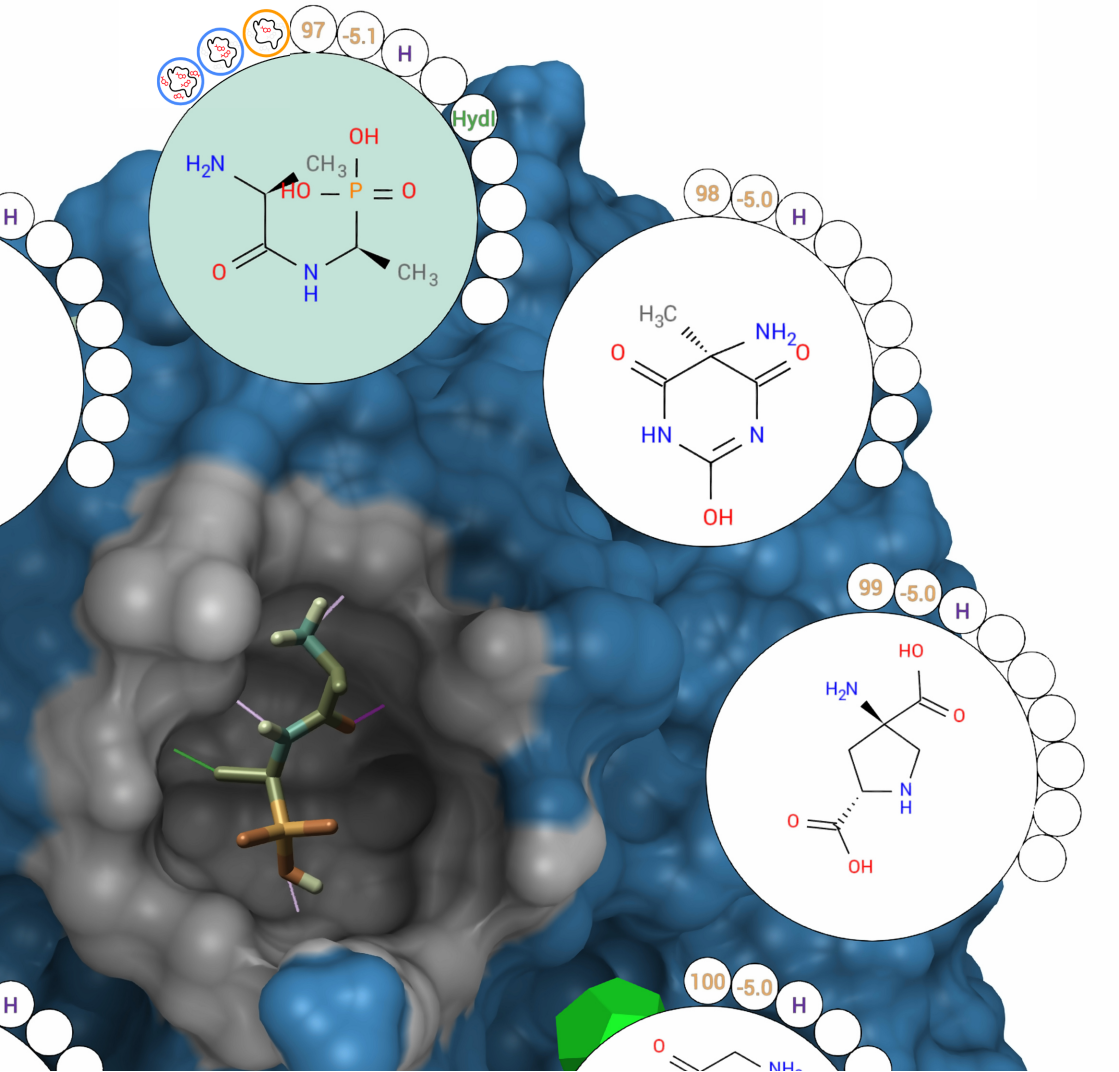
Interactive Visual Analysis of Protein-Ligand Interactions

Marco Schäfer



Interactive Visual Analysis of Protein-Ligand Interactions

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

Marco Schäfer

aus Marburg

Tübingen

2024

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	17.06.2025
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Michael Krone
2. Berichterstatter:	Prof. Dr.-Ing. Oliver Kohlbacher

Acknowledgments

First and foremost, I want to thank my supervisor, Michael Krone, for his continuous support, guidance, and inspiration throughout my doctoral studies. It was always a pleasure to work with you, and I truly enjoyed the time. I also want to thank my second supervisor, Oliver Kohlbacher, for his valuable feedback and support during the course of my research.

Special thanks go to my colleague and coauthor Nicolas Brich for the many engaging, interesting, and often amusing discussions that made this time even more enjoyable. I am especially grateful to my project partners and coauthors from Masaryk University in Brno, Barbora Kozlíková and Jan Byška, for the excellent and pleasant collaboration on the PROLINT project.

Of course, I would also like to extend my gratitude to all my other coauthors and collaborators from various institutions who contributed to this work in numerous ways: David Bednář, Patrick C. F. Buchholz, Thomas Ertl, Gloria Fackelmann, Valerio Ferrario, Florian Frieß, Juan J. Franco-Moreno, Pedro Hermosilla, Matej Lang, Sérgio M. Marques, Jürgen Pleiss, Timo Ropinski, Alexander S. Rose, Tobias Ritschel, Karsten Schatz, Philipp Thiel, and Pere-Pau Vázquez. I also want to thank my colleagues at IBMI, especially Kay Nieselt and the members of her IT group, who made it a pleasure to work there.

This research was funded by the German Research Foundation (DFG) through the projects PROLINT (project number 391088465) and IVM (project number 437702916), with additional support from the University of Tübingen. I am thankful for all the financial support that enabled this work.

Last but not least, my heartfelt thanks go to Daniela, my sister Nadine, and my friends and parents for their constant support throughout this journey. Countless thanks to all of you for always believing in me and encouraging me.

Contents

List of Abbreviations and Acronyms

ANN	Artificial Neuronal Network
AO	Ambient Occlusion
API	Application Programming Interface
ATP	Adenosine Triphosphate
BLAST	Basic Local Alignment Search Tool
CLISD	Compressed Ligand-Interaction Sequence Diagram
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
D3	Data-Driven Documents
DBSCAN	Density-Based Spatial Clustering
DLSS	Deep Learning Super Sampling
DNA	Deoxyribonucleic Acid
EC	Enzyme Commission
EM	Electron Microscopy
fps	Frames Per Second
GCNN	Graph CNN
GLSL	OpenGL Shading Language
GPGPU	General Purpose Computation on GPUs
GPU	Graphics Processing Unit
GUI	Graphical User Interface
GVF	Gradient Vector Flow
HLSL	High-Level Shading Language
HMM	Hidden Markov Model
kNN	k-Nearest Neighbor
LOD	Level Of Detail
LSTM	Long Short-Term Memory
MD	Molecular Dynamics
MDS	Multidimensional Scaling
MLP	Multilayer Perceptron
MM	MultiMer
mmCIF	Macromolecular Crystallographic Information File
mRNA	Messenger RNA
MRSA	Methicillin-resistant *Staphylococcus aureus*
MSA	Multiple Sequence Alignment

NLP Natural Language Processing

NMR Nuclear Magnetic Resonance

OpenCL Open Computing Language

OpenGL Open Graphics Library

PCA Principal Component Analysis

PDB Protein Data Bank

PDBQT Protein Data Bank (PDB), Partial Charge (Q), Atom Type (T)

PK Phosphoketolase

PLIP Protein-Ligand Interaction Profiler

PLY Polygon File Format

PP Pyrophosphate

PYR Pyrimidine

RAM Random-Access Memory

RBD Receptor Binding Domain

RCSB Research Collaboratory for Structural Bioinformatics

RMSD Root Mean Square Deviation

RMSF Root Mean Square Fluctuation

RNA Ribonucleic Acid

RNN Recurrent Neural Network

RT Ray Tracing

SAS Solvent Accessible Surface

SCOP Structural Classification of Proteins

SDF Structure Data File

SES Solvent Excluded Surface

SIMD Single Instruction, Multiple Data

SIMT Single Instruction Multiple Thread

SM Streaming Multiprocessor

SSBO Shader Storage Buffer Object

TK Transketolase

TKC Transketolase C-terminal

TM Template Modeling

tRNA Transport RNA

UPGMA Unweighted Pair Group Method with Arithmetic mean

VA Vertex Array

VBO Vertex Buffer Object

vdW Van der Waals

VRAM Video RAM

Abstract

Protein-ligand interactions play a crucial role in many biological processes, including signal transduction, gene regulation, and enzymatic reactions. Understanding these interactions is fundamental and yet challenging since they are a vast and diverse field. To address this, abstract and interactive visualizations are required to enable visual analysis that facilitates both intuitive understanding and access to complex protein-ligand data. Due to their inherent three-dimensional nature, the interaction data and the protein and ligand structures are particularly well-suited for spatial visualization. However, to answer certain research questions, derived abstract 3D and 2D representations are utilized as well. An interactive depiction of the complex protein-ligand interplay provides profound insights into various research areas. It enhances the understanding of conformational changes of proteins, biochemical processes in and between cells, and disease mechanisms, which is pivotal in medical research. Furthermore, it supports the investigation of ligand transport processes, such as how a ligand approaches the active site of an enzyme and which conditions must be fulfilled for certain interaction types to occur. This also comprises valuable information for protein engineers and ligand designers, enabling targeted mutations of amino acids to improve the binding affinity of protein-ligand complexes or to increase catalytic rates of enzymes, which are essential for advancements in biotechnology and drug development.

This thesis focuses on developing interactive visualization methods for studying protein-ligand interactions, facilitating the exploration and visual analysis of complex and large data sets. Existing free and commercial tools provide visualizations of interactions and the molecular structures involved. However, they lack specialized functionality and features, often offering no or only basic capabilities for evaluation and visual analysis, e.g., of results from virtual screenings or molecular dynamics simulations. Furthermore, these solutions are also not designed to handle large datasets effectively, and they are typically specialized in a detailed ligand-centered analysis, frequently neglecting the protein perspective. However, a comprehensive approach also needs to include the investigation and analysis of protein surfaces and their physico-chemical properties, since it is not only the shape but the chemical composition as well that determines possible interactions.

In this thesis, various methods are discussed that enable the interactive visual analysis of protein-ligand interactions and elucidate the field from different

points of view. Individual methods use machine learning to extract specific structural features of proteins for classification. In addition, the protein structure is also considered together with its physico-chemical properties, which enables the search for functionally similar proteins. These methods enable deriving common features of interactions and the learning of conditions that are crucial for the formation of certain interaction types. Subsequently, the thesis focuses on dynamic data from computational chemistry, particularly approaches visualizing molecular dynamics simulations. One application uses a highly GPU accelerated implementation of a protein surface visualization method, allowing an exploratory analysis on current consumer computers. A further method uses an aggregation approach to create a dashboard-like presentation of entire molecular dynamics simulations. It supports identifying binding sites and most interacting amino acids, highlighted in a novel sequence diagram. Furthermore, molecular docking is also considered to study interactions. The visual analysis application InVADo was designed to enable a ligand-centered, comprehensive analysis and evaluation of docking results. It guides scientists to areas of interest and supports users in verifying existing hypotheses by enriching the results through post-docking analysis.

In addition, the applicability and usefulness of the approaches were underpinned by expert evaluations. The methods developed for the interactive visual analysis of protein-ligand interactions contribute to gaining new insights into the functions of the molecular machinery of life and to acquiring a more sophisticated understanding of the underlying mechanisms, which is crucial for systems biology, biotechnology, and pharmaceutical research.

German Abstract
—Zusammenfassung—

Protein-Liganden Interaktionen spielen eine entscheidende Rolle in vielen biologischen Prozessen, wie der Signaltransduktion, der Genregulation und enzymatischen Reaktionen. Das Verständnis dieser Wechselwirkungen ist von grundlegender Bedeutung und stellt zugleich eine besondere Herausforderung dar, da es sich um ein weites und vielfältiges Gebiet handelt. Um diese Herausforderung zu bewältigen, sind abstrakte und interaktive Visualisierungen erforderlich, welche eine visuelle Analyse ermöglichen, die sowohl ein intuitives Verständnis als auch den Zugang zu komplexen Protein-Liganden Daten erleichtert. Aufgrund ihrer inhärenten dreidimensionalen Natur eignen sich die Interaktionsdaten sowie die Protein- und Liganden-Strukturen besonders gut zur räumlichen Visualisierung. Jedoch werden zur Beantwortung bestimmter Forschungsfragen auch abgeleitete abstrakte 3D- und 2D-Repräsentationen verwendet. Eine interaktive Darstellung des komplexen Protein-Liganden Zusammenspiels ermöglicht tiefere Einblicke in verschiedene Forschungsbereiche. Sie verbessert das Verständnis von Konformationsänderungen von Proteinen, biochemischen Prozessen in und zwischen Zellen, sowie von Krankheitsmechanismen, was in der medizinischen Forschung von zentraler Bedeutung ist. Des Weiteren unterstützt sie die Untersuchung von Liganden-Transportprozessen, z. B. wie sich ein Ligand einer aktiven Stelle eines Enzyms nähert und welche Bedingungen für das Auftreten bestimmter Interaktionsarten erfüllt sein müssen. Dies umfasst zugleich wertvolle Informationen für Protein-Ingenieure und Liganden-Designer, die es ihnen ermöglichen, gezielte Mutationen von Aminosäuren durchzuführen. Diese Veränderungen können die Bindungsaffinität von Protein-Liganden Komplexen verbessern oder katalytische Raten von Enzymen erhöhen, was von entscheidender Bedeutung für Fortschritte in der Biotechnologie und Arzneimittelentwicklung ist.

Die Arbeit befasst sich mit der Entwicklung interaktiver Visualisierungsmethoden zur Untersuchung von Protein-Ligand-Interaktionen, die die Erforschung und visuelle Analyse komplexer als auch großer Datensätze erleichtern. Bestehende kostenlose und kommerzielle Lösungen ermöglichen die Visualisierung von Wechselwirkungen und den beteiligten molekularen Strukturen, jedoch fehlen ihnen spezialisierte Funktionen und Eigenschaften. Sie bieten oft keine oder nur grundlegende Möglichkeiten zur Evaluierung und visuellen Analyse von, z. B. Ergebnissen eines virtuellen Screenings oder Molekulardynamik (MD)-

Simulationen an. Des Weiteren sind diese Lösungen nicht darauf ausgelegt, große Datensätze effektiv zu verarbeiten und sie sind in der Regel auf eine detaillierte, liganden-zentrierte Analyse spezialisiert, die häufig eine Betrachtung der Proteinperspektive vernachlässigt. Ein umfassender Ansatz muss jedoch auch die Untersuchung und Analyse von Proteinoberflächen und ihren physikalisch-chemischen Eigenschaften umfassen, da nicht nur die Form, sondern auch die chemische Komposition mögliche Wechselwirkungen bestimmt.

In dieser Arbeit werden verschiedene Methoden diskutiert, die eine interaktive visuelle Analyse von Protein-Ligand-Interaktionen ermöglichen und das Forschungsfeld aus verschiedenen Blickwinkeln beleuchten. Einzelne Methoden nutzen maschinelles Lernen, um spezifische Strukturmerkmale von Proteinen zur Klassifizierung zu extrahieren. Zusätzlich wird die Proteinstruktur auch zusammen mit ihren physikalisch-chemischen Eigenschaften betrachtet, was die Suche nach funktionell ähnlichen Proteinen erlaubt. Diese Methoden ermöglichen es, gemeinsame Merkmale von Interaktionen abzuleiten und Bedingungen zu lernen, die für die Bildung bestimmter Interaktionstypen entscheidend sind. Darauffolgend befasst sich die Arbeit mit dynamischen Daten aus der computergestützten Chemie, insbesondere mit Ansätzen zur Visualisierung von MD-Simulationen. Eine Anwendung nutzt dazu eine stark GPU-beschleunigte Umsetzung einer Proteinoberflächen-Visualisierungsmethode, die eine explorative Analyse auf aktuellen Consumer-Computern ermöglicht. Eine weitere Methode verwendet einen Aggregationsansatz, um eine dashboard-ähnliche Darstellung ganzer MD-Simulationen zu erstellen. Sie unterstützt die Identifizierung von Bindungsstellen und der am meisten interagierenden Aminosäuren, die in einem neuartigen Sequenzdiagramm hervorgehoben werden. Außerdem wird zur Untersuchung der Interaktionen auch molekulares Docking betrachtet. Die visuelle Analyseanwendung InVADo wurde so konzipiert, dass sie eine liganden-zentrierte, umfassende Analyse und Evaluierung von Docking-Ergebnissen ermöglicht. Sie lenkt den Fokus der Wissenschaftler zu interessanten Bereichen und unterstützt Benutzer bei der Verifizierung bestehender Hypothesen, indem sie die Ergebnisse durch Post-Docking-Analysen weiter anreichert.

Zudem wurde die Anwendbarkeit und Nützlichkeit der Ansätze mittels Experten-Evaluationen untermauert. Die für die interaktive und visuelle Analyse von Protein-Ligand-Interaktionen entwickelten Methoden, tragen dazu bei, neue Einblicke in die Funktionsweise der molekularen Maschinerie des Lebens zu erhalten und ein differenzierteres Verständnis der zugrunde liegenden Mechanismen zu erlangen. Dies ist von entscheidender Bedeutung für die Systembiologie, die Biotechnologie und die pharmazeutische Forschung.

CHAPTER 1

Introduction

> Visualization gives you answers to questions you didn't know you had.
>
> *Ben Shneiderman, University of Maryland* [Kir12]

Proteins are the building blocks of life. From giving structure to our cells to enabling us to walk, they are involved in almost all chemical reactions in living organisms. Their wide range of crucial functions is made possible primarily by their interplay with small molecules called ligands, besides interactions with other proteins. These protein-ligand interactions constitute a significant part of the complex biological relations and regulations that enable life. Therefore, their investigation enables us to get a more sophisticated understanding of how the molecular machinery works together. It supports gaining comprehensive insights into the biological processes and pathways, which in turn enables comprehension of disease mechanisms and may lead to new treatments.

Thanks to the tremendous development in computational power, bioinformatic technologies, and efficient algorithms in the last decades, more protein data can be obtained, and more complex molecular systems can be simulated than ever before. Additionally, millions of ligand structures are nowadays available. These developments led to the creation of large databases such as the RCSB Protein Data Bank (PDB) [Ber+00] providing currently ~207,000 structures, and the ZINC database [Irw+20] providing ~1.4 billion ligands. On the algorithmic

side, the investigation of protein-ligand interactions is often performed using methods like Molecular Dynamics (MD) simulation, which predicts their behavior over time and tests their response at various environmental conditions, and molecular docking, which determines the ligand conformation, orientation, and binding affinity. The MD simulation can be preceded by molecular docking, whereby the top-scored ligand conformations are selected and further investigated.

These virtual experiments are significantly more cost-effective than wet lab experiments. This advantage becomes even more important with the aforementioned constantly increasing number of available proteins and ligands that can be investigated to study interactions or to screen in silico for new drug candidates. Hence, the investigation of protein-ligand interactions provides the theoretical foundation and makes it crucial in drug discovery, design, and the search for novel drug targets. It contributes to solving current issues, such as the increasing problem of antibiotic resistance [CM21] and the development of better cancer therapy approaches like cancer immunotherapy [FDB16]. Moreover, studying protein-ligand interactions is indispensable in the field of structural and systems biology to get more insights into pathways, cellular signaling as well as immune reactions, gene regulation, and protein functions. Those interactions also play an important role in analytical chemistry. An example is the frequently used biotin-avidin bond, which is the strongest non-covalent protein-ligand interaction. It is used in affinity chromatography, a method of separating a protein from a mixture, which enables the purification of protein solutions. Protein-ligand research has its industrial application in biotechnology as well, as it is used to produce biofuel or other valuable substances such as detergents and drugs.

As mentioned earlier, studying the nanoscopic protein-ligand interaction helps to gain insights into their macroscopic behavior and to determine their binding affinity. This is especially interesting for protein and ligand designers. Possible scenarios are aiming to optimize the binding affinity of a drug working as an inhibitor of a pain receptor, increasing the catalytic rate of an enzymatic reaction, or searching for potential reaction or interaction partners.

The protein-ligand interaction data received from virtual experiments are complex. This is partly due to the number of possible interaction types and the number of molecules observed. Also contributing to the complexity is the temporal component when studying the behavior of molecular systems. Due to these challenges and the spatial complexity, a visual inspection is required to analyze these data and evaluate the results. It is important to take advantage of

the special imprint of the human being on the sense of seeing. Via visualization, these data can be studied and presented more easily than a description with words or plain text. Generally, the goal of visualization consists of generating a graphical representation of raw data that enables an observer to comprehend these data, gain new insights, and derive decisions from it. That is, to enable a comprehensive understanding of complex concepts and the multifaceted relationships between biomolecules, especially of protein-ligand interaction.

1.1 Motivation

Visual analysis of biomolecules is common and essential in academic and industrial (e.g., pharmaceutical) research. It is crucial to understand their interactions and to evaluate the results of tools that serve the necessary data for studying complex biomolecular relationships. Out of this area, this dissertation focuses on the interactive visual analysis of protein-ligand interactions. They are a vast and diverse field and understanding them is essential and simultaneously challenging. For the visualization of molecules, there are established representations called molecular models, each representing different molecule properties. They range from simple models such as the ball-and-stick to complex ones such as molecular surfaces, e.g., solvent-based models [Ric78] that show the interface of a protein interacting with its environment. A detailed and comprehensive overview of visualizing biomolecular structures is given in the state-of-the-art report by Kozlíková et al. [Koz+17].

Examples of established tools that apply various of these representation models and enable the visualization of protein-ligand interactions are *PyMol* [SD20], *VMD* [HDS96], *Chimera* [Pet+04], and *Mol** [Seh+21]. Several tools focus specifically on protein-ligand interactions and their abstraction as 2D representations, including *ProteinPlus* [Fäh+17] with *PoseView* [SR10] as well as *LigPlot+* [LS11]. There are also commercial tools that provide visualization of interactions and molecular structures, focusing on drug discovery, molecular docking, and molecular design. These include *SAMSON Connect*[1], *Molsoft* (ICM Modelling, ICM Browser Pro)[2] and *Schrödinger Maestro* (Ligand Designer, Glide)[3].

[1] *SAMSON Connect*. https://www.samson-connect.net (accessed 07/10/2024).
[2] *Molsoft LLC.*: https://www.molsoft.com/icm_browser_pro.html (accessed 07/10/2024).
[3] *Schrödinger LLC.*: https://www.schrodinger.com/products/maestro (accessed 07/10/2024).

These tools are widely used and continuously extended, but they lack specialized functionality and features. They are not interactive for large data sets and partly provide basic interaction visualizations. In addition, they often have no or only basic possibilities for evaluation and visual analysis of virtual screening results or MD simulations. The aforementioned tools are designed for detailed analysis and lack the handling of these large data sets. Furthermore, they focus on the ligand perspective and consider the protein perspective less, e.g., the investigation and comparison of protein surfaces and their physicochemical properties.

When considering the protein perspective, the study of the protein surface is important since it is the interface to the outer environment and plays a critical role in interaction. Not only their chemical composition determines possible interactions, but it is their shape as well that needs to be investigated. Therefore, it is important to elucidate approaches that consider the protein sequence as well as their three-dimensional shape. Thereby, machine learning methods and novel comparison approaches can be used to perform a general and hierarchical classification of proteins. In addition, the physico-chemical properties need to be taken into account, as they are a most crucial factor in mediating interactions, along with shape. Together, this enables the comparison of proteins and allows learning and deriving common features of interaction, fold classification, and function determination, all contributing to a comprehensive understanding of protein-ligand interactions.

On the other hand, focusing on protein-ligand interactions from a more ligand-centered perspective requires developing approaches for visual analysis and evaluation. The objective is to analyze the temporal behavior of proteins and ligands, assess their binding affinity, identify the types of interactions involved, and comprehend the transport mechanisms occurring across the surface. An example is the investigation of how a ligand approaches the active site of an enzyme. The utilized visualization methods need to enable interactive visualization. This is crucial when working with dynamic three-dimensional data as obtained from MD simulation or complex spatial data when creating a comprehensive view of thousands of molecular docking results obtained from virtual screening experiments. For example, the user needs to inspect the spatial data from different perspectives, which is enabled by adjusting a virtual camera. Methods to investigate protein-ligand interaction can produce huge data sets with several terabytes, which are hard to visualize and where it is challenging to get the important information. This requires approaches that consider aggregation, clustering, filtering, and efficient rendering techniques to

allow an interactive visualization. These interactive visual analysis approaches contain special views on complex and large molecular interaction data, which are combined with targeted data enrichment, enabling facilitated and improved decision-making.

In this thesis, the research directions mentioned focus solely on interactive visualization and investigation of proteins and ligands, focusing mainly on their interactions. That means the approaches going to be presented will not cover interactions where covalent bonds are formed, that is, no chemical reactions. They do not consider intermolecular interactions and include no new methods of interaction force calculations. They will be based on known interaction types, including hydrogen bonds, hydrophobic interactions, and others. The non-covalent interactions are described in Section 2.3.2 and a detailed description of them is given by Berg et al. [Ber+15] and by Biedermann and Schneider [BS16]. The new approaches aim to enable a visually comprehensive and interactive inspection of protein-ligand interactions. They will contribute to gaining new insights into how the molecular machinery works and to get a more sophisticated understanding of the underlying mechanisms crucial for systems biology, medical chemistry, and pharmaceutical research.

1.2 Outline and Contribution

This section gives an overview of the structure of this thesis and summarizes the contributions of the author to the discussed publications. Figure 1.1 represents the logical structure of the following chapters and sections, following the idea of investigating protein-ligand interactions from different perspectives. Therefore, the publications go successively from a protein to a ligand perspective. All subsequent publications can be taken from List 1.1 and were co-authored by the author's Ph.D. advisor, Michael Krone. The detailed contribution shares of the individual co-authors can be found at the end of this section in Tables 1.1 to 1.5.

Chapter 2 (Fundamentals) provides the theoretical background that is necessary throughout the rest of the dissertation. It begins with the fundamentals of visualization and its technical application in the form of rendering. This is followed by a closer view of general-purpose computations on graphics hardware, which is instrumental in solving biological research questions and visualizing the results. A biological background provides the required understanding for those research questions by elucidating the proteins' structure and their interactions with ligands. This is complemented by a brief introduction to machine learning and an overview of molecular models commonly used to visualize proteins

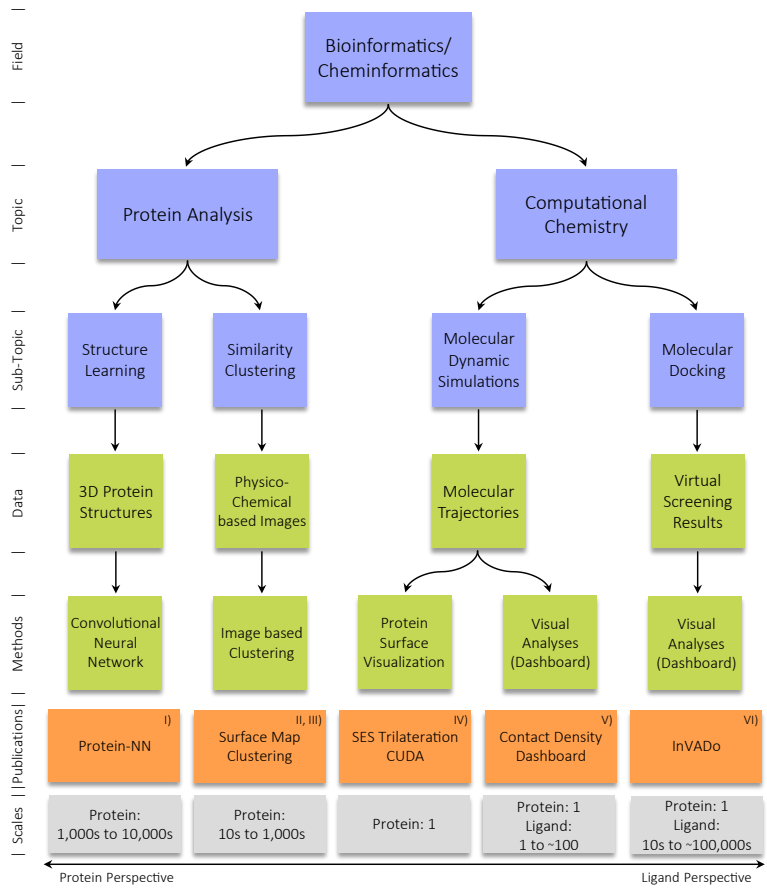


Figure 1.1 — Schematic representation of the publication relations. The publications (orange) are arranged horizontally, starting with a work focusing on the protein perspective and moving to works increasingly focusing on a ligand perspective (gray). The sub-topics (blue) range from *protein structure learning* to *protein similarity clustering* (Protein Analysis) and go to *molecular dynamic simulations* and *molecular docking* (Computational Chemistry). The main data and methods used are listed below the topics (green). The contributing works are marked with abbreviations that indicate their topic. Details of the numbered publications are given in List 1.1. - **I** Protein NN (Section 3.2); **II, III** Surface Map Clustering (Section 3.3), **IV** SES Trilateration CUDA (Section 4.2), **V** Contact Density Dashboard (Section 4.3), **VI** InVADo (Section 4.4)

and ligands. In this context, a student work co-supervised by the author of this thesis is included, which is about an enhanced interactive protein sequence diagram. It has a peer-reviewed abstract and was presented as a poster on the VCBM 2020 (see Section 2.5). This first part of the chapter reviews the aforementioned subjects based on previous work found in the literature. The chapter closes with a description of how the visualization research was enabled by introducing the prototyping framework *MegaMol* [Gro+15; Gra+19] and giving a rough overview of the used development environment in a web-based context.

Chapter 3 (Protein Analysis) presents methods focusing on the protein perspective and their classification as shown in the left main branch of Figure 1.1. They aim to compare, analyze, and learn the proteins' structure and certain properties, including protein functions, fold classes, enzyme classes, and various physico-chemical properties. The first method [Her+21] uses a deep neural network with a convolution and pooling approach for learning the protein sequence as well as the 3D structure (Section 3.2). It was successfully tested for assigning protein fold and enzyme classes. The author of this dissertation contributed to the conceptualization, training data preparation, and evaluation.

In Section 3.3, a comparison approach is discussed based on protein surface images [Sch+20b]. It hierarchically clusters protein surface maps using various methods and builds on the earlier work of Krone et al. [Kro+17]. They presented a method for summarizing the complex protein surface as a 2D image representation called a Molecular Surface Map. This map can encode topology and various physico-chemical properties, e.g., hydrophobicity, mapped as color onto the protein surface. The Molecular Surface Map-based clustering method was further extended and applied to protein domain analysis in a consecutive publication [Sch+21b] showing its feasibility. The dissertation's author primarily contributed to the evaluation.

Chapter 4 (Computational Chemistry) considers more the ligand perspective. It focuses on the visualization and analysis of molecular trajectories obtained from MD simulations and virtual screening results derived from thousands of molecular dockings. While MD simulations try to forecast molecular behavior, molecular docking aims to predict the optimal orientation and conformation of a ligand in a binding site of a protein. The first two publications of this chapter deal with dynamic MD data using different strategies. Section 4.2 discusses a fast algorithm for the interactive visualization of a certain protein surface, namely the Solvent Excluded Surface (SES) [SK19]. The method is designed for dynamic molecular data that require a recalculation of the biochemically

interesting but complex SES in each time step. Since this is computationally expensive, the method is based on a highly parallel Graphics Processing Unit (GPU) algorithm. The author of this thesis, who is the first author of this paper, contributed to the idea and implemented and evaluated the method.

In contrast to the mentioned GPU-accelerated SES computation, which visualizes each time step, another idea is presented in Section 4.3 that follows an aggregation approach [Sch+21a]. In a decoupled preprocessing step, several terabytes of MD simulations of protein-ligand interactions are aggregated. Thereby, the focus is on ligand paths along the protein surface, as well as summarizing ligand atom contacts with residues or the time steps they are contacted. This data is interactively visualized on a rigid 3D protein surface together with a linked and enhanced 2D sequence diagram, both embedded in a web application. The author of this dissertation mainly contributed to the design and implementation of the web application, which includes an enhanced sequence diagram linked with a 3D protein surface representation. Subsequently, a co-supervised student work is briefly discussed, also dealing with MD simulations but utilizing a comparative visual analysis approach (see Section 4.3.7). The work has a peer-reviewed abstract and was presented as a poster on the VCBM 2022.

This is followed by a work that highly focuses on the ligand perspective. Section 4.4 presents a visual analysis tool for virtual screening results, which are produced by tens to hundreds of thousands of molecular dockings [Sch+24]. The tool named InVADo aims to enable scientists to evaluate their docking results by providing multiple linked 2D and 3D views. InVADo filters and spatially clusters the data and enriches it with a post-docking analysis. This includes the determination of functional groups and interaction types for the individual docking poses of the ligands. As the first author of this extensive work, the dissertation's author is the main contributor to the scientific idea and methods, its implementation, evaluation, and analysis.

Chapter 5 (Discussion and Outlook) concludes the dissertation by exploring the potential opportunities and advantages offered by the presented methods and evaluates their applicability in addressing current research questions. Finally, the chapter concludes with prospective avenues for future research, building upon the findings of this dissertation.

List 1.1 — List of Publications

I Pedro Hermosilla[6], **Marco Schäfer**[1,2], Matej Lang[8], Gloria Fackelmann[5], Pere-Pau Vázquez[11], Barbora Kozlikova[8], Michael Krone[1,2], Tobias Ritschel[12], and Timo Ropinski[6]. "Intrinsic-Extrinsic Convolution and Pooling for Learning on 3D Protein Structures." In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=l0mSUROpwY.

II Karsten Schatz[3], Florian Frieß[3], **Marco Schäfer**[1,2], Thomas Ertl[3], and Michael Krone[1,2]. "Analyzing Protein Similarity by Clustering Molecular Surface Maps." In: *EG Workshop on Visual Computing for Biology and Medicine*. 2020, pp. 103-114. DOI: 10.1111/cgf.14386.

III Karsten Schatz[3], Florian Frieß[3], **Marco Schäfer**[1,2], Patrick C. F. Buchholz[4], Jürgen Pleiss[4], Thomas Ertl[3], and Michael Krone[1,2]. "Analyzing the Similarity of Protein Domains by Clustering Molecular Surface Maps." In: *Computers & Graphics*. 2021, pp. 114-127. DOI: 10.1016/j.cag.2021.06.007.

IV **Marco Schäfer**[1,2] and Michael Krone[1,2]. "A Massively Parallel CUDA Algorithm to Compute and Visualize the Solvent Excluded Surface for Dynamic Molecular Data". In: *EG Workshop on Molecular Graphics and Visual Analysis of Molecular Data*. 2019. DOI: 10.2312/molva.20191094.

