets, the direction of an arrow indicates the propagation of the chain), and spirals (alpha-helices) and is assigned a unique colour from a preselected set of colours. The WebGL JavaScript

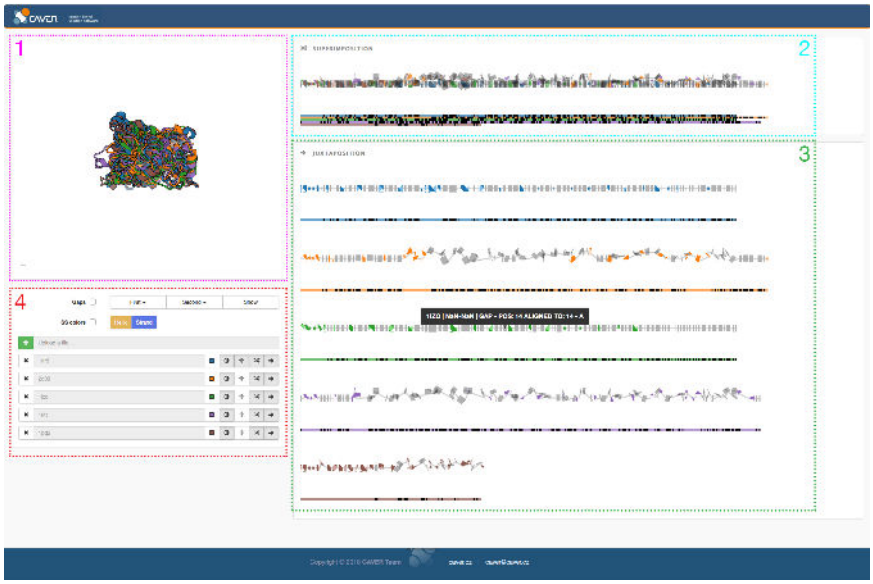

Figure 4.1: The full layout of the former web application with four separated sectors: 1 = PV viewer, 2 = Superimposition view, 3 = Juxtaposition view, 4 = control panel.

API [43] is responsible for rendering 3D representations of molecules in this view. It allows the viewer to support visual effects, such as anti-aliasing filter, level of detail rendering, near and far clipping planes, outline rendering for better contrast, and fogged objects in larger depth. Used combination of these techniques gives us an explicit representation of structurally aligned protein chains. The use of the PV viewer module will be discussed further in section 6.2.1.

The bottom left of the canvas contains a text field, which displays the information about the amino acid the cursor is currently hovering above. This amino acid also temporarily changes its colour to red until the cursor is moved away from the visible surface of the amino acid, or if it is clicked on it. In that case, the colour of not only the selected amino acid but the whole secondary structure it belongs to is changed to contrast green (Figure 4.3). The same colour is also used for the selected secondary structures in the other visualization views.

To the right of the PV viewer, the Juxtaposition and Superimposition visualization views are placed (in Figure 4.1 the former is labelled as 2, the latter as 3). Both of them render all protein chains as a sequentially aligned progression of protein structures and gaps, using D3.js graphic library [44]. While the Juxtaposition view displays each protein chain separately, in case of the Superimposition view the sequences are rendered all in one position, overlapping each other. The orientation and offset of the secondary structures with respect to the reference chain are calculated according to the real differences between secondary structure positions in the 3D view, efficiently presenting the distinct regions of the compared sequences.

The line consisting only of black (gaps) and coloured (actual chain) segments is inserted below every sequence. This line serves two purposes: it indicates the sequential alignment calculated by the gap insertion algorithm, and upon click, it sets the range of rendered sequence, i.e., zooming to the sequence (bottom part of Figure 4.2). The sequence alignment lines below the Superimposition view are aggregated into the group, while in the Juxtaposition view the lines are separated and placed below the corresponding chains. All the interaction procedures are quick to respond and easy to use.

Finally, the control panel, positioned below the PV viewer (part 4 in Figure 4.1 labelled by the number 4), is a general mechanism for global manipulation with the compared protein chains. It is designed

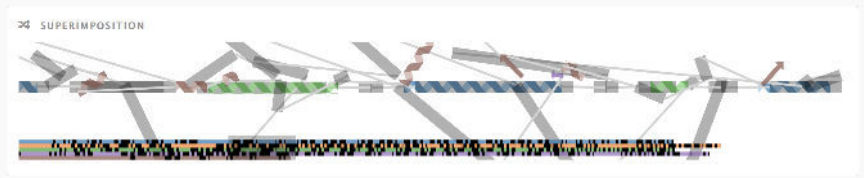

Figure 4.2: Defects caused by positions of secondary structures and gaps in the Superimposition view. The visualized sequence is often deformed into the incomprehensible shape, reversing the direction and running out of the space allocated for rendering.

to provide a robust control over the already shown proteins, their deletion, and upload into the tool. In addition, it presents the possibility to temporarily remove one or more proteins from any of the visualization tools, avoid rendering of gaps, render only one or no secondary structures ribbons, manage colours of the protein chains, and center to one of the proteins, setting it as a reference chain the other ones are aligned to. However, none of the initially sought functionality indeed works with the visualization views.

4.2 Drawbacks of the Former Application

Despite all the bright ideas that were put into the former comparative visualization tool, its unfinished state is immediately noticeable. Even if we overlook the missing or false functionality in the control panel, the Juxtaposition and Superimposition views render the compared sequences too densely. The indication of gaps in the sequences is too severe and often overdraws the surrounding molecular structures. On a closer look, even these structures do not look like they are positioned in the rendered canvas accurately. Indeed, the computed sequence alignment is not optimal for some parts of the compared protein chains (Figure 4.2).

One of the main drawbacks is the performance of the gap insertion algorithm. Its output often contains more gaps than necesssary, and the length and frequency of the inserted gaps are also questionable. Finally, the overall layout contains repetitive elements, specifically the navigation lines of the sequence alignment. Although they provide an

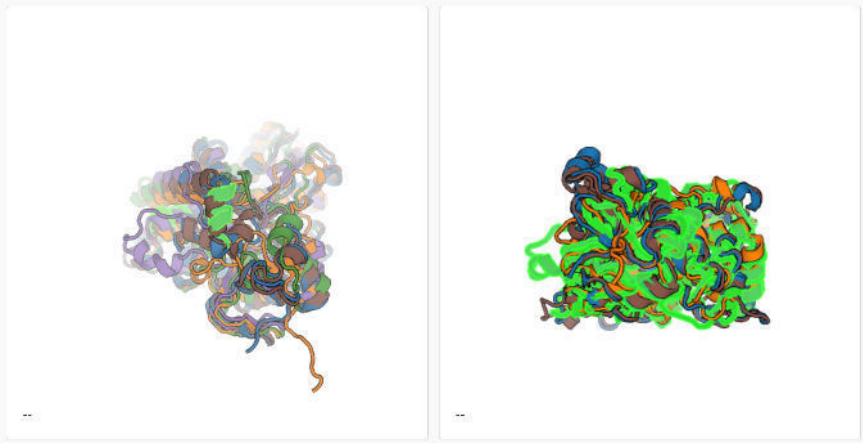

Figure 4.3: Defects of the PV viewer. On the left: after the selection of an amino acid in the 3D viewer, the view is centred to this amino acid and the rest of the 3D structure is suppressed from the view using fog. On the right: selecting more secondary structures, five, in this case, may result in very cluttered visualization.

excellent tool for interaction, they do not convey the information about the inserted gaps very legibly and occupy too much space (Figure 4.1 – 1200 height pixels are not enough for complete rendering of five protein chains).