V Karsten Schatz[3], Juan José Franco-Moreno[11], **Marco Schäfer**[1,2], Alexander S. Rose[13], Valerio Ferrario[4], Jürgen Pleiss[4], Pere-Pau Vázquez[11], Thomas Ertl[3], and Michael Krone[1,2]. "Visual Analysis of Large-Scale Protein-Ligand Interaction Data." In: *Comput. Graph. Forum*. 2021, pp. 394-408. DOI: 10.1111/cgf.14386.

VI **Marco Schäfer**[1,2], Nicolas Brich[1,2], Jan Byška[7,8], Sérgio M. Marques[9,10], David Bednář[9,10], Philipp Thiel[13], Barbora Kozlíková[8], and Michael Krone[1,2]. "InVADo: Interactive Visual Analysis of Molecular Docking Data." In: *IEEE Trans. Visual. Comput. Graphics*. 2024, pp. 1984-1997 DOI: 10.1109/TVCG.2023.3337642.

List 1.2 — List of Author Affiliations

[1] Big Data Visual Analytics in Life Sciences (BDVA), University of Tübingen, Tübingen, Germany

[2] Institute for Bioinformatics and Medical Informatics (IBMI), University of Tübingen, Tübingen, Germany

[3] Visualization Research Center (VISUS), University of Stuttgart, Stuttgart, Germany

[4] Institute of Biochemistry and Technical Biochemistry, University of Stuttgart, Stuttgart, Germany

[5] Institute of Evolutionary Ecology and Conservation Genomics, Ulm University, Ulm, Germany

[6] Visual Computing Group, Ulm University, Ulm, Germany

[7] Department of Informatics, University of Bergen, Bergen, Norway

[8] Department of Visual Computing - Visitlab, Masaryk University, Brno, Czech

[9] Loschmidt Laboratories, Department of Experimental Biology and RECETOX, Masaryk University, Brno, Czech

[10] International Center for Clinical Research, St. Anne's University Hospital Brno, Czech

[11] Virtual Reality and Graphics Interaction (ViRVIG), University of Catalonia, Barcelona, Spain

[12] Virtual Environments and Computer Graphics (VECG), University College London, London, United Kingdom

[13] RCSB Protein Data Bank, San Diego Supercomputer Center, University of California, La Jolla, San Diego, California, USA

Table 1.1 — Contributions Manuscript I

Author Position & Author	Scientific Ideas	Data Generation	Analysis & Interpretation	Paper Writing
1. Hermosilla, Pedro	30 %	55 %	40 %	50 %
2. Schäfer, Marco	15 %	25 %	15 %	5 %
3. Lang, Matěj	5 %	10 %	10 %	5 %
4. Fackelmann, Gloria	5 %	10 %	10 %	5 %
5. Vázquez, Pere Pau	5 %	0 %	5 %	5 %
6. Kozlíková, Barbora	10 %	0 %	5 %	5 %
7. Krone, Michael	10 %	0 %	5 %	5 %
8. Ritschel, Tobias	5 %	0 %	5 %	5 %
9. Ropinski, Timo	15 %	0 %	5 %	15 %
Titel of Paper:	Intrinsic-Extrinsic Convolution and Pooling for Learning on 3D Protein Structures			
Publication Status:	accepted as conference paper [Her+21] (reviewed) at the International Conference on Learning Representations (ICLR), 2021			

Table 1.2 — Contributions Manuscript II

Author Position & Author	Scientific Ideas	Data Generation	Analysis & Interpretation	Paper Writing
1. Schatz, Karsten	30 %	55 %	40 %	50 %
2. Frieß, Florian	25 %	10 %	15 %	15 %
3. Schäfer, Marco	10 %	35 %	20 %	20 %
4. Ertl, Thomas	10 %	0 %	10 %	5 %
5. Krone, Michael	25 %	0 %	15 %	10 %
Titel of Paper:	Analyzing Protein Similarity by Clustering Molecular Surface Maps			
Publication Status:	accepted as conference paper [Sch+20b] (peer-reviewed) at Visual Computing for Biology and Medicine (VCBM), publisher: Eurographics Association, 2020			

Table 1.3 — Contributions Manuscript III

Author Position & Author	Scientific Ideas	Data Generation	Analysis & Interpretation	Paper Writing
1. Schatz, Karsten	30 %	40 %	35 %	45 %
2. Frieß, Florian	20 %	5 %	10 %	10 %
3. Schäfer, Marco	10 %	25 %	15 %	15 %
4. Buchholz, Patrick	5 %	25 %	15 %	15 %
5. Pleiss, Jürgen	5 %	5 %	10 %	5 %
6. Ertl, Thomas	10 %	0 %	5 %	5 %
7. Krone, Michael	20 %	0 %	10 %	5 %
Titel of Paper:	Analyzing the similarity of protein domains by clustering Molecular Surface Maps			
Publication Status:	accepted as journal paper [Sch+21b] (peer-reviewed) in Computers & Graphics (CG), 2021			

Table 1.4 — Contributions Manuscript IV

Author Position & Author	Scientific Ideas	Data Generation	Analysis & Interpretation	Paper Writing
1. Schäfer, Marco	50 %	95 %	80 %	75 %
2. Krone, Michael	50 %	5 %	20 %	25 %
Titel of Paper:	A Massively Parallel CUDA Algorithm to Compute and Visualize the Solvent Excluded Surface for Dynamic Molecular Data			
Publication Status:	accepted as conference paper [SK19] (peer-reviewed) at MolVA workshop at conference EuroVis, publisher: Eurographics Association, 2019			

Table 1.5 — Contributions Manuscript V

Author Position & Author	Scientific Ideas	Data Generation	Analysis & Interpretation	Paper Writing
1. Schatz, Karsten	30 %	35 %	40 %	45 %
2. Franco-M., Juan J.	10 %	15 %	5 %	5 %
3. Schäfer, Marco	10 %	25 %	10 %	15 %
4. Rose, Alexander S.	0 %	5 %	0 %	5 %
5. Ferrario, Valerio	5 %	10 %	10 %	5 %
6. Pleiss, Jürgen	15 %	10 %	10 %	5 %
7. Vázquez, Pere-Pau	10 %	0 %	5 %	5 %
8. Ertl, Thomas	5 %	0 %	5 %	5 %
9. Krone, Michael	15 %	0 %	15 %	10 %
Titel of Paper:	Visual Analysis of Large-Scale Protein-Ligand Interaction Data			
Publication Status:	accepted as journal paper [Sch+21a] (peer-reviewed) in Computer Graphics Forum (CGF), 2021			

Table 1.6 — Contributions Manuscript VI

Author Position & Author	Scientific Ideas	Data Generation	Analysis & Interpretation	Paper Writing
1. Schäfer, Marco	68 %	84 %	65 %	75 %
2. Brich, Nicolas	0%	10 %	5 %	0 %
3. Byška, Jan	5 %	0 %	5 %	6 %
4. Marques, Sérgio M.	0 %	3 %	5 %	3 %
5. Bednář, David	0 %	3 %	5 %	3 %
6. Thiel, Philipp	7 %	0 %	0 %	0 %
7. Kozlíková, Barbora	5 %	0 %	5 %	3 %
8. Krone, Michael	15 %	0 %	10 %	10 %
Titel of Paper:	InVADo: Interactive Visual Analysis of Molecular Docking Data			
Publication Status:	accepted as journal paper [Sch+24] (peer-reviewed) in IEEE Transactions on Visualization and Computer Graphics (TVCG) , 2024			

CHAPTER 2

Fundamentals

This chapter lays the theoretical foundation on which this thesis is built. It begins with the fundamentals of scientific visualization and its technical application by explaining the rendering pipeline as well as the structure of modern graphics hardware. After this, an introduction to the biological background is given by discussing the structure of proteins and their possibilities to interact with small molecules (ligands). This is followed by the methods of supervised and unsupervised machine learning utilized to work with biological data. Finally, the basics of visualizing molecular structures are elucidated, which is complemented with an overview of the used software infrastructure, the biochemical file formats and preprocessing steps.

2.1 Visualization and Rendering

Visualization is the creation of graphical representations of data. The purpose of visualization is to show complex information in a way that makes it easy to understand and analyze [Bro+92; MDB87]. The data can be acquired from different kinds of sources like measurements or simulations. Visualization can be categorized into two fields, including *scientific visualization* and *information visualization*. In the context of visualizing biomolecular structures, the data is numerical, spatial and potentially time-dependent. This field belongs to *scientific visualization*, which deals with methods to visualize spatial data. In contrast,

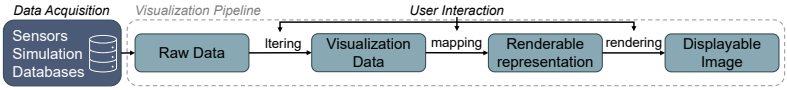


Figure 2.1 — The visualization pipeline as described by Weiskopf [Wei07]. The *data acquisition* serves the *raw data* for the visualization pipeline. It runs through different processing steps, including *filtering*, *mapping*, and *rendering* that can be adjusted by *user interaction*. The final result is a *displayable image*.

information visualization uses methods for visualizing non-spatial data. The following sections will outline the visualization process from the raw data until the rendering of the final image, and will provide an insight into a commonly used rendering pipeline. The used examples are tailored to a biochemical context.

2.1.1 Visualization Pipeline

The visualization pipeline summarizes the steps of the visualization process. Most visualization applications apply the scheme by Weiskopf [Wei07] presented in Figure 2.1, which is an extended version of the reference model described by Haber and McNabb [HM90]. It is enhanced with the data acquisition step, which is essential but not part of the actual visualization pipeline. This is due to the visualization pipeline being independent of the utilized *raw data*.

The data acquisition is the upstream process of the visualization pipeline. It provides the *raw data*. This can be obtained from many sources such as *databases* (RCSB PDB) [Ber+00], from *sensors*, e.g., a biosensor used for glucose monitoring in the context of diabetes, or from *simulations* such as MD simulations. The *raw data* are the first instance of the visualization pipeline. In order to serve as input for it, these data might need to be further filtered and prepared to obtain the *visualization data*. Possible preparation includes de-noising, clipping, interpolation, segmentation, and classification, to name a few. The following stage is the mapping of *visualization data* to a *renderable representation*. This can be a transformation to graphical primitives such as points and lines. In general, mapping uses a combination of geometric primitives (e.g., points, lines, surfaces) and visual channels such as size, shape, color, and position to express data [Mun15]. For instance, atom positions of a molecule can be mapped to spheres that are approximated by triangles that form a surface mesh. Additionally, the temperature factor (B-factor) can be assigned as color to the atom spheres using a transfer function. This will visually encode how flexible

certain atoms or atom groups are. Another example is a vector field produced by streaming simulations or real measurements, which can be mapped to arrows encoding both the flow direction of particles by their orientation and the velocity by their color.

The final step in the visualization pipeline is to generate a *displayable image* as the result of the rendering of the *renderable representation*. In the context of computer graphics, rendering is the generation of a computer-generated image, for example, from a scene consisting of a 3D model, a virtual camera, and a light source. Thereby, the final image is generated under consideration of an illumination model used to calculate the lighting and shading of the 3D model. Scientific visualization is often an iterative process that should enable the user to refine and adjust different steps of the visualization pipeline. Due to that, the pipeline allows the user to interact with three different processes, which are filtering, mapping, and rendering. This could be the adjustment of thresholds and clustering parameters for filtering, changing color or size during mapping to highlight certain features or particles, and changing the virtual camera during rendering.

2.1.2 OpenGL Rendering Pipeline

Taking up the rendering as the last process of the visualization pipeline (cf. Figure 2.1) and setting it in the context of visualizing spatial data, the goal is to get interactive real-time visualizations. Usually, hardware-accelerated computer graphics is used for this purpose, which is made accessible through an Application Programming Interface (API). The most commonly used ones are Open Graphics Library (OpenGL), Direct3D, Vulkan, and Metal. This section illustrates the concept by explaining the rendering pipeline for modern OpenGL, which was used for most of the methods presented in Chapters 3 and 4.

In computer graphics, 3D objects are typically represented as a mesh built from polygonal primitives like triangles or quads. A primitive, in turn, is defined by connected vertices. One vertex is a point in space, and a connection between two vertices is called an edge. In the case of a triangle primitive, three vertices have closed connections between them, and the enclosed area is called a face. A vertex can have multiple attributes such as color values, texture coordinates for texture mapping, and normals defining a direction vector used for shading.

In a nutshell, the main stages of the rendering pipeline are the transformation of the vertex data, followed by rasterization and shading. To perform these steps, a modern GPU is built from highly optimized hardware. The majority

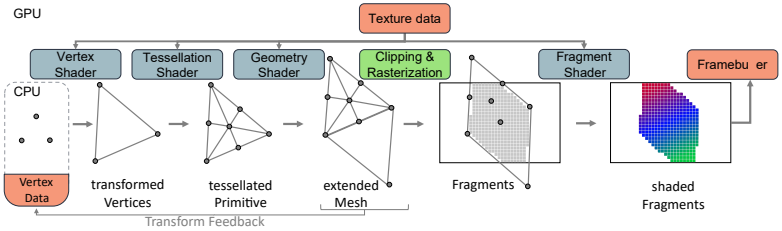


Figure 2.2 — The simplified version of the OpenGL 4.6 pipeline. All data (orange) is held in the GPU memory except the *vertex data*, which is initially sent via the CPU from the main memory to the GPU memory. There, it is further processed and modified. This is done via the shader stages (blue) followed by the *rasterization* (green), where the vertex data is converted to fragment data. The fragment data is then modified by the *fragment shader* and stored in the framebuffer to display it on a monitor.

of operations that have to be performed are matrix and vector calculations. For each vertex, equal instructions needed to be executed as transformations between different spaces, such as model space and camera space. This includes setting the position of an object in a 3D scene and projecting the 3D scene to a 2D image so that it can be presented on a monitor. These vertex operations and other subsequent steps are of a similar nature, which can be easily executed in parallel. Therefore, GPUs are optimized for such Single Instruction, Multiple Data (SIMD) operations.

To make use of this highly specialized hardware, the OpenGL API is utilized. The most recent version of OpenGL is 4.6, released in 2017 [SA22]. A simplified version of its pipeline is presented in Figure 2.2 focusing mainly on the programmable stages. Almost all stages are programmable using the OpenGL Shading Language (GLSL). Besides these, there are also non-programmable stages with fixed functions such as rasterization. The programmable stages include the *vertex shader*, *tessellation shader*, *geometry shader* and the *fragment shader*. The shaders have fixed built-in variables and allow the definition of further custom in- and output variables. They all have read access to the texture memory, where arbitrary data can be stored in different texture formats.

Initially, the process starts with vertex data that is sent from the Central Processing Unit (CPU) side to the GPU memory. There, the rendering pipeline begins with the *vertex shader*. It modifies the input values, which are the vertex positions and further attributes (e.g., color, normal, texture coordinates). Typi-

cally, at this stage, the transformation from object space to clip space takes place by applying the model-view-projection matrix, which is composed of three matrices. The first is the model matrix that transforms from object space (local space) to world space. This corresponds to the placement of an object in the scene, including rotation, scaling, as well as shearing. To transform from world space to the camera space (eye space) the view matrix is applied. It transforms the scene into the reference system of the virtual camera. The projection matrix is then used to project the 3D world to the 2D clip space, usually the screen coordinates of a PC monitor. Applying the projection matrix is similar to the function of a camera lens projecting the real world onto a sensor. The function of a *vertex shader* is exemplarily depicted in Figure 2.2. There are three vertices defining a triangle primitive, which are sent from the CPU to the GPU to be transformed. The shown transformation result is an enlargement and a rotation of the triangle primitive.

As an optional stage, the *tessellation shader* is used to subdivide primitives as triangles or quads into a set of points, lines, or triangles. This is done by the interplay of three parts. One part is the *tessellation primitive generator* with a fixed function and two programmable shaders for controlling and evaluating the generator. The first is the *tessellation control shader*, which controls the level of subdivision and ensures continuity across faces, avoiding gaps and breaks in the mesh. The second is the *tessellation evaluation shader*, which performs on each vertex the interpolation of the vertex attributes and determines the final position [SA22]. As displayed in Figure 2.2 the transformed vertices serve as input primitive for the *tessellation shader*, which subdivides it further into six triangle primitives. A common use case of *tessellation shaders* is controlling the Level Of Detail (LOD) of objects.

The *geometry shader* is optional as well. It emits zero or multiple primitives for every input primitive. Since the input primitive is always discarded, this stage can be used to remove unnecessary geometry. A *geometry shader* is more flexible in the type of operation performed than the *tessellation shader* but can emit fewer primitives. Figure 2.2 illustrates the operating principle of this stage. It extends the primitive by adding a vertex, resulting in a diamond-shaped mesh.

After these stages, it is possible to read back the final vertex positions from the GPU-Memory into the main memory, which is enabled by the *transform feedback*. This allows the vertex data to be further processed by the CPU and stored for other usages. The next stage is *clipping* of primitives that are not in the field-of-view (frustum) of the virtual camera. This is combined with the *rasterization* that outputs a set of fragments for every primitive. For each

fragment, the position and depth values are given, as well as the interpolated vertex attributes such as color, normal, texture coordinates, or any other defined attribute. In the example Figure 2.2, the camera frustum is represented by a rectangle. As the mesh of the 2D diamond shape does not entirely fit into the frustum, the spikes of the shape are clipped. The remaining part is rasterized, illustrated as small gray squares.

The output values of the rasterization stage are further modified by the *fragment shader*. Usually, the *fragment shader* determines the color and depth for each fragment. Optionally, fixed function operations like blending can be applied, where the alpha value of the color given in red, green, blue, and alpha (RGBA) format can be used for a transparency effect. The final values are written to the framebuffer in case of direct rendering or to a buffer object for offscreen rendering.

2.2 General Purpose Computation on GPUs

GPUs were originally designed for fast image rendering using the SIMD architecture as explained in Section 2.1.2. However, beyond that, the SIMD architecture has been used for General Purpose Computation on GPUs (GPGPU) applications for more than 20 years. Modern GPUs have thousands of specialized cores, which is contrasted by consumer CPUs, only having around ten physical cores. Utilizing this tremendous potential of parallelism accelerates parallelizable algorithms and can reduce runtimes many times over. In general, problems are suitable for SIMD parallelization if the same computations need to be executed for a large set of data points independently. Possible scenarios are calculations for molecular visualizations and simulations as well as particle simulations for the virtual investigation of fluid and gas streams. Another application example is the reconstruction of computer tomography scans in the medical field. Access to this parallelization potential is given by the graphic APIs OpenGL and DirectX offer *compute shaders*, which are decoupled from the traditional graphics pipeline. As the other programmable shader stages of OpenGL, *compute shaders* can also be written in the high-level language GLSL (or HLSL in the case of DirectX). An alternative to *compute shaders* is utilizing Open Computing Language (OpenCL), which is an API enabling parallel computations on CPU, GPU and further processors.

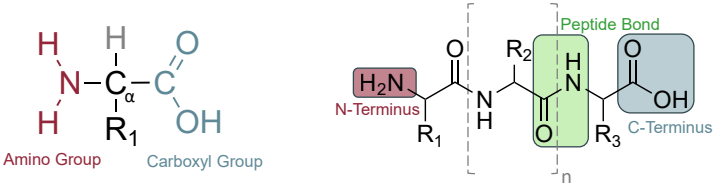
Compute Unified Device Architecture (CUDA) is another high-performance parallel computing specialized hardware and software platform. It is developed by the NVIDIA Corporation and provides a development environment based

on the programming languages C/C++. Its usage is solely limited to NVIDIA graphics hardware. CUDA offers features like atomic operations, ensuring data integrity when working with multiple threads and unified memory that enables the GPU to access the main memory directly. In general, the CUDA toolkit offers supporting tools for the development of graphics hardware-accelerated programs. In the context of parallelization, CUDA uses the Single Instruction Multiple Thread (SIMT) model. Therefore, CUDA cores are aggregated to Streaming Multiprocessors (SMs), and one SM directs the same instruction to all its cores.

The usual sequence of steps while using CUDA starts with a parallelizable problem that is subdivided into smaller tasks (threads). This is followed by transferring the data from the CPU (host) to the GPU (device). The data is then processed in each thread by executing CUDA code as a so-called kernel that represents a function. The threads are organized in blocks, which are divided in turn into warps. The warp size matches the number of CUDA cores that are aggregated to a SM. Finally, the processed data is transferred back to the host. The overall goal is to reduce these host-device data transfers to a minimum due to their relatively high time costs. Different algorithms in this work were implemented using CUDA as described in Chapters 3 and 4. CUDA was used, for instance, to perform fast neighbor searches and solve sorting problems. A more extended introduction to CUDA is given by Storti and Yurtoglu [SY16].

2.3 Biological Background

Proteins are macromolecules of living systems that accomplish crucial tasks. They are involved in almost all biological processes like *transportation, digestion, chemical synthesis* and *storage*. An example is the hemoglobin of blood cells that binds the oxygen that is distributed to all cells in the body. Proteins fulfill *mechanical tasks*, e.g., they give cells their structure and form elastic blood vessels. They enable *coordinated motion* facilitated by the interplay of myosin and actin filaments in muscles. To steer body movements, the necessary *signal transduction* is also enabled by the participation of proteins. In the case of chemical signal transduction, receptor proteins are important. The electric signal transduction is enabled, among others, by protein tunnels in the cell membranes of nerves that control the stream of ions. A subset of those tunnels are active ionic pumps playing an essential role in signal transduction. The energy needed for those pumps or muscle contraction is provided by *catalyzed chemical reactions* performed by enzymes. This specific class of proteins is of



(a) Amino Acid **(b)** Peptide

Figure 2.3 — (a) Exemplary structural formula of an amino acid. It consists of an α-carbon (C_α) with connections to an amino group (red), a hydrogen (gray), and a carboxyl group (blue). The fourth connection is with a residue (R_1), which is an individual group defining one of the various amino acids. **(b)** A peptide built from three amino acids with undefined residues (R_n). The colors correspond to the groups as in (a). The green area frames a peptide bond between two amino acids.

interest to industry and biotechnology as well. There they are utilized to split and transform substances and to facilitate complex synthesis. Examples are the production of detergents, biofuels, food, or drugs.

The following subsections briefly introduce the biological principles on which the further chapters are based. They will focus on protein structure, how they are synthesized following the genetic code, and which types of interaction they enter with ligand molecules. A more detailed explanation of this and various related topics can be found in the books of Alberts et al. [Alb+22] and Berg et al. [Ber+15].

2.3.1 Protein Structure

The structure of proteins can consist of one or multiple chains. On average, these chains consist of 50 to 2000 amino acids [Ber+15]. Amino acids have a common structure scheme as illustrated in Figure 2.3 (a). A central α-carbon (C_α) atom is connected with four groups consisting of single or multiple atoms. Three of those are always identical in all amino acids and are built of an amino group (-NH_2), a hydrogen (H), and a carboxyl group (-COOH). Amino acids are differentiated by the residue (R), which is one of 20 different chemical groups defining the 20 proteinogenic amino acids as building blocks of all proteins. Two amino acids in a chain are called a peptide as well and are connected via a peptide bond formed during protein synthesis. A peptide bond is created

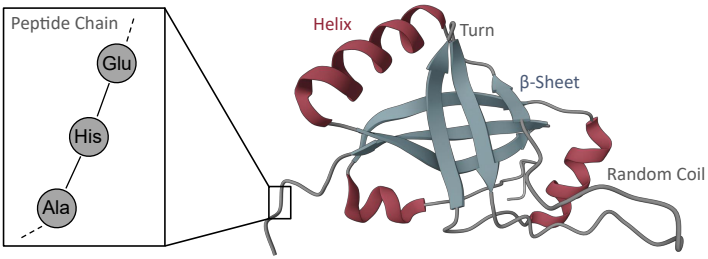


Figure 2.4 — On the right side, an abstract *tertiary structure* representation of a protein is shown. A zoomed-in section of its *primary structure* is shown on the left side, presenting a peptide chain segment (gray). The *secondary structure* elements α-helices (red) and β-sheets (blue) are labeled and shown as cartoon representations. Unordered chain regions are represented as gray lines. (RNA binding protein of a bacterium - PDB ID: 1K8H)

by a condensation reaction of the aforementioned amino and carboxyl group under the elimination of water (H_2O). These bonds between amino acids form the backbone of a protein. The general scheme of a polypeptide is presented in Figure 2.3 (b). A polypeptide always has an amino group on one end (N-terminus) and a carboxyl group on the other end (C-terminus).

The order of the amino acids in a peptide chain (cf. Figure 2.4) is determined by the DNA that stores the protein blueprint in a gene, which are specific sequences of DNA bases. The bases are read in triplets known as codons, each of which corresponds to a distinct amino acid. There are special start- and stop-codons, which do not encode an amino acid. These codons flank the region of a gene that is transcribed to RNA. The DNA *transcription* to RNA is part of the protein biosynthesis and is followed by the *translation* process, where RNA codons are translated to amino acids. In eukaryotes (e.g., mammalians), the DNA is stored in the cell core in a condensed form as chromosomes. During the *transcription*, chromosomes are disentangled to enable transcription of certain genes from DNA to Messenger RNA (mRNA). The mRNA is transported through the cell core pores into the cytoplasm, where the mRNA is *translated* to a peptide. The *translational* process is done by a protein complex called ribosome, consisting of two assembled subunits. There the mRNA is read, and a peptide is built. The amino acids needed for that process are transported to the ribosome with Transport RNA (tRNA). There are many different tRNA molecules, where each has a specific combination of a bound amino acid and anti-codon, which must

be complementary to the currently read codon of the mRNA to be used for the peptide elongation. While the ribosome moves the mRNA forward, the amino acids are decoupled from the tRNA and are connected via peptide bonds.

The resulting polypeptide is a linear chain. This sequence of amino acids is called the *primary structure* of a protein (see Figure 2.3). The folding of this chain into the energetically most stable conformation determines the spatial structure. The structure of a protein is crucial for its function, and a misfolding can lead to known diseases such as Alzheimer's, Parkinson's, and Huntington's disease [CP06]. The *primary structure* serves as the base of various *secondary structure* elements such as β-sheets, α-helices, turns, and random coils (cf. Figure 2.4). The β-sheets are almost flat structures of neighboring chain segments. In contrast, α-helices are chain segments wound in the shape of a rod. Both structures are stabilized by hydrogen bonds (*H-bonds*). The α-helices and β-sheets can be connected via turns, and the remaining unordered regions of a protein are named random coils. Moreover, the *secondary structure* folds into a higher order three-dimensional structure, which is the *tertiary structure* representing the final arrangement of a protein chain (see Figure 2.4). The folding process can be supported by special proteins, which are named chaperons. The *tertiary structure* is also stabilized by H-bonds and, additionally, by covalent bonds as disulfide bridges formed between two specific residues (cysteines). If a protein is a complex assembled of multiple chains, then such an arrangement is called the *quaternary structure*. Hemoglobin, for example, is a heterotetramer consisting of two different subunits assembled into a *quaternary structure* of four subunits in total.

2.3.2 Protein-Ligand Interactions

As introduced at the beginning of Section 2.3, proteins interact with their environment to fulfill a broad spectrum of functions. Protein interactions are a wide and diverse field that includes protein-protein as well as protein-ligand interactions. In this work, the focus is on the latter, whereby existing interaction force calculation methods are used, and the primary emphasis is on their visualization. The aim is to enable the investigation of how ligands affect proteins or vice versa. The mentioned ligands are small molecules, which can be, for instance, *cofactors* enabling an enzymatic reaction. One such cofactor is Adenosine Triphosphate (ATP), which is the primary cellular energy source and is needed for the phosphorylation of enzymes. The process of enzyme phosphorylation often leads to their activation. Furthermore, ligands can be *substrates* like the milk sugar lactose, which is split into its two monosaccharides

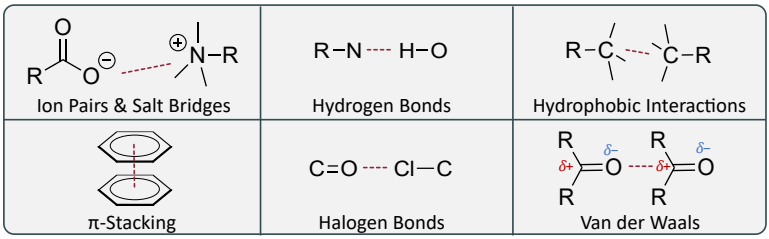


Figure 2.5 — Non-covalent interaction types. For each, an example of a partially complete set of subtypes is shown as a structural formula (black) with a simplified interaction between the molecules (red). The symbols ⊖, ⊕ indicate ions and $\delta+$, $\delta-$ are partial charges.

by the enzyme lactase. Moreover, they are *hormones* as well, which can bind to a receptor protein and play a crucial role in chemical signaling. An example is insulin, which enables the intake of glucose into cells. This is only a subset of ligand types, but it already highlights both their importance and their diversity.

How proteins interact or which functions they have is determined by their spatial structure and the residues of the individual amino acids forming their chains. The residues can be roughly divided into *polar*, *nonpolar*, *acidic*, and *basic*, which defines their physico-chemical properties. On certain protein surface areas, the residues form binding sites for ligands, often located in pockets or clefts, which can be connected through tunnels to their outer environment [CS10]. In an enzyme, the binding site enabling a chemical reaction is named its active site. If a ligand binds to an enzyme, for example, as a cofactor, it can inhibit or promote an enzymatic reaction by triggering a conformational change of the protein. During a binding event, the interactions between the residues and the ligand are of a weaker and non-covalent nature in comparison to the covalent peptide bonds, where electrons are shared between atoms by forming hybrid orbitals. One example function of a conformational change would be the opening or closing of an ion channel formed by a membrane protein.

The non-covalent interaction types are exemplarily depicted in Figure 2.5 and are comprised of H-bond, halogen bonds, π-interaction, hydrophobic interaction, Van der Waals (vdW) forces, ion pairs, and salt bridges. The salt bridges are a special type since they are a combined electrostatic interaction type of an ionic bond and an H-bond. This set of non-covalent interactions gives an overview and can be partially divided into further subtypes. For instance, π-interactions

consist of π-stacking, cation-π interaction, and further. The interaction types often depend on the angle of the interacting chemical groups and physicochemical properties, such as polarity, charge, geometric or steric effects, and flexibility. Additionally, they also depend on environmental conditions, such as the solvent and its pH value. These conditions and properties all contribute to mediating protein-ligand interactions. A more detailed description of individual non-covalent interactions is given by the book of Berg et al. [Ber+15] and with a stronger focus on proteins, examples, and a more medical chemist view by Biedermann and Schneider [BS16] and Bissantz et al. [BKS10].

2.4 Machine Learning

To study proteins and ligands and to gain a comprehensive understanding of structures, functions, and mechanisms, as mentioned in the introduction, a huge amount of biological data is available nowadays and continues to increase due to technological progress. Because of the amount and the complexity, machine learning is a viable method to handle and gain new insights into these data. These learning methods help to extract common features, group the data by clustering, or to reduce the dimensions to facilitate further data analysis. Moreover, classification problems exist, such as assigning proteins into fold or function classes and predicting the correct folding with only known primary structures. A famous example of using machine learning methods in this context is *AlphaFold* [Jum+21] released by Google DeepMind in 2021. It is one of the best tools attempting to solve the problem of protein folding. *AlphaFold* allows an accurate secondary and tertiary structure prediction using a neural network-based model.

Neural networks, as exemplarily depicted in Figure 2.6, belong mainly to *supervised learning* as well as classification and regression [Bur18]. The methods mentioned, such as clustering and dimensionality reduction, are assigned to *unsupervised learning* methods. More use cases are the prediction of binding affinity of ligands forming protein-ligand complexes or to cluster proteins by structure similarity to identify common features for ligand binding. A main difference between the fields of *supervised* and *unsupervised machine learning* is that supervised methods need labeled data for training a model [Bur18]. In the following two subsections, more details about the fields will be outlined. A more detailed explanation of machine learning, including regression, dimensionality reduction, neural networks, and cluster algorithms, among other related topics, is given by Zhou and Liu [Zho21] and with a stronger focus

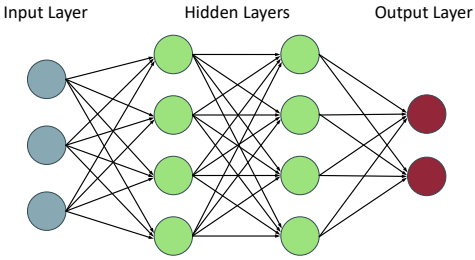


Figure 2.6 — The neurons of the exemplary artificial neural network are represented as circles and ordered in layers. The data goes into the neurons of the *input layer*. The input values are passed to the computational neurons of the *hidden layers*. The final neurons are in the *output layer*, giving the result.

on neural networks by Wennker [Wen20]. Furthermore, the domain-tailored review about machine learning by Zhu, Chen et al. [Zhu+22] provides a general overview of the assignment of a wide variety of machine learning methods to the main categories of supervised, unsupervised, and hybrid models, which are divided even finer in this review.

2.4.1 Supervised Learning

Supervised models consist mainly of *regression*, *classification*, and *mixed* forms. The *regression* works for numerical data and investigates how two variables depend on each other [Bur18]. For example, linear regression can be applied to given values that describe the time and distance a car drives. Regression attempts to find an equation/model and adjust its parameters to fit the (temporal) development approximately. The goal for the time and distance example can be to calculate the travel distance for a certain time point in the future based on the regression equation that, e.g., allows determining which city could be reached. Another example considers enzymatic reactions, where it is of interest how the reaction rate evolves. The goal can be to calculate the expected amount of product for a certain time point and given substrate concentration. This calculation is based on the catalytic rate, which can be determined with an equation called the *Michaelis-Menten equation*, whose enzyme-specific parameters are adjusted by regression.

Instead of regression, which is used to predict continuous values, the *classification* predicts discrete categorical values, as in the aforementioned example of assigning fold and function classes to proteins. Therefore, thresholds can

be used. A widely used protein classification is Structural Classification of Proteins (SCOP) [Lo +00]. It consists of multiple hierarchy levels, including family and superfamily for evolutionary relationships, and the third level describes the protein fold on *secondary structure* content and topology. For example, assigned classes can be *all-*α, when a protein consists of only α-helices and some unordered regions or $\alpha + \beta$ if the protein contains segregated α-helices and β-sheets [Lo +00].

A often used method of supervised learning is a *neural network* that is trained with labeled data. It can handle both regression and classification tasks [Bur18]. An Artificial Neuronal Network (ANN) is basically built of neurons ordered in three different types of layers as depicted in Figure 2.6. The data is processed in the neurons or nodes, which emit a signal dependent on an activation function.