Then there are some smaller flaws, which underline the incomplete state of the application. The selected secondary structure cannot be unselected in the PV viewer, only replaced by another one. The text field on the bottom of the viewer canvas is empty at first, but after the completed information revelation, it illustrates two dashes. After the double-click action, the PV viewer will centre the camera focus on the selected amino acid (Figure 4.3) left, unable to recenter to the whole model, only by choosing another amino acid. Proceeding to the sequential visualization tools, they often run out of the SVG coordinate space, dropping molecule structures that are too far from the baseline. The interaction elements give an impression of a prototype, considering non-aligned horizontal sliders and widely changing cursor style.

However, the most significant drawback is the absence of the upload and management of already uploaded PDB files, which makes the original application unusable. It renders only the records that are hard-coded into the source code as the absolute paths to the PDB files. In addition to that, the proteins have to be structurally prealigned; otherwise, the PV viewer would render the 3D protein structures far apart, thus inefficient for the comparative visualization. Therefore, here lies one of the essential requirements for our new implementation – including automatic alignment method which would perform structural alignment for PV viewer as well as the sequence alignment. However, this task turned out to be more difficult than expected, mostly because of the source code of the original application and outdated technologies used in it.

4.2.1 Application Design Drawbacks

For the implementation design, the application is built upon multiple web technologies, including Gulp [45], Webpack [46], and various JavaScript modules, such as minify, browserify, and babelify [47]. While these extensions vastly simplify the development process for a skilled JavaScript programmer, they make the understanding of the code much more difficult. Many of them got updated to the version that differs from the used one too much or became deprecated, so it would be time-consuming to rewrite them into the present-day state. Also, even the sole JavaScript architecture is hard to comprehend. It consists of six classes, and even though the pipeline executed by the main init.js class is clear and straightforward (Figure 4.4), the more profound understanding became much more complicated, mostly because of unnecessary dependencies between the JavaScript classes (Figure 4.5).

JavaScript is a weakly typed programming language, which means that the variables are typed dynamically during the execution of the program, and they do not need to be statically specified. The keyword *var* declares all containers for storing data values. Therefore, it is necessary to have access to the documentation about the architecture design to understand the data flow and the structures used for calculation of the visualization models. Unfortunately, there was no documentation provided.

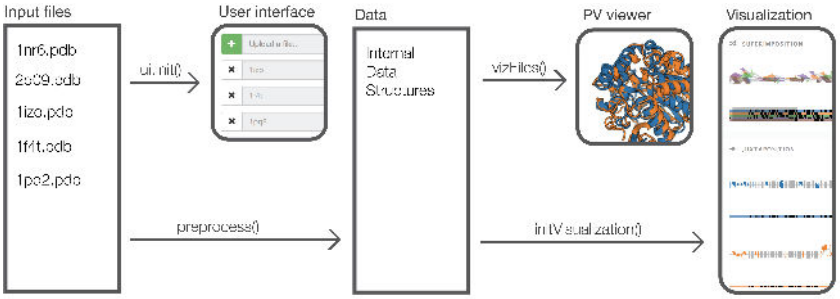

Figure 4.4: The former application front-end pipeline. The idea is very simple: initialise the user interface according to the input files, process the input files, and calculate the visualization views upon the processed data.

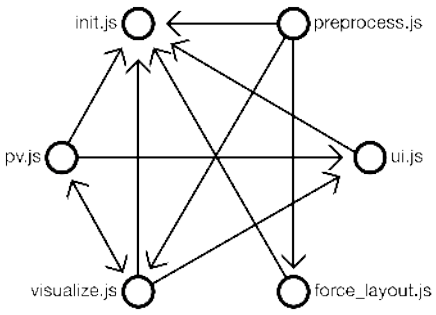

Figure 4.5: Dependencies of all used JavaScript classes in the former application. Every use of the functionality from a foreign method is marked as an arrow. The diagram shows the very complicated code structure design, which is hard to read.

Another characteristic of a JavaScript application is that all its tasks run asynchronously. Many techniques can be used to serialize some processes (e.g, calculate the Juxtaposition only after the complete execution of the gap insertion algorithm), such as callback functions and Promises [48]. The Promise is a JavaScript object that represents the eventual completion (or failure) of an asynchronous operation, and its resulting value. However, the use of multiple nested Promises may degrade the readability of the source code, which also happened in this case.

4.2.2 Summary of Implementation Drawbacks

In summary, the most important flaws of the former solution are the following:

- Missing upload/management of PDB files
- Inefficient and incorrect gap insertion algorithm
- Missing structural alignment
- Unclear representation of the compared secondary structures
- Non-functioning GUI
- Complicated architecture design and no documentation

Therefore, the main aim of our newly proposed design and implementation was to overcome these limitations and create a fully working tool for comparative visualization of protein secondary structures.

5 CVASS – Application Design and Implementation

The primary goal of this thesis is to present a functioning solution, which uses the principles of modern visualization of protein structures, it builds upon the fundaments of the state-of-the-art protein alignment algorithms. It benefits from the qualities of the former demo application, while it redesigns and repairs all its defects and flaws. In contrast to the former application, we aimed for the minimalistic appearance and easy-to-use interface. Its design does not provide any client-server communication and relies purely on absolute paths to the preprocessed data. However, the client-server approach is essential for an appropriate alignment computation, since it is too difficult to be computed in a browser with moderate results.

This section will describe the detailed description of the design and implementation of the proposed Comparative Visualization and Analysis of Secondary Structures (CVASS) application, starting with the client-server scheme, the alignment computation and file handling on the server side, and visualization methods and webpage design in client's browser.

5.1 Design of the Client-Server Application

The newly proposed application should not only serve for the demonstration purposes, but it also should be able to handle the uploaded PDB files, align them, and calculate the comparative visualization views upon them. Therefore, the application's structure ought to be implemented as a client-server model, where a client communicates with a server through the World Wide Web. The server's API should be able to react to either a request for a home page or a request to upload the PDB files for alignment. After that, the home page adapts to the new set of compared proteins and recalculates the visualization views.

The final, minimalistic design of the CVASS architecture, is shown in Figure 5.1. The figure depicts a client-server model, that is responsible for uploading the PDB files to the server, execution of the alignment

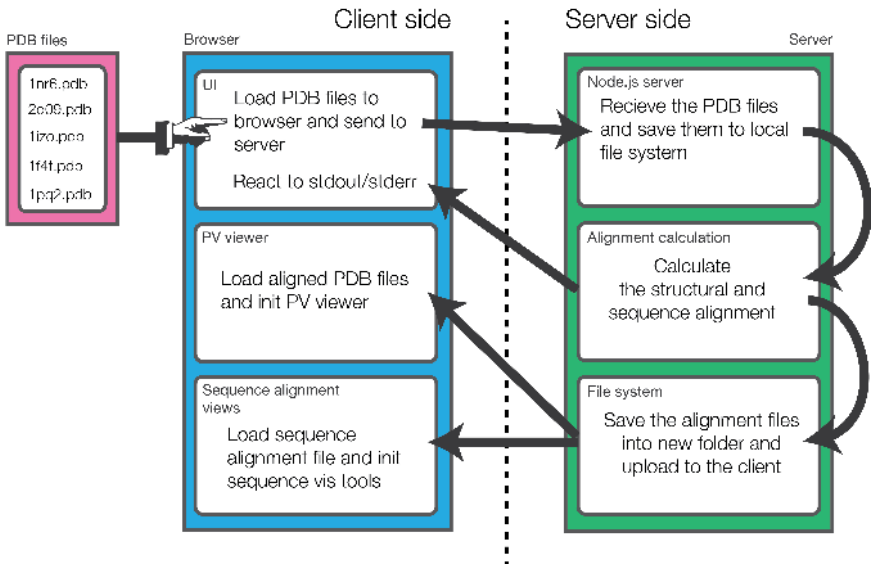

Figure 5.1: Design of the Client-Server web application. The input PDB files are submitted by the client through the browser interface and sent to the server. Afterwards, the alignment is computed, and its result is sent back to the browser for recomputing the visualization views.

pipeline script, loading the result of the alignment process back to the client's side, and calculating the visualization views.

In order to achieve the server-side functionality, while keeping the simplicity, the Express.js [49] framework for Node.js [50] was chosen as the fundamental technology for the CVASS application. The simple sample of Express.js application and its basic structure can be seen in Figure 5.2. The further development was very straightforward, mostly because of the Node Package Manager (npm) [51]. The npm is currently the world's largest software registry of building blocks of code and it makes the process of adding more functionality very simple. There were several packages installed in the CVASS, including the Express.js framework.

Additional packages from npm used in the CVASS:

- FileSystem – used for file operations

```
1  var express = require('express');
2  var app = express();
3
4  app.get('/', function(req, res) {
5    res.send('Hello World!');
6  });
7
8  app.listen(3000, function() {
9    console.log('Example app listening on port 3000!');
10 });
```

Figure 5.2: Example of the Express.js server application. The sample imports the Express module and creates the Express application object. This object then specifies the route definition for an HTTP *GET* request, which returns a string as the response. Finally, it starts up the server, listening on a specified port.

- BodyParser – node.js middleware used for parsing incoming request bodies
- Multer – node.js middleware responsible for handling multiple uploading files
- Exec – used for calling a child process, in our case shell script containing the alignment computation pipeline

The final implementation of the server-side API is thus simple. After accessing the port (i.e, on *GET* HTTP method, client requests data from a specified source), the server will send the requested front-end HTML file (see section 6.1), which contains the visualization views. If the data is sent to the server (i.e, the *POST* HTTP request is called), the BodyParser module extracts the names from the upcoming protein files and fills the list of all files to be uploaded. After that, the FileSystem module creates the specified upload file and saves the input list, then the Multer module receives and saves the data in that file. The number of the possibly uploaded PDB files is currently restricted to eight. Finally, the server pipeline is called by the Exec module.

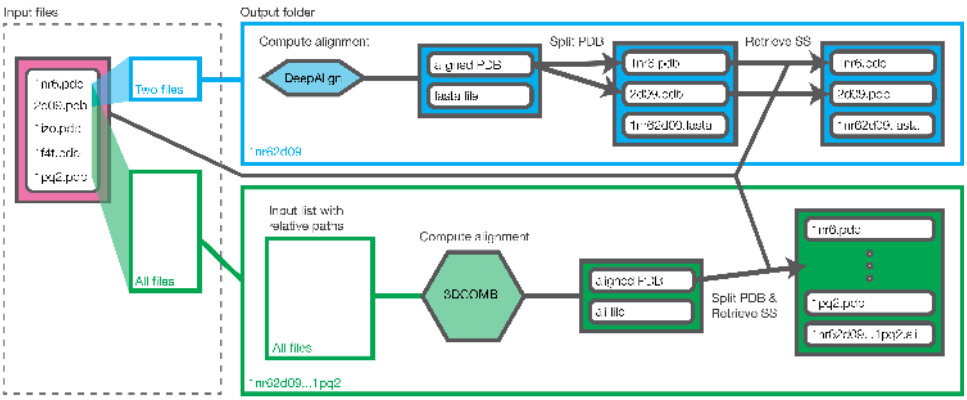

Figure 5.3: Server pipeline illustration. With uploaded PDB files and their list saved in text file taken as an input, and according to the number of proteins to be aligned, it executes one of the two branches. The output consists of the output folder, structurally aligned PDB files and fasta/ali file containing the sequence alignment, all placed into the output folder.

5.2 Server Pipeline

Once the PDB files and their list are stored in the server memory, the alignment process begins, and the server pipeline is initialised (Figure 5.3). The pipeline consists of a short bash script, which decides wheather to run the DeepAlign (in case of two PDB files on input) or 3DCOMB (in case of more than two input files) binary. It is also responsible for the creation of expected output folder and the correct placement of output files into this folder. Finally, the script takes care of the proper assignment of the secondary structures into the aligned PDB files (discussed in Section 3.3.2).

The bash script is executed only after the successful completion of the writeFile Promise. After that, the Node.js web server runs the shell executable file [52], which reads the input list of PDB files. First, a folder with the name composed of all uploaded PDBs is made in the path *./public/uploads*, with the root of the path in server's home folder. Consequently, the alignment process is started. In case of two input files, it executes the DeepAlign binary on these two PDB files,

generating four output files and storing them into the newly created folder, sharing the same name as the folder. If the alignment process fails, the error message is sent to the client's side of the application and will be displayed as an alert in the client's browser.

The DeepAlign output files are the following:

- .pdb – PDB file with ATOM records of the new coordinates after the structure alignment
- .fasta – pairwise structure alignment in the FASTA format
- .local – more readable structure alignment format, containing the RMSD (Root Mean Square Deviation) and the conformational letter
- .score – output score and rigid-body transformation

In case of the three and more input PDB files, it executes 3DCOMB binary with the following five output files:

- .pdb – PDB file with ATOM records of the new coordinates after the structure alignment
- .ali – pairwise structure alignment in the FASTA format
- .rmt – rigid-body transformation matrix for every input sequence
- .rms – column conservation and RMSD records
- .score – length of the chains, RMSD, and average TM score after sequence alignment

Only the first two output files are suitable for the purpose of this thesis. The file containing the sequence alignment (.fasta or .ali) contains just a name of the compared protein with a sequence of characters (either gaps or amino acids) and does not require more adjustments. On the other hand, the output PDB file merges all compared protein chains into one multi-sequence structure and does not include the relative position of the secondary structures; thus the additional processing is required. First, the file is split into new PDB files using the text processing language AWK [53]. Then, the information about the

Table 5.1: Time performance of the server pipeline from the moment it receives the input PDB files until it results in success. Tested on MacBook Pro with 2,6 GHz Intel Core i7 processor.