Neural networks are frequently used for image processing and pattern recognition. For example, recognition of handwritten numbers. In this case, the data is an image of a single number. The data goes to the *input layer*, and the number of neurons in that layer corresponds to the number of pixels [Bur18; Alb18]. The gray value from the pixels is passed to the neurons of a number of *hidden layers*. The neurons are connected via edges having different weights. In these neurons, a decision over firing a signal to the next layer is made by passing a weighted sum of the input neurons to an activation function, which often is a logistic or sigmoid function. This function has a result range from zero to one, corresponding to the probability that a neuron is inactive or active.

The neurons have a different static value as well, which is a *bias* influencing if a neuron gets activated or not. In the case of number recognition, the interconnection of weights and biases allows the recognition of certain parts of handwritten numbers. This process continues from layer to layer until the final result is given by the neurons of the *output layer*. The exemplary, fully connected neural network in Figure 2.6 is also named a *Multilayer Perceptron (MLP)*. Following the example, the number of output neurons is ten because of ten possible handwritten numbers (0-9). The number of *hidden layers* and the number of neurons of each layer are partly empirically determined. A neural network with a large number of *hidden layers* is also called *deep neural network*.

Lastly, a neural network is a computational model inspired by the structure of the human brain that transforms data through layers of interconnected neurons, often calculating probabilities or values to support decision-making. The input data, the layers, and neurons, as well as their weights and biases, can be represented as matrices and vectors. They are added and multiplied with

each other, which are calculations a GPU is made for, as explained in Section 2.2. This means the execution of neural networks can be accelerated with GPUs and is especially useful during the training of those networks. A widely used framework applying this is *TensorFlow* [Aba+16]. During the training step, it is necessary to perform many thousands of network executions while automatically adjusting weights and biases to fit the labeled training data. The adjustment of weights and biases are applied using *back propagation*, and the process of one network execution and the adjustment step is called an *epoch* [Wen20]. After the training step, the network's accuracy is evaluated with a test set, which is from an initial split of the data into a training and a test set.

Among other neural network variants, such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM) [Zhu+22], there are several other established methods of supervised learning available including *support vector machines* [Hea+98] and *random forests* [Bre01].

2.4.2 Unsupervised Learning

The unsupervised models include *clustering* and *dimensionality reduction* [Bur18]. The general idea is to find structure in the data. No training or testing is applied compared to supervised learning with neural networks. Examples of dimensionality reduction methods are Principal Component Analysis (PCA), preserving the covariance, and Multidimensional Scaling (MDS), which preserves the distance. More details can be found in the mentioned book of Zhou and Liu [Zho21]. The following cluster algorithms and approaches are only a subset relevant to this work. There are various cluster approaches comprising centroid-based (e.g., k-means [Ste56]), density-based (e.g., Density-Based Spatial Clustering (DBSCAN) [Est+96]), hierarchy-based (e.g., Unweighted Pair Group Method with Arithmetic mean (UPGMA) [SM58; DR00]) and others.

The hierarchical cluster algorithm UPGMA is often used in bioinformatics to construct a phylogenetic tree [FM67] based on a distance matrix. A phylogenetic tree is a variant of a dendrogram that is used to visualize evolutionary relations between proteins, genes, organisms, species, and further (cf. Figure 2.7). It assumes there is a common molecular clock and all taxa have a similar velocity of evolution. This is in contrast to another hierarchical cluster algorithm called neighbor-joining [SN87], which does not assume a similar velocity of evolution and does not always create rooted trees. In general, the UPGMA algorithm of Sokal and Michener [SM58] builds a tree starting at the leaves towards the

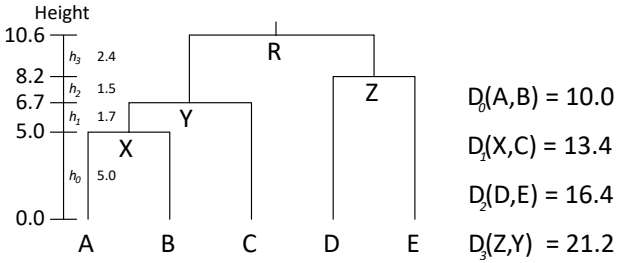


Figure 2.7 — A rooted tree constructed from five nodes (leaves) A to E using the UPGMA algorithm [SM58]. The shortest distances of the distance matrix of each iteration are shown on the right side (D_0 *to* D_3). The edge lengths between the nodes are computed as $h_0 = {}^{D_0}/_2$ for the first level and $h_i = {}^{D_i}/_2 - {}^{D_{i-1}}/_2$ for the other levels. - modified figure © Elsevier 2021 [Sch+21b]

root [Dur+98]. It iteratively merges two clusters with the shortest distance found in the distance matrix and creates a new node (cluster) from them. This continues until only one cluster is left in the distance matrix. A new node is placed above the two clusters, and its height is based on the original cluster distance (e.g., $D_0(A, B)$), which can exemplarily be seen in Figure 2.7, where cluster A and B are combined to the new node X. The new node is added to the cluster distance matrix, replacing the two clusters. Then, the new distance values of the new cluster C_N to all existing clusters C_i are determined as defined in Equation (2.1). The equation shows that the distance value $d(C_N, C_i)$ between C_N and any Cluster C_i corresponds to the arithmetic mean of all pairwise node distances [Sch+21b; Dur+98]. The length (h_i) of connecting edges between the nodes (e.g., X, Y) results from the difference in their height, whereby the initial length is $h_0 = {}^{D_0}/_2$. The initialization step of UPGMA consists of treating each leaf node as a one-element cluster (e.g., A to E), which is followed by the described cluster merging to build a rooted tree such as in Figure 2.7.

$$d(C_N, C_i) = \frac{1}{|C_N| \cdot |C_i|} \sum_{x \in C_N} \sum_{y \in C_i} d(x, y) \tag{2.1}$$

In contrast to UPGMA following a hierarchical-based approach, another clustering method that uses a centroid-based approach is the k-means algorithm. It is suitable for large sample sizes and needs only the number of clusters as an input parameter, which is a drawback when the number of clusters is unknown. The used metric is the distance between points. The DBSCAN algorithm also uses

distances but requires two additional input parameters, which are ϵ defining the search distance around a point and *minPts* setting the minimum number of points that must be present to form a cluster. DBSCAN also defers from k-means, though, that it is not as sensitive to outliers, which it marks as noise. More details about cluster algorithms, their parameters, and use cases can be found in the documentation of the python library scikit-learn [dslea23; Ped+11] and in the survey of Xu and Wunschll [XW05].

Further well-established unsupervised learning methods include *autoencoders*, *single value decomposition*, *mixture-of-Gaussian* clustering, among others. More details are given in the book of Zhou and Liu [Zho21].

2.5 Molecular Visualization

Molecular visualization is a vital method to investigate protein-ligand interactions. Most methods focus on the interactive visualization of molecular data, i.e., the user can interact and manipulate it. Each method or representation model highlights different aspects of the molecule's physical and chemical properties, depending on the context of the research question. The depictions help to understand the structure, the function, and how the molecules interact with each other. They allow visual analysis of sequence and structure data of proteins and ligands.

Various spatial and non-spatial molecular visualizations are available, enabling a better understanding and more insights to be gained. An example of a non-spatial visualization is the structural formula (see Figure 2.5), which is a 2D depiction of a chemical compound showing the different atom types as characters. The atoms are connected via one or multiple lines (A═B) representing covalent bonds (e.g., Fischer projection). Variants of the structural formula representation as the Natta projection also attempt to visualize the spatial arrangements of the atoms by using, in addition to solid lines (A—B) also dashed, broadening, solid lines (A⋯B), and broadening lines (A◀B) to indicate a forward or backward spatial orientation of chemical elements. Structural formulas are limited when visualizing large molecules as proteins, where they take up much space and lead to visual clutter. A protein sequence diagram is another example of a non-spatial visualization that overcomes this limitation. It simplifies the protein representation by ignoring the spatial arrangement of the peptide chain and showing only the primary structure as a sequence of characters, using the one-character code of amino acids (cf. Figure 2.8).

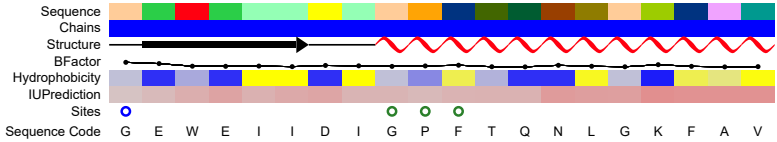


Figure 2.8 — An enhanced interactive protein sequence diagram[1], where the categorical attributes *Sequence Code* (characters) and *Sequence* (color code), *Chain* (chain ID), and *Sites* (binding sites) a represented by colored rectangles or circles. The quantitative attributes *Hydrophobicity*, and *IUPrediction* (predicted intrinsic disorder) are visualized as rectangles as well using linear and diverging color scales. The also quantitative *B-factor* (temperature factor) is encoded as a line chart. The secondary structure is depicted by different types of helices and arrows. It shows a part of the plant protein named monellin, PDB ID: 5LC6.

In bioinformatics, a simple form of a sequence diagram can be utilized to represent the result of a Multiple Sequence Alignment (MSA). There, the color-coded amino acids of multiple protein sequences are depicted together along the x-axis to easily show where areas appear with high congruence (conserved regions) or to see positions where a deletion, an insertion, or a replacement happened. Further details about a MSA are given in the article of Edgar and Batzoglou [EB06].

The interest is often on a single protein. Usually, a sequence diagram is then extended with additional relevant properties of amino acids to enable a more comprehensive representation of a protein. That, for example, includes the hydrophobicity value of the amino acids or binding site affiliation. An example of an enhanced and interactive sequence diagram was developed in two co-supervised bachelor theses. The result is depicted in Figure 2.8 and was presented as a poster at Eurographics VCBM[1]. It can not only depict the attributes stored in a PDB file but also enrich the available information using external analysis tools. The quantitative attributes B-Factor, hydrophobicity, and IUPrediction [Dos+05] can also be interactively switched to a bar chart or a line chart representation (see B-Factor Figure 2.8). It allows hiding attributes and shows a tooltip with detailed values while hovering. A further example of a use case of a single protein sequence diagram is the well-known and widely used RCSB PDB [Ber+00]. There, a 2D protein sequence diagram is combined with

[1] M. Schäfer, A. Cremer, M. Aktürk, and M. Krone. "Towards an Enhanced Interactive Protein Sequence Diagram." *Eurographics VCBM Posters*. 2020

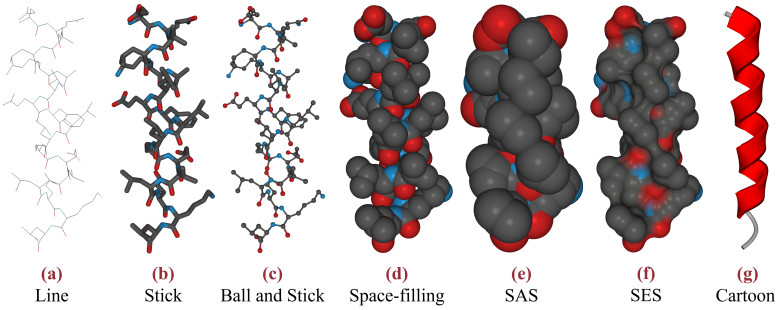


Figure 2.9 — Simple molecular models are shown in (a) to (c) (line, stick, ball-and-stick) followed by surface models (d) to (f) (space-filling, Solvent Accessible Surface (SAS), Solvent Excluded Surface (SES)) and ended with an α-helix as an abstract cartoon representation (g). The atoms are colored according to their types (carbon - black, oxygen - red, nitrogen - blue). (synthetic protein - PDB ID: 1PEF; Images created with *MegaMol* [Gro+15])

3D protein visualization, which is provided by utilizing *Mol** [Seh+21]. *Mol** is a web-based toolkit for visualization and analysis of large-scale molecular data.

Molecular data are often extensive, in three-dimensional space, complex, and sometimes time-varying as well, such as in the context of MD simulations. Two main classes of 3D representation models exist to visualize these data. In contrast to the mentioned 2D-sequence diagrams, they also can take into account different levels of spatial arrangement by considering the 3D positions of atoms, amino acids, and peptide chains. They can be divided into *atomistic models* and *surface models* (cf. Figure 2.9) as introduced in the state-of-the-art report of Kozlíková et al. [Koz+17], where are a detailed description of the following models and further related visualization can be found. In addition to these models, there exist cartoon representations of proteins, which provide an abstraction showing the high-level, functional structure of a protein.

2.5.1 Simple Models

The simple models are bond-centric and include representations such as lines, sticks, and ball-and-stick model, as depicted in Figure 2.9 (a) to (c). This representation of covalent bonds helps to infer various chemical properties. The atom positions of the *line* and *stick* model are encoded in the models as corners

only. Their position becomes more clearly encoded in the *ball-and-stick* model, where atoms are represented with small spheres having a smaller radius than their natural atomic radius. This leads to sufficient space for visualizing the bonds as well as the atom positions, which otherwise would lead to overlapping spheres and hidden bonds (see Figure 2.9 (d)).

The geometric primitives needed to be rendered are only lines, cylinders, and spheres. Normally, they are tessellated into triangles (see Section 2.1.2). That is, a very high number of triangles for molecules with up to millions of atoms have to be rendered, which can lower the interactivity. To overcome this problem, the fact that these primitives can be easily described mathematically is exploited. It allows an implicit description of the surface, which can be rendered using ray casting. The basic idea is that a simple form, often a quad, is rasterized, and a view ray is cast for each fragment. Its intersection with the geometric primitive is then calculated in the fragment shader. If the current ray does not hit the implicit surface, the fragment will be discarded. This method of modern GPU-based glyph ray casting was presented by Gumbold [Gum03]. It leads to a much faster rendering of a high number of surfaces compared to the traditional rendering of tessellated geometries [Koz+17]. Moreover, it is additionally also a pixel-precise rendering for each zoom level of a surface compared to the dependency on the tessellation level of traditionally rendered geometries.

2.5.2 Surface Models

Parts of this section have been published in:

- M. Schäfer and M. Krone. "A Massively Parallel CUDA Algorithm to Compute and Visualize the Solvent Excluded Surface for Dynamic Molecular Data." EG Workshop on Molecular Graphics and Visual Analysis of Molecular Data, 2019. DOI: 10.2312/molva.20191094

The surface models focus on the interface of molecules and their outer environment, as opposed to simple models that aim to represent atomic bonds. Various examples are depicted in Figure 2.9 showing the *space-filling* model, *Solvent Accessible Surface* (SAS) and *Solvent Excluded Surface* (SES). A widely used and simple one is the *space-filling* model or *calotte* model, which encodes atoms as spheres using the atom position and radius. The atom radius is defined as the halved distance between the nuclei of two covalent bond atoms. The surface then consists of the union of all outer spheres (cf. Figure 2.10). A further very similar surface is the *vdW surface* [Ric77], which utilizes the vdW radius to

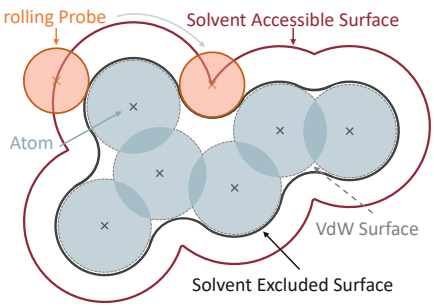


Figure 2.10 — Schematics of three surface model definitions. Atoms are depicted as blue semi-transparent Van der Waals (vdW) circles forming the molecule's vdW surface (gray dashed line). A probe (orange) rolling over the vdW surface defines the Solvent Excluded Surface (black) and the Solvent Accessible Surface (red).

define the atom spheres. This radius is half of the closest distance between two nuclei of atoms they can take without forming a chemical bond (equilibrium of vdW attraction and electrostatic repulsion) [Ber+15; Koz+17].

The more complex surface models of Figure 2.9 are described by a rolling probe sphere, which can represent a solvent molecule. The solvent can be a water molecule, for instance, whose geometry can be approximated by a sphere to simplify the calculation. The SAS [LR71] is defined by following the probe's center while rolling over the *vdW surface* and is, therefore, an extension of it. Effectively, the SAS is the same as the *vdW suface* with the difference of combining the vdW atom radius and the probe radius. The resulting surface shows the interface of the molecule that is accessible by a solvent equal to or smaller than the radius of the probe. The SAS is cheap to compute if it is waived to remove the interior parts, but the SAS has the drawback that it creates a non-smooth surface and does not faithfully reflect the volume of the molecule compared to the *vdW surface*. This can lead to unwanted occlusions and intersections with other molecules when visualizing MD simulations. Because of that, it is less suited for the analysis of molecular interactions.

For this purpose, the SES [Ric77] is more feasible. It is equally defined as the SAS, but tracing the touching boundary of the probe instead of its center point [Koz+17] (see Figure 2.10). The resulting smooth surface combines the advantages of the surface accessibility visualization of the SAS with a better volume representation of the *vdW surface*. The SES is well suited for visualization of MD simulations or molecular docking results because cleft, pockets, and tunnels of proteins are clearly visible because of the mentioned properties.

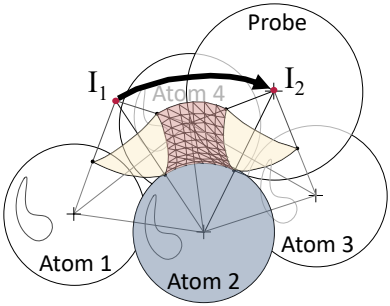


Figure 2.11 — Three-dimensional genesis of the three different SES patches including (●) spherical triangle patch, (●) toroidal patch, and the (●) spherical patch. The points I_1, I_2 show the center points of the probe sphere where it is in a fixed position. - modified figure © John Wiley & Sons 1996 [SOS96]

Greer and Bush [GB78] gave a further, equivalent definition, that the SES can be defined as the union of all probe spheres, which do not intersect the vdW spheres of the atoms. They also coined the term Solvent Excluded Surface. Another name for the SES is the Connolly surface since Connolly was the first to present the analytical equations to compute the three surface patches [Con83]. These surface patches, which are shown in Figure 2.11, can be derived from different states of the rolling probe sphere: 1.) When the probe is in contact with just one vdW sphere, then the part of the vdW sphere which is in contact with the probe can be described as a convex spherical patch as part of the SES (most left probe Figure 2.10). 2.) When the probe is in contact with two vdW spheres, then it can roll around the axis given by the two vdW sphere centers. This results in tracing out a torus. The part that lies between the two vdW spheres is a toroidal patch that contributes to the SES as well. 3.) The third case is when the probe is in a fixed position, which means it is in contact with three vdW spheres. From this position, it is not able to roll in any direction without losing contact with at least one of the spheres. This leads to a concave spherical triangle patch, which is the probe surface between the three points of contact.

For this thesis, the mentioned surface and simple models are among the most important molecular visualizations. In contrast to these more *"realistic"* models, which often represent different and partly derived physical properties, there also exist abstract and simplifying cartoon representations as shown for an α-helix in Figure 2.9 (g). This representation focuses especially on abstracting the secondary protein structure elements, where α-helices are represented as spirals and β-sheets as curved arrows (see Figure 2.4). The remaining structure elements, such as turns, random coils, and other unordered regions, are simplified into curved tubes, connecting the aforementioned cartoon representations.

2.6 Software Infrastructure

This section will explain the biological data formats and software that served as the basis for processing and visualizing data in this thesis. In this context, the visualization prototyping framework *MegaMol* [Gro+15; Gra+19] is introduced as used for methods described in Sections 3.3 and 4.1. Additionally, the desktop framework *MegaMol* was used in combination with various web technologies, which are briefly outlined as well. A web and desktop combination was used in diverging ways in visual analytics applications discussed in Sections 4.3 and 4.4.

2.6.1 The MegaMol Framework

MegaMol [Gro+15; Gra+19] is a prototyping framework for visualizing large scientific data sets. Initially, it was a joint research project of physicists, biologists, material scientists, and visualization research experts to enable working and visualizing large particle data sets as they arise by MD simulations. Now, the software has further evolved and provides information visualization, improved software implementations, and a scripting interface to facilitate accessibility for domain scientists. With various modules and rendering techniques, it supports 2D and 3D visualization and data preprocessing. *MegaMol* is a cross-platform desktop framework for Windows and Linux. It is written in C++ and uses OpenGL as its main graphics API.

In the context of visualizing protein-ligand interactions, there exist various comparable tools as *PyMOL* [SD20], *VMD* [HDS96], and *Chimera* [Pet+04], which provide molecular visualization capabilities, but they are more tailored and do not have this degree of flexibility for more comprehensive visualization research. *MegaMol* already offers a wide range of molecular visualizations and the necessary data structures to work with data like atoms, molecules, or volumetric data as electron density maps. The visualizations include all molecular representations depicted in Figure 2.9 and volume rendering as well. Additionally, *MegaMol* enables it to directly work interactively with molecular trajectories gained from MD simulations.

The framework aims to give a thin layer on top of 3D APIs. It provides data type and problem-tailored data structures, which offer and allow for high-performance GPU-accelerated graphics. This, for example, interactive rendering of particle data and creating information visualization [Gro+15; Gra+19]. In this context, the framework intends to exploit hardware capabilities by efficiently using computing resources, including CPU and GPU. *MegaMol* is designed

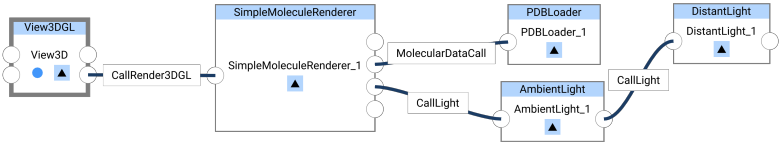


Figure 2.12 — A module graph for rendering a ball-and-stick model of a molecule created with the *MegaMol* Configurator. Each box represents a module with its static module name in the light blue area and a centered custom name. All modules are connected via calls. The graph starts with the *View3DGL* calling for rendering an image and ends with the *PDBLoader* (providing the protein data) and the *distantLight* (providing light and its direction).

to support the rapid development of visualization prototypes, which is also applied by a Lua-based scripting interface. Moreover, it provides a processing pipeline aiming to achieve good scalability as well.

To reach these goals, the *MegaMol* framework follows three paradigms, which are flexibility, reusability, and zero-copy data management [Gro+15; Gra+19]. What this means in detail becomes clear by taking a closer look at the modular design of the *MegaMol* framework. *MegaMol* builds up on a core library, a front end, and one or more chooseable plugins. The core library acts as an interface between the front end and the back end. It generates and manages GPU resources, loads the plugins, and manages the logic of the module graph as presented in Figure 2.12. Plugins consist of multiple modules, which are interconnected via calls following a pull pattern. A call models an intent, which is the reason for invoking it. For instance, an intent can be a request for new data or data extents. From a technical perspective, a call represents an interface of a module. Since a call can model multiple intents, it can connect to multiple callback functions, e.g., collect and convert the requested data.

This call and module structure is exemplified in the module graph of Figure 2.12. The presented set of modules is part of the protein plugin of *MegaMol*. The given module interconnection arrangement enables the rendering of various molecular representations as discussed in Section 2.5.1. The algorithmic idea encoded by this acyclic graph begins with the *View3DGL* module that requests the *SimpleMoleculeRenderer* to render an image. That, in turn, asks the *PDBLoader* via the *MolecularDataCall* for data to build the 3D representations based on positional information of the individual atoms. Through the *CallLight* connection, the *SimpleMoleculeRenderer* also requests light parameters needed for the intended shading of the molecular representations.

As mentioned, the *SimpleMoleculeRenderer* is able to render different molecular representations. To allow switching between them is one example of a parameter that modules have exposed for external input [Gro+15; Gra+19]. This input can be defined by a configuration file or comes from Graphical User Interface (GUI) elements allowing to set, e.g., file paths, thresholds, colors, or to enable or disable render options like transparency. Thus, modules have three different slot types, which are parameter slots, caller slots, which are located on the right side as the outgoing part to connect a call, and callee slots, which are the ingoing connection slots on the left. To build the module graph, *MegaMol* has a graphical configurator. Using this configurator, modules can be put together and connected with calls in the needed order based on their possible call types. Furthermore, it allows the setting of module-specific parameters.

As it can be derived from the explanation of the depicted module graph, there are no restrictions on complexity, granularity, or functionality encapsulated in modules to give complete freedom while developing visualization prototypes [Gro+15; Gra+19]. All this insight about the modular structure explains the implementation of the flexibility and re-usability paradigm. The application of the zero-copy data management paradigm is shown in how data is handled and owned by modules. Data produced by a module is owned by the module, and only a pointer or a reference is forwarded to other modules when they request data via a call. That aims to reduce memory consumption and allows better scalability. Further details about how computationally data copies and redundant computations are avoided, besides more specific design and implementation details, can be found in the cited publications about the *MegaMol* framework.

Web Development Environment As indicated in the introduction of the enclosing section, desktop frameworks were combined with modern web techniques to create interactive visualizations of protein-ligand interactions. There is no specialized framework like *MegaMol*, which is utilized for that purpose. However, many frontend and backend programs and libraries are assembled to process and visualize the molecular data interactively. Not all libraries and frameworks can be named and explained here due to their amount, but the most essential will be mentioned. They include the backend environments *Node.js* as server-side JavaScript runtime environment and *flask* as a Python framework, both used to create web servers and host web applications.

As a frontend framework, *Vuetify* [Lei+23] is used to build the general web page layout and design with interactive components. *Vuetify* is a client-side Vue.js component framework. The components can be extended with any further content. For example, this can be visualizations using the JavaScript library d3.js [BOH11], which stands for Data-Driven Documents (D3). This library is specifically designed for data visualization, e.g., to create interactive scatter plots, heatmaps, bar charts, and many other advanced visualizations as well.

To create interactive molecular visualization in the web context, *Mol** [Seh+21] was used and extended to handle SES meshes. For the interactive visual analysis tool InVADo (described in Section 4.4), an additional web-socket communication was implemented to connect the advantages of non-web and desktop-based *MegaMol* framework with a web application to get used to the capabilities of the mentioned and further web libraries and frameworks.

2.6.2 Biochemical File Formats and Data Preparation

While working in the field of protein-ligand interaction, various file formats arise that are tailored to different data properties and application scenarios. These include types to store static molecular data (e.g., single proteins as PDB), dynamic molecular data (e.g., MD simulations as XTC), and files containing multiple models respectively, conformations of a single molecule (e.g., docking results as PDBQT).

Static Molecular Data - Proteins The mentioned PDB file format stores static data of biological macromolecules and is mainly designed to store three-dimensional coordinates of atoms and how they are connected. A widely used source of PDB files is the RCSB PDB [Ber+00] which currently stores ~$2.07 \cdot 10^5$ structures. The structures are experimentally determined using methods such as X-ray crystallography, Nuclear Magnetic Resonance (NMR) spectroscopy, and Cryo-Electron Microscopy (EM). The PDB format is one of the most well-known formats to store protein data, but nucleic acids (DNA, RNA) that, in certain conditions, assemble with proteins, are stored as PDB files as well.

Each PDB file has a unique four-digit identifier. For example, the identifier *8H9T* stands for the human ATP synthase. PDB is an ASCII file format storing the data as plain text. Each line starts with a keyword defining the type of entity followed by attributes describing it. For instance, the central information of a PDB is stored in lines starting with the "ATOM" keyword, which is followed by

attributes with fixed positions (column ranges). The attributes include information about the atom name, number, and its Cartesian coordinates in ångström ($1\ \text{Å} = 1 \cdot 10^{-10} m$), the residue name and number of the protein where the atom belongs, the chain identifier of the residue and further attributes including optional properties such as charge and structural data. A detailed description of the attributes and their positions can be found in documentation [Wan+23] of the python MD ANALYSIS toolkit [Gow+16]. A further example of keywords is "REMARK", which can contain arbitrary information, such as details about the above-mentioned experimental methods and authorship, or the keyword "CONECT", where information about covalently connected atoms is stored.

Besides the PDB format, which is no longer being extended to support new information content and is limited in the number of atom entries that can be stored, there exists a newer file format called Macromolecular Crystallographic Information File (mmCIF) [Wes+06]. It is an extension of the PDB format and allows for storing more metadata about experimental and structural data and is continually modified to support new content.

Multiple Molecule Models - Preparation for Molecular Docking In the context of molecular docking, a file format is needed that is able to store multiple models/conformations of the same small molecule. The reason is justified in the nature of the goal of docking, which is to find the optimal phyisco-chemical fit of a small molecule for a certain region on a larger target as a protein. The docking is often performed with the widely used tools *AutoDock* (3, 4, Vina) [Mor+98; Hue+07; Ebe+21], which uses its PDB, Partial Charge (Q), Atom Type (T) (PDBQT) format [Wan+23; Mor+14] to store its results. This format is similar to the PDB format but adds partial charges to the atoms, *AutoDock* atom types, and a flexible representation via defining rotatable bonds. The results of the docking tools are multiple possible conformations and positions of a single small molecule. The small molecules used for docking are also called ligands and can be stored in various file formats, for example, as PDB file, or as the also frequently arising text-based formats like MOL [Dal+92] or Structure Data File (SDF), where a SDF file can contain multiple MOL entries. Nevertheless, both the docking target as well as the ligand must be converted to the PDBQT format when using *AutoDock*. Besides *AutoDock*, there exist various further docking software mentioned in Section 4.1.1, but in this work, the focus is on *AutoDock* because of its high degree of dissemination and as it is mainly used to generate data for development and visualization purposes in this thesis.

When docking, the result depends decisively on the correct preparation of the target (e.g., a protein-receptor) and the ligands (e.g., an inhibitor). One reason is that there can be missing atoms in the PDB file, e.g., missing atoms in protein residues or missing atoms charges. To have all atoms with their correct coordinates and charges is important because some docking algorithms, such as the one of *AutoDock 4*, use force fields to calculate the interactions between the target and the ligand atoms. An exemplary preparation and docking protocol is given by Morris et al. [MHO08], where the *AutoDockTools* [Mor+09] are used to perform the preparation tasks and to get the needed files for molecular docking with *AutoDock* [Mor+98]. A more recent overview of *AutoDock* distributions, tools, and protocols focusing on virtual drug screening is given by Forli et al. [For+16].

In general, the preparation tasks consist of removing disturbing water molecules if they are contained in the molecular data set. Furthermore, hydrogens are added to complement the molecule because hydrogens are usually omitted when storing molecular data. Additionally, a check for missing atoms is performed, which can be identified and added by using *AutoDockTools* as well. Moreover, hetero atoms as cofactors are removed, and partial charges are determined by adding Gasteiger charges [GM78], which is a further functionality of the *AutoDockTools*. The final step consists of converting the original molecular data to the PDBQT file format while saving the preparation results. A configuration file is needed as well to perform the docking, which sets the 3D space as a box to specify an area on the target in which the docking is to be performed.

Molecular structures are not rigid bodies and have rotatable bonds. Therefore, the possibility exists to set a limited amount of rotatable bonds that will be considered during the docking process. To store this information, the PDBQT format follows an idea of a tree, where the rigid core of a molecule is the *root* and the flexible parts are *branches* that can be nested as well [Mor+14]. Due to that, the keywords are ROOT, ENDROOT, BRANCH, and ENDBRANCH, which are complemented by the keywords ATOM, REMARK known from the PDB format. In addition, the keyword MODEL sets the number of a specific molecule conformation, and the keyword TORSDOF specifies the number of torsional degrees of freedom. Latter is used to evaluate the conformational entropy when using *AutoDock* 4.

Dynamic Molecular Data Assuming the purpose is to investigate how a path of a moving ligand to the active site of an enzyme looks like and whether conformational changes of the enzyme appear. Therefore, it is necessary to

physically simulate the behavior of molecules and to track their positional changes, such as done by MD simulations. To store this information, using the PDB format alone is not feasible because it is designed for static data and only stores the topology of the molecule. Notwithstanding this, the topology of the MD simulation results can be stored in a static PDB file because MD simulations do not include chemical reactions. Thus, the molecular topology will not change via the simulation. However, the dynamic data is often written into an additional file. An example is the XTC format [dGRO23] as used by the MD simulation software *GROMACS* [Abr+15; Hes+08]. That means that the result of a MD simulation is stored in a combination of the trajectory information of the atoms and the molecular topology, e.g., as XTC and PDB file.

CHAPTER 3

Protein Analysis

The investigation of protein-ligand interactions is a vast field that can be considered from different perspectives, including protein-centered and ligand-focused perspectives (cf. Figure 1.1). Proteins are of utmost importance when studying protein-ligand interactions because their shape and physico-chemical properties enable the mediation of interactions in the first place. Therefore, this chapter takes a protein-centric perspective and is concerned with the analysis of protein structures and their properties. It discusses various comparison methods and presents a CNN-based approach for 3D protein structure learning. The learned features are linked with knowledge about, e.g., protein function and fold classes. This chapter will also show a new comparison concept based on protein surface maps, including the possibility of considering their physico-chemical properties as well. Those methods support inferring how interactions between proteins and ligands work and can lead to findings about common structural elements for interactions. Those elements can include certain physico-chemical arrangements, such as protein domains that form a binding site. Protein domains are conserved regions with the same or very similar amino acid sequence and a specific structural motif that can be found across many proteins [Jin+09]. Those domains are involved in mediating molecular interaction or enabling catalysis. They can be seen as building blocks of proteins. Finding similar domains across proteins indicates evolutionary relations and supports classification tasks. The comparison of proteins with known structure and function helps to gain a comprehensive understanding of how protein-ligand interactions work in general and which conditions must be fulfilled from the protein perspective.

3.1 Sequence and Structure Comparison Methods

As mentioned in Chapter 1, nowadays huge amounts of protein and ligand data are available (e.g., RCSB PDB [Ber+00], scPDB [Des+15], PDBbind [Wan+04], ZINC [Irw+20]) as well as many technologies and algorithms to process these data and gain new insights. It is essential to compare and understand protein structure to understand biological function and protein-ligand interactions. Finding similar proteins is an important task for many application areas, such as in biomedical research. However, the manual search for similar proteins is a time-consuming task, and there are several possible variations regarding what can be compared. That resulted in the development of many protein comparison methods focusing on different aspects. This section will give a brief overview of some established methods and tools for protein comparison, comprising similarity search-based ones that use sequence and/or structural data. They include alignment methods based on DNA or amino acid sequence. These alignment methods can be grouped in the widely used sequence alignments for two or multiple sequences (MSA [EB06]), e.g., *Basic Local Alignment Search Tool (BLAST)* [Alt+90] or *BLASTp* (protein BLAST [Alt+90]), profile alignments, e.g., *PSI-BLAST* [Alt+97], and Hidden Markov Model (HMM) [BP66] based alignments, e.g., *HMMER* [FCE11] and *Clustal Ω* [SH14].