Proteins	Number of proteins	Challenging alignment computation	Time in seconds
1izo, 1nr6	2	*No*	0.86 s
1nr6, 2d09	2	*Yes*	1.06 s
1nr6, 1izo, 1f4t	3	*No*	0.78 s
1nr6, 2d09, 1pq2	3	*Yes*	1.81 s
1nr6, 1izo, 1f4t, 2d09, 1pq2	5	*Yes*	3.48 s

secondary structure is extracted from the input PDB files and inserted into the output PDB files. Finally, the final adjustments are made, such as linking secondary structure records to the correct PDB chain, utilising Unix utility for text parsing and transformation sed [54]. At this point, all the data necessary for the calculation of visualization views is ready and the server results with *success*.

The performance of this pipeline can be seen in Table 5.1. As we can see, even on a personal computer with a mobile processor, the calculation does not require a long computation time. However, both algorithms for the protein alignment require full PDB header and correct file format with all termination symbols for successful execution. While it does not have problems with any file downloaded from the Protein Data Bank [55], it failed to process some of the test PDB files provided by the CAVER Analyst tool [10].

6 CVASS – Visualization Design and Implementation

The visual part of the CVASS application is vastly influenced by the former application, maintaining the idea of the combination of the 3D viewer and quasi-sequential view with hinted relative position of the protein secondary structures. On the other hand, some elements of the former solution, such as the overcomplicated control panel and repetitious sequence alignment lines, are superfluous and should be altered. Because of the complicated nature of the original source code, as well as no documentation of the code, it was decided to redesign the whole front-end part from scratch and make more minimalistic form, which would be easier to understand.

6.1 Front-end Pipeline

The new solution also inherited the same technologies, namely JavaScript for the front-end architecture and communication with Node.js server, D3.js for the visualization views and the PV viewer for the 3D structural alignment view. Unlike in the former solution, the CVASS was built solely on the newly written code and no additional modules or libraries were added. This approach resulted in the clear and simple web application, which is easy to extend and optimise. The code architecture design can be seen in Figure 6.1.

Figure 6.2 describes the concept of initialisation of all visualization structures, which can be executed repetitively. The front-end JavaScript application will load the data processed on the server side directly from the server's memory, using *express.static* built-in middleware function in Express.js [56]. The path to a folder, for which the function is invoked, is specified and it enables the client to have access to the folder. The whole computation is called from the main *redrawAll()* function, which calls methods located in the external JavaScript source code files, logically divided according to their function. The data structures (see Section 6.1.1 for the detailed description) are invoked in the prime *index.html* file, other methods operate with these data structures.

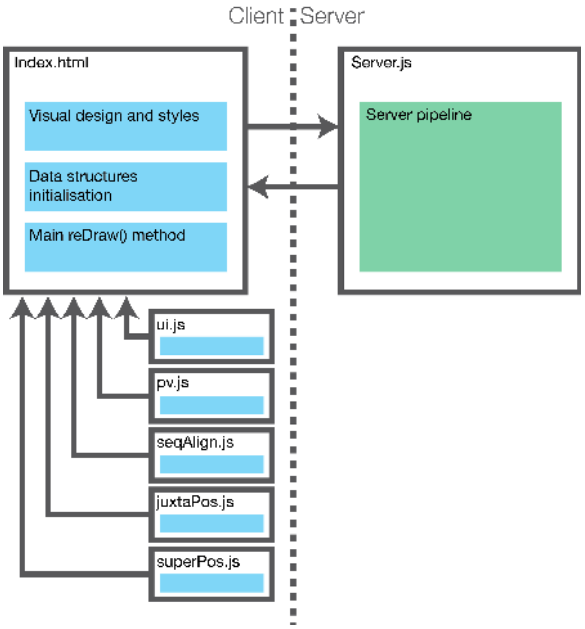

Figure 6.1: Architecture design of the CVASS application. The server side is responsible just for the alignment process, the client's side handles the upload and calculation of visualization views.

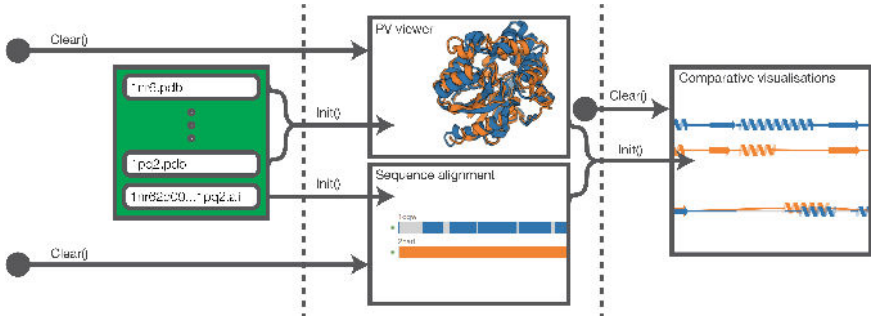

Figure 6.2: CVASS visualization pipeline. On the contrary to the former front-end pipeline (Figure 4.4), the *preprocess* and the *user interface* stages are omitted. The new pipeline is more suitable for the repetitive execution, because it requires less amount of data to load – only the PDB files for the PV viewer and .fasta file for the Sequential alignment view, the rest is derived from this data.

6.1.1 Data Structures

One of the main difficulties of understanding the former implementation was the absence of the data structure documentation. As mentioned before in section 4.2, JavaScript is a weakly typed language, which makes it problematic to track the flow of data inside the application. To overcome this obstascle in the CVASS, the global data structures are either primitive data types or arrays, all of them are initialized at the beginning of the main method, and their explanation will be given in this chapter. The data structures (Figure 6.3) can be divided into four major categories:

- Data structures changed by user interaction, by uploading a new set of proteins or selecting some secondary structures.
- Data structures initialized by the PV viewer – positions of secondary structures, coordinates of amino acids in 3D space, differences of these coordinates with regard to the reference chain.
- Data structures initialized by the sequence alignment viewer – positions of gaps in sequentially aligned structures.
- Data structures derived from the previous two categories – positions of secondary structures in sequentially aligned structures, conversion tables between structure and sequence alignment.

Most of the arrays are two dimensional, having an array of record for every sequence. Some structures, such as *seqAlignmentRects*, are optimisations for the better performance: after selection, only the affected rectangles of the sequential alignment view will be changed, instead of redrawing the whole canvas. This approach maintains the simplicity of the code and enables fast implementation of new features. However, as the solution will grow in complexity, it will require to aggregate the data into composite structures and report it to detailed documentation.

6.1.2 Layout Design

On the first run, the application shows the comparative visualization of two prealigned proteins [42, 57], indicating the design of the user

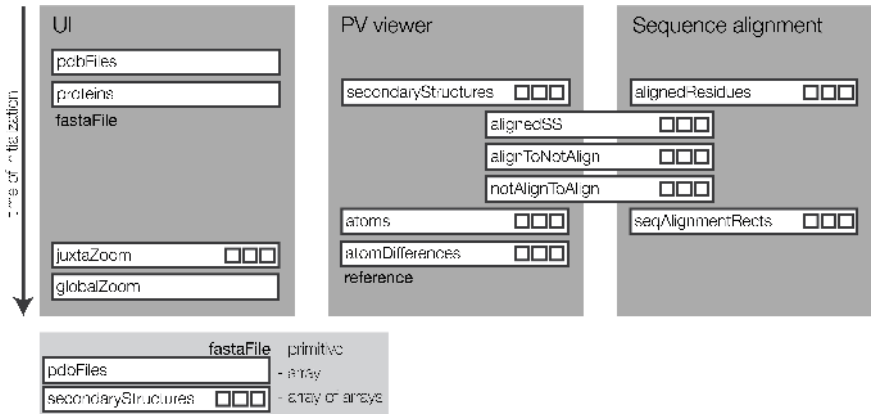

Figure 6.3: Diagram of the data structures used in the CVASS, their time of initialisation, and a the process that invoked them.

interface and visualization tools. The layout of the application (Figure 6.4) is rendered as a single page in the visually minimalistic form. The header and the footer are the Bootstrap [58] elements, presenting the name of the application and the university affiliation. Right under the header is located the upload form, composed of the input button, the informative read-only text field, and the upload button, aligned in a row. Below the input form, the body of the web page contains four different protein alignment views:

- PV viewer – top left
- Sequence alignment view – top right
- Juxtaposition view – middle right, under the Sequence alignment view
- Superimposition view – bottom

The comparative alignment views are not visually divided in the layout design and share the same colour scheme, which immediately indicates their connection. Once one or more of the secondary structures are selected and highlighted in one view, they are automatically highlighted in the other views as well. The front-end design has minimum control elements, just small fading dots between the PV viewer

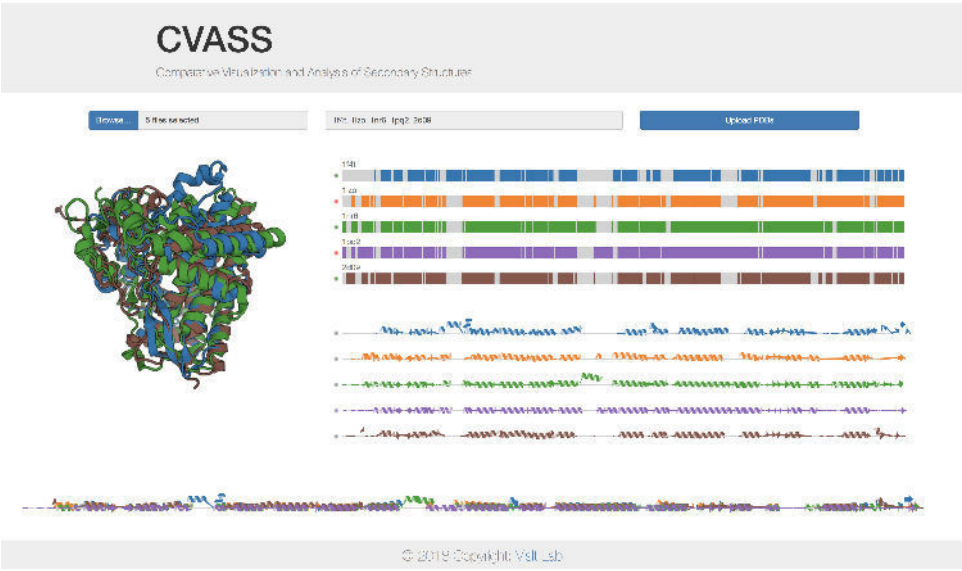

Figure 6.4: Web layout design of the CVASS application. Right below the header is located the upload form with the information text field. The 3D representation is rendered using the PV viewer and next to it and below there are the Sequential alignment view, the Juxtaposition and the Superimposition views. The views are not visually divided and labelled. Up to five protein chains can be displayed in the FullHD resolution without the need of scrolling.

and the sequence alignment views. The dots next to the Sequence alignment view, coloured either green or red, indicate whether the protein sequence is displayed in the PV viewer or not. On click, it changes the sequence's state and its colour. It helps the orientation in the PV viewer in case of many compared proteins and a cluttered 3D view.

The dots next to the Juxtaposition view, coloured either grey or the colour of the protein sequence they belong to, indicate the reference chain for the comparative visualization. Only one sequence can be chosen at one time; thus only one dot at once does not have the grey colour (i.e., it serves as a radiobutton choosing the reference chain). Once again, the change of the reference chain can be done by a simple click action.

6.2 Visualization Methods

The following section contains the description of design and implementation of all visualization methods used in the CVASS application, including the brief explanation of the interaction techniques used in these methods.

6.2.1 PV Viewer for Structure Alignment Visualization

To visualize the structure alignment and 3D models of compared proteins (seen in Figure 6.4 on the top left position), the same JavaScript module as in the former solution, the PV viewer [8], is used. Even the same factory settings were preserved – proteins are displayed as Cartoon ribbons, the zoom level is adjusted such that all objects are visible on screen and occupy as much space as possible, the view camera is centred to the last added protein structure. However, the process of loading the protein structures into the module is significantly simplified. Instead of preprocessing the PDB files into the complicated data structures and then initialising the PV viewer on these structures, we load PDB files directly into the module.

The native PDB loader in the PV viewer has many advantages over the one written in the former application. It went through the long process of development and thus there is a higher chance it will be more efficient and better optimized. Also, it simplifies the process of linking the sequentially aligned sequences (with gaps) to structurally aligned sequences (without gaps). The information about the position of secondary structures, together with respective positions and references to the 3D model, is extracted directly during the PV viewer loading process from the uploaded structures, saving computational time, which would normally be used by the preprocessing stage.

Lastly, many imperfections from the previous version of PV viewer used in the former application were fixed, starting with the option to deselect the previously selected secondary structure. The viewer can no longer focus on one selected amino acid and is always centred on the best possible view. The text field, displaying the information about the selected amino acid, is now excluded and this kind of information is shown in the informative text field on the top of the page.

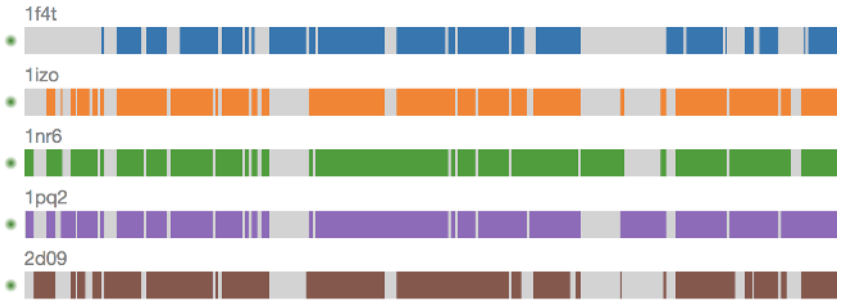

Figure 6.5: The Sequence alignment view. The colour of the segments indicates whether it is a sequence structure or a gap. The dot on the left controls whether the protein sequence is rendered in the PV viewer or not.