Some methods also take the spatial structure into account. Spatial comparisons often use atomic coordinates to find proteins with similar layouts and sizes. A measurement for spatial or conformational similarity is the Root Mean Square Deviation (RMSD) as used by *Template Modeling (TM)-align* [Zha05] and *MultiMer (MM)-align* [MZ09]. Further examples of tools performing protein structure comparisons are *DaliLite* [Hol+23] and *3D-Surfer 2.0* [Xio+14]. *MM-align* considers the protein sequence as well as the protein structure, enabling comparisons of entire protein complexes consisting of multiple chains. It first aligns the proteins before comparing them, in contrast to *3D-Surfer 2.0*, which is an example of working on unaligned proteins. *3D-Surfer* computes a feature vector containing 3D Zernike Descriptors to find similarities, which are invariant under rotation. An overview with more details of computational methods for protein comparison is given in the review by Chen et al. [Che+18] and the work of Kufareva and Abagyan [KA12].

3.1.1 Sequence-based Methods

Sequence-based alignments are fundamental and widely used in areas such as bioinformatics, molecular biology, and phylogenetics. A general assumption

and idea behind primary structure alignments is that similar protein sequences indicate a common evolutionary relation and similar conformation and function. The protein function, in turn, is mediated by interactions, which often take place with ligands. This means it provides insight into the evolution of certain protein-ligand interactions, contributing to a comprehensive understanding of how they generally work, e.g., allowing the identification and investigation of protein domains and their physico-chemical properties.

Dynamic Programming-based Alignments When comparing two sequences, one class of applied algorithms are dynamic programming algorithms. These include the well-known algorithm by Needleman and Wunsch [NW70] performing a global alignment and the one by Smith and Waterman [SW81] used for local alignments. The latter algorithm is suitable for sequences with a higher difference in length, whereas the approach of Needleman and Wunsch is more suitable for sequences of similar length [Che+18]. Local alignments make it possible to find similar regions or sequence motifs. These algorithms determine a sequence of edit operations needed to transform one sequence into another. These operations comprise insertions, deletions, and substitutions of amino acids and are assessed with different penalty scores. That results in an alignment score, which provides information on how similar the two sequences are. Finding optimal alignments is computationally expensive, especially when searching a database for similar sequences, requiring thousands of alignments. Therefore, heuristic algorithms are used, which reduce the accuracy but increase the alignment speed. A very frequently used tool applying a heuristic approach and performing database searches is *BLAST* [Alt+90] utilized for DNA, RNA (*BLASTn*) and protein sequence alignments (*BLASTp*). For the aforementioned penalties for edit operations, *BLAST* uses specific scoring matrices for substitutions, such as BLOSUM 62 [HH92] defining different substitutions penalties for the proteogenic amino acids. More details about the *BLAST* algorithm and scoring matrices are given in the book of Yandell and Bedell [KYB03].

Profile Alignments The second class of alignments is an optimization and is named profile alignments [Che+18]. A profile contains the extracted evolutionary information and is constructed based on a MSA by searching a database. A representation of such a profile can be a position-specific scoring matrix [Alt+97]. This profile can be used to search a database. A variant of *BLAST* that uses profile alignments is called *PSI-BLAST* [Alt+97]. More variants and details about profile alignment approaches are given by Chen et al. [Che+18].

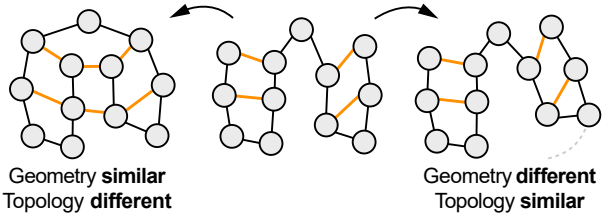


Figure 3.1 — Invariance present in protein structures. The gray circles represent amino acids connected via peptide bonds (black lines). The structures stabilizing H-bonds are shown as yellow lines. - Image source: [Her+21]

HMM-based Alignments: The third group of alignments uses Hidden Markov Models (HMMs) [BP66]. They are constructed based on a MSA and represent the transition probabilities between different amino acids [Che+18]. They further reduce computational costs and are applied by tools such as *PHMMER* [FCE11] or *Clustal Ω* [SH14], which uses guide trees and HMM profile-profile methods for alignments.

3.1.2 Structure and Spatial-based Methods

Besides protein sequence alignments, another possibility for protein comparison is to align the spatial structure of proteins. That can be important because, as shown by Alexander et al. [Ale+09], proteins sharing nearly identical primary structures, i.e., differing only by a few amino acids, can adopt completely distinct conformations upon folding [Her+21]. Another very similar case is that identical primary structures can have differing tertiary structures. That is often a misfolding, where the same proteins can fold into different three-dimensional structures under certain circumstances. These misfolded proteins, known as *prions*, can be dysfunctional and cause severe diseases such as Alzheimer's disease or Creutzfeldt-Jakob disease. On the other hand, Agrawal and Kishan [AK01] discussed proteins with similar tertiary structure that can have significantly differing primary and secondary structures (cf. Figure 3.1) [Her+21]. Those proteins can also have similar functions, which can be exploited for drug discovery [KW05], or to reveal distant evolutionary relationships between organisms [Koe01]. The described phenomenon of invariant protein structures needs to be considered, depending on which aspect of proteins is to be compared.

If it is intended to measure how similar the tertiary structures of two proteins are, this can be done using the Root Mean Square Deviation (RMSD). It is frequently used in validating and analyzing results of computational chemistry, e.g., as obtained from MD simulations or molecular docking (see Section 4.1). The structures are superimposed before the RMSD is calculated, which is the averaged deviation of the distance between the atoms of two structures [KA12; Kab76]. The RMSD is optimized during the alignment or superimposition by transposing and rotating the structures until the RMSD does not decrease any further. The result is expressed in ångström (Å), where 1 Å corresponds to $1 \cdot 10^{-10} m$. Aligning structures using the RMSD minimization is feasible for structures with equal or similar sequence lengths, which makes it unfeasible to search distant homologs or similar conformation motifs, where the sequence similarity can be low. Proteins with low sequence similarity can nevertheless have a similar conformation, as aforementioned in the context of the publication of Alexander et al. [Ale+09]. The *DALI* web server [Hol+23] is a tool that overcomes this drawback. It performs a distance matrix-based protein comparison, where matrices are generated by distance determination of the secondary structures [Hol+23]. It searches for local similarities, which is useful for investigating protein domains that can be surrounded by non-similar structures. For more details on *Dali*, see also the corresponding book chapter of Holm [Hol20].

3.1.3 MM-align - Sequence & Spatial Comparison

Parts of this section have been published in:

- K. Schatz, F. Frieß, M. Schäfer, P. C. F. Buchholz, J. Pleiss, T. Ertl, and M. Krone. "Analyzing the Similarity of Protein Domains by Clustering Molecular Surface Maps." *Computers & Graphics*. 2021, pp. 114–127. DOI: 10.1016/j.cag.2021.06.007

In contrast to the methods mentioned in Sections 3.1.1 and 3.1.2, the application *MM-align* [MZ09] follows a hybrid approach combining the protein sequence as well as the conformational information. *MM-align* was used extensively in this thesis (cf. Sections 3.2 and 3.3). As indicated, this tool considers the amino acid sequence information as well as the protein conformation and even enables the comparison of whole protein complexes consisting of multiple chains. *MM-align* expresses the similarity as a single value called the *TM-score* [ZS04], which is a widely used protein similarity measurement score (see, e.g., [FT20; GKJ19]). The returned similarity value can be seen as a global fold similarity score. When

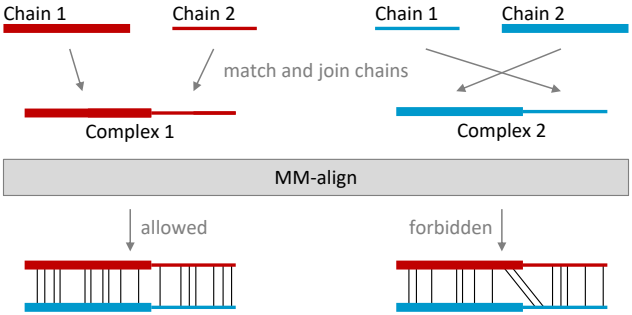


Figure 3.2 — Example of the chain alignment and joining procedure by *MM-align*. Two chains (thick and thin) of two different dimeric protein complexes (red, blue) are joined to a contiguous artificial chain (complex) for each protein. The complexes are aligned, whereby *MM-align* only allows the alignment of amino acids of one chain against one other chain (bottom left). That is, cross-chain alignments are forbidden, where amino acids of one chain segment are aligned against multiple chain segments of another complex (bottom right). - modified figure from [MZ09] as used in © EG 2021 [Sch+20b]

using *MM-align*, the TM-score is first computed for pairs of single protein chains from one protein with the chains of the other protein [MZ09]. This is part of one of three general steps. The *first step* is the selection of chains and their order for chain-joining. The *second step* is the construction of different initial alignments, which is followed by the *third step* consisting of an optimization of the TM-score. Therefore, a heuristic iteration of the protein superposition is executed.

The first step continues by reordering the chains of one of the proteins to find an optimal alignment between corresponding amino acids in the two protein complexes. Based on this sequence alignment, a heuristic, iterative algorithm is used to superimpose the structures. For the reordering, *MM-align* first permutes all chains of the protein with more or equal chains ($n \geq m$) by selecting m chains. As a time optimization for proteins with $m > 3$, a combination of those chains is used, where the sum of the previous TM-scores, which were determined for the individual monomer chain comparisons, is $> 90\%$ of the maximum sum of the TM-scores of the individual monomers. After the chain selection, permutations are generated where the sequences of the chains are joined to a new contiguous artificial sequence. This sequence is then aligned while avoiding cross-chain alignments (cf. Figure 3.2). Therefore, a modified

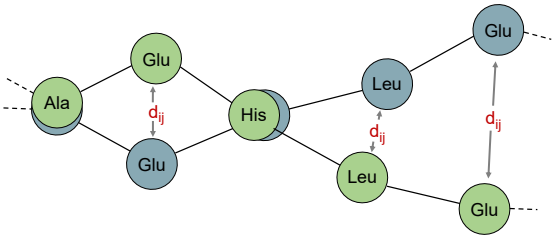


Figure 3.3 — Example of a superposition between two short parts of aligned amino acid chains (blue and green circles). $\mathbf{d_{ij}}$ denotes the distance between the C_α atoms of two aligned amino acids **i** and **j**. - modified figure © Elsevier 2021 [Sch+21b]

Needleman-Wunsch dynamic programming algorithm is used. This is followed by five different variants of sequence alignments that are performed on these contiguous sequences (for more details about the alignment algorithm, please refer to the original publications of *MM-align* [MZ09] and the TM-Score [Zha05]). For each of these alignments, an inter-complex distance matrix is computed. This guides a heuristic, iterative superposition algorithm that tries to optimize the TM-score. The iterative superposition algorithm stops when the alignments and, consequently, the TM-score converges to a stable number. The final found TM-score combines the *"measure of the accuracy and coverage of the structure superposition"* [MZ09]. The TM-score is computed as follows:

$$\text{TM-score} = \max \left[\frac{1}{L} \sum_{i=1}^{L_{align}} \frac{1}{1 + d_{ij}^2 / d_0^2(L)} \right] \tag{3.1}$$

The variable L denotes the length of all chains of the target complex, and L_{align} is the number of aligned protein pairs. d_{ij} is the Euclidean distance between the C_α atoms of the aligned amino acids i and j (cf. Figure 3.3) and d_0 is a scale to normalize the match difference: $d_0(L) = 1.24\sqrt[3]{L-15} - 1.8$ [ZS04; MZ09]. Thus, the final TM-score combines the similarity of the aligned regions and the alignment coverage as one similarity value. Its value range is $[0 \dots 1]$, where higher values denote higher similarity. A TM-score above 0.5 indicates similar topology, chain orientation, and the same fold class based on SCOP/CATH [Mur+95; Ore+97], which makes it useful for protein classification tasks [XZ10]. As mentioned in the beginning, the briefly introduced methods are only a small selection of the field of protein structure comparison. For more

details, please refer to the review by Chen et al. [Che+18] and the work of Kufareva and Abagyan [KA12].

3.2 Structure Learning Using Neural Networks

Parts of this section have been published in:

- P. Hermosilla, M. Schäfer, M. Lang, G. Fackelmann, P.-P. Vázquez, B. Kozlikova, M. Krone, T. Ritschel, and T. Ropinski. "Intrinsic-Extrinsic Convolution and Pooling for Learning on 3D Protein Structures." International Conference on Learning Representations. 2021. https://openreview.net/forum?id=1OmSURQpwY

Methods based on Artificial Neuronal Networks (ANNs) (Section 2.4.2) have significantly evolved in recent years, which has also been accompanied by improvements in specialized hardware used to train and execute these networks, such as *Tensor Cores* [Mar+18] integrated in GPUs. This enabled a new class of approaches for analyzing, comparing, and classifying proteins. A well-known example of the potential of these approaches is the protein folding tool *AlphaFold* [Jum+21], which allows an accurate secondary and tertiary structure prediction. The presented methods in Section 3.1, such as sequence-based (MSA) or RMSD-based ones, use various fixed metrics for protein comparison and similarity search. In contrast to these stands the *learning* of the protein sequence as well as its 3D structure. This section presents a newly developed method [Her+21] for protein structure learning, which applies deep neural networks combined with a new convolution and pooling approach.

As previously explained, it is of key importance to consider all structural levels of proteins to avoid misinterpretations due to ambiguities. That includes the primary, secondary, tertiary, and quaternary protein structure (details in Section 2.3.1). The new method is capable of considering all these structural levels in contrast to most state-of-the-art methods that use ANNs. Combining the information from these four levels contributes to a comprehensive understanding of interrelations between structure and function, interactions, and, e.g., enables solving protein fold classification or enzyme classification tasks.

This section will discuss the architecture of a new method comprising a graph representation of the protein structure from which certain geometric features are derived. These are used with a new convolutional operator and pooling method to learn all structural levels of proteins. Additionally, the new method's performance is evaluated at fold and enzyme classification compared to state-of-the-art methods, showing an overall better mean accuracy.

When investigating previously state-of-the-art methods, it can be recognized that they were dominated by methods based on hand-crafted features, usually extracted from MSA tools [Alt+90] or annotated databases [El-+19]. In recent years, these have been outperformed by protein learning algorithms in different protein modeling tasks such as protein fold classification [HAC18; Rao+19; BB19; All+19; Min+20] or protein function prediction [Str+20; Gli+19; KKH18; KH20; Ami+18]. This can be attributed to the ability of machine learning algorithms to learn meaningful representations of proteins directly from the raw data. However, most of these techniques consider only a subset of the relevant structural levels of proteins and thus can only create a representation from partial information.

Due to the high amount of available protein sequence data, most techniques rely solely on sequence data as input. Early works on learning protein representations [AM15; Yan+18] used word embedding algorithms [Mik+13], as employed in Natural Language Processing (NLP) and applied for protein learning [Rao+19; All+19; Min+20; Str+20].

Other approaches have used 1D Convolutional Neural Networks (CNNs) to learn protein representations directly from an amino acid sequence for tasks such as protein function prediction [KKH18; KH20], protein-compound interaction [Tow+19], or protein fold classification [HAC18]. Recently, researchers have applied complex NLP models trained unsupervised on millions of unlabeled protein sequences and fine-tuned them for different downstream tasks [Rao+19; All+19; Min+20; Str+20; BB19]. While representing proteins as amino acid sequences during learning is helpful when only sequence data is available, it does not leverage the full potential of spatial protein representations that become more and more available with modern imaging and reconstruction techniques.

To learn beyond sequences, approaches have been developed that consider the 3D structure of proteins. Various methods have sampled protein structures to regular volumetric 3D representations using 3D CNNs. This was used to assess the quality of the structure [Der+18], classify proteins in enzymes classes [Ami+18], predict the protein-ligand binding affinity [Rag+17]

and the binding site [Jim+17], as well as the contact region between two proteins [Tow+19]. This is attractive because 3D grids allow for unleashing the benefits of all approaches developed for 2D images, such as pooling and multi-resolution techniques. Unfortunately, those grids do not scale well to fine structures or many atoms, and even more importantly, they do not consider proteins' primary and secondary structure.

Another approach that makes use of a protein's 3D structure is representing proteins as graphs and applying Graph CNNs (GCNNs) [KW17; HYL17]. Works based on this technique represent each amino acid as a node in the graph, while edges between them are created if they are at a certain Euclidean distance. This approach has been successfully applied to different problems. Classification of protein graphs into enzymes, for example, has become part of the standard data sets used to compare GCNN architectures [GJ19; Yin+18]. Moreover, other works with similar architectures have predicted protein interfaces [Fou+17] or protein structure quality [Bal+20]. However, GCNN approaches suffer from over-smoothing, i.e., indistinguishable node representations after stacking several layers, which limits the maximum depth usable for such architectures [CW20].

Few attempts have been made to consider more than one structural level of proteins in the network architecture by, for instance, using Long Short-Term Memory to encode the primary structure and applying GCNNs to capture the tertiary structure [Gli+19]. Also, the work from [Ing+19] proposes an amino acid encoder that can capture primary and tertiary structures in the context of protein generative models. Unfortunately, none of these previous methods can incorporate all structural protein levels of proteins simultaneously. In contrast, a common approach is to process one structural level with the network architecture and the others indirectly as input features ([Bal+20] or [HAC18]).

3.2.1 Protein Representation and Convolution

When working and learning on protein data, it is necessary to have a suitable data structure. Protein data consist of atoms connected via covalent bonds or H-bonds. As aforementioned in the context of GCNNs, these can be represented using graphs (see Figure 3.4). To be able to take into account all structural levels of proteins during learning, they are expressed by a multi-graph $\mathcal{G} = (\mathcal{N}, \mathcal{F}, \mathcal{A}, \mathcal{B})$. This graph represents atoms as nodes associated with their 3D coordinates, $\mathcal{N} \in \mathbb{R}^{n \times 3}$ capturing the tertiary structure. Additionally, the atoms are associated with features, $\mathcal{F} \in \mathbb{R}^{n \times t}$, whereby n is the number of atoms, and t is the number of features. The per-atom features comprise: 1.) the

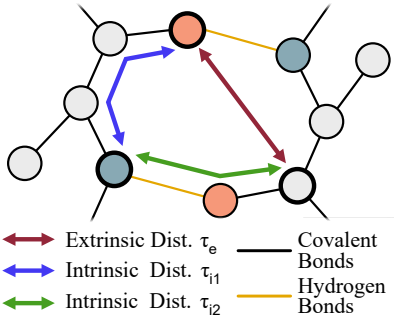


Figure 3.4 — Exemplary part of the protein graph. The graph nodes represent the protein atoms which are (●) carbon atoms, (●) oxygen, and (●) nitrogen. Three geometric distances are depicted as arrows between the atoms. They include the extrinsic τ_e distance and two types of intrinsic distances τ_{i1} and τ_{i2}. - modified figure [Her+21]

covalent radius, 2.) vdW radius, 3.) atom mass, and three further features 4.) to 6.). The last three are of an atom type embedding learned together with the network, which are certain geometric distances as described in the following Section 3.2.2. Moreover, the two different adjacency matrices $\mathcal{A} \in \mathbb{R}^{n \times n}$ and $\mathcal{B} \in \mathbb{R}^{n \times n}$ represent the connectivity of the graph or if there are connections between atoms at different structural levels of the protein. The elements of matrix $\mathcal{A}$ represent the connectivity of the primary structure and are defined as $\mathcal{A}_{ij} = 1$ if there is a covalent bond between atom i and atom j, and $\mathcal{A}_{ij} = 0$ otherwise. Similarly, the elements of matrix $\mathcal{B}$ are defined as $\mathcal{B}_{ij} = 1$ if there is a covalent or hydrogen bond between atom i and atom j, and $\mathcal{B}_{ij} = 0$ otherwise, thus covering connections of secondary protein structure (hydrogen bonds) and of the tertiary structure (covalent bonds such as disulfide bridges).

Based on this graph representation of the protein, the next step consists of creating a convolution kernel that enables to learn structural patterns. For this, certain intrinsic and extrinsic geometric properties of protein structures were chosen. Differential geometry of surfaces [Pog73] defines intrinsic geometric properties as those that are invariant under isometric mappings, i.e., under deformations preserving the length of curves on a surface. On the other hand, extrinsic geometric properties are dependent on the embedding of the surfaces into the Euclidean space. Analogously, in the protein multi-graph, the intrinsic geometric properties are defined as those that are invariant under deformations, preserving the length of paths along the graph, i.e., deformations that preserve the connectivity of the protein atoms. In addition, extrinsic geometric properties are defined as those that depend on the embedding of the protein into the Euclidean space, which means depending on the protein's 3D conformation.

Using this terminology, three distances in the multi-graph are defined to capture the structural relations between protein atoms, comprising one extrinsic distance and two intrinsic ones (see Figure 3.4).

The extrinsic distance τ_e is defined by the protein conformation in Euclidean space. Therefore, the Euclidean distance between atoms is used to capture the tertiary and quaternary structures of the protein. The intrinsic distances are inherent to the protein and independent of the actual 3D conformation. For the first intrinsic distance τ_{i1}, the shortest path between two atoms is used that is found along the adjacency matrix $\mathcal{A}$ of the graph, capturing the primary structure. The second intrinsic distance τ_{i2} is defined as the shortest path between two atoms along the adjacency matrix $\mathcal{B}$, capturing thus the secondary structure as well as partly the tertiary structure (disulfide bridges).

The key idea consists of considering the multiple invariances, as described at the beginning of Section 3.2, during the learning process. Therefore, based on the success of convolutional neural networks for images [KSH12] and point clouds [Cha+17b], a convolution operator for proteins was defined that can capture these invariances effectively. To this end, a convolution on 3D protein structures was proposed, which is inspired by conventional convolutions as used to learn on structured images. First, the neighborhood of an atom is defined as all atoms at an Euclidean distance smaller than m_e. In addition, the convolution kernel is defined as a single Multilayer Perceptron (MLP), which takes the three described distances as input and outputs the values of all kernels. This enables the convolution to learn kernels based on one or multiple structural levels of the protein. Thus, the proposed convolution operator is defined as:

$$(\kappa * F)_k(\mathbf{x}) = \sum_{i \in \mathcal{N}(\mathbf{x})} \sum_{j=1}^{t} F_{i,j} \cdot \kappa_{j,k} \left(\frac{\tau_e(\mathbf{x}, \mathbf{x}_i)}{m_e}, \frac{\tau_{i1}(\mathbf{x}, \mathbf{x}_i)}{m_1}, \frac{\tau_{i2}(\mathbf{x}, \mathbf{x}_i)}{m_2} \right) \tag{3.2}$$

where $\mathcal{N}(\mathbf{x})$ are the atoms at Euclidean distance $d < m_e$ from $\mathbf{x}$, $F_{i,j}$ is the input feature j of atom $\mathbf{x}_i$, $\kappa_{j,k}$ is the aforementioned kernel that maps $\mathbb{R}^3 \rightarrow \mathbb{R}$,$\tau_e(\mathbf{x}, \mathbf{x}_i)$ is the Euclidean distance between atom $\mathbf{x}$ and atom $\mathbf{x}_i$, τ_{i1} and τ_{i2} are the two intrinsic distances, and m_e, m_1, and m_2 are the maximum distances allowed ($m_1 = m_2 = 6$ hops in all experiments while m_e is layer-dependent). All normalized distances are clamped to $[0, 1]$. The convolution, performed independently for every atom, is illustrated on a model protein in Figure 3.5, where the distances between neighboring atoms are shown. This operation has all properties of the standard convolution, i.e., locality and translational invariance, and at the same time, rotational invariant.

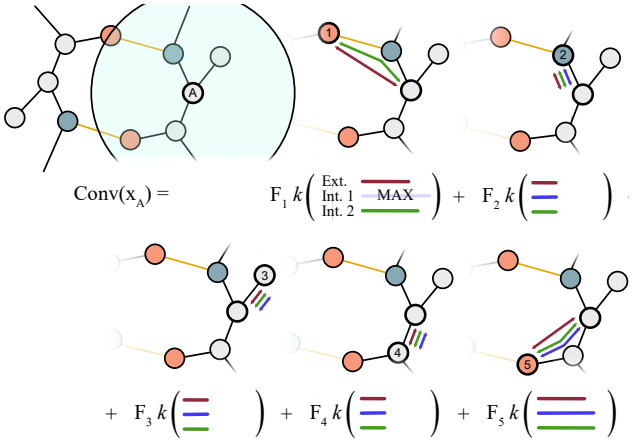


Figure 3.5 — Intrinsic-extrinsic convolution on the multi-graph for atom A. First, the neighboring atoms involved in the convolution are detected using a ball query (a function returning all nodes closer than r to a point $\mathbf{x}$). For each atom, the extrinsic (red) and two intrinsic (blue and green) distances are input into the kernel, and the result is multiplied by the atom's features. Lastly, all contributions from neighboring atoms are summed up. - modified figure [Her+21]

This operation is stated as an unstructured convolution as done in 3D point cloud learning [Her+18; GWH18; WQF19; Xio+19; Tho+19], it can also be understood in a message passing framework [KW17] where hidden states are atoms, edges are formed based on nearby atoms and messages between atoms. The key difference to standard GCNNs is that it has edges for multiple kinds of bonds, a hierarchy of graphs is used, and the message passing function is learned.

3.2.2 Hierarchical Pooling

The approach considers proteins at atomic resolutions. This allows identifying the spatial configuration of the amino acid side chains, which is crucial for active site detection. However, proteins can have a large number of atoms, preventing the usage of large receptive fields in the convolutions (computational restriction) and limiting the number of learned features per atom (memory restriction).

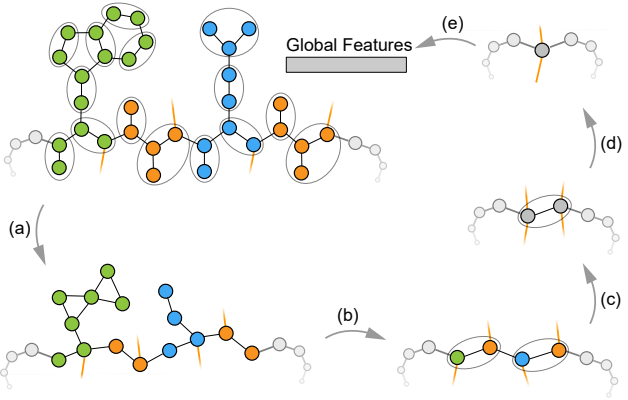


Figure 3.6 — Hierarchical protein pooling: The protein is segmented into amino acids (blue, orange, green). First, (a), a spectral clustering on each independent amino acid graph is applied. Then, (b), each resulting amino acid is pooled to its α-carbon. After that, two backbone pooling operations are applied, (c) and (d). Lastly, the symmetric operation average is applied, (e), to obtain the final feature vector. - modified figure [Her+21]

Pooling is a technique commonly used in CNNs, as it hierarchically reduces the dimensionality of the data by aggregating local information [KSH12]. It is able to overcome computation and memory restrictions when learning on images, but it cannot be directly applied to proteins, as conventional pooling methods rely on discrete sampling. On the other hand, some of the techniques developed for unstructured point cloud data [Qi+17; Her+18] could be applied to learn the 3D coordinates of atoms. However, it is unclear what the edges of that new graph representation would be.

Therefore, a set of operations is defined that successively reduces the number of nodes in the protein graph. First, it iteratively reduces the number of atoms per amino acid to its alpha carbon. Afterward, it reduces the number of nodes along the backbone. Figure 3.6 illustrates this process, which is described in the following paragraphs.

Amino Acid Pooling Simplified representations of amino acids have been previously used in MD simulations in order to lower the number of computations to a manageable level. Nevertheless, in contrast to this approach, these techniques do not use a uniform pooling pattern. Rather, while the atoms belonging to the backbone are not simplified, most of these approaches [Sim+97] simplify all atoms of the side chains to a single node. Other more conservative approaches, such as PRIMO [Kar+13] or Klein's models [DeV+09], represent side chains with a variable number of nodes using a lookup table. However, they do not perform a uniform simplification, resulting in a high variance in the number of atoms per cluster. Moreover, these methods need to manually update the lookup table to incorporate rare amino acids.

In contrast, in this work, the common practice in CNN for images is followed, where a uniform pooling is used over the whole image. The proposed new method is able to reduce the number of nodes in the protein graph by half. To this end, an independent graph for each amino acid is generated using the covalent bonds as edges and spectral clustering [vLux07] is applied to reduce the number of nodes by half. Since the number of amino acids is finite (20 appearing in the genetic code), these pooling matrices are reused among all proteins (cf. Figure 3.6 (a)). However, since the method only requires a graph as input, it can be directly applied to synthetic or rare amino acids.

From the amino acid pooling matrices, a protein pooling matrix $\mathcal{P} \in \mathbb{R}^{n \times m}$ was created, where n is the number of input atoms in the protein and m is the number of resulting clusters in the simplified protein. This matrix is defined as $\mathcal{P}_{ij} = 1$ if the atom i collapses into cluster j, and $\mathcal{P}_{ij} = 0$ otherwise. Using $\mathcal{P}$ a simplified protein graph can be created, $\mathcal{G}' = (\mathcal{N}', F', \mathcal{A}', \mathcal{B}')$, with the following equations:

$$\mathcal{N}' = \mathcal{D}^{-1}\mathcal{P}\mathcal{N} \qquad F' = \mathcal{D}^{-1}\mathcal{P}F \qquad \mathcal{A}' = \mathcal{P}\mathcal{A}\mathcal{P}^T \qquad \mathcal{B}' = \mathcal{P}\mathcal{B}\mathcal{P}^T \tag{3.3}$$

where $\mathcal{D} \in \mathbb{R}^{m \times m}$ is a diagonal matrix with $\mathcal{D}_{jj} = \sum_{i=0}^{n} \mathcal{P}_{ij}$. Note that the resulting matrices $\mathcal{A}'$ and $\mathcal{B}'$ might not be binary adjacency matrices, i.e., on the diagonal, they have non-zero values or values greater than one. Therefore, zeros were assigned to the diagonals and clamp edge values to one. These matrices are computed using sparse representations in order to reduce the memory footprint.

Alpha Carbon Pooling In a second pooling step, the protein graph is simplified to a backbone representation, whereby all nodes from the same amino acid are clustered into a single node. $\mathcal{P}$ is defined accordingly, and the number of clusters m is equal to the number of amino acids, while $\mathcal{P}_{ij} = 1$ if node i belongs to amino acid j, and $\mathcal{P}_{ij} = 0$ otherwise. Equation (3.3) is used to compute F', $\mathcal{A}'$, and $\mathcal{B}'$. However, $\mathcal{N}'$ is defined as the alpha carbon positions of each amino acid since they better represent the backbone and the secondary structures.

Backbone Pooling The last pooling steps are simplifications of the backbone chain. In each pooling step, every two consecutive amino acids in the chain are clustered together, effectively reducing by half the number of amino acids. Therefore, to compute $\mathcal{N}'$, F', $\mathcal{A}'$, and $\mathcal{B}'$, $\mathcal{P}$ is defined accordingly and use Equation (3.3). For single-chain proteins, for example, $\mathcal{P}_{ij} = 1$ if $\lfloor i/2 \rfloor = j$, or 0 otherwise.

3.2.3 Network Architecture

By facilitating protein convolution and pooling, it was possible to realize a deep architecture that enables learning on complex structures. In particular, the proposed architecture encodes an input protein into a latent representation, which is later used in different downstream tasks.

The input of the network is a protein graph at the atom level with a 6D input feature vector per atom as described in Section 3.2.1. The input is then processed by five layers, which iteratively increases the number of features, each composed of two ResNet [He+16] bottleneck blocks followed by a pooling operation (see Figure 3.7). The respective receptive fields are [3, 6, 8, 12, 16] ångström using [64, 128, 256, 512, 1024] features. The size of the receptive fields has been chosen to include a reduced number of nodes in it. Several proteins have missing H-bonds, which can be computed using *DSSP* [KS83].

As the architecture is fully convolutional, it can process proteins of arbitrary size, but after finitely many steps, it obtains an intermediate result of varying size. Hence, to reduce this to one final result vector, a symmetric aggregation operation average is used. Lastly, a single-layer MLP with 1024 hidden neurons is used to predict the final probabilities.

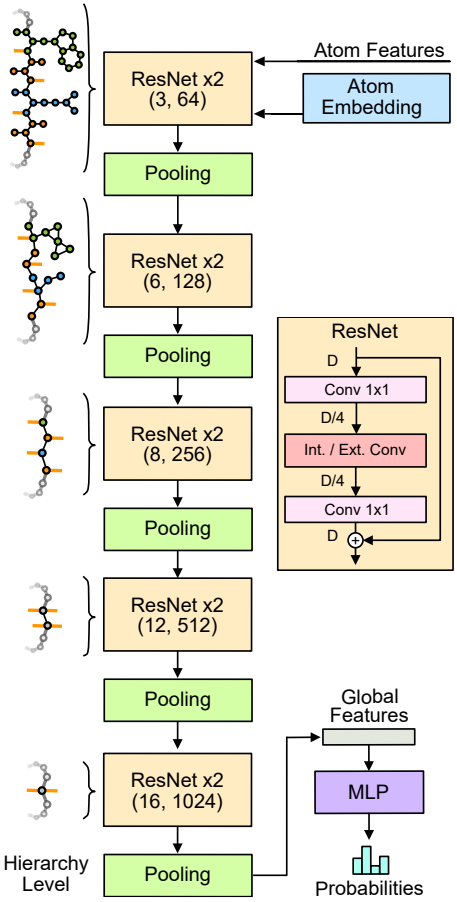


Figure 3.7 — Hierarchical protein pooling: The architecture of the protein structure learning model. The input is composed of atom features and an atom embedding learned together with the network. Each layer is composed of two ResNet [He+16] bottleneck blocks, for which the radius in ångström and the number of features are indicated in parentheses, followed by a pooling operation. An illustration of a single ResNet bottleneck block is presented on the right in the middle for D input features (before each convolution batch normalization and a biologically motivated activation function named Leaky ReLU is used). The global protein features are processed by an MLP, which computes the final probabilities. Protein graphs used at each level are indicated on the left. - modified figure [Her+21]

3.2.4 Training Data Preparation

In addition to a sophisticated network architecture capable of capturing all structural levels of proteins, it is equally important to have a sufficient and well-prepared set of training data. These need to be split into three sets, including a training, testing, and validation set. For this work, it is needed to guarantee that, based on a similarity-based clustering, all possible identified similarity or fold classes are covered in each split of the data set when assigning parts of the found clusters to the training, testing, and validation set. These fold classes may include, for example, those based on SCOP/CATH [Mur+95; Ore+97]. Thus, it is necessary to have already labeled data to perform the network training. Therefore, the two databases scPDB [Des+15] and PDBbind [Wan+04] were clustered based on similarity. Both databases are a collection of proteins or biomolecular complexes from the RCSB PDB. The scPDB contains annotated proteins with drugable binding sites, and the PDBbind database consists of protein-ligand complexes with experimentally measured binding affinity.