6.2.2 Sequence Alignment Visualization Methods

More noticeable changes were made in the sequential alignment visualizations. The new arrangement proposes to utilize three sequence visualizations instead of two, combining all lines that represent gaps in the sequence alignment from the former solution into the interactive view, placed right next to the PV viewer (Figure 6.5). The primary motivation was to efficiently indicate gaps coming from the alignment process because they are not always easily visible in the other two composite representations. It also allows to zoom and control the zoom of more sequences at one time, which was not possible before. On top of that, the combined view saves the screen space and allows more effective comparisons between proteins.

Unlike the Juxtaposition and Superimposition views, which render the sequences as secondary structures, the sequence alignment tool displays every amino acid in the protein chain, allowing additional, more precise type of selection. Instead of highlighting a series of residues of a secondary structure, only the selected amino acid (if chosen in the PV viewer) or last amino acid of the secondary structure (in the sequential visualization tools) is highlighted (Figure 6.6). It is suitable for exploring molecular structures in the vicinity of inserted

Figure 6.6: Example of the amino acids highlighted in the sequence alignment view.

gaps. The view also writes the name of the protein chain above the protein's sequence alignment visualization.

The remaining two visualization tools are very much like the concept presented in the reference paper. They both render the secondary structures as cartoon ribbons in a similar way as the PV viewer. The visualization model makes protein's secondary structures, their position and length easily recognisable. Additionally, the adjusted position and rotation against the reference chain causes the differences between the reference and compared chain effortlessly understandable. Due to the constant threshold, it is impossible for a ribbon to run out of the rendering canvas. This was one of the drawbacks of the former application.

The visualization view is continuous, without any unexpected twists and dubious parts. Because of the sequence alignment viewer, we can afford to display gaps less apparently, so now they do not interfere with explicit representation of secondary structures (see Figure 6.7). The Superimposition view (see Figure 6.8) is located in the bottom of the page, below the PV viewer (instead of the right side), allowing it to extend the rendering canvas to the whole width of the screen.

The orientation of every secondary structure ribbon is calculated as the offset of the beginning and ending amino acid of the secondary structure. The offset is the Euclidean distance in the 3D coordinate system position. Once both offsets are computed, the length of the

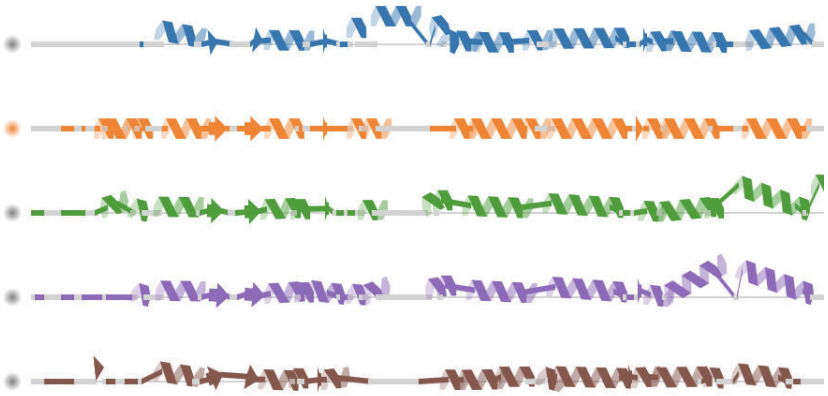

Figure 6.7: The Juxtaposition view example. The further is the secondary structure from the baseline, the more significant is the real distance between these parts of molecule. Dots on the left indicate the reference chain, the orange sequence is the reference one in this case.

Figure 6.8: The Superimposition view composed of five protein sequences.

ribbon is adjusted according to their positions and the glyph is translated and rotated to match these positions. Only alpha-helices and beta-sheets are shifted, coils and gaps (represented as lines) are for the better readability always converging to the baseline. All chains, except for the reference one, also contain a semi-transparent line to indicate the zero difference from the reference chain.

6.2.3 Interaction with the Comparative Visualizations

The option to select a chosen secondary structure in any view and highlight it in all secondary structure will be highlighted in all the views is the fundamental form of interaction. The selections can be easily

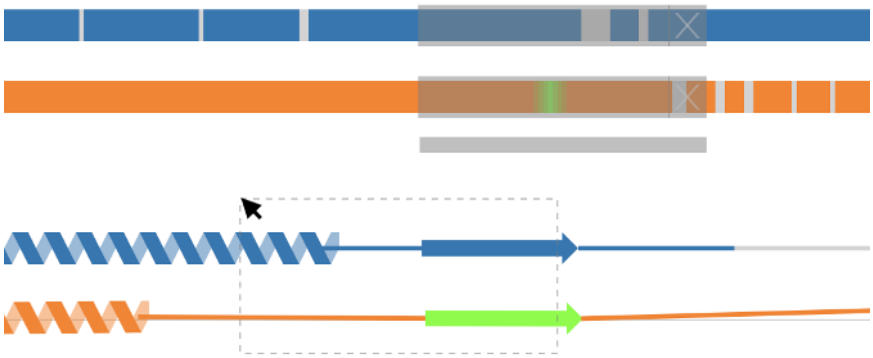

Figure 6.9: An example of the interaction in the comparative visualizations. The beta-sheet secondary structure in the orange sequence is highlighted in both Juxtaposition and Sequence alignment views. The Juxtaposition view is also zoomed in, and the zooming range is indicated in the Sequence alignment view as a semi-transparent rectangle. Finally, the transparent selection area with the dashed stroke is illustrated in the Juxtaposition view, defining its shape by dragging the mouse cursor through the view.

deselected by clicking on the highlighted structure one more time (in the PV viewer by a double-click). As mentioned above, there are more ways to interact with the application, such as removing/reinserting a protein into the PV viewer, changing the reference sequence for the comparative sequence alignment views, and adjusting the views.

All views, with the exception of the Sequence alignment view, can be zoomed (Figure 6.9). While in the case of the PV viewer all the adjustments (zooming and rotation) are handled automatically by the JavaScript module, the Juxtaposition offers an innovative approach to view interaction. The range of the displayed protein sequences can be set from the full-range view by a selection tool. It has a rectangular shape and is transparent with the dashed line as a stroke. The selection starts rendering once the mouse cursor is pressed in the area of the Juxtaposition view, it changes its rectangular shape according to the position of the dragged cursor and defines the selection after the mouse cursor is released. After that, the Juxtaposition view is redrawn, and all protein sequences that were inside the selection will be zoomed into the selection range. Once the protein chain is zoomed in, it has to be reset before being able to be zoomed in once again.

For every zoomed sequence, a semi-transparent handle appears in the Sequence alignment view. The position of the handle is determined by the range of the selection and can be moved along the whole length of the sequence, translating also the range of the view for a selected sequence. The range can be reset by clicking on the handle. If two or more sequences are zoomed in, another handle is rendered under the Sequence alignment view, which controls all the handles above it. It also defines the range of visualization of the Superimposition view.

The performance of the visualization pipeline can be seen in Table 6.1.

Table 6.1: Time performance of the visualization pipeline from the moment it receives the aligned PDB files until it renders all visualization views. Tested on MacBook Pro with 2,6 GHz Intel Core i7 processor.

Proteins	Number of proteins	Challenging to align	Time of alignment computation	Time of visualization computation
1izo, 1nr6	2	*No*	0.86s	0.63s
1nr6, 2d09	2	*Yes*	1.06s	0.66s
1nr6, 1izo, 1f4t	3	*No*	0.78s	0.89s
1nr6, 2d09, 1pq2	3	*Yes*	1.81s	0.97s
1nr6, 1izo, 1f4t, 2d09, 1pq2	5	*Yes*	3.48s	2.63s

7 Conclusion

In this work, we analyzed the ideas and the implementation of the work of Kocincová et. al [1], pointing out the innovative approach of their comparative visualization of protein sequences, which highlights the significance of secondary structures in these proteins. Then we identified the drawbacks of the former method and proposed alternative solutions. In addition, we described the drawbacks of the former gap insertion algorithm and introduced the new modern approach of sequential and structural alignment calculation, DeepAlign for the pair-wise alignment and 3DCOMB for the multiple alignment.

Based on these foundations, we built the CVASS application – a web tool for comparative visualization and analysis of protein secondary structures. The architecture design of the application is adapted for the upload of new to-be-compared PDB protein files, the server-side alignment process and the interactive visualization tools, which combine 1D and 3D views of aligned protein sequences. The text part of the thesis provides the detailed description of the web tool, which is written in the JavaScript programming language, using the D3.js library for the rendering of the visualization views.

The application is designed in a minimalistic, scalable fashion, leaving space for the optimisation and the customisation according to the release requirements. At the time of submission of this thesis, the CVASS is available on one of the faculty servers for the public usage. However, the testing on a large scale with multiple users is yet to be completed. Also, the work is based solely on the computer science oriented literature; therefore testing by the biochemists is needed prior to the deployment of the application.

Bibliography

1. KOCINCOVÁ, Lucia; JAREŠOVÁ, Miroslava; BYŠKA, Jan; PARULEK, Julius; HAUSER, Helwig; KOZLÍKOVÁ, Barbora. Comparative Visualization of Protein Secondary Structures. *BMC Bioinformatics*. 2017, vol. 18. ISSN 1471-2105. Available from DOI: `http://dx.doi.org/10.1186/s12859-016-1449-z`.
2. BERMAN, Helen M.; WESTBROOK, John; FENG, Zukang; GILLILAND, Gary; BHAT, T. N.; WEISSIG, Helge; SHINDYALOV, Ilya N.; BOURNE, Philip E. The Protein Data Bank. *Nucleic Acids Research*. 2000, vol. 28, no. 1, pp. 235–242. Available from DOI: `10.1093/nar/28.1.235`.
3. BERMAN, Helen; HENRICK, Kim; NAKAMURA, Haruki. Announcing the worldwide Protein Data Bank. *Nature Structural Biology*. 2003, vol. 10.
4. KOZLOWSKI, Lukasz P. *Nucleic Acids Research*. Proteome-pI: proteome isoelectric point database. 2017.
5. B. FULTON, Alice; B. ISAACS, William. Titin, a huge, elastic sarcomeric protein with a probable role in morphogenesis. 1991, vol. 13, pp. 157–61.
6. RICHARDSON, Jane S. The Anatomy and Taxonomy of Protein Structure. In: ANFINSEN, C.B.; EDSALL, John T.; RICHARDS, Frederic M. (eds.). Academic Press, 1981, vol. 34, pp. 167–339. Advances in Protein Chemistry. ISSN 0065-3233. Available from DOI: `https://doi.org/10.1016/S0065-3233(08)60520-3`.
7. GUEX, Nicolas; PEITSCH, Manuel C.; SCHWEDE, Torsten. Automated comparative protein structure modeling with SWISS-MODEL and Swiss-PdbViewer: A historical perspective. *ELECTROPHORESIS*. 2009, vol. 30, no. S1, pp. S162–S173. ISSN 1522-2683. Available from DOI: `10.1002/elps.200900140`.
8. BIASINI, Marco et al. SWISS-MODEL: Modelling protein tertiary and quaternary structure using evolutionary information. 2014, vol. 42.
9. TELLER, Swizec. *Data Visualization with D3.Js*. Packt Publishing, 2013.

10. KOZLÍKOVÁ, Barbora et al. *Caver Analyst 1.0*. 2014. Available also from: `http://www.caver.cz`.

11. PARISI, Tony. *WebGL: Up and Running*. 1st. O'Reilly Media, Inc., 2012.

12. ZHOU, P.; SHANG, Z. 2D molecular graphics: a flattened world of chemistry and biology. *Brief. Bioinformatics*. 2009, vol. 10, no. 3, pp. 247–258.

13. HUTCHINSON, E. G.; THORNTON, J. M. HERA–a program to draw schematic diagrams of protein secondary structures. *Proteins*. 1990, vol. 8, no. 3, pp. 203–212.

14. STERNBERG, M. J.; THORNTON, J. M. On the conformation of proteins: the handedness of the beta-strand-alpha-helix-beta-strand unit. *J. Mol. Biol.* 1976, vol. 105, no. 3, pp. 367–382.

15. VEERAMALAI, M.; GILBERT, D. A novel method for comparing topological models of protein structures enhanced with ligand information. *Bioinformatics*. 2008, vol. 24, no. 23, pp. 2698–2705.

16. LASKOWSKI, R. A. PDBsum new things. *Nucleic Acids Res.* 2009, vol. 37, no. Database issue, pp. D355–359.

17. RICHARDSON, J. S. beta-Sheet topology and the relatedness of proteins. *Nature*. 1977, vol. 268, no. 5620, pp. 495–500.

18. STIVALA, A.; WYBROW, M.; WIRTH, A.; WHISSTOCK, J. C.; STUCKEY, P. J. Automatic generation of protein structure cartoons with Pro-origami. *Bioinformatics*. 2011, vol. 27, no. 23, pp. 3315–3316.

19. SCHÄFER, Tim; MAY, Patrick; KOCH, Ina. Computation and Visualization of Protein Topology Graphs Including Ligand Information. In: *German Conference on Bioinformatics (GCB'12)*. 2012, pp. 108–118.

20. STOLTE, C.; SABIR, K. S.; HEINRICH, J.; HAMMANG, C. J.; SCHAFFERHANS, A.; O'DONOGHUE, S. I. Integrated visual analysis of protein structures, sequences, and feature data. *BMC Bioinformatics*. 2015, vol. 16 Suppl 11, pp. S7.

21. Bioinformatics: Sequence and Genome Analysis. Second Edition. By David W Mount. *The Quarterly Review of Biology*. 2005, vol. 80, no. 1, pp. 109–109. Available from DOI: `10.1086/431054`.

22. NOTREDAME, Cédric; HIGGINS, Desmond G; HERINGA, Jaap. T-coffee: a novel method for fast and accurate multiple sequence alignment1. *Journal of molecular biology*. 2000, vol. 302, no. 1, pp. 205–217.

23. LATHROP, R. H. The protein threading problem with sequence amino acid interaction preferences is NP-complete. *Protein Eng*. 1994, vol. 7, no. 9, pp. 1059–1068.