The goal of using these two databases is to enable the network to classify or determine binding sites or binding affinities due to the provided protein annotations. The second reason is that a wider variety of proteins is needed to ensure that no protein fold class based on SCOP/CATH is missed. Furthermore, integrating a database with annotated binding sites also allows for learning more specific properties or features of protein-ligand interactions. For the preparation of the training data, it was intended to take into account the comprehensive information of the protein amino acid sequence as well as the protein 3D structure. The method used should also enable the comparison of whole protein complexes consisting of multiple chains.

A wide range of methods and tools exist for protein comparison as described in Section 3.1, but only very few fulfill the mentioned criteria. A tool covering them is *MultiMer (MM)-align* [MZ09], which expresses the combined information of protein sequence and conformation as a global fold similarity score, which is named TM-score [XZ10]. A TM-score above 0.5 indicates a similar topology, chain orientation, and the same fold class based on SCOP/CATH, which is useful for clustering. For more information about *MM-align* and the TM-score, see Section 3.1.

MM-align was used to build a similarity or distance matrix for clustering the scPDB and PDBbind databases. Therefore, the TM-score between all proteins was determined. A necessary all-versus-all comparison is a very time-consuming task, which would take several weeks or months. For that reason,

the comparisons were optimized and reduced according to the basic bioinformatic assumption that protein sequence similarity also means structure and function similarity. Based on that, all comparisons were prevented where the individual sequence lengths differed more than 15%, which is a heuristically determined value where it is assumed that the compared proteins are different enough to be not rated as similar. This led to 16,494,287 (scPDB) and 15,557,882 (PDBbind) protein comparisons, which correspond to only 6% of all possible comparisons for both databases. The used data set from the PDBbind database contains 16,126 protein-ligand pairs and the data set from the scPDB contains 17,595 sets. Some circumstances lowered the number of proteins that were used. One is that there were 980 protein-ligand sets in the scPDB where the same proteins were contained multiple times, leading to fewer unique proteins. The final TM-score matrices contained 16,574 proteins (scPDB) and 16,126 protein-ligand pairs (PDBbind). The computations were done in parallel using a Python script executed on an Intel i9-9820x with ten cores and 20 threads. Despite the optimization step and the parallel execution, the calculations took approximately seven days for each database.

After the similarity matrices were determined, a clustering step followed to group the proteins according to their similarity and to be able to split the data sets into proper training, testing, and validation sets. The used algorithm was DBSCAN (see Section 2.4.2). The similarity matrix of the TM-scores was passed to the clustering as a precalculated distance matrix. The TM-score value range is between zero and one, where higher values equal higher similarity. To be sure that the TM-scores are treated as distances, the scores were inverted by subtracting the TM-score from one. DBSCAN needs two input parameters, including the neighbor search distance (ε) and the minimum number of neighbors forming a cluster (*minPts*). The ε value was set to 0.5 because proteins with a TM-score > 0.5 (distance $<= 0.5$) belong to the same fold in SCOP/CATH [XZ10].

To find a proper value for *minPts* k-Nearest Neighbor (kNN), distance plots were used. Normally, they support optimizing the ε value, whereby a good ε can be read from a kNN-distance plot from that point, where the curvature reaches its maximum or searching for knee of the elbow [Est+96; Sch+17]. But as mentioned, the ε value is already known, and for that reason, the kNN distance plot was used for optimizing the *minPts* instead of ε. Therefore, multiple kNN plots were created for different *minPts* with the goal of finding a graph where the maximum curvature is near to ε of 0.5. Multiple *minPts* values were found that fulfilled that criterion. One exemplary result of a clustering based on the

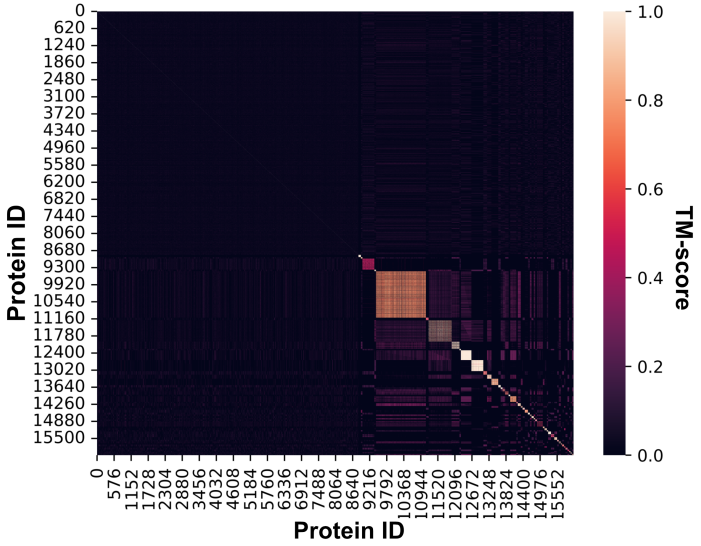


Figure 3.8 — Heatmap of the TM-score matrix created of the PDBbind data set using *MM-align*. The proteins are clustered using DBSCAN and are sorted according to their cluster ID. 8,835 of 16,574 proteins are not assigned to any cluster (upper-left corner 0 to 8,835). Right to the heatmap is the Magma color legend used to encode the TM-score (higher values equals higher protein structure similarity). Used cluster parameters are: $\varepsilon = 0.5$; minPts = 40.

found parameters is shown in the heatmap of Figure 3.8, showing the TM-score matrix of the PDBbind data set, where the proteins are sorted according to the found clusters. From this data set, $\sim$9,000 proteins were not assigned to any cluster located in the area in the upper-left corner of the heatmap, with darker colors indicating low similarities. The cluster size distribution can be seen in Figure 3.9 showing a small number of larger clusters and many smaller ones. The results were used to guarantee that all similarity clusters are covered when assigning parts of each cluster to the training, testing, and validation set.

TensorFlow [Mar+15] was used to train the network. The results initially looked promising. However, after evaluating the results with another similar approach named Kalasanty [SZS20], the accuracy was not better or even worse in some cases. Additionally, it was found that the minimum cluster distances were too

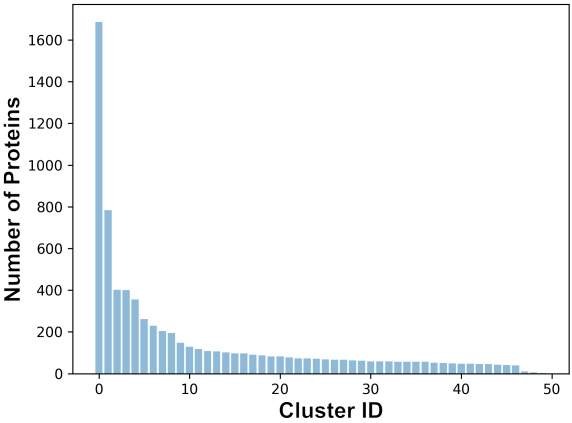


Figure 3.9 — Bar chart showing the distribution of cluster sizes based on the DBSCAN clustering of the TM-score matrix created of the PDBbind data set using *MM-align* (see also Figure 3.8). The bars are sorted according to cluster size, where 8,835 of 16,574 proteins are not assigned to any cluster and are not included in the chart. Used cluster parameters: $\varepsilon = 0.5$; $minPts = 40$.

high, i.e., the highest similarity found between proteins of two clusters was too high. This means the cluster separation was insufficient. Finally, this led to discarding the results because they could not be relied upon for the mentioned reasons.

The later used data sets varied depending on two different downstream tasks including fold classification and enzyme-catalyzed reaction classification, for which the network architecture was successfully tested. According to the first downstream tasks, the fold class is predicted for a given protein, whereby the performance is measured as mean accuracy. Therefore, the SCOPe 1.75 data set of [HAC18] was used to train the network. This data set consolidated 16,712 proteins from 1,195 folds. The 3D structures of the proteins were obtained from the SCOPe 1.75 database [Mur+95]. The data set provides three different test sets: *Fold*, in which proteins from the same superfamily are not present during training; *Superfamily*, in which proteins from the same family are not seen during training; and *Family*, in which proteins of the same family are present during training.

As aforementioned, the second downstream task was enzyme-catalyzed reaction classification. For this task, proteins are classified based on the enzyme-catalyzed reaction according to all four levels of the Enzyme Commission (EC) number [Web92]. Therefore, EC annotations from the SIFTS database [Dan+19] were utilized. This database contains EC annotations for entries of the RCSB PDB [Ber+00]. From the annotated enzymes, the protein chains were clustered using a 50 % sequence similarity threshold, as provided by PDB. The performance is again evaluated as mean accuracy. Here, a total of 37,428 proteins were collected from 384 enzyme commission numbers. The data was then split into 29,215 instances for training, 2,562 instances for validation, and 5,651 for testing. Note that all proteins have less than 50% sequence-similarity in between splits. A full description of the data set is provided in the appendix of the publication [Her+21].

3.2.5 Evaluation and Discussion

To evaluate the proposed protein learning approach, a deep architecture (Section 3.2.3) is specified, which is compared to state-of-the-art methods mentioned in the following paragraph. The entire evaluation focuses on two downstream tasks comprising protein fold classification and enzyme classification. An overview of all results showing the mean accuracy for these tasks is given in Table 3.1. The data sets used for training are described at the end of Section 3.2.4 The new network architecture is compared to state-of-the-art learners designed for similar downstream tasks. First, it is compared to the latest sequence-based methods pre-trained unsupervised on millions of sequences: [Rao+19; BB19; All+19; Str+20; Eln+21]. Second, it is compared to different methods that take only the 3D protein structure into account: [KW17; Die19; Der+18]. Lastly, the architecture is also compared to two recent hybrid methods: Gligorijevic et al. [Gli+19], which process primary structure with several LSTM cells first and then apply GCNN, and Baldassarre et al. [Bal+20], which indirectly process primary and secondary structures by using distances along the backbone as edge features and secondary structure type as node features as input in a GCNN setup. When available, the results reported in the original publications are given, while more details of the training procedures are provided in the appendix of our publication [Her+21].

One of the tested downstream tasks is **fold classification**. This task consists of the fold class prediction for a given protein, with performance measured as mean accuracy. As mentioned, protein fold classification is of key importance when studying the relationship between protein structure and function as well

Table 3.1 — Comparison of the new network architecture to other methods on the two downstream tasks comprising protein fold (FOLD) and enzyme catalytic reaction classification (REACT). They are measured as mean accuracy, where all methods are outperformed by the new network. [Her+21]

	Architecture	# params	FOLD			REACT
			Fold	Super.	Fam.	
HHSuite			17.5 %	69.2 %	98.6 %	82.6 %
[HAC18]	1D CNN	1.0 M	40.9 %	50.7 %	76.2 %	
	1D ResNet	41.7 M	17.0 %	31.0 %	77.0 %	70.9 %
[Rao+19]*	LSTM	43.0 M	26.0 %	43.0 %	92.0 %	79.9 %
	Transformer	38.4 M	21.0 %	34.0 %	88.0 %	69.8 %
[BB19]*	LSTM	31.7 M	17.0 %	20.0 %	79.0 %	74.3 %
[BB19]†	LSTM	31.7 M	36.6 %	62.7 %	95.2 %	66.7 %
[All+19]*	mLSTM	18.2 M	23.0 %	38.0 %	87.0 %	72.9 %
[Str+20]*	LSTM	22.7 M	14.9 %	21.5 %	83.6 %	73.9 %
[Eln+21]*	Transformer	420.0 M	26.6 %	55.8 %	97.6 %	72.2 %
[KW17]	GCNN	1.0 M	16.8 %	21.3 %	82.8 %	67.3 %
[Die19]	GCNN	1.0 M	12.9 %	16.3 %	72.5 %	57.9 %
[Der+18]	3D CNN	6.0 M	31.6 %	45.4 %	92.5 %	78.8 %
[Gli+19]*	LSTM+GCNN	6.2 M	15.3 %	20.6 %	73.2 %	63.3 %
[Bal+20]	GCNN	1.3 M	23.7 %	32.5 %	84.4 %	60.8 %
Ours		9.8 M	**45.0** %	**69.7** %	**98.9** %	**87.2** %

*Pre-trained unsupervised on 10-31 million protein sequences.

†Pre-trained on several supervised tasks with structural information.

as protein evolution. In the different fold classes, proteins are grouped that have similar secondary structure compositions, orientations, and connection orders (cf. Section 2.4.1 and SCOP/CATH [Mur+95; Ore+97]). The results of the comparison to the other tools are reported in Table 3.1. It can be observed that the new architecture performs better by a large margin without using additional features as input or pre-trained on extra data. At the top of Table 3.1, the architecture is compared to the state-of-the-art results reported in the publication [HAC18], which are all outperformed.

The second task is **enzyme catalyzed reaction classification**. This is the classification of proteins based on the enzyme-catalyzed reaction according to all four levels of the Enzyme Commission (EC) numbers [Web92]. The performance is again evaluated as mean accuracy, which showed that the new architecture also outperforms previous works on this task (cf. Table 3.1, column REACT).

The last task is a certain type of testing in the context of protein fold classification named **one-shot fold classification**. This task aims to answer the binary question of whether a pair of proteins is in the same fold or not, which were both not observed during training. This is also known as one-shot learning. Therefore, the siamese architecture from Koch et al. [KZS15] was adopted, which uses the same protein encoder as used in the presented new architecture. The same data set was used as in the aforementioned fold classification task but withheld a random subset of 50 folds out of the original 1195 folds during training. While testing, all proteins contained in the test set were compared to all proteins contained in a reference set that comprises all proteins of the original training set and thus represents all 1195 folds.

From each test protein, the pair with a higher probability was selected to predict the fold. To analyze how accuracy is affected by folds unseen during training, the fold test set was divided into two different test sets. The *seen* test set contained 419 proteins from 86 folds seen during training, while the *unseen* test set contained 299 proteins from the 50 folds not seen during training. When analyzing how many proteins could be predicted correctly, an accuracy of 39.0% can be obtained for *seen* and 31.9% for *unseen*. This indicates that the new method is able to generalize to protein folds that have not been seen during training. It also indicates that the new method is flexible enough to take concepts like the one from Koch et al. [KZS15], which uses hierarchical image convolutions and generalizes them to proteins. To view the resulting accuracies in context with the other previously reported values, the architecture has been trained for the standard classification of the fold classification task with the same training set composed of 1145 folds. It can be obtained that an accuracy of 46.5% was reached when testing against the 86 folds compared to the 39 % of the one-shot training.

Additionally, four different axes of ablations of the new approach were studied, including the *convolution*, *neighborhood*, *pooling*, and the *representation*. It was shown that each part of the new approach is important for an increased performance. A more detailed description and the explicit accuracy values are given in our publication [Her+21]. Despite the reported success a limitation of the new approach is that it requires the 3D structure of each protein, which

could be alleviated by advances in protein structure determination. Moreover, the convolution operator is invariant to rotations but also to chirality changes, which could be solved by incorporating directional information into the input of the kernel.

In conclusion, the new architecture provides a substantial basis for investigating protein-ligand interactions by learning the geometric features of proteins as an elementary component for mediating interactions with ligands. In addition, the chemical composition, in particular of the protein's surface, is important, as well as the depth and accessibility of a binding site. Therefore, it is necessary to use a comparison and classification method that also considers such features to reach a comprehensive understanding. An image-based approach that pursues this idea is discussed in the following Section 3.3.

3.3 Protein Surface Map Similarity Clustering

Parts of this section have been published in:

- K. Schatz, F. Frieß, M. Schäfer, T. Ertl, and M. Krone. "Analyzing Protein Similarity by Clustering Molecular Surface Maps." *EG Workshop Vis. Comput. Biol. Med.* 2020, pp. 103–114. DOI: 10.2312/vcbm.20201177
- K. Schatz, F. Frieß, M. Schäfer, P. C. F. Buchholz, J. Pleiss, T. Ertl, and M. Krone. "Analyzing the Similarity of Protein Domains by Clustering Molecular Surface Maps." *Computers & Graphics*. 2021, pp. 114–127. DOI: 10.1016/j.cag.2021.06.007

Protein-ligand interactions depend on geometric as well as chemical properties that determine how they interact with each other. The previously discussed approach in Section 3.2 considers all structural levels of proteins and is able to learn their geometric features, which contributes to analyzing the spatial or steric components of mediating interactions between proteins and ligands. This section presents a new image-based method that takes the protein topology into account and also incorporates physico-chemical aspects that were not sufficiently considered in the previous approach. The new method focuses on the protein's surface, as it represents the interface of the protein interacting with its external environment and having a major influence on its function. It is also discussed how molecular feature surface maps are created and how the features are extracted, which are used for distance measurement and clustering. Moreover, the new method is evaluated by comparing the results with the BRENDA databases [SCS02] and proteins compared by *MM-align* [MZ09]. Additionally, its feasibility is shown with an ensemble of protein domains from enzymes belonging to the carbohydrate metabolism.

The new approach is a protein comparison method that analyzes protein similarity by hierarchically clustering Molecular Surface Maps [Kro+17] as shown in Figure 3.10. The used maps incorporate the protein's topology or physico-chemical properties. The approach follows the basic assumption that similar shape and similar physico-chemical properties, as encoded in the surface maps, mean similar surface maps and, thus, mean similar function or interaction with ligands [BJ09; TL12]. This approach combines both geometric and chemical aspects. This is necessary when investigating evolutionary relations or searching for proteins with similar functions as it is of interest in drug development,

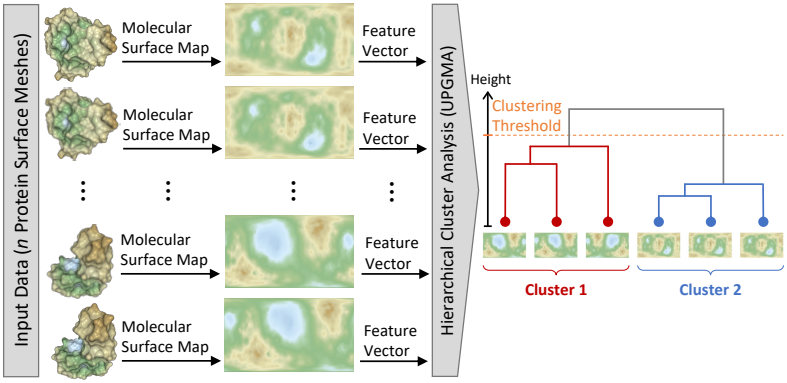


Figure 3.10 — Schematic overview of the hierarchical protein clustering approach (from left to right). As input serves an ensemble of proteins. For each protein, a three-dimensional molecular surface representation (SES mesh) is computed, which is subsequently transformed to a two-dimensional Molecular Surface Map [Kro+17]. From each of these maps, a descriptive feature vector is calculated using either *Image Moments*, *Color Moments*, or a Convolutional Neural Network. Based on the distances between these feature vectors, a hierarchical clustering is performed using the UPGMA algorithm [SM58]. - modified figure © Elsevier 2021 [Sch+21b]

systems biology, and biotechnology. To avoid misinterpretations and to not overlook proteins, which can be the case when, for example, only considering the primary structure, it is necessary to consider multiple aspects because of the phenomenon of invariant protein structures as described in Section 3.1.

3.3.1 Molecular Surface Maps

The key element of the comparison method is a map representation of the protein surface, which is discussed in this section. As mentioned before, the molecular surface of a protein has a significant influence on its function, but molecular surfaces are visually complex and suffer from occlusion, as all three-dimensional depictions. Krone et al. [Kro+17] presented Molecular Surface Maps to solve the 3D occlusion problem and to reduce visual complexity. It was shown that these two-dimensional representations, showing physico-chemical properties as well as the topography, can be used for overview and comparison.

In general, map-like protein representations are primarily based upon a spherical description of the geometry. Rahi and Sharp [RS07] re-parametrize the surface vertices of a molecule into spherical coordinates. This allows the surface to be mapped onto a sphere in order to compare multiple proteins more easily. The actual comparison, however, is only visual and does not provide any countable distance measures.

Hasegawa and Funatsu [HF12] created a further protein mapping method, which uses a spherical self-organizing map to achieve a projection from the protein surface onto a sphere. Due to their algorithm, the results are not necessarily smooth and may contain holes that can get larger when the size of the projected protein decreases. Hass and Koehl [HK14], on the other hand, use conformal mapping to measure the roundness of a given protein. Opposed to the new approach, they do not incorporate further properties of the protein surface. In their Structuprint method, Kontopoulos et al. [Kon+16] project the protein onto a sphere by casting a straight line from the center point through the surface points. This approach, therefore, can lead to undesired overlaps of projected surface parts, which is the case for protein tunnels or cavities that are also the subject of mapping methods. Kolesár et al. [Kol+16] unfold occurring tunnels into a map-like representation. They also use the *Image Moments* method to compare different tunnels, for which it is later shown that it is not suitable for the needs of the new approach. To further resemble the shape of the original cavity, Schatz et al. [Sch+19] choose a hat-like shape as primitive, which can be further simplified to a map in the form of a disk. While depicting the original shape better, this can lead to heavy surface distortions.

The Molecular Surface Maps method of Krone et al. [Kro+17] builds up on the approach of Rahi and Sharp [RS07], and further transforms their spherical descriptions. This approach is used in the new image-based comparison method. The Molecular Surface Map approach uses conventional map projection methods that are normally used in cartography to generate two-dimensional maps of protein surfaces. Additionally, as opposed to other methods, occurring tunnels of the protein are closed before any projections are applied to achieve an artifact-free representation. Their results sometimes suffer from the drawbacks of the selected map projection method, as there exists no mapping between a sphere and a map that is able to conserve area as well as distance perfectly.

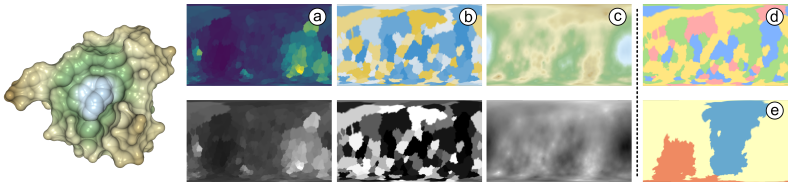


Figure 3.11 — Overview of the different maps for the pyrimidine (PYR) domain of a transketolase from *Pseudomonas aeruginosa* (PDB ID: 4XEU), which is shown to the left as molecular surface. The color-coded maps (a – c) in the top row were generated using the scalar maps that are shown in the bottom row (black and white maps). The two maps to the right are based on nominal data and thus have no value maps. The maps display: (a) B-factor coloring, using the Viridis color map ( ). (b) Hydrophobicity interpolated between blue (hydrophilic), white (neutral), and yellow (hydrophobic) ( ). (c) Cartography-inspired coloring by elevation, i.e., the distance in Å between the surface and the centroid of the protein ( ). (d) different physico-chemical properties of amino acids: ( ) basic, ( ) acidic, ( ) polar, or ( ) unpolar. (e) coloring by contact areas where the surface of the ( ) PYR domain is in contact with the two other domains (( ) PP domain and ( ) TKC domain). The values of the scalar maps have been normalized for better visibility. - modified figure © Elsevier 2021 [Sch+21b]

3.3.2 Algorithm Overview

The algorithm of the new approach consists of four steps. First, the proteins are aligned using the most feasible method. In the second step, three Molecular Surface Maps are created for each protein of the input data set (see Figure 3.10). Each map contains the scalar values of the associated surface property, including the B-factor, hydrophobicity, and topology encoded as heightmap as proposed by Krone et al. [Kro+17]. An example set of the three mentioned surface properties is shown in Figure 3.11 as colored maps, and directly below them, the corresponding scalar values are depicted as black and white images. The hydrophobicity describes how energetically (un-)favorable an interaction with water molecules would be, and the temperature factor (B-factor) is an indicator of the flexibility of the protein. It is specified per atom, and it can also be understood as the uncertainty of the atom's position or atom's mobility. In the third step, a feature vector for each generated map is computed, resulting in three feature vectors for each protein. They can be concatenated to a vector containing two or three of the original vectors. The final similarity comparison is based on descriptive feature vectors, where three different methods are tested

to compute them. They include a calculation either using *Image Moments* [Hu62; Flu00], *Color Moments* [MSM09], and a CNN [San+18]. For the *Color Moments* a color mapping from the scalar values to RGB values is used (see Figure 3.11). The other two methods work directly on the scalar maps. In the fourth step, the distances between the feature vectors are used for a hierarchical clustering performed by using UPGMA as described in Section 2.4.2, and the results are visualized as phylogenetic trees [FM67].

3.3.3 Protein Alignment and Map Creation

The visual appearance of the Molecular Surface Maps highly depends on the orientation of the protein. Even for very similar proteins, however, the orientation can differ widely in the Protein Data Bank (RCSB PDB). Therefore, it is necessary to align the proteins as well as possible before the mapping step. For similar proteins, minimizing the RMSD between corresponding atoms is one of the most commonly used alignment methods (see Section 3.1). However, since the input data can be highly heterogeneous, this approach is not always feasible, as it is impossible to establish the necessary per-atom correspondence for highly different proteins. Thus, a more general approach based on PCA of the atom positions is used in this case. Each protein is rotated in such a way that the first principal component is aligned to the x-axis, and the second principal component is aligned to the y-axis. This fast method leads to proteins with similar shapes being oriented similarly, regardless of the underlying chemical sequence. This works for elongated as well as globular proteins. A possible flipping in either x- or y-direction is later handled by the rotationally invariant feature extraction methods. However, since the established RMSD method is more accurate, it should be applied to all data sets where possible (e.g., similar protein domains).

The new comparison method is imaged-based and the used figures are created using a modified version of the *Molecular Surface Map* algorithm of Krone et al. [Kro+17]. This original method comprises four main steps, as exemplary depicted in Figure 3.12. The first step is the computation of the molecular surface, followed by tunnel detection and tunnel closing to get a hole-free surface. The third step is a morphing of the received *Molecular Surface Globe* into a sphere. The last step applies a map projection as used in cartography to perform a mapping of the sphere to a 2D *Molecular Surface Map*.

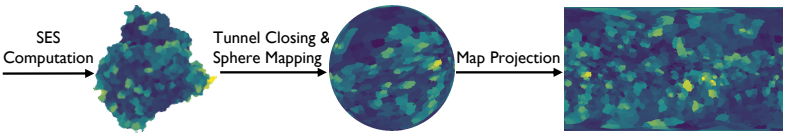
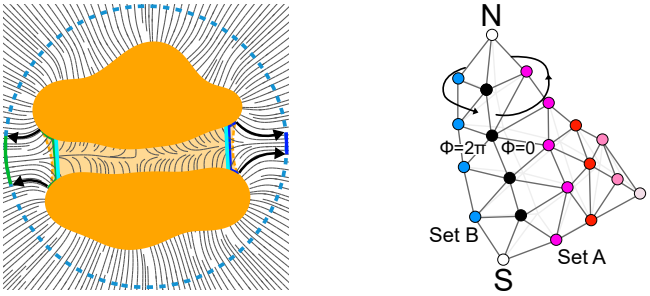


Figure 3.12 — Schematic overview of the steps involved in the generation of a Molecular Surface Map. From left to right: Solvent Excluded Surface (SES), Molecular Surface Globe, and Molecular Surface Map. The protein is colored by temperature factor (B-factor) using the Viridis color map ( ). The coloring indicates the flexibility of the protein (PDB ID: 1AJN). - modified figure © Elsevier 2021 [Sch+21b]

SES Computation The Molecular Surface Map algorithm uses the Solvent Excluded Surface (SES). As described in Section 2.5, it depicts the molecular surface with respect to a smaller molecule like a ligand or, more often, with respect to a solvent such as water. It intuitively shows the protein interface and is useful for detailed analysis. To be able to manipulate the surface more easily in the following steps, an explicit triangulated surface in the form of a mesh is needed (see Figure 3.13 (b)). For the creation of the SES mesh, Krone et al. [Kro+17] used the *MSMS* tool of Sanner et al. [SOS96].

Tunnel Closing The Molecular Surface Map algorithm uses an intermediate object, which is the Molecular Surface Globe. This is because the globe allows the use of established map projection algorithms utilized in cartography. To morph the SES into the Molecular Surface Globe, the surface that can be of genus n has to be converted to genus zero. That means a surface with holes, such as a torus, is converted to a surface with zero holes, such as an ellipsoid or sphere. Proteins can have tunnels or holes, e.g., transmembrane proteins, for ion transport, which results in a non-zero surface. To enable the transformation to genus zero, tunnels have to be detected and closed. Therefore, they used a Ambient Occlusion (AO)-based tunnel detection method presented by Krone et al. [Kro+13]. To reduce the distortion when mapping the surface to a sphere, tunnels are cut near their entrance, such as shown in Figure 3.13 (a). The tunnel is removed, and the entrances are closed using a triangle fan.

Sphere Mapping After the tunnels are closed and the molecular surface is of genus zero, the morphing into a Molecular Surface Globe is performed. In the approach of Krone et al. [Kro+17], two different methods are proposed to achieve



(a) Force-Directed Mapping **(b)** Assignment of Spherical Coordinates

Figure 3.13 — Schematic representation of the two used methods for mapping a molecular surface to a sphere as available in the implementation of Molecular Surface Maps of Krone et al. [Kro+17] (see Figure 3.12). (a) shows a force-directed approach [Sch+14] for mapping the molecular surface (yellow) via forces (gray lines) to a sphere (blue dashed circle). The turquoise lines show where the protein tunnel is cut and closed to get a genus zero surface for sphere mapping. (b) depicts a re-parameterization of the molecular surface by assigning spherical coordinates to the vertices. This parameter-based method [RS07] assigns Φ values. The vertices of *Set B* have as Φ values 2π and the vertices of *Set A* have as Φ values 0 assigned. The boundary meridian (black vertices) connects the North Pole (N) and South Pole (S). - © IEEE 2017 [Kro+17]

the sphere mapping. The methods comprise a force-directed mapping and a variant, where the surface is re-parametrized by assigning spherical coordinates to the vertices of the surface mesh (see Figure 3.13). The force-directed approach can be seen as having a surrounding sphere and the molecular surface is pulled to it. The pulling forces are indicated as gray lines in Figure 3.13 (a). For their force-directed approach, they used a modified Gradient Vector Flow (GVF) [XP98] as also used in an earlier work of Scharnowski et al. [Sch+14]. It calculates the external forces by diffusing the molecular surface normals and the normals of the surrounding sphere. The alternative parameterization-based approach assigns spherical coordinates (z, Φ, θ) to the vertices of the surface mesh. For this approach, they are using the algorithm of Rahi and Sharp [RS07], whose principle is depicted in Figure 3.13 (b). In the original algorithm, the poles are determined by the vertices with the biggest Euclidean distance. In their adjusted variant, the domain scientists determine the original orientation of the protein, which is used to assign the North and South Pole. Then the value

$z = +Z$ is assigned to the neighbor vertices of the North Pole and $z = -Z$ to the neighboring vertices of the South Pole [Kro+17]. The vertices between these get the averaged z values of their neighbors, where Z is a number greater than zero. The θ values are determined based on the z values using inverse Mercator projection. For assigning the Φ values, the boundary meridian between the poles is determined following the steepest slope of the connections between the vertices (cf. Figure 3.13 (b)). Their results rate the parameter-based approach as good for complex surfaces and the force-directed method as more suitable for globular surfaces without deep cavities or large protrusions.

Map Projection The final step of the Molecular Surface Map algorithm consists of the projection of the Molecular Surface Globe to a 2D map. In theory all possible map projections can be used, but for their approach they tested only a couple of reasonable ones. This means, those with favorable properties like equal area, as they are familiar to many users, i.e., they are more suited for visual analysis. Thus, two projection variants of the cylindrical equal-area projection are offered, including the Lambert and Hobo-Dyer projection, and a cylindrical equirectangular projection according to Plate Carée is also available. Both create very similar results and introduce a low error through the projection. This is important since the sphere mapping leads to unavoidably non-uniform point distributions and can not fully preserve the distance of the vertices. This concerns especially points of *valleys* and *mountains* and their surrounding points of the molecular surface. The projections lead to a flattening of the valleys and mountains, which is a problem in cartography as well [Kro+17].

Adjustments The previous paragraphs outline the individual steps of the original Molecular Surface Map algorithm, which can be summarized as the computation of the molecular surface as SES, the tunnel detection, and tunnel closing. This is followed by the morphing of the molecular surface to a sphere (Molecular Surface Globe) and the final projection to a 2D map (Molecular Surface Map). For more details, especially regarding the implementation, please refer to the original publication of Krone et al. [Kro+17].

For the new image-based approach, the original Molecular Surface Map algorithm was adjusted. One change affects the generated map so that it now directly contains the scalar values of the chosen physio-chemical property. That is, suitable color scales can be applied subsequently. Additionally, the computationally expensive tunnel detection and closing step was replaced with a new variant, which is fast and, thus, results in lower computation times for large

protein ensembles. The new variant iteratively increases the probe radius used to compute the SES mesh. As proposed by domain experts in biochemistry, it starts with a radius of 2.4 Å and increases it in steps of 0.2 Å until 4.0 Å is reached, or the mesh is of genus zero. For cases where the new faster approach fails, i.e., if the mesh is still not of genus zero when reaching the maximum radius of 4.0 Å, the original, slower variant by Krone et al. [Kro+17] is used. However, for the used test data, this applied only for less than 10% of the proteins. The faster approach for the tunnel closing can result in a comparison of surfaces generated for different probe radii. As a change of the probe radius only affects small local concave parts of the surface, the effects on the resulting surface are minimal.

While the map for most coloring modes is derived directly from the properties of the depicted atoms or amino acids, the contact map (cf. Figure 3.11 (e)) requires an additional pre-computation step. Contact maps are computed by calculating atom-atom distances between the viewed protein and one or more contacting molecules. A contact is defined as the undercutting of a user-defined threshold distance, where a default value of 5 Å was used in most cases. If this is the case, the surface color is changed to the color of the contacting molecule. If two or more atoms are in contact, the map is colored according to the closest one.

3.3.4 Feature Determination and Clustering

To cluster the Molecular Surface Map images generated in the previous step, a descriptive feature vector is assigned to each of them. To generate the descriptive values, three different approaches were tested, which comprise *Image Moments*, *Color Moments*, and *MobileNetV2* as a CNN-based feature computation. The different methods calculate a feature vector, whose size differs from method to method, including a vector with seven, nine, and 1792 elements used to represent the image uniquely. The evaluation of the different feature vector calculation methods is described in Section 3.3.6.

For the *Image Moments* approach and the *MobileNetV2*-based features, the calculated maps can be directly used to determine the descriptive values. In contrast, when using *Color Moments*, a color scale must first be applied to the scalar values of the maps. This, in turn, means that the quality of the result heavily depends on the quality of the used color scale, but without this step, there would not be enough features to represent the image accurately. When using one of these approaches, a feature vector is assigned to each Molecular Surface

Map calculated in the first step. A more unique and meaningful map description can be created by increasing the number of different features describing it. Therefore, multiple Molecular Surface Maps representing different quantities, as shown in Figure 3.11 for B-factor, hydrophobicity, and the heightmap of the same protein, can be combined by concatenating the individual feature vectors of the maps. An advantage of concatenating the feature vectors is that it can take mutations of proteins into account that only change certain quantities. An example would be a protein where a hydrophobic amino acid contributing to the surface is replaced with a hydrophilic amino acid of similar size. This would not affect the heightmap-like Molecular Surface Map but would change the hydrophobicity-based one.

Image Moments The *Image Moments* are derived from the intensity value of a pixel $I(x,y)$. Therefore, the Molecular Surface Maps can be used without further preparation. In order to get descriptive values that are invariant under translation, rotation, and scaling, the Hu moments invariants [Hu62] are computed. From these seven *Image Moments* ($I_{1,\dots,7}$), the moment I_3 depends on other moments, and because of this, it is replaced by the additional calculation of I_8, as suggested by Flusser [Flu00]. This results in a feature vector F_{im} for each Molecular Surface Map containing seven *Image Moments*:

$$F_{im} = (I_1,\ I_2,\ I_4,\ I_5,\ I_6,\ I_7,\ I_8) \tag{3.4}$$

Color Moments As the second feature extraction method the *Color Moments* are used. They are computed as proposed in the work of Maheshwari et al. [MSM09] using the color information as input. Therefore, the scalar values of the map are color-encoded as depicted in the top row of Figure 3.11. The first three elements in the resulting feature vector are the mean values ($E_{R,G,B}$) of each RGB color channel, followed by the standard deviation ($SD_{R,G,B}$) of the individual RGB channels. The final three components represent the skewness ($S_{R,G,B}$) of each color channel, which can be understood as a measure of the degree of asymmetry in the distribution. The final feature vector F_{cm} of the *Color Moments* contains nine elements uniquely describing the associated map.

$$F_{cm} = (E_R,\ E_G,\ E_B,\ SD_R,\ SD_G,\ SD_B,\ S_R,\ S_G,\ S_B) \tag{3.5}$$

MobileNetV2 The third feature computation method is *MobileNetV2*, which is the name of a certain CNN. It was developed by Sandler et al. [San+18] and is provided as a *TensorFlow* module. An introduction to Artificial Neuronal Networks (ANNs) can be found in Section 2.4.1. The *MobileNetV2* network is not applied as a whole to the map images since its output is not a feature vector. Due to its modular structure, the authors provide a smaller version of the network that only calculates image feature vectors, which are used as input for the subsequent calculations of the new Molecular Surface Map-based clustering approach. Compared to the two previously explained feature extraction methods, the feature vectors retrieved from *MobileNetV2* are significantly more extensive. The obtained feature vector F_{nn} for each map contains 1792 elements. The feature vector is also invariant under translation, rotation, and scaling.

$$F_{nn} = (F_0, F_1, \ldots, F_{1791}) \tag{3.6}$$

Tile-based Feature Computation: As an additional approach to the previously mentioned feature computation methods, a tile-based feature computation was implemented. This enables a local-to-global comparison, which can lead to better results, especially for proteins that contain local distortions but overall exhibit the same surface structure and properties. Comparing the maps as a whole can result in a greater global distance even though the function is very similar. For such cases, a tile-based approach was developed that subdivides the maps into overlapping smaller images. That is, the small tiles of two maps are compared individually, leading to a not necessarily same global distance value when compared to the distance that is calculated by a comparison of the maps as a whole.

In detail, the approach consists of subdividing the value images into smaller overlapping tiles of fixed size. When aiming to achieve the highest accuracy, this overlap should be as large as possible as the viewed feature might be contained in the other map, only moved by a few pixels. In the current implementation, the overlap is set to 50%, as it is a compromise between computational cost and accuracy. The size of individual tiles can further depend on the use case and, therefore, on the size of the displayed proteins. The general target is to keep the tiles small enough to avoid the aforementioned problems and big enough to contain at least the footprint of one amino acid. As a default value, a tile is of the size of 160×160 px, which fits the default map resolution of 2560×1440 px, and the average enzyme size. After all tiles are generated, the feature vectors are determined for each of them. Then, a comparison is performed for each tile of a map A individually against map B. A feature vector of one tile of map A

is compared to all tile feature vectors of map B by determining the Euclidean distance between them. From these distances, the smallest one is chosen as the representative distance for the current tile of map A. The final distance between the two maps is then calculated by averaging the thus-determined distances.

While this approach can lead to more accurate results for certain use cases (see Section 3.3.6), it has a significantly higher computational cost than comparing whole maps. Due to the increased computational complexity, it is mainly feasible for smaller data sets (<100 proteins).

Hierarchical Clustering One of the most well-known clustering applications in biology is the phylogenetic analysis using UPGMA as described in Section 2.4.2. For the new method, UPGMA is also used since it constructs rooted trees, and the results can be interpreted easily, showing the evolutionary relationship of the proteins. Thus, after the feature computation, the hierarchical clustering with UPGMA is the next step in the algorithmic pipeline. This is performed on n input proteins using their feature vectors computed as described before. The Euclidean distance d_{ij} is calculated between each pair of feature vectors in order to get a $n \times n$ distance matrix. This matrix is used as input of the hierarchical clustering algorithm UPGMA described in Section 2.4.2. This clustering computes a binary tree containing all surface maps or proteins respectively (cf. Figure 3.10).

Initially, each protein is treated as a leaf node (cluster) as in the standard UPGMA algorithm. A particularity is that these leaf nodes contain a feature vector and the corresponding Molecular Surface Map. During the following agglomerative clustering, new cluster nodes are created by merging two clusters with the closest distance found in the distance matrix. Additionally, to the normal UPGMA procedure, the newly created node gets a descriptive feature vector assigned as well, which is computed as the centroid of the feature vectors of all leaf nodes contained in the two merged subtrees. Also, a representative Molecular Surface Map is determined for this new node or subtree. This is done by comparing each feature vector of the leaf nodes against the centroid feature vector. The Molecular Surface Map, which is the most similar to the centroid feature vector, is then used as the representative map. The whole process is continued iteratively until only one cluster remains, and a binary tree has been formed.

The UPGMA algorithm is able to generate trees where the distance between nodes corresponds to the distance in the distance matrix. Such a tree allows for an intuitive visual analysis, as the distance values can be estimated directly

from the visualization. Using a user-defined similarity threshold T_c the final clusters are determined based on the UPGMA tree. All nodes that belong to a subtree of less than the height T_c (dashed orange line in Figure 3.10) belong to the same cluster.

As a restriction, the clustering can only be applied to map types where the values represent quantitative or ordinal data (e.g., hydrophobicity). If maps contain only nominal or categorical data with no inherent order, it is not possible to determine the distance between them. That means a distance function necessary to perform the clustering cannot be formulated. As the physico-chemical type map (d) and the contact map (e) in Figure 3.11 contain nominal data, they are only used for display and analysis purposes and are not used for Molecular Surface Map clustering. The physico-chemical type maps display nominal data by showing the most prominent physico-chemical property for each amino acid comprising the categories basic, acidic, polar, and nonpolar.

3.3.5 Interactive Result Visualization

Based on the *MegaMol* visualization framework (see Section 2.6.1), a prototypical application was implemented to visualize the clustering results. The visualization consists of three linked views, enabling an interactive exploration and analysis of the hierarchical clustered proteins. For a detailed comparison, two proteins can be selected individually, which are presented in the detail-views as shown at the bottom of the application overview in Figure 3.14. The size of the views can be adjusted by moving the gray separation lines. All visualizations were implemented using OpenGL to achieve a fast rendering.

The top view shows the UPGMA hierarchical cluster results as a dendrogram. A categorical coloring is applied to the different subtrees to encode the cluster affiliation. The leaf nodes represent the proteins used in the clustering. When the mouse pointer is moved over a leaf node, the map of the corresponding protein is displayed, and for non-leaf nodes, the most representative map of the subtree is presented (see Figure 3.14, `2R8O_PYR`). By default, the heightmaps are displayed, as they are more easily distinguishable for humans, but the user can also choose to see the other map types, such as presented in Figure 3.11. As mentioned, two proteins can be selected for a detailed comparison and analysis, enabled by the two detail-views below the dendrogram. They show the 3D molecular surface as SES of the selected proteins. The tree's leaf nodes corresponding to the selected proteins are highlighted in red (cf. Figure 3.14). In addition to the SES, the detail-view shows the corresponding Molecular Surface

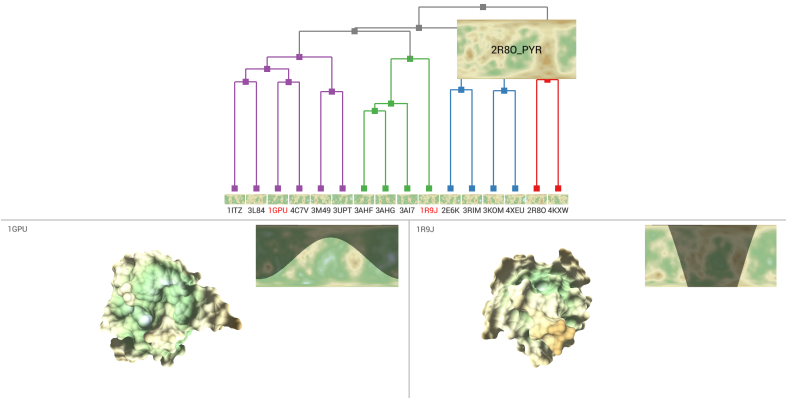


Figure 3.14 — Overview of the visualization application for analyzing the clustering result. The top view shows the hierarchical binary tree, colored according to the clustering. Selected proteins are marked with red text. At the bottom, two detail-views show the molecular surfaces of the proteins corresponding to the selected maps. Each surface view contains an inset that shows the corresponding map, where the area is highlighted according to the 3D surface part the user is looking at. Both the maps and the surfaces are displayed using the heightmap coloring. In this example, the two detail-views are unlinked to allow an individual adjustment of the camera for each detail-view. - © Elsevier 2021 [Sch+21b]

Map as well, which is placed as an inset in the top-right corner, together with the protein ID in the top-left corner. To provide better orientation for the user, the part of the 3D molecular surface facing the viewer is highlighted on the map, comparable to the common depiction of the terminator on a world map. The highlighting of the viewer facing SES was implemented using a shader-based approach utilizing the view direction and the inverse of the map projection as used for map generation. Moreover, zooming and panning are possible in all views, and in addition to that, the two 3D views support rotations as well. The 3D views can be linked to become a coordinated view, i.e., the aligned proteins will rotate and translate simultaneously.

3.3.6 Evaluation and Discussion

The new image-based approach for hierarchical clustering proteins aims to create clusters of proteins with similar structure and function. In this section, the new method and the three different similarity measures (*Image Moments*, *Color Moments*, *MobileNetV2*) are validated according to this goal using two different resources. One is Enzyme Commission (EC) classification, which is a naming and numbering convention [Web92] classifying enzymes based on their function (catalyzed reaction). The second one is *MM-align* determining a structural similarity score between two proteins. Moreover, the feasibility of the new method is shown for an application-driven use case by a comparative analysis of the domains of functionally similar proteins.

Two different protein data sets are used to test the approach. The first data set consists of ten proteins retrieved from the PDB, which are selected based on sequence-based comparison search (*BLAST*) so that they form two clusters and contain one outlier. This data set is used to evaluate the correctness and feasibility of the proposed method. The second data set is an ensemble of 50 functionally similar proteins provided by the domain scientists who also co-authored the original publication.

Enzyme Commission (EC) Numbers To test whether the new approach is able to find clusters of proteins with similar functions, established pre-clustered results are used for verification. Therefore, the EC classification is utilized, which is a naming and numbering convention [Web92] that can be obtained from sources such as the BRENDA database (BRaunschweig ENzyme DAtabase [SCS02]). The EC numbers group the enzymes based on their function or the catalyzed chemical reaction. An enzyme is described by a hierarchical code of EC numbers consisting of four digits. Since codes of EC numbers describe a chemical reaction, enzymes from different organisms catalyzing the same reaction have the same enzyme code. An example code for the enzyme *hydroxynitrile lyase* from almond (PDB ID: 1JU2) is `EC 4.1.2.10`. The first number indicates the general role of the enzyme and is the main class, such as oxidoreductase, transferase, lyase, etc. [Web92]. The meaning of the second two numbers depends on the first one. For example, one number could indicate the donor of the reaction and another the acceptor. A general description of the last number is not available, as it just further subdivides the pre-defined classes. Therefore, this classification does not represent a true hierarchical clustering, as the order of the inner values is defined as-is and could be interchangeable. The aforementioned example enzyme code of EC numbers assigns the enzyme to

Table 3.2 — Possible matching cases are presented that result in the comparison of two EC numbers of the form `EC 4.1.2.10` for two proteins. Each matching case is assigned a distance value that allows comparing proteins classified by EC numbers. The column headers represent their position in the hierarchical order. The table lists a ✓ when the classes on the same position match and a × otherwise. - modified table © EG 2020 [Sch+20b]

EC Number Position ①	②	③	④	assigned distance
×	×|✓	×|✓	×|✓	4
✓	×	×	×|✓	3
✓	✓	×	×|✓	2
✓	×	✓	×|✓	2
✓	✓	✓	×	1
✓	✓	✓	✓	0

the main class of lyases (`EC 4`), the subclass of carbon-carbon lyases (`EC 4.1`), the sub-subclass of Aldehyde-lyases (`EC 4.1.2`) and as further specification, it describes the enzyme as hydroxynitrile lyase of almond (`EC 4.1.2.10`). To enable a comparison between two enzyme codes, a distance value for specific configurations of two enzyme codes is assigned as shown in Table 3.2. The exact scalar distance values are arbitrary, as it was necessary to set numerical distance values for the comparison of categorical enzyme codes.

MM-align & TM-score As the second method for evaluation, the tool *MM-align* [MZ09] is used, which computes a similarity score between two proteins. It takes the sequence information as well as the spatial arrangement of the protein chains into account. The similarity score given by *MM-align* is the established TM-score, which is described in detail as well as *MM-align* in Section 3.1.3.

Method Evaluation First, a small data set of ten proteins was constructed, which contains two clusters and one outlier. This was achieved by selecting three sufficiently different proteins: myoglobin, lysozyme, and luminescent protein. For myoglobin and lysozyme, a sequence-based similarity search was performed. Out of the search results, the most similar proteins were picked, resulting in a ten-protein data set comprising a myoglobin cluster (1MLQ, 1CIO, 108M, 101M, 2SPO) and lysozyme cluster (115L, 108L, 107L, 102L) and one outlier the luminescent protein (4I38). On this data set, all three feature extraction methods were tested using different Molecular Surface Maps and combinations of them, as exemplary presented in Figure 3.15. All tests were evaluated in terms of their agreement with the ground truth, with the results shown in Table 3.3.

(a) Heightmap, MobileNetV2

(b) Hydrophobicity, MobileNetV2

(c) B-Factor, MobileNetV2

(d) Heightmap, Image Moments

(e) Hydrophobicity, Color Moments

(f) Heightmap & Hydrophobicity, MobileNetV2

Figure 3.15 — Comparison of similarity trees created with UPGMA using the ten protein data set (cf. Section 3.3.6). The individual captions note the used physicochemical property (coloring mode: Heightmap, B-factor, Hydrophobicity) and the utilized feature extraction method (*MobileNetV2*, *Color Moments*, *Image Moments*). Each subfigure displays the following items named in order from top to bottom: the generated tree, the Molecular Surface Maps of each leaf node, the respective PDB ID, and the enzyme code (EC numbers) if it is available. Proteins with an enzyme code should form a cluster and the remaining proteins another. Protein 4I38 is considered an outlier. - modified figure © EG 2020 [Sch+20b]

Table 3.3 — Color-coded results of the clustering for the ten protein evaluation set (cf. Section 3.3.6) using each of the three feature extraction methods: *Image Moments* (IM), *Color Moments* (CM) and *MobileNetV2* (MN) and different map coloring modes: Heightmap (H), Hydrophobicity (Y), B-factor (B) as well as combinations H-Y and H-Y-B. The expected result is two clusters, C1 and C2, as well as one outlier O. Configurations with (■) *proper clustering* are marked blue and (■) *unsuccessful configurations* are colored orange. (■) *Borderline cases* got yellow assigned, where a wrong protein was added to a cluster at a later point in time, indicating a large distance to the rest of the cluster. For H-Y and H-Y-B multiple feature vectors of the maps are combined either without or with the B-factor features, respectively. - © Elsevier 2021 [Sch+21b]

	IM			CM			MN		
Map	C1	C2	O	C1	C2	O	C1	C2	O
H	–	–	–	+	+	○	+	+	+
Y	+	+	+	+	+	+	+	+	+
B	–	–	–	–	–	+	+	–	–
H-Y	+	+	+	+	+	○	+	+	+
H-Y-B	–	–	–	–	–	+	+	+	+

As ground truth, the information about the functional similarity of enzymes was used, which is provided by the EC numbers. In addition, a UPGMA-based clustering was performed on a similarity matrix created with *MM-align*, which formed exactly the clusters and outliers mentioned above.

From Figure 3.15, it can be seen that using *MobileNetV2* leads to a result where the two clusters and the outlier are generally well preserved (Figure 3.15 (a) and (b)). The only exception is the B-factor map in (c), where only one cluster is preserved properly. The same problem occurs with the B-factor map for Image and *Color Moments* as well. In contrast to *MobilenetV2*, the method with *Image Moments*, as shown in (d), is not able to correctly categorize proteins according to their similarity, but the *Color Moments* method (e) leads to similarly good results as *MobileNetV2*. Figure 3.15 (f) shows the combination of two map types (heightmap, hydrophobicity) that mutually compensate for the shortcomings of each map type.

When comparing the final trees of the different methods as shown in Figure 3.15, it can be noticed that *Image Moments* and *Color Moments* tend to have a more diverse distance value distribution than the *MobileNetV2* features. That is, the distance between the closest proteins is far smaller compared to the distance between the farthest ones. Therefore, many merges happen at the very bottom

of the dendrogram as opposed to the *MobileNetV2* results due to the length of the feature vectors. As the feature vector given by *MobileNetV2* has 1792 entries, even small distances for single vector elements add up to larger numbers. Although the base difference values are generally high, the quality of the results is superior to the other two methods.

To cover multiple features of the protein, including spatial and further physicochemical properties, three different map types were chosen. The heightmap is one of these map types representing the geometry of the protein surface. A further map type encodes hydrophobicity that influences solubility and binding behavior, e.g., the potential to form hydrophobic interactions between the protein and a ligand. The third map type considers the B-factor, indicating the flexibility of the protein, its amino acids, and chains. It also influences binding behavior and can affect the catalytic rate of an enzyme. However, as it can be seen in Table 3.3 the B-factor map is not even sufficient to characterize proteins, which are very distinct, resulting in wrong or missing clusters for the ten protein data set.

Furthermore, it is not decisive for the protein function, and the B-factor also depends on environmental conditions that influence the stability of the protein, such as the pH value or temperature. These facts and the poor performance led to the decision to omit the B-factor maps for other data sets. This choice was also discussed with a biochemist, who confirmed the assessment that the B-factor or other flexibility measures are usually not considered to be expedient for a larger comparison of different proteins. Only in very specific cases, such as when investigating changes in the catalytic rates of an ensemble of similar enzymes, e.g., different mutants of an enzyme, clustering based on flexibility could be interesting. However, this level of detail analysis is not the focus of the new image-based clustering approach.

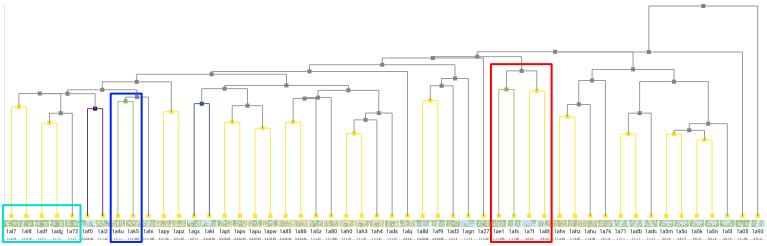
For the other two map types, it was determined that the clustering quality depends on the feature extraction method. *Image Moments* were successfully used by Kolesár et al. [Kol+16] to calculate similarities for maps of protein cavities. In the use case presented here, however, they performed worse than *Color Moments* and *MobileNetV2*. The *Image Moments* even considered one of the enzymes as a clear outlier while merging the actual outlier early during the tree construction. It is assumed that Kolesár et al. obtained good results since their maps consisted of large distinct patches that differed only slightly between the maps. Although *Color Moments* strongly depend on the chosen color scale, they performed considerably better than *Image Moments*. However, they exhibit weaknesses in the detection of outliers (cf. Table 3.3). *MobileNetV2* produced

the overall best results, especially when considering that the results of using *Color Moments* features might get worse when choosing a different color scale.

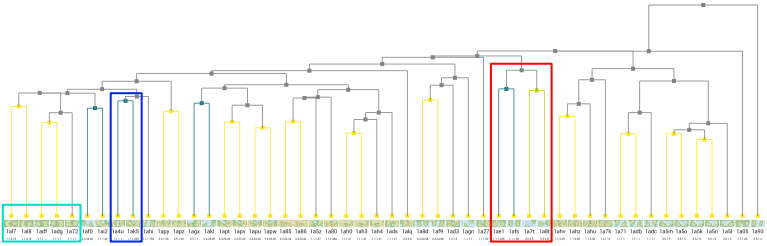
As mentioned in Section 3.3.4, the approach can also combine multiple map types of each protein to perform the clustering. This can be useful since multiple properties like hydrophobicity and the shape of the surface influence the function of a protein. Their combination increases the number of features, which allows a more detailed characterization of the proteins and their function. As shown in Table 3.3 and Figure 3.15 (f), the *MobileNetV2*-based clustering produces good results, which exactly correspond to the ground truth. Since *MobileNetV2* achieved the best results on the small verification data set, it is the only method used for the larger data set and the protein domain use case. It is always applied to the combined features of the heightmap and the hydrophobicity map.

Comparison with EC classes Figure 3.16 (a) shows results for the ensemble of 50 proteins. This data set consists of short, randomly chosen sections of enzymes, where each section consists of enzymes with a similar EC classification. This ensured that the resulting set of 50 proteins contained multiple clusters of enzymes with similar function. When inspecting the clustering results, it can be observed that both visually and functionally similar enzymes are merged early in the tree, which can be seen in the cyan-colored box, for example. These proteins were identified as being similar by the new method and are also classified as being functionally similar according to the EC classification. Note that only the branches are colored where leaf nodes are directly merged since those are most easily interpretable. For the other cases, it is unclear which value to assume as the basis for proper coloring. Most of the clustering results are consistent with the EC classification. This means that proteins of the same class and, therefore, the same function are located in the same cluster. Two outlier pairs are visible: 1AGX and 1AKL, and 1AF0 and 1AI2. The first pair belongs to the same main class, and their shape is at least roughly similar, which explains the clustering by the new method. That means the remaining pair (1AF0, 1AI2) is the only real outlier, as their heightmaps are different and they are not in the same EC class.

Comparison with TM-score Figure 3.16 (b) shows a comparison with the results of *MM-align* (TM-scores) determined for the aforementioned ensemble of 50 proteins. While a generally good consensus between the new method and the TM-scores (yellow nodes) can be observed, the clusters in the blue and red



(a) Direct merges colored by enzyme classification



(b) Direct merges colored by TM-score

Figure 3.16 — Clustering of 50 different proteins using combined *MobileNetV2* features of heightmap and hydrophobicity map. Visually similar maps are clustered together. Merges of two leaf nodes are colored by the quality of the merge, which is based on two established external scores or classifications. Direct matches are colored using the Viridis color map (), where yellow corresponds to a good match and purple to a poor one. (a) applied as ground truth a given enzyme classification based on EC numbers and (b) applied the TM-score determined with *MM-align*. The cyan box (■) contains five enzymes belonging to the same class, where *MM-align*, as well as the new method, detect a high similarity. The red (■) and blue (■) boxes are cases where the new method gives a more suitable similarity score than *MM-align*. - © Elsevier 2021 [Sch+21b]

boxes show abnormalities. Here, the TM-score indicates a high dissimilarity between these proteins, although they are functionally very similar according to their EC classification. With the new method, these cases are clustered early but typically later than cases where all four enzyme classes match. For example, the enzymes of the blue box are both dehydrogenases, but they significantly differ in length (329 to 508 amino acids), which is an explanation for the deviation of the result of the new method to the result of *MM-align*. *MM-align* compares the

sequence and the spatial arrangement without a focus on the protein surface or further properties, which would make it more sensitive for function comparison. Additionally, the TM-score was not able to catch the similarity of 1A7T and 1A8T completely, although they have very similar structures, almost identical lengths, and fall into the same EC class (right half of the red box). Although the heightmaps do not look very similar, the new method was able to cluster them correctly. Deviations of the similarity detections between the new approach and the TM-score can also happen where the interior of two proteins is different, but the surfaces are similar. The new approach will determine a high similarity while the TM-score indicates a high difference. Since the surface of a protein is generally more important to its function, matching proteins with similar surfaces but dissimilar interiors is desirable.

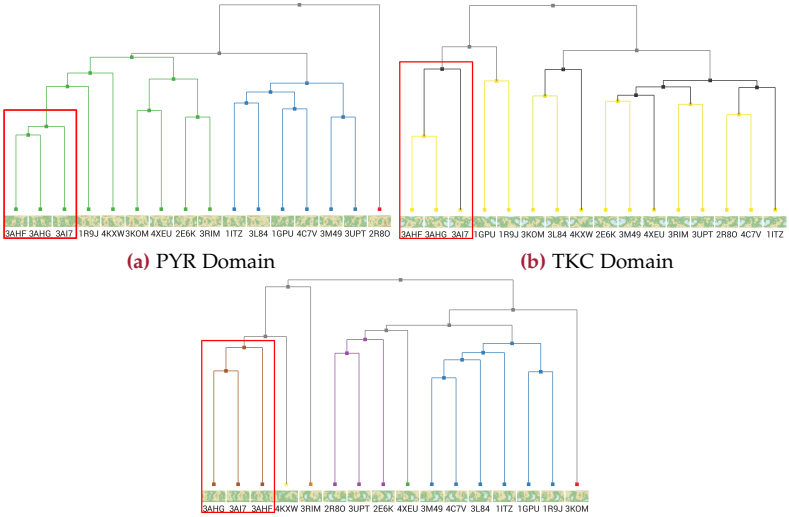
Case Study: Analysis of Protein Domains The new method was used to analyze an ensemble of carbon metabolism-involved enzymes consisting of thirteen Transketolases (TKs) and three Phosphoketolases (PKs) that were selected previously for a representative profile MSA of both protein families using *Clustal Ω* (see supplementary material of [Bai+18]). TKs are important enzymes in carbohydrate metabolism that catalyze the cleavage of carbon-carbon bonds in a ketose and the transfer of the remaining two-carbon unit onto an aldose [KS14]. PKs are enzymes similar to TKs that occur in a variation of the carbon metabolism that was found in certain fermentative bacteria such as *Bifidobacterium* sp. [Sán+10]. Both TKs and PKs require the cofactor molecule vitamin B1 (thiamine diphosphate or thiamine pyrophosphate) to catalyze the reaction. Both share a common arrangement of protein domains despite local differences in their structures [Dug06; VP14]. A TK or PK monomer includes an N-terminal Pyrophosphate (PP) binding domain, a central Pyrimidine (PYR) binding domain, and a Transketolase C-terminal (TKC) domain. The PP domain and the PYR domain interact with vitamin B1 and are thus required for the catalytic function, whereas the role of the TKC domain is not completely elucidated [CWD07].

The sixteen representative TKs and PKs used herein served as a test case for clustering Molecular Surface Maps of individual protein domains. It is generally assumed that the structure of a protein domain is folded independently and that protein domains serve distinct functions, such as interaction with other molecules or with other protein domains. In addition, protein domains are often seen as modules that can be combined or rearranged with other protein domains during evolution. When protein sequences or structures are aligned,

usually the complete monomers are compared to each other, and thus similarities between individual protein domains are often neglected or overlooked. The comparison of protein domains can be helpful in the elucidation of evolutionary relationships (cf. introduction Chapter 3) and the search for novel protein functions, especially in cases where overall similarity between complete monomers is rather low. Therefore, different clusterings were created that allowed the various domains to be compared separately, as well as the impact of different alignment methods. A RMSD-based alignment of the separate domains is used, which is possible since their sequences only differ slightly. This results in a tree for each of the three domains, showing that the domains of the PKs (PDB ID: 3AHG, 3AI7, 3AHF) are well conserved in all three cases. Additionally, the clustering, especially of Figure 3.17 (b), is in high agreement with the TM-score with only negligible differences.

The domain structures of the TKs (PDB IDs: 3AHF, 3AHG, 3AI7) form a cluster that is separated from the remaining thirteen TK representatives (cf. Figure 3.17). This is to be expected since protein structures from the PK subfamily differ from TK structures, for example, in the loop regions of the PYR domain [VP14]. The same cluster formation can be observed for each individual domain tree. Slight deviations between the cluster results of the trees are observed (see Figure 3.17). They can be explained by different degrees of conservation, whereby, for instance, protein sequences of PYR domains usually being more conserved than protein sequences of PP domains [VP14]. As an example, the PP domains (PDB IDs: 3AI7, 3AHG) are clustered in one subgroup, which means being more similar to each other than to also quite similar PP domain of PDB ID 3AHF. In contrast to this, the PYR domains of these two PDB entries are split into two subgroups, as the PYR domains (PDB IDs: 3AHF, 3AHG) are clustered in one subgroup instead. All three investigated PK structures originated from *Bifidobacterium* sp., while the selected TK representatives originated from more diverse taxa, ranging from bacteria to eukaryotes. However, the taxonomic hierarchy of the source organisms does not necessarily imply an equivalent hierarchical clustering of the separate protein domains, as the functional and visual differences might not reflect that directly.

Tile-based Approach Evaluation To evaluate the tile-based approach, presented in Section 3.3.4, the PP domains of the carbon metabolism data set were used. The results are compared with the standard approach of the new method and additionally with clusterings based on results of *MM-align* and results of the sequence alignment tool *Clustal Ω*, which is a widely used HMM-based

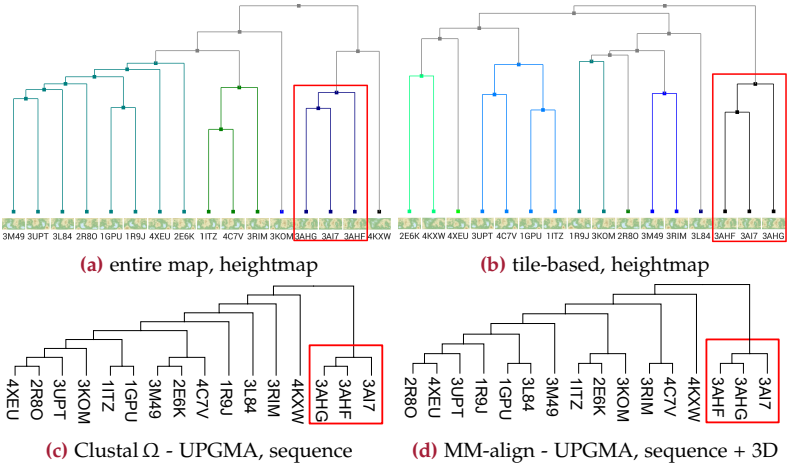


(a) PYR Domain **(b)** TKC Domain

(c) PP Domain

Figure 3.17 — Clustering hierarchy of the Pyrimidine (PYR) domain (a), the Transketolase C-terminal (TKC) domain (b) and the Pyrophosphate (PP) domain (c) of the carbon metabolism involved enzymes (3 phosphoketolases (PKs), 13 transketolases (TKs)). The domains are clustered based on the combined features of heightmap and hydrophobicity map. The three PKs are always marked with a red box. (a) The PKs are clustered together early, while the evolutionary closest (3AHF and 3AHG) are merged first. (b) The leaf node merges are colored according to the TM-score using the Viridis color map (see Figure 3.16). (c) The clustering of the PP domains of the PKs differs slightly from the one for PYR and TKC domains; however, all three PKs are still in the same cluster (brown). - modified figures © Elsevier 2021 [Sch+21b]

profile alignment tool in bioinformatics (see Section 3.1.1). It can be seen that the tile-based approach leads to a more distinct clustering of the PKs against the TKs as shown in Figure 3.18 (a) and (b). The tree of the solely sequence-based *Clustal Ω* delivers an even more distinct clustering of the three PKs as well as with *MM-align*, both similar to the tile-based approach. The resulting trees generated based on the results provided by *Clustal Ω* and *MM-align* are more similar to each other than to the trees created by the new method. However,



(a) entire map, heightmap **(b)** tile-based, heightmap

(c) Clustal Ω - UPGMA, sequence **(d)** MM-align - UPGMA, sequence + 3D

Figure 3.18 — Clustering results of the PP domain of different carbon metabolism involved enzymes. (a) Normal clustering of the new Molecular Surface Map based clustering approach only by using heightmaps. (b) The same clustering using the proposed tile-based clustering approach. (c) A sequence-based clustering using distances calculated by *Clustal Ω* [Lar+07]. (d) Sequence and 3D structure based clustering using distances calculated by *MM-align* [MZ09]. - modified figures (a), (b), (c) © Elsevier 2021 [Sch+21b], supplemented figure (d)

solely judging by protein function, the domain scientists argued that a strong clustering of the PKs is essential and should indicate a good result, as shown by the distinct clustering provided by the tile-based approach.

An adaptive tile-based approach could be used to improve the comparison accuracy of the new approach further. That is, the approach should consider the difference in the surface size of two compared proteins. Therefore, a high protein surface size should generate a higher resolution surface map, but the used tile size will still have fixed pixel dimensions.