24. KRISSINEL, Evgeny. On the relationship between sequence and structure similarities in proteomics. *Bioinformatics*. 2007, vol. 23, no. 6, pp. 717–723. Available from DOI: 10.1093/bioinformatics/btm006.

25. WANG, Sheng; PENG, Jian; XU, Jinbo. Alignment of distantly related protein structures: algorithm, bound and implications to homology modeling. *Bioinformatics*. 2011, vol. 27, no. 18, pp. 2537–2545. Available from DOI: 10.1093/bioinformatics/btr432.

26. HENRY, Vincent J; BANDROWSKI, Anita E; PEPIN, Anne-Sophie; GONZALEZ, Bruno J; DESFEUX, Arnaud. OMICtools: an informative directory for multi-omic data analysis. *Database*. 2014, vol. 2014.

27. HOLM, Liisa; SANDER, Chris. Protein structure comparison by alignment of distance matrices. *Journal of molecular biology*. 1993, vol. 233, no. 1, pp. 123–138.

28. SHINDYALOV, Ilya N; BOURNE, Philip E. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein engineering*. 1998, vol. 11, no. 9, pp. 739–747.

29. ZHANG, Yang; SKOLNICK, Jeffrey. TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic acids research*. 2005, vol. 33, no. 7, pp. 2302–2309.

30. KIM, Changhoon; LEE, Byungkook. Accuracy of structure-based sequence alignment of automatic methods. *BMC bioinformatics*. 2007, vol. 8, no. 1, pp. 355.

31. KABSCH, Wolfgang; SANDER, Christian. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*. Vol. 22, no. 12, pp. 2577–2637. Available from DOI: 10.1002/bip.360221211.

32. HOLM, Liisa; SANDER, Chris. Protein Structure Comparison by Alignment of Distance Matrices. *Journal of Molecular Biology*. 1993, vol. 233, no. 1, pp. 123–138. ISSN 0022-2836. Available from DOI: https://doi.org/10.1006/jmbi.1993.1489.

33. WANG, Sheng; MA, Jianzhu; PENG, Jian; XU, Jinbo. Protein structure alignment beyond spatial proximity. *Scientific Reports*. 2013, vol. 3, pp. 1448. Available also from: http://dx.doi.org/10.1038/srep01448.

34. LAFFERTY, John; MCCALLUM, Andrew; PEREIRA, Fernando CN. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

35. WANG, Sheng; ZHENG, Wei-Mou. CLePAPS: fast pair alignment of protein structures based on conformational letters. *Journal of bioinformatics and computational biology*. 2008, vol. 6, no. 02, pp. 347–366.

36. WOUTER G TOUW Coos Baakman, et al. A series of PDB related databases for everyday needs. *Nucleic Acids Research*. 2015, vol. 43(Database issue), pp. D364–D368.

37. KABSCH, W; SANDER, C. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*. 1983, vol. 22, no. 12, pp. 2577–2637. ISSN 0006-3525. Available from DOI: 10.1002/bip.360221211.

38. CMBI. *DSSP Explanation* [online]. 2012 [visited on 2018-03-24]. Available from: http://swift.cmbi.ru.nl/gv/dssp/.

39. SZABÓ, Tibor. *Dynamická vizualizácia sekundárnych štruktúr v molekulách proteínov [online]*. 2010. Available also from: https://is.muni.cz/th/dljy8/.

40. FRISHMAN, Dmitrij; ARGOS, Patrick. Knowledge-based protein secondary structure assignment. *Proteins: Structure, Function, and Bioinformatics*. Vol. 23, no. 4, pp. 566–579. Available from DOI: 10.1002/prot.340230412.

41. NEWMAN, Janet et al. Haloalkane Dehalogenases: Structure of a Rhodococcus Enzyme, *Biochemistry*. 1999, vol. 38, no. 49, pp. 16105–16114. Available from DOI: 10.1021/bi9913855. PMID: 10587433.

42. BENARROCH, Delphine; EGLOFF, Marie-Pierre; MULARD, Laurence; GUERREIRO, Catherine; ROMETTE, Jean-Louis; CANARD, Bruno. A Structural Basis for the Inhibition of the NS5 Dengue Virus mRNA 2-O-Methyltransferase Domain by Ribavirin 5-Triphosphate. *Journal of Biological Chemistry*. 2004, vol. 279, no. 34, pp. 35638–35643. Available from eprint: `http://www.jbc.org/content/279/34/35638.full.pdf+html`.

43. WEBGL-OPENGL, ES. *2.0 for the Web*. Khronos Group, http://www.khronos. org/webgl, 2014.

44. MYATT, G.J.; JOHNSON, W.P. *Making Sense of Data III: A Practical Guide to Designing Interactive Data Visualizations*. Wiley, 2011. Books 24x7 IT PRO. ISBN 9781118121603. Available also from: `https://books.google.cz/books?id=nTpjoNgJQOMC`.

45. MAO, Jed. *Developing a gulp edge : the streaming build system*. Santa Rosa, CA: Bleeding Edge Press, 2014. ISBN 978-1-939902-14-6.

46. VEPSLINEN, Juho. SurviveJS-Webpack and React: From apprentice to master. 2016.

47. SMIRNOV, Egor. 2015. Available also from: `http://egorsmirnov.me/2015/05/25/browserify-babelify-and-es6.html`.

48. PARKER, D. *JavaScript with Promises: Managing Asynchronous Code*. O'Reilly Media, 2015. ISBN 9781491930793. Available also from: `https://books.google.cz/books?id=G7rBCQAAQBAJ`.

49. MARDAN, Azat. *Express. js Guide: The Comprehensive Book on Express. js*. Azat Mardan, 2014.

50. TILKOV, Stefan; VINOSKI, Steve. Node. js: Using JavaScript to build high-performance network programs. *IEEE Internet Computing*. 2010, vol. 14, no. 6, pp. 80–83.

51. HERRON, David. *Node Web Development*. Packt Publishing Ltd, 2013.

52. *Node.js v10.0.0 Documentation*. Available also from: `https://nodejs.org/api/child_process.html`. Accessed: 2018-03-30.

53. STUTZ, Michael. *Get started with GAWK: AWK language fundamentals*. September, 2006.

54. MCILROY, M Douglas. A Research Unix reader: annotated excerpts from the Programmer's Manual. 1971.

55. BERMAN, HM; WESTBROOK, J; FENG, Z; GILLILAND, G; BHAT, TN; WEISSIG, H; SHINDYALOV, IN; BOURNE, PE. The Protein Data Bank Nucleic Acids Research, 28, 235-242. *URL: www. rcsb. org Citation*. 2000.

56. *Express.js Static Files*. 2018. Available also from: `https://expressjs.com/en/starter/static-files.html`. Accessed: 2018-04-20.

57. FRANKEN, S. M.; ROZEBOOM, H. J.; KALK, K. H.; DIJKSTRA, B. W. Crystal structure of haloalkane dehalogenase: an enzyme to detoxify halogenated alkanes. *EMBO J.* 1991, vol. 10, no. 6, pp. 1297–1302.

58. BHAUMIK, Snig. *Bootstrap Essentials*. Packt Publishing Ltd, 2015.