3.4 Summary and Conclusion

Parts of this section have been published in:

- P. Hermosilla, M. Schäfer, M. Lang, G. Fackelmann, P.-P. Vázquez, B. Kozlikova, M. Krone, T. Ritschel, and T. Ropinski. "Intrinsic-Extrinsic Convolution and Pooling for Learning on 3D Protein Structures." International Conference on Learning Representations. 2021. https://openreview.net/forum?id=l0mSUROpwY
- K. Schatz, F. Frieß, M. Schäfer, P. C. F. Buchholz, J. Pleiss, T. Ertl, and M. Krone. "Analyzing the Similarity of Protein Domains by Clustering Molecular Surface Maps." *Computers & Graphics*. 2021, pp. 114–127. DOI: 10.1016/j.cag.2021.06.007

In this chapter, the field of protein-ligand interactions was first approached from the protein perspective by discussing two methods aiming to extract protein features used to classify proteins (cf. Figure 1.1). Proteins are an essential part of studying protein-ligand interactions because their shape and physico-chemical properties, especially of their molecular surface, enable the mediation of interactions in the first place. The two methods described in this chapter contributed to this by comparing, analyzing, and learning the proteins' structure and further physico-chemical properties to perform a characterization according to specific aspects. These aspects include protein structure (e.g., fold classes) and protein function (e.g., enzyme reaction classes), which are determined based on analyzing the protein sequence and its spatial structure. They also focus on the topology of the molecular surface and physico-chemical properties such as hydrophobicity. One approach utilizes an ANN to learn structural features at all structural levels of proteins for classification. The other method focuses on the protein surface and uses visualization combined with machine learning for feature extraction to enable structure analysis and image-based clustering.

The CNN based approach, as discussed in Section 3.2, uses 3D protein structures defined by atom coordinates to learn protein features for classification. A neural network architecture was proposed based on the multi-level structure of proteins that can process all structural levels. The method was developed since common algorithms in protein learning are not explicitly designed for protein data and, therefore, cannot capture all relevant structural levels. The presented architecture takes advantage of primary, secondary, tertiary, and

quaternary protein structures to learn a novel convolution operator. For this, it uses n-D convolutions defined by Euclidean distances and multiple intrinsic and extrinsic distances between the atoms in a multi-graph. Moreover, a set of pooling operations was presented that enable the dimensionality reduction of the input, mimicking the designs used by CNNs on images. The evaluation has shown that incorporating all the structural levels in the designs leads to a network architecture that can outperform existing state-of-the-art protein learning algorithms on two downstream tasks, comprising protein fold and enzyme classification. The new architecture enables enhanced precision, i.e., more precise and correct classification of proteins, thereby facilitating the elucidation of their function, which is advantageous for systems biology. It also helps to understand and derive structural features for specific types of protein-ligand interactions, since it also provides a higher accuracy in enzyme reaction class classification. This can provide valuable information for protein and ligand designers, e.g., to design and improve drug candidates.

The second method follows a different approach, solely focusing on the protein's surface, representing the interface that interacts with its outer environment, including solvents and ligands. To conclude, Section 3.3 presented a new image-based approach for identifying and visually comparing proteins with similar function that builds on hierarchical clustering of Molecular Surface Maps. These maps are planar representations of the complex molecular surface SES. They represent different protein properties, such as the topology or physicochemical properties. For creating Molecular Surface Maps, the method of Krone et al. [Kro+17] was adopted. The resulting maps serve to characterize proteins by extracting feature vectors from them. These feature vectors are computed utilizing the CNN-based *MobileNetV2*, which proved to achieve the best results in comparison to the tested *Image* and *Color Moments*. Multiple maps with different properties of the same protein are combined and used to construct a dendrogram (phylogenetic tree) representing the clustering hierarchy (evolutionary relationship).

Experts can explore and analyze the results of the hierarchical clustering interactively with a prototypical visualization application created using the visualization prototyping framework *MegaMol* (see Section 2.6.1). Therefore, the visualization consists of a dendrogram linked and complemented by two 3D views, showing the protein and highlighting the actual surface view area on the corresponding surface map. The discussed method was developed in close collaboration with domain experts to ensure that it meets their requirements. The evaluation showed that the new approach clusters proteins together that

also exhibit a functional similarity, and the results are in high agreement with enzyme reaction classification based on Enzyme Commission (EC) numbers and with the established TM-score determined by *MM-align*. It produces only a few outliers compared to the EC classification that is used as a ground truth. Thus, the new method can be applied for clustering functionally similar proteins, and its results can be rated as highly suitable overall. This makes the tool interesting for biomedical applications, such as drug discovery and protein engineering, where it can be utilized to identify proteins with similar functions or proteins with specific properties.

As mentioned at the summary's beginning, this chapter approached protein-ligand interactions from the protein perspective. In the following chapter, this perspective moves successively to a more ligand-centered perspective, therefore leaving the field of protein analysis and moving on to computational chemistry.

CHAPTER 4

Computational Chemistry

This chapter continues the idea of investigating protein-ligand interactions from two perspectives as introduced in Chapter 1. While the previous Chapter 3 focused on the protein-centered view, this second main chapter is concerned with the ligand-centered view, which is realized by using and developing methods of computational chemistry. Computational chemistry, also known as molecular modeling, includes all theoretical as well as computational methods used to model or predict the behavior of molecules [Lew11; Lea01]. In this discipline, computer programs solve chemical problems by implementing and automating methods of theoretical chemistry, for example, to perform MD simulations and molecular docking. Thereby, theoretical chemistry provides mathematical descriptions of chemical problems and processes. Computational chemistry combines various fields, comprising the aforementioned chemistry as well as physics, mathematics, and computer science. The methods used are correspondingly diverse and extend from molecular mechanics, over force fields, to quantum mechanics, semi-empirical methods, and *ab initio* calculations (Latin: *from the start*). The latter two are based on the physical theory of quantum mechanics and aim, among others, to solve the Schrödinger equation. The Schrödinger equation mainly describes the behavior of electrons in a molecule, whereas the semi-empirical methods do more approximations in solving it. The mentioned methods cover a vast field, which is even further extended by density functional calculations (electron density function). From this broad field, this work uses two suitable methods for investigating protein-ligand interactions

that are described in Section 4.1. A more detailed description of computational chemistry and its methods can be found in the books of Lewars [Lew11] and Leach [Lea01]. Furthermore, a general overview of different approaches for visualizing molecular dynamics trajectories was recently given by Belghit et al. [Bel+24].

It is important to investigate the ligand's dynamic behavior to understand protein-ligand interactions, including the analysis of ligand movement and determining the interaction types they form while interacting with a protein. This chapter starts by considering the dynamics of molecules and their structure. In this context, the first section is about a fast molecular surface algorithm contributing to the sub-topic of MD simulations (cf. Figure 1.1). It enables interactive visualization and allows evaluating molecular trajectories from MD simulations. Therefore, it uses a fast, massively parallel CUDA algorithm to compute the analytically correct SES. In turn, calculating the molecular surface supports inferring ligand moving concepts and helps to show and understand ligand paths along a target protein. The second method also contributes to the topic of MD simulations by discussing an approach for visualizing ligand behavior. In contrast to the direct visualization of the simulation results of the method mentioned above, the second method follows a MD simulation aggregation approach. It maps the condensed simulation information onto the protein surface and combines it with a Compressed Ligand-Interaction Sequence Diagram (CLISD). Continuing the aggregation approach, a method of a co-supervised student work fits in this context as well, focusing on conformational changes of a molecule such as a protein during co-factor binding. It uses dimensionality reduction to project each time step into 2D space and enables a comparative analysis of time steps. This is followed by a work assigned to molecular docking, which focuses on protein-ligand interaction types and spatial binding frequencies. It is a visual post-docking analysis application that enables evaluation and decision-making by analyzing and enriching, e.g., virtual screening results such as those used in drug discovery.

4.1 Molecular Dynamics and Docking

This work focuses on two computational chemistry techniques, which are Molecular Dynamics (MD) simulation and molecular docking, both suitable for examining protein-ligand interactions. MD simulation utilizes molecular mechanics to model small biological systems and large molecules as proteins. It focuses on the motion of atoms over time. However, molecular docking aims to

predict the orientation, conformation, and binding affinity of a small molecule (ligand) when bound to a target molecule (receptor), which is often a protein. In contrast to MD simulations, which create trajectories of molecular motion on an atomistic level, molecular docking attempts to solve an optimization problem that consists of looking for a geometrically and energetically optimal fit of a ligand in a binding site. Molecular docking uses different methods dependent on the various docking algorithms, which include molecular mechanics as well.

This section will give a brief overview of the concepts of molecular docking and MD simulation. A detailed overview of various docking algorithms, their differences, and the tools using them, as well as details about virtual screening, are given by the review of Yuriev et al. [YHR15]. A further detailed view of molecular docking and virtual screening can be found in the extensive work of Azevedo et al. [Aze19]. More details about quantum mechanics and molecular mechanic simulations are given in the review of Hollingsworth and Dror [HD18] and the books by Leach [Lea01] and Frenkel and Smit [DB02].

4.1.1 Molecular Docking

Parts of this section have been published in:

- M. Schäfer, N. Brich, J. Byška, S. M. Marques, D. Bednář, P. Thiel, B. Kozlíková, and M. Krone. "InVADo: Interactive Visual Analysis of Molecular Docking Data." *IEEE Trans. Visual. Comput. Graphics*. 2024, pp. 1984–1997. DOI: 10.1109/TVCG.2023.3337642

Molecular docking has become an important technique in structural biology and computer-aided drug discovery. As mentioned, molecular docking aims to predict the orientation, conformation, and binding affinity (free energy) of a ligand in the binding site of the target protein. During the search process, many different ligand conformations are perturbed in the protein binding site or, in the case of blind docking, on the whole protein surface. These conformations of the ligand are called binding *poses* or *modes* [Tor+19]. When a ligand is bound to its target, it can perform a specific biological function, such as activating or inhibiting an enzymatic reaction (cf. Section 2.3.2). Docking has been used for over four decades and still plays an essential role in high-throughput virtual screening, especially in drug discovery [WZ16; SM18]. Ligand-target binding characteristics are investigated to find geometrically and chemically optimal fits, i.e., with a high binding affinity. For this, individual poses are identified using stochastic search algorithms that are evaluated by scoring functions, which

enable ranking, for instance, of possible hits or leads [Ban+19]. The term *hit* is used in the context of biomedical research. It describes a ligand with high binding affinity, which is also a so-called active compound. In this context, a *lead* is a ligand that is a drug candidate. A lead is a hit or is derived from a hit that has undergone a physico-chemical optimization [Syk24].

The general docking procedure consists of the target and ligand selection, their preparation, and the docking calculations, followed by the evaluation of the results [ML08]. The preparation step comprises, for example, the addition of hydrogen atoms, the removal of water from the structure data, and the calculation of charges. A tool enabling these operations is, for instance, *AutoDockTools* [Mor+09] as part of *MGLTools*. There are many commonly used tools to perform the actual docking, such as *AutoDock Vina*, *Gold*, *DOCK 6*, *MOE Dock*, *Glide* and *rDock* [Ebe+21; Jon+97; All+15; Che24; Hal+04; Rui+14] (see Zhang et al. [Zha+22]). The static coordinates of the protein and ligand atoms are used as input data. The molecular docking output is typically a multi-model or multi-poses file storing different orientations and conformations of a ligand. A widely used file format is the PDBQT format, which is one of several formats discussed in the context of molecule preparation for docking in Section 2.6.2.

Virtual Screening As already indicated, molecular docking is an essential component of virtual screening as it is used in biomedical research. Virtual screening is an early stage of drug discovery used for *hit* finding and *lead* identification [Syk24]. It improves efficiency by reducing the number of wet lab experiments and aims to find effective therapeutics with minimal side effects. The general idea of virtual screening is to search for desired ligand physico-chemical properties, comprising shape, electrostatics, and pharmacophores, where a pharmacophore can be a composition of multiple functional groups in a specific spatial position (see Section 4.4.1).

Virtual screening includes different approaches, such as ligand-based and structure-based [Syk24]. The ligand-based one uses a ligand with known properties, e.g., a known inhibitor of an enzyme. This ligand serves as a template to search a database for ligands with similar properties (e.g., shape, pharmacophores). The structure-based one uses the known structure of the target molecule and the ligand. It performs computationally more expensive molecular docking and allows finding hits or leads without a prior known template.

Following the structure-based approach, the general virtual screening procedure consists of having a target/receptor molecule and dozens to hundreds of

thousands of different ligands for which a search for a spatially and chemically optimal fit is performed. A fit is a ligand pose with a good docking score (high affinity), indicating that the ligand potentially binds well to the target. The docking result is a list of *ligand binding poses* that have the lowest free binding energy/lowest score, i.e., a list of the energetically most favorable ligand-target complexes. Molecular docking uses various search algorithms and scoring functions to predict and rank binding poses. Although docking still has limitations regarding its accuracy, it is commonly used to screen large databases of ligands to narrow down possible drug candidates [WZ16]. Due to the speed of modern docking algorithms and computing hardware, this is much faster and more cost-effective than wet lab experiments for the screening process. Once the ligand candidates have been identified, a MD simulation can follow to evaluate the findings and determine a more accurate binding affinity (cf. Section 4.1.2). Additionally, the verification of such drug candidates can be carried out in a wet lab experiment.

An example of the docking tool mentioned above, which is also used for virtual screening, is *AutoDock* [Mor+09]. Over the last 30 years, *AutoDock* has been one of the most widely used docking tools and is still continuously improved [TO10; Goo+21; Ebe+21]. It uses a Lamarckian genetic algorithm and Monte Carlo simulations [Ban+19] to solve the docking task. Since molecular docking algorithms are a complex and extensive topic that is beyond the scope of the thesis, please refer to the original publications and the literature mentioned above for more information (cf. Section 4.1 [YHR15; Aze19]).

4.1.2 Molecular Dynamic Simulations

Molecular Dynamics simulation has many applications, including drug discovery, molecular flexibility, protein folding, and structure refinement [HD18]. Over the past years, the usage of MD simulations increased significantly due to broader accessibility. This was enabled by enhanced algorithms and strongly improved hardware that nowadays allows to run MD simulations on a single consumer GPU that previously had to run on costly supercomputers [Sto+16; HD18]. The basic idea of MD simulation consists of calculating the force exerted on each atom of a small biological system [HD18]. This system is, for example, a protein solvated in water containing a part of a cell membrane (lipid bilayer). The mentioned force is computed in dependence on all other atoms, enabling the prediction of the spatial position of each atom as a function of time. The force is then used to update the atom positions by using Newton's laws of

motion. This results in a trajectory similar to a 3D movie describing the atom motion by the atom position for every time step of the simulated time interval.

As earlier mentioned, the behavior of the atoms depends on quantum effects, and an exact calculation, so-called *ab initio* calculations, would take too much time. To reach acceptable computation speeds, they are approximated by using force fields mimicking the quantum effects [HD18]. These force fields, for instance, simulate electrostatic interaction between the atoms. One restriction is that atoms in MD simulations cannot form or break a covalent bond. Still, their widespread use shows their useful strengths in gaining biological insights into molecular systems. One significant advantage is the possibility of tracking the position and motion of every atom, which would be difficult with other experimental techniques. Additionally, MD simulations have known and precisely controllable experimental conditions, including protein conformation and protonation state, bound ligands, environmental molecules, mutations, post-translational modifications (e.g., glycosylation), and more. This allows for identifying the system behavior for various molecular perturbations. Widely used tools for MD simulations are *GROMACS*, *AMBER*, *CHARM*, and *OpenMM* [Abr+15; Cas+05; Bro+09; Eas+17]. Their usage allows for gaining insights about interactions, thermodynamics, and kinetics. Additionally, they are also used for system behavior studies, structure validation, and structure refinement [Gen+17]. Further use cases are lead optimization as part of drug discovery and to simulate ligand binding paths [HD18]. Furthermore, MD simulations are utilized for testing binding poses and for predicting a more accurate ligand binding affinity, both of which are obtained beforehand from a molecular docking (see Section 4.1.1).

To ensure numerical stability when performing MD simulations, the time steps must be very short, typically in the scale of femtoseconds (10^{-15}s) [HD18]. Biological events such as structural changes in proteins take nanoseconds, microseconds, and sometimes longer, which leads to billions of time steps. Each time step contains millions of interatomic interactions, which means that the simulations are very computationally intensive. However, the improvements in the past years enabled simulation time intervals up to milliseconds. As mentioned, the result of MD simulations are molecular trajectories. This dynamic data is often stored using the XTC format, which describes the changes over time of the static molecule topology, usually stored as PDB file (cf. Section 2.6.2). For more details on MD simulations, please refer to the previously mentioned literature (cf. Section 4.1 [HD18; Lea01; DB02]).

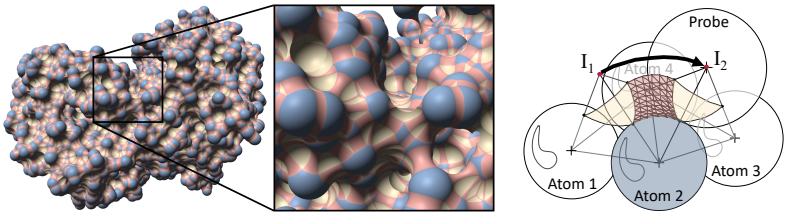
4.2 GPU-Accelerated Molecular Surface Computation

Parts of this section have been published in:

- M. Schäfer and M. Krone. "A Massively Parallel CUDA Algorithm to Compute and Visualize the Solvent Excluded Surface for Dynamic Molecular Data." EG Workshop on Molecular Graphics and Visual Analysis of Molecular Data, 2019. DOI: 10.2312/molva.20191094

In the previous chapter, methods were discussed mainly focusing on feature extraction and the comparison of multiple proteins. Now we are going further to a more detailed analysis of a single protein, also considering its interactions with ligands. Such detailed analyses are MD simulations of proteins, their solvent, and ligands. Visualizing MD simulations is a crucial part of the analysis. Therefore, an interactive representation of the protein surface is required.

Since MD simulations try to compute the behavior of a system over time, they provide molecular trajectories that describe the movement of each atom (see Section 4.1.2). By this positional time-dependent description, a trajectory covers conformational changes of, e.g., a protein during cofactor binding. A fast computation of the molecular surface is needed to represent those changes and to provide an interactive visualization for evaluation and further analysis of MD trajectories. There exist various molecule and surface representations as discussed in Section 2.5, but for the mentioned purposes, an analytically correct surface representation is needed, which is the Solvent Excluded Surface (SES) described in Section 2.5.2. The SES is the most commonly used surface representation because it is suitable for protein-solvent interaction or docking. This complex surface type needs to be recomputed for every time step of the MD analyses. The analysis of MD simulations is often done on normal consumer PCs, which requires a fast and efficient implementation of the SES calculation. Therefore, a massively parallel CUDA algorithm is presented, computing the analytically correct SES. Calculating the molecular surface is instrumental for identifying cavities and tunnels, since they are important for understanding ligand docking and transport processes. High-quality visualization of smooth molecular surfaces combined with a coloring that illustrates physico-chemical properties that influence interaction can greatly assist the understanding and knowledge discovery. It enables a visual inspection of the ligand paths along the protein surface and the formed interactions, allowing for an evaluation and



(a) SES of Hemoglobin **(b)** SES Patches

Figure 4.1 — (a) shows a screenshot of the SES visualization method presenting the haptoglobin-hemoglobin complex (PDB ID: 4XOL) colored by SES patch type: (●) spherical triangle patch, (●) toroidal patch, (●) spherical patch. The cutout to the right shows that the method handles all singularities correctly and creates a pixel-perfect image. The probe radius was set to 1.4 Å approximating water. - © EG 2019 [SK19] - (b) presents the schematic genesis of the three SES patches following the rolling probe definition (cf. Section 2.5.2). The patch coloring corresponds to subfigure (a). The points I_1, I_2 show the center points of the probe sphere where it is in a fixed position. - modified figure © John Wiley & Sons 1996 [SOS96]

detailed investigation of the MD simulation results. The mentioned cavities or pockets are potential binding sites of ligands and are interesting spots for determining and analyzing interaction types. To achieve an interactive visualization, the GPU-accelerated algorithm computes the SES by searching for intersection points of atom sphere triplets. These triplets are vdW spheres of the protein atoms. The intersection points are used to derive the different SES surface parts, as the SES is built from three different surface patches (see Figure 4.1). These patches will be rendered utilizing quick, modern, and pixel-precise GPU-based glyph ray casting [Gum03; KBE09] (cf. Section 2.5.1).

The computation of the SES is quite complex and time-consuming. Dynamic molecular data, such as the MD simulation trajectories mentioned, typically contain several thousand to tens of thousands of frames. Thus, precomputing the SES for all frames, such as some molecular viewers do, is not a feasible approach for large, dynamic data. A method that computes the surface out-of-core is crucial to enable the visualization of the SES for dynamic data at interactive frame rates. Hence, many approaches for the interactive computation and rendering of the SES have been proposed over the last 15 years.

An efficient method to render the surface patches of the SES is the GPU-based ray casting presented by Krone et al. [KBE09]. The implicit description of the different patches is sent to the GPU, where shader programs are used to compute the ray-patch intersections (see Section 2.5.1). This method is well-suited, as it combines high image quality and low rendering times. Lindow at al. [Lin+10] presented a CPU-parallelized version of the *Contour-Buildup* algorithm by Totrov and Abagyan [TA95] to compute the SES interactively. Subsequently, Krone et al. [KGE11] adapted the *Contour-Buildup* to run efficiently on the GPU. Their implementation is, to date, one of the fastest ways to compute the SES analytically. Jurcik et al. [Jur+16] adapted their approach to enable semi-transparent SES rendering, allowing, for instance, simultaneously rendering inner voids of the protein.

Another class of algorithms to compute the SES discretizes the space into voxel grids and samples the atoms to this grid. The SES can then be derived using an Euclidean distance transform [XZ09]. Hermosilla et al. [Her+17b] presented an efficient GPU implementation of this approach. By computing the grid progressively, they achieve interactive computation times even for huge molecular complexes. Egan and Gibou [EG18] presented a method that uses octrees [Mea80] as an efficient data structure to refine the grid. This was implemented to run on massively parallel compute clusters. While the quality of the resulting SES is very good, the computation takes several seconds for larger data sets and is, thus, not interactive. In general, grid-based methods can usually be parallelized well and have low computation times. However, the grid resolution limits the quality of the SES. Therefore, the new method focuses on the analytical computation of the SES.

4.2.1 Algorithm Overview

This section gives an overview of the new, highly parallel, and analytic approach for the efficient calculation and rendering of the SES. In general, the algorithm computes all intersection points of three-way combinations of SAS spheres that represent the atoms of a protein. The radius of a SAS sphere is the vdW radius of the corresponding atom plus the probe radius, e.g., the radius 1.4 Å of a probe sphere approximating a water molecule. The intersections represent the center of a probe sphere in a fixed position, that is, a probe in contact with three vdW spheres, which defines a spherical triangle (cf. I_2 Figure 4.1 (b)). Only a fixed probe sphere that does not intersect with any other atom is in a valid position, and all other fixed probe spheres are discarded. To calculate those valid intersection points, the new algorithm is specifically designed to exploit

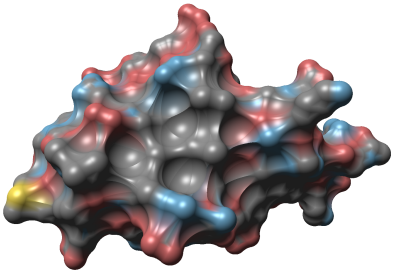


Figure 4.2 — Solvent Excluded Surface of an isomerase (PDB ID: 1OGZ) colored by element. The probe radius was set to 2.4 Å. - © EG 2019 [SK19]

the massive parallel computational potential of modern GPUs. The central part is the parallel computation of the intersection points of all combinations of three SAS spheres to derive the SES. From these intersections, the three graphical primitives that constitute the SES can be derived, including spherical patches, spherical triangle patches, and toroidal patches (see Figure 4.1). After this, the SES can be rendered. An exemplary result showing the SES of a protein colored according to the individual atom types or elements is shown in Figure 4.2.

To order to utilize the tremendous potential of modern GPUs, the algorithm has to be divided into multiple tasks with a moderate computational load. A GPU can perform a massive number of tasks in parallel in comparison to a CPU due to the much higher number of cores, but the resources per core are much more limited. The crucial SIMD architecture of GPUs, which is described in more detail in Section 2.2, is used to run an instruction in the form of a CUDA kernel on multiple data via different threads [Fly72]. Therefore, all steps of the new algorithm are designed to run in a highly parallel environment.

Figure 4.3 gives an overview of the new algorithm. The algorithmic pipeline's first step is to find each atom's neighboring atoms. To do this efficiently, a spatial acceleration structure needs to be constructed. A grid-based data structure is a good choice since it maps well to the GPU and can be computed fast, which is essential when dealing with dynamic data. Using this *neighbor search grid*, the *nearest neighbor search* is executed for each atom a_i. N_i is the set of all neighbor atoms a_j that are within a radius of $r_i + r_j + 2 \cdot r_p$, where r_i is the vdW radius of atom a_i, r_j is the vdW radius of atom a_j, and r_p is the probe radius. That is, N_i contains all neighboring atoms a_j that are no more than one probe diameter away from atom a_i, which means that the SAS spheres of two atoms are at least touching each other.

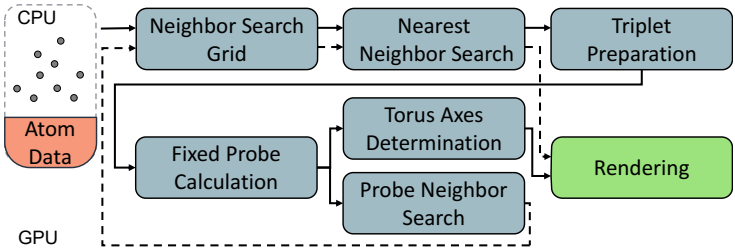
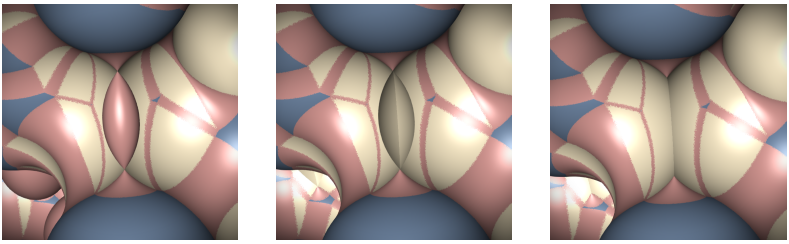


Figure 4.3 — Flow chart of the algorithmic pipeline. At the beginning, the data (orange oval) is uploaded to the GPU. All subsequent processing steps are executed entirely on the GPU by CUDA kernels (blue boxes). The final rendering step uses OpenGL (green box). - modified figure © EG 2019 [SK19]

The neighbor information is required for the next step, consisting of finding all combinations of three SAS spheres that might intersect each other. For the *triplet preparation* for each atom a_i, two neighbors $a_j \in N_i$ and $a_k \in N_i$ with $i \neq j$ are chosen from the list of neighbors N_i. If the SAS spheres of these atoms are also at least touching ($\|a_j - a_k\| \leq r_j + r_k + 2 \cdot r_p$), a potential triplet of intersecting atoms has been found. This is followed by the *fixed probe calculation*. Trilateration is used to calculate the two intersection points. Therefore, the sphere center coordinates have to be transformed for S_i to $(0,0,0)$, for S_j to $(d,0,0)$ and for S_k to $(e,f,0)$ as a necessary simplification that allows formulating the following three equations for the spheres [Mah15]:

$$r_1^2 = x^2 + y^2 + z^2 \tag{4.1}$$
$$r_2^2 = (x-d) + y^2 + z^2 \tag{4.2}$$
$$r_3^2 = (x-e) + (y-f) + z^2 \tag{4.3}$$

After solving the equations, the coordinates have to be transformed back to get the correct coordinates [Mah15]. As mentioned before, both intersection points are potentially the centers of probes in a fixed position (cf. Figure 4.5). An alternative approach to compute the two potential fixed probe positions was given by Connolly [Con83]. To check if a probe position p is valid, it has to be tested for intersection with all other neighboring atoms $a_l \in N_i : l \neq j, l \neq k$. This intersection test is a simple distance test: $\|p - a_l\| > r_p + r_l$. Only valid fixed probe positions are stored for further processing, and all others are discarded.



(a) Spindle Torus **(b)** Intersecting Patches **(c)** Singularity Free

Figure 4.4 — SES singularities as also shown and discussed in [KDE10]. Each subfigure shows the same SES part colored regarding the three types of surface patches (cf. Figure 4.1). The singularity presented in (a) is a spindle torus. (b) shows intersecting spherical triangle patches and (c) presents the singularity free SES.

As mentioned above, the valid fixed probes found in the previous step define the spherical triangles. Each spherical triangle is mathematically defined by a fixed probe and the three corresponding atoms a_i, a_j, and a_k. To render the spherical triangles correctly, it is necessary to take care of the so-called *singularities*, which occur due to probe-probe intersections. These intersections can lead to spherical triangle patches that intersect each other (see Figure 4.4 (b)). The singularities are the parts of the spherical triangles that are within the other probe and have to be removed. For this *probe neighbor search*, a second nearest neighbor search is used to find all intersecting fixed probes for each fixed probe (cf. dashed lines Figure 4.3). Therefore, the nearest neighbor search radius of $2 \cdot r_p$ is used.

In addition, the information about the spherical triangles can be used to derive the toroidal patches. Each pair of atoms $(\left(a_i, a_j\right), (a_i, a_k), \left(a_j, a_k\right))$ defines a torus axis. The analytic equations to compute the parameters for the tori were also given by Connolly [Con83]. A second type of singularity can appear in dense protein regions, like in tunnels and pockets, called spindle torus (see Figure 4.4 (a)). A spindle torus appears when the surface contributing atoms lie opposite each other, and the distance of their not touching vdW spheres is $< 2 \cdot r_p$. This type of singularity can be avoided by simply checking if the inner radius (tube thickness) of the torus is greater than the outer radius ($r > R$) [KDE10]. Finally, the convex spherical patches of the SES are simply the vdW spheres of the atoms. Thus, the SES can be constructed by rendering the spherical triangles, the toroidal patches, and the vdW spheres of the atoms.

All steps of the algorithmic pipeline are designed to run in parallel on the GPU. In the next section, a possible implementation of this algorithm is described.

4.2.2 Implementation Details

The prototypical implementation of the algorithm described in Section 4.2.1 runs entirely on the GPU. For the computational part, CUDA is used, and the rendering is performed using OpenGL with GLSL shaders. The implementation is designed for maximum speed while avoiding data duplication and unnecessary copying of data between the host (CPU) and the device (GPU). Especially memory transfers between host and device can still be a bottleneck. Therefore, the only data transferred to the GPU are the atomic positions, atom radii, and per-atom colors. All further operations use these and derived data, which are only processed by the GPU.

Neighbor Search Grid After uploading the atomic properties (position, radius, color) to the GPU, the first step is to sort them into a three-dimensional grid for fast nearest neighbor retrieval. This neighbor search grid spans the entire bounding box of the molecule, that is, the maximum and minimum values of the Cartesian coordinates in (x, y, z) direction have to be determined. Therefore, a uniform grid is used that is built up from cubic grid cells that have a side length

$$g_{dim} = 2 \cdot r_p + 4 \tag{4.4}$$

where r_p is the probe radius. The goal is to construct the grid in such a way that a neighboring atom a_j satisfying the abovementioned neighbor criterion

$$\|a_i - a_j\| \leq r_i + r_j + 2 \cdot r_p \tag{4.5}$$

is either located in the same grid cell as atom a_i or in one of the neighboring cells. That is, to search for the nearest neighbors, it is only necessary to test the distance to atoms located in the grid cell of a_j and the surrounding grid cells, which is a $3 \times 3 \times 3$ sub-cube of the grid. Therefore, the constant of 4 is a conservative estimate of the sum of atom radii $r_i + r_j$, which is chosen because the algorithm focuses on processing protein data sets, i.e., the vdW radius of all atoms will be below 2 Å.

The construction of the neighbor search grid follows the *counting sort* CUDA implementation proposed by Hoetzlein [Hoe14]. A CUDA kernel is executed in parallel for all atoms to assign each atom the corresponding grid cell index. Each grid cell is addressed via an individual hash value. In this step, the number

of atoms per cell is determined by using an atomic operation for addition. This is necessary to guarantee the avoidance of race conditions, meaning that no other thread can increase the counter for the atoms before the current thread finishes its incrementation[1].

The result from the previous step is a list of all the atoms and their assigned grid cell together with the total count of atoms per cell. This is followed by sorting the atoms according to the assigned grid cells. Therefore, a parallelized prefix sum over the atom counts per cell is computed using the `inclusive_scan` function provided by the Thrust library[2], which is part of the CUDA Toolkit. The following *counting sort* step uses this sum to place the atoms in the correct regions of the sorted array. The result of the prefix sum provides an array index range for the atoms located in a specific grid cell. An atom a_i is assigned to cell g_j, which has the prefix sum s_j. That is, the atom a_i will be stored at position s_j, and s_j will be decreased by one. Note that only the index of an atom is stored in the sorted array to save memory. To later access the atoms located within a specific grid cell, the start, and end of each grid cell need to be stored in a separate array. More details about the construction of the counting sort grid are given in the original description by Hoetzlein [Hoe14].

Nearest Neighbor Search Based on the acceleration data structure built before, a nearest neighbor search is executed in parallel for all atoms, resulting in a list of all neighboring atoms written to an array. Since the number of neighbors can be different for each atom, this step, in theory, requires a dynamic data structure. Since this would not map well to the CUDA API, space is reserved for 150 possible neighbors for each atom. This empirical number has proven to be sufficient in the tests conducted. For each atom a_i, the neighbors a_j are searched for in the grid cell that contains a_i and the directly neighboring cells ($3 \times 3 \times 3$ grid cells). The neighbor criterion used is defined as before in Equation (4.5). That is, each potential neighbor a_j found within one of the visited grid cells satisfying this criterion is added to the list of actual neighbors. Two separate lists of neighbors are written, where the first one is the complete list of all neighbors, and the second only contains neighbors where the atom index j is greater than the index i of the actual atom. That is an optimization to avoid redundant computations in later steps. The neighbors that satisfy the condition $j > i$ are referred to as *reduced neighbors* throughout the rest of the paper.

[1] *CUDA C++ Programming Guide: 7.14 Atomic Functions*. https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#atomic-functions (accessed 02/07/2024).

[2] *Thrust - Parallel Algorithms Library*. https://thrust.github.io/ (accessed 02/07/2024).

SAS Sphere Triplet Preparation The total number of reduced neighbors allows to compute the number of theoretically possible combinations of three intersecting SAS spheres. In general, the number of possible k-tupels from a set of n items can be calculated using the combination formula $\binom{n}{k}$. In the present case, this means to compute the number of triplets for a_i, i.e., determining all possible combinations of two neighbors ($k = 2$). The number of theoretically possible SAS sphere triplets for a_i can then be calculated as follows:

$$\binom{n}{2} = \frac{n \cdot (n-1)}{2} \tag{4.6}$$

where n is the number of neighbors of a_i. This number of triplets is computed in parallel for each atom and is stored in an array. Subsequently, a prefix sum over this array is computed using `thrust::inclusive_scan`. This prefix sum is again required to get the total amount of memory that is needed to be allocated for storing the actual triplets, as well as to get the array indices where these triplets are stored. The triplets are determined using the reduced neighbor lists and two nested for-loops for iterating over them. The first position of the combination is the current atom a_i, and the remaining two positions are filled with its neighbors. Note that redundant triplets are avoided by using the reduced neighbors. For example, if atom a_3 has the neighbors a_1 and a_6, the triplet (a_3, a_1, a_6) is not generated, since $1 < 6$, that is, neighbor a_1 would not be available in the reduced neighbor list. This triplet is already generated for a_1 as triplet (a_1, a_3, a_6), which is sufficient since the order of the atoms does not matter for the following calculation.

Computation of Fixed Probe Positions This step computes the actual intersection tests for the previously stored triplets of atom indices. A triplet of SAS spheres is defined as S_i, S_j, S_k corresponding to the atoms a_i, a_j, a_k. They are at the same position as the atoms a, but the radius of these spheres is the vdW radius of the atom plus the probe radius r_p. An obvious requirement for an intersection of three SAS spheres is that they all overlap with each other (cf. Figure 4.5). Since it is known that S_j and S_k are neighbors of S_i, it is only necessary to test whether S_j and S_k overlap. Triplets that do not fulfill that criterion are excluded from further computations. Next, it is checked whether the two intersection points I_1 and I_2 (see Figure 4.5) for the current triplet $\left(S_i, S_j, S_k\right)$ actually exist flowed by computing their position as described in Section 4.2.1. For each intersection point, it is mandatory to check whether it lies within another neighboring SAS sphere S_l. Therefore, the list of all neighbors

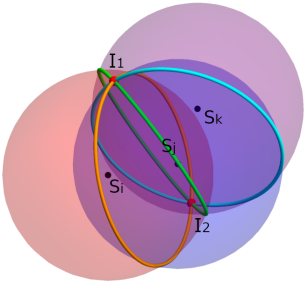


Figure 4.5 — Transparent spheres representing the SAS spheres of three atoms with their center points (S_i, S_j, S_k). The three colored circles show the course of the cut surfaces of the spheres. They meet each other in two points (I_1, I_2), which are the intersection points. - image is based on [Mah15] © EG 2019 [SK19]

has to be used instead of the reduced neighbor list (see *Nearest Neighbor Search*). Only intersections outside all neighboring SAS spheres are valid fixed probe positions that are saved in a list for further processing.

In theory, it is necessary to reserve sufficient memory to store both intersections for all triplets, which would require a large amount of memory. However, most triplets do not intersect at all, or the intersection points are not valid probe positions since neighboring SAS spheres cut them away. Therefore, the required size for the fixed probe position array is based on an empirically determined value. The final number of valid probe positions in the tests is less than one percent of the theoretical number of triplets (see *SAS Sphere Triplet Preparation*). Consequently, only sufficient memory is allocated to store the fixed probe positions for one percent of the triplets. Since the intersections of all triplets are computed in parallel, it is mandatory to ensure that no intersection is lost due to race conditions between concurrent CUDA kernels. That means, an atomic counter is used again to store the fixed probe positions in a global array (see *Neighbor Search Grid*). If a valid fixed probe position is found, the atomic counter is incremented, and the position is written to the previously stored index.

Probe Neighbor Search for Singularity Handling To avoid singularities of the spherical triangles in the SES rendering, it is important to find fixed probes that intersect with each other. The same neighbor search procedure is used here as for the atoms. The fixed probe positions are sorted into a uniform neighbor search grid (see *Neighbor Search Grid*), and then the neighbor search is performed (see *Nearest Neighbor Search*). That is, for each probe p_i, all intersecting neighboring probes are stored. This probe neighbor list is then used during the rendering to avoid the singularities of intersecting spherical triangles.

Torus Axes The torus axes are determined using the fixed probe positions. The atom index triplet (i, j, k) is stored for each fixed probe. All three pairs of atoms define an axis of a torus (i.e., $\left(a_i, a_j\right)$, (a_i, a_k), $\left(a_j, a_k\right)$). Thus, all these torus axes are written to an array using a further CUDA kernel. This leads to duplicates since each combination will occur twice. To get rid of these duplicates, the torus axes are first sorted using `thrust::sort` and then the duplicates are removed by using `thrust::unique`, which removes consecutive duplicates in an array. Since all Thrust functions run parallelized on the GPU, these steps are executed very fast.

GPU Memory Management and Rendering The new algorithm is implemented as a new plugin of the open-source visualization framework *MegaMol* (see Section 2.6.1). *MegaMol* already provides IO routines for molecular data (e.g., PDB file loading) and SES rendering using the fast GPU-based ray casting presented by Krone et al. [KBE09; Kro+12]. Instead of triangulating the patches of the SES, special GLSL shaders compute the actual ray-object intersections during rendering, resulting in a pixel-perfect image of the patches. For the three types of SES patches (spherical, spherical triangles, toroidal), three different GLSL shaders are used. For details, please refer to the original publication [KBE09]. This section explains how these shaders were modified for the implementation.

Traditionally, a Vertex Array (VA) or a Vertex Buffer Object (VBO) is used to pass data to the GLSL shader, but the new implementation uses a Shader Storage Buffer Object (SSBO). A SSBO has the convenient property that a GLSL shader has random access to its stored values. After allocating a SSBO via OpenGL, it can be registered by CUDA to get a resource (`cudaGraphicsGLRegisterBuffer`). This resource can then be mapped to CUDA (`cudaGraphicsMapResources`). From the mapped resource, a device pointer can be obtained (`cudaGraphics ResourceGetMappedPointer`) that allows a CUDA kernel to have read and write access to the SSBO memory. Before rendering, the resource must be unmapped again to be able to bind the SSBO for the GLSL shader. Via the CUDA kernels, all the information that is required for rendering is directly written to a SSBO, which allows to completely avoid costly host-device memory copies or data duplication on the GPU. A further optimization is that the CUDA arrays and SSBOs are only resized if the current computation needs more memory, otherwise, the already allocated memory is re-used. Using dynamic data such as MD trajectories, where the molecule can deform over time, leads to varying memory requirements. All arrays and buffer objects are allocated with 10%

excess to reduce the number of necessary reallocations in these cases. Since the amount of memory usually does not increase much, this is generally sufficient to avoid a costly reallocation in every time step of a simulation.

The previous SES implementation available in *MegaMol* uses VAs and VBOs to transfer data to the GPU memory. The existing shaders using this way were modified to utilize SSBOs instead. Two SSBOs are created for the spherical patches. One stores the atomic coordinates and vdW radii, and the second contains the atom colors. These data are used to render all atoms as colored spheres, which are the spherical patches of the SES.

Previously, all parameters required for rendering, including the spherical triangles and toroidal patches, are precomputed and passed to the GPU as VA or VBO. For the new SES rendering, these parameters are calculated in the vertex and fragment shaders, which reduces the required amount of memory. As mentioned in Section 4.2.1, the necessary equations for this rendering type are given by Connolly [Con83]. The data for the spherical triangle patches consists of five SSBOs that store the atom positions + radii, atom colors, fixed probe positions, the corresponding triplets, and the list of probe neighbors for singularity handling. It has to be noted that the triplet SSBO only contains the atom indices used to access the actual positions stored in the first SSBO. The rendering of the toroidal patches requires three SSBOs, which are the atom positions + radii, atom colors, and the torus axes. The torus axes SSBO also only stores atom indices. All shaders reuse the two SSBOs containing the atom positions + radii and the colors, thus further reducing the amount of memory, which has to be transferred to the GPU for rendering.

4.2.3 Results and Discussion

The performance of the new implementation was measured on a test system running Windows 10. It was equipped with an Intel i5-8600k CPU ($6 \times \sim 3.6$ GHz), 16 GB Random-Access Memory (RAM), and an NVIDIA GeForce GTX 1080 (8 GB Video RAM (VRAM), 2560 CUDA cores). Table 4.1 shows the timings for the test data sets of various sizes obtained from the PDB [Ber+00]. The resolution was set to FullHD (1920×1080) to measure the rendering performance. Please note that the data were treated as if they were dynamic, i.e., the complete SES computation was executed in each frame. Only the memory allocation was not done for every iteration. The new method is able to compute the SES interactively for molecular complexes of more than 20 k atoms, where the test system was able to maintain a frame rate of more than 12 fps. Even for the

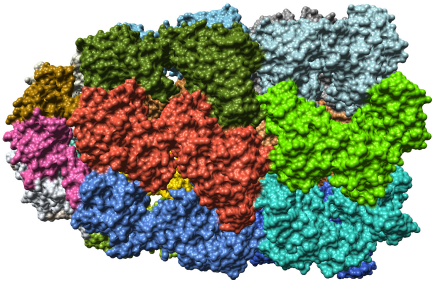


Figure 4.6 — SES of a chaperonin complex (PDB ID: 1AON) determined with a probe radius of 1.4 Å. - © EG 2019 [SK19]

Table 4.1 — Performance measurements for different data sets. *Atoms:* number of atoms; *SES Share:* percentage of atoms that are part of the SES; overall *Run Time* of the SES calculation. Additionally, the required GPU memory (VRAM) and the Frames Per Second (fps) are listed. Note that the fps include the time for recomputing the SES in each frame. - © EG 2019 [SK19]

Data Set	Atoms [10^3]	SES Share [%]	Run Time [ms]	VRAM [GB]	FPS
1OGZ	0.9	69	6	0.12	118.7
1VIS	2.5	60	11	0.13	78.6
4X01	4.2	64	14	0.15	60.1
1AF6	10.0	61	26	0.18	33.1
1MMO	21.3	49	82	0.28	12.0
1AON	58.7	68	147	0.46	6.7
5TZS	98.5	92	125	0.63	7.8
6HIV	99.4	71	200	0.69	4.9
4v4J	147.1	69	365	1.01	2.8
ribo01	147.2	65	366	1.03	2.7
CCMV	214.4	62	479	1.36	2.1

largest test data set (CCMV), a virus capsid of more than 200 k atoms, the new implementation takes less than 500 ms for the SES computation, reaching more than 2 fps. Figure 4.6 shows one of the test data sets, which is the chaperonin complex 1AON with ∼58 k atoms.

Table 4.2 and Figure 4.7 show a more detailed breakdown of the timings for the individual steps of the algorithmic pipeline. It can be observed that the parallel computation of fixed probe positions is the most time-consuming step, which is also the core of the algorithm. This is because, in this step, all the possible

Table 4.2 — Timings for the individual steps of the new algorithmic pipeline (all measurements in milliseconds). *Grid*: inserting the atoms into the grid for neighbor search, *Neighbor*: nearest neighbor search, *Triplet*: preparing the list of potentially intersecting three SAS spheres, *Probe*: find valid fixed probe positions, *Tori*: derive torus information from fixed probe positions, *PN*: search all neighboring probes for each fixed probe position. - © EG 2019 [SK19]

Data Set	Grid	Neighbor	Triplet	Probe	Tori	PN
1OGZ	2.04	0.89	0.70	1.38	0.32	0.94
1VIS	3.58	1.00	0.92	3.98	0.39	1.05
4X01	4.00	0.96	0.80	6.90	0.35	1.30
1AF6	4.68	0.96	1.81	17.06	0.36	1.53
1MMO	5.91	1.68	12.30	59.70	0.43	1.88
1AON	19.29	3.86	28.44	88.63	1.17	5.72
5TZS	21.14	3.73	25.72	58.21	2.04	14.39
6HIV	13.28	5.72	43.40	126.64	1.70	9.61
4V4J	26.74	9.08	80.50	231.53	2.25	14.97
ribo01	16.43	9.48	79.69	244.34	2.32	13.36
CCMV	31.71	13.23	103.68	309.49	2.94	18.15

triplets of SAS spheres have to be checked for each atom. Thus, the other parts play a subordinate role computationally. On average, preparing the triplets and computing the fixed probes takes more than 80% of the whole computation time. It can be seen that the total runtime of our algorithm scales linearly with the number of atoms (coefficient of determination $= 0.994$). An exception is the data set 5TZS, which has a very special atomic configuration containing protein and also RNA (see Figure 4.8) and, therefore, exhibits an irregular runtime. This outlier case is discussed in *Conformational Dependency*.

The VRAM consumption was measured as well (see Table 4.1). In general, the memory requirements of the new method are reasonably low and depend linearly on the number of atoms (coefficient of determination $= 0.998$), except for data set 5TZS. This outlier is again due to the fact that the memory consumption also depends on the conformation of the atoms within the molecule (see *Conformational Dependency* for details).

Conformational Dependency As mentioned above, the runtime and memory consumption increases linearly with the number of atoms. This applies for all test data sets (see Table 4.1), except for the data set 5TZS. Here, the computation

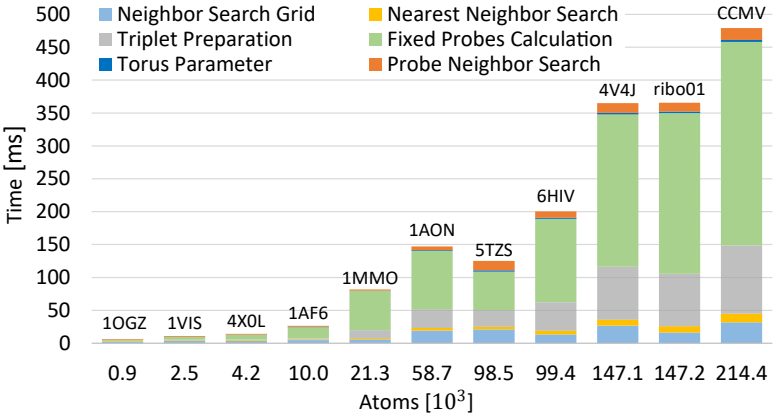


Figure 4.7 — The atoms per molecule [10^3] are plotted against the SES calculation time [ms]. Each stacked bar is labeled with the PDB ID except *ribo01*, which is a ribosome consisting of 2WDL, 2WDK and except *CCMV* that is a virus capsid consisting of multiple 1CWP units. - © EG 2019 [SK19]

time is reduced by about 17% compared to 1AON, although 5TZS has ∼68% more atoms. The computation time of the only ∼1% larger data set 6HIV is even about 60% higher than the one of 5TZS. A closer look at Table 4.2 reveals that the main difference is due to the considerably shorter execution time of the step that prepares the triplets and finds the valid fixed probe positions. This can be explained by the conformation, i.e., the atomic configuration of 5TZS. This data set has more atoms as well as a higher relative number of atoms contributing to the SES. In detail, 92% of the atoms in 5TZS contribute to the SES (see *SES Share* in Table 4.1) compared to only 68% in 1AON.

Additionally, 5TZS also has a larger relative spatial extent with respect to the total number of atoms. The larger spatial extent and the higher *SES Share* are due to the fact that this data set is much sparser than a typical protein complex. This means it has a lot of empty space in between and contains long strands of RNA that stick out (see Figure 4.8). These two aspects result in a configuration where the atoms have fewer neighbors.

The lower number of neighbors per atom for 5TZS lowers the run time for preparing the triplets and finding the fixed probes compared to the similar-sized 6HIV (see Figure 4.9). The number of triplets depends quadratically on

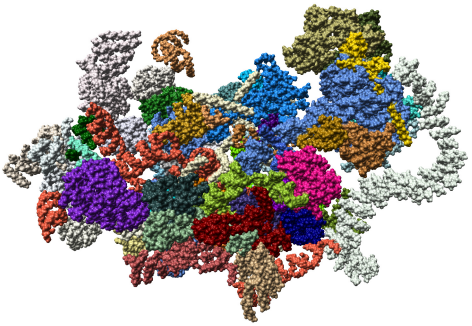


Figure 4.8 — SES of the yeast small subunit processome [Cha+17a] (PDB ID: 5TZS, r_p = 1.4 Å). Each chain is colored differently. - © EG 2019 [SK19]

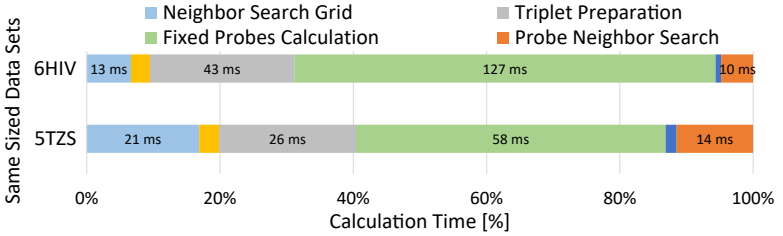


Figure 4.9 — The relative time consumption [%] of the different parts of the new SES algorithm compared to the total time is shown for the two similar-sized data sets 6HIV and 5TZS. The bar length of the individual algorithm parts represents the relative time consumption. Additionally, the labels present the absolute time [ms] for crucial parts. See Figure 4.7 for a complete description of the color usage. - modified figure © EG 2019 [SK19]

the number of neighbors (see Equation (4.6)). Consequently, a lower number of neighbors leads to significantly fewer triplets per atom, which in turn means that much fewer triplets have to be checked while searching for valid fixed probe positions. In addition, far fewer neighboring atoms need to be checked for intersection for each potential probe position, which is needed to ensure that a probe is in a valid position. The slightly longer calculation time for the *fixed probe neighbor search* can be explained by the higher total number of valid fixed probes. Since the atoms in 5TZS are distributed more sparsely, more triplets will generate valid fixed probe positions.

Discussion of the Optimization Process In this paragraph, the optimization process of the implementation is discussed with respect to memory consumption as well as computation speed. Different approaches were tested until the final solution, described in Section 4.2.2, was reached. The aim was to find an approach that allocates the minimum amount of required VRAM and enables a fast execution time.

1. Exact Method: In this approach, the calculation routine for finding fixed probe positions needs to be executed twice. In the first run, the triplets for each atom are examined regarding the fixed probes and the number of valid fixed probes. This number is stored for each atom. Then, a prefix sum is calculated to get the total number of fixed probes. This determines the amount of memory that needs to be allocated in order to store all valid fixed probes, which is done in the second run of the computation. Using this approach, only the necessary memory size is reserved, but as the fixed probe calculation is the most time-consuming part of the new algorithm, running it two times is computationally too expensive.

2. Sort And Unique Method: The second method is to allocate an array twice the size of the number of all possible SAS sphere triplets. This is sufficient to store both possible intersections for each triplet, that is, potentially all possible fixed probe positions. This is followed by the routine for the fixed probe calculation, which is executed only once, directly writing only the valid fixed probe positions to the array. Thereafter, this solution array will be sorted to get all empty entries of the array to a coherent block prior to running `thrust::unique`. It leads to a compact array and gives the number of fixed probes. This method is faster than the first approach but requires a huge amount of memory.

3. Heuristic AtomicAdd Method: The third method is described in *Computation of Fixed Probe Positions*, which uses the `atomicAdd` operation to write valid fixed probe positions. Using atomic operations is the fastest way to solve the problem when comparing all three approaches. This approach, therefore, not only has the best performance but also has a very similar memory consumption to the first one due to the heuristic estimation of required memory.

Comparison with Previous Work In this paragraph, the new method is compared to the GPU-parallelized *Contour-Buildup* algorithm by Krone et al. [KGE11], which is available in the open source visualization framework *MegaMol* [Gro+15], discussed in Section 2.6.1. The computation speed of the *Contour-Buildup* algorithm is only slightly faster than the new method when considering small data sets. But, for larger data sets, the Contour Buildup is con-

Table 4.3 — Comparison of the overall performance and memory requirements (VRAM) of the new implementation and the CUDA *Contour-Buildup* (*CB*) algorithm presented by Krone et al. [KGE11].

Data Set	Atoms	Performance		VRAM	
	[10^3]	new	CB	new	CB
1VIS	2.5	78.6 fps	86 fps	0.13 GB	0.4 GB
1AF6	10.0	33.1 fps	51 fps	0.18 GB	1 GB
1AON	58.7	6.7 fps	20 fps	0.46 GB	5 GB

siderably faster and exhibits better scaling (cf. Table 4.3). However, the Contour Buildup implementation requires a disproportional amount of GPU memory with increasing numbers of atoms (see Table 4.3, *VRAM*). The chaperonin 1AON is the largest of the used test data sets, which was able to be visualized using the *Contour-Buildup* algorithm, since the GPU ran out of memory for the larger ones. For this data set, the new implementation still requires only ∼460 MB of VRAM, while the *Contour-Buildup* requires more than 10× the memory (5 GB). That is, the memory requirements of the CUDA *Contour-Buildup* implementation currently available in *MegaMol* is comparable to the second approach described in *Discussion of the Optimization Process*, which makes it inapplicable to large data sets, although the computation speed would still be sufficient. In contrast, the new implementation still requires only 1.36 GB of VRAM even for the largest test data set, the virus capsid, which has more than 214 k atoms.

In contrast to previous approaches, the new algorithm has overall linear behavior in computation time and memory consumption and can render large molecular data sets interactively using a single consumer GPU.

4.3 Visual Analysis of Large-Scale Protein-Ligand Interactions

Parts of this section have been published in:

- K. Schatz, J. J. Franco-Moreno, M. Schäfer, A. S. Rose, V. Ferrario, J. Pleiss, P.-P. Vázquez, T. Ertl, and M. Krone. "Visual Analysis of Large-Scale Protein-Ligand Interaction Data." *Comput. Graph. Forum.* 2021, pp. 394–408. DOI: 10.1111/cgf.14386

The fast SES computation and visualization from the previous Section 4.2 is an elementary 3D protein structure visualization. It allows analyzing interactively the accessibility of the protein by a specific ligand to investigate the formed interactions, even for dynamic data such as retrieved from MD simulations. This enables a detailed analysis of MD trajectories or docking results, but the simulation size remains a crucial problem. The question is where and when an event of interest happens, i.e., which time points and spatial regions are of interest for a detailed analysis. As explained, MD trajectories can consist of millions of time steps with hours of playtime. It is very hard for researchers to see important changes or to get a general impression of how, e.g., a ligand moves along the surfaces and reaches the active site.

While existing approaches for the visualization of protein-ligand interactions, like the one by Vázquez et al. [Váz+18], typically concentrate on single ligand molecules, global approaches to analyze ligand movement are rare. Due to the size, streaming, or aggregation approaches are the most viable options. Streaming approaches typically use animation and are often preferred by domain scientists as they are easy to understand. However, it is well known that analyzing animations, especially long ones, is more difficult for humans than analyzing static images [TMB02]. An aggregated representation of the simulation can be more beneficial for adequate visualization. On the other hand, systems using aggregated data only keep track of a limited number of represented parameters. Moreover, they do not calculate information such as contact counts, which are highly relevant for the domain experts who collaborated in the requirement analysis and the evaluation of the new approach. The domain experts also co-authored the publication. They want to understand how and where ligand molecules approach the active site of an enzyme. Therefore, they generate extremely large simulations, up to terabytes of data that cannot be tackled directly by almost any existing software. For example, packages

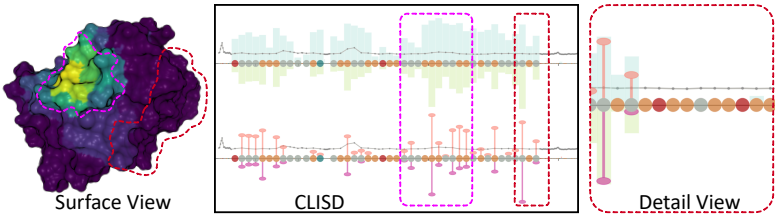


Figure 4.10 — The three main views of new protein-ligand interaction visualization. The *surface view* on the left side can be colored by different aggregated quantities (here contacted atoms) using the Viridis color map (; 0 to maximum). The middle part shows the CLISD representing the aggregated quantities (for a detailed description see Figure 4.13). Amino acids represented as circles on the x-axis are de-emphasized if only a few contacts to a ligand were detected. All values can be inspected in a *detail view*, alongside additional bond counts. The corresponding protein area of the detail view is marked by red dashed lines in the three views. The protein area with high numbers of contacts is marked by purple dashed lines. - modified figure © EG & Wiley 2021 [Sch+21a]

such as *MegaMol* (cf. Section 2.6.1) would be able to reproduce the animation but do not provide tools for the analysis. Other packages with a scriptable interface support any dataset indirectly, but they require upfront work by the user and visual programming expertise, which cannot be expected from domain scientists. To effectively inspect those simulations, domain experts are interested in the interactions/contacts between the enzyme and the ligand, for example, to find spots for protein engineering to increase the catalytic rate of an enzyme.

To achieve an effective inspection of the simulation and to address the mentioned issue of representing a long simulation as an animation, an approach is to show only the *important* information, which is an obvious but effective approach. In contrast to the direct visualization of the simulation results, this can be done by the mentioned aggregation, consisting of summarizing and deriving certain features of the MD simulation and visualizing them in an easy-to-understand and comprehensive way. Therefore, the new approach includes a preprocessing pipeline and a web-based visualization with multiple linked 2D and 3D views facilitating a visual and explorative analysis of the aggregated data (see Figure 4.10). It uses a mapping of the condensed simulation information onto the protein surface and combines it with a Compressed Ligand-Interaction Sequence Diagram (CLISD). The goal of this approach is not only to ease the

analysis of the simulation results, but also to provide a tool that supports the work of protein engineers by making the understanding of simulations clearer and more accessible. Apart from the physico-chemical surface properties, the access of a ligand to the active site is to a significant amount a geometrical problem, where the new approach supports the visual inspection of the protein geometry.

The improvements in simulation algorithms and computational power continuously challenge previously developed visualization techniques. On the one hand, the models are increasing in size and complexity. This means, the number of elements needed to be visually represented, such as atoms or cells, surpassed the order of millions quite some time ago [Gro+15; Le +15]. On the other hand, the non-geometric information researchers need to explore and analyze is continuously increasing too, for instance, the further growing number of time steps in molecular simulations [Dur+19], or a large number of properties to represent such as water trajectories [Vad+17], or physico-chemical properties and interactions within protein cavities [Fur+17; Byš+16; Byš+15]. This together leads to a data complexity problem. In general, visualization research often focuses on the usage of GPUs for accelerating the rendering of complex 3D structures using different strategies, such as LOD [Ibr+18] or highly parallel architectures [Riz+15; Kno+14] (cf. Section 4.2). However, when analyzing visualization approaches that address the problem of data complexity, increasingly intelligent data abstractions and aggregation techniques can be found that aim to explain more in a smaller space. Thus, it is focused on two different areas, which are the aggregation of the massive input data to extract meaningful information and the design of a multi-view visualization system that enables the exploration of many variables at once.

Data Aggregation Approaches Visual analysis is at the core of all visualization techniques [HS12]. When the data becomes very large, e.g., when visualizing whole ensembles, several strategies can be employed to reduce the required space. Animation, for instance, reuses the same space by modifying the visual depictions with the time. However, for large sequences, animation can be less effective than static graphs [TMB02], as it relies on the short-term memory of viewer [Wan+19]. The opposite approach is to aggregate the information so that it can be digested and facilitates a detailed exploration of elements of interest. In molecular visualization, *abstraction* and LOD techniques have been used extensively (e.g. [Mia+18]) since they enable reducing clutter when the size or density of elements requires a sizable screen footprint. Other methods

have been developed to represent full molecules [PRV13; Par+14], as well as additional information such as solvent pathlines near protein cavities [Bid+08]. *Data aggregation* has also been used for non-geometric properties, such as energy plots. For instance, Duran et al. [Dur+19] presented simplified energy charts using a hierarchical exploration tool that supports the investigation of detailed parts with few clicks. Data aggregation can be very useful if it adequately provides an informative overview of the whole data set. Vázquez et al. [Váz+18] aggregate energy data from a entire Monte Carlo-based molecular simulation to facilitate the quick exploration of interaction areas. However, their approach deals with relatively short trajectories containing several hundreds of steps and does not include other interesting information such as contacts. Bidmon et al. [Bid+08] cluster the paths of solvent molecules near the active site of a protein to simplify the exploration of complex solvent movement patterns. In contrast, Skånberg et al. [Skå+18] facilitates the exploration of large MD data with an extensive use of semantic *filtering* operations. Byška et al. [Byš+19] detect interesting spatio-temporal events in very large simulations to allow the users to concentrate on these potentially interesting parts and filter out uninteresting portions of the simulations. Alharbi et al. [ALC16] facilitate the exploration of MD simulations by path filtering based on geometric properties of the paths, such as edge lengths or curvatures. In a subsequent work [Alh+19], they presented a tool that extracts and visualizes protein-protein as well as protein-lipid interactions from MD simulations. However, their method is tailored to membrane simulations and cannot be applied directly to protein-ligand interactions.

Visualization of Multiple Variables Most visualization systems use multi-view and overlaying techniques to increase the number of variables that are shown at once. In molecular visualization, for example, Vázquez *et al.* [Váz+18] create a compact visualization for the depiction of protein-ligand binding simulations. They overlay multiple variables simultaneously, including molecular backbone, per-residue energy, minimum and maximum energies, H-bonds, and more. They combine both single-step and aggregated information. Hermosilla et al. [Her+17a] display multiple energy-related information and make extensive use of filtering operations to display the interactions between the ligands and the proteins in a simulation on a step-by-step basis. However, this method does not give an overview of the whole simulation. Lichtenberg et al. [Lic+18] propose the *Residue Surface Proximity* map, which allows users to visually analyze the closeness of each amino acid of a protein to the surface throughout a simulation. This corresponds to the potential exposure to ligands,

which is in contrast to the new approach presented here, which focuses on the actual behavior of the ligand molecules. Furmanová et al. [Fur+19] facilitate the exploration of protein-protein contacts through a combination of charts and a 3D view. However, they concentrate on the analysis of contacts and provide views for illustrating the individual residues' interaction in a node-link view. In the context of the new approach, it is of more interest to show the entire path in an explorative way and to provide additional information, such as the number of steps in contact. Mostly, a contact is defined as spatial proximity without the forming of covalent bonds.

The strategy of the new approach consists of coping large amounts of data by using aggregation. That enables the generation of informative overviews of the entire simulation, and additionally, the resulting visualization can be run on commodity hardware. Exploratory visualization is achieved through data superposition, filtering, and compaction to maximize the amount of information that can be displayed at the same time.

4.3.1 Tasks and Requirements

The goal is to create a visual analysis application for protein-ligand interactions that is able to visualize even the largest biomolecular data sets, e.g., ensembles of MD simulation data. In order to ensure that the application is effectively usable by domain experts and fits smoothly into their analysis workflow, the application requirements were formulated in close cooperation with actual target users. After numerous refining iterations, the following seven requirements were identified.

The first requirement R1 is to provide a comprehensive overview resulting from the vast amount of data that the domain experts are working with. They typically use animations, which was identified to be nearly impossible for the current amount of data. Even with filtering approaches, tens of thousands of time steps remain relevant. Limiting animations to solely those remaining parts leads to animations that move too fast for a human to analyze the presented data in full detail without numerous animation repetitions, especially when the subsequent requirements have to be considered as well.

An unavoidable drawback of aggregated visualizations is that a part of the original information is lost. The following four requirements mostly specify what domain scientists are interested in, restricting which information is allowed to be omitted during the aggregation process. A point of interest is enabling the identification of protein areas with many detected contacts with

ligands. Therefore, R2 enforces the inclusion of the location, duration, and frequency of contacts between protein residues and ligand molecules. As the sequence, as well as the folding of its underlying amino acid chain, influences the behavior of a protein, the requirement also enforces the usage of certain visual representations (e.g., SES).

The most interesting part for domain scientists is investigating the actual ligand paths on the surface of the protein. Tools the experts usually use either do not work or produce too much visual clutter. To investigate the hypothesis that ligands crawl along the surface, the landing spots are of keen interest, followed by the actual path taken by the ligands (R5 and R3). However, as different kinds of ligands form different kinds of bonds or interaction types (cf. Section 2.3.2), it is important to consider different kinds of bonds or interaction types depending on the research question (R4). Besides relevant distance-based contact definitions, specialized definitions should also be allowed.

Two further requirements stem from the workflow of the domain scientists. Although they can obtain some of the analysis data with existing tools such as *GROMACS* [Abr+15] or the MDAnalysis library [Mic+11; Gow+16] by analyzing distances and interactions from the simulation trajectory, it is currently not possible to extract all of the information presented by the new approach. As far as is known, all the available tools either require loading the full trajectory into memory, which is only possible on dedicated hardware with much RAM, or they require extensive coding efforts. Dedicated solutions reducing the memory requirement are currently not available in the existing analysis libraries. Furthermore, dedicated tools for aggregating and visualizing the analyzed data are not available, which makes the systematic analysis of large simulation data complex and time-consuming. In addition, a graphical representation of aggregated data is missing in the available tools that are typically tailored towards computing clusters. For visualization, the domain scientists normally use *PyMOL* [SD20] or *VMD* [HDS96]. While both tools provide many options for surface visualizations, features for sequence diagrams are limited or not present at all. Additionally, both programs have neither a dedicated way to input aggregated data nor are they able to depict longer trajectories without loading them completely into RAM.

As it is mostly impossible for domain scientists to occupy a special machine for visualization tasks, a further requirement was formulated (R6). It describes the need to have an application able to perform the preprocessing as well as the visualization on a consumer machine. The implementation of this leads to more needed precalculations, i.e., longer initial waiting time. This is not an

issue, as domain experts are used to longer calculation periods of their MD simulations. As long as these calculations are not necessary each time the visualization application runs, and it allows presenting the visualization results within a few seconds to collaborators, they are acceptable (R7). To sum up, the seven requirements are the following:

R1 The visualization should give an overview of the whole simulation, instead of showing the whole simulation as an animation.

R2 Areas where contacts are most common should be identifiable in a surface representation as well as in the amino acid sequence.

R3 Ligand paths on the surface of the protein should be identifiable.

R4 Depending on the use case, different kinds of contact definitions need to be considered (e.g., distance-based, or hydrogen bonds).

R5 Identification of areas/amino acids where ligands get initially into contact with the protein (*"landing spots"*).

R6 The preprocessing as well as the visualization should be possible on a commodity machine.

R7 It should be possible to present visualization results to collaborators without having to wait more than a few seconds.

Requirement R1 to R5 are describing what the final visualizations should enable the user to analyze and requirement R6 and R7 restrict the technical implementation. Please note that R3 and R5 do not apply to protein docking data, as they are tailored to the use case of simulation data.

4.3.2 Approach and Application Overview

In the context of the discussed requirements, especially R1, an aggregation approach is used instead of filtering only. The representation of the temporally aggregated values in a static visualization eases the analysis since no animation is required. A further advantage is that it can tremendously reduce the amount of data required for the visualization compared to the original simulation data size, as detailed below. It also makes it possible to completely decouple the aggregation as a preprocessing step from visualization.

The visualization is implemented as a web-based application, as this is the most convenient way for domain scientists. This also addresses requirement R6 since

it enables them to access the visualization via a browser from all devices without any installation needed. For the final application, the focus is primarily on incorporating visualization methods that are familiar to users from the field of biology and chemistry, further reducing the barriers and facilitating an intuitive and fast analysis process. This includes a commonly used three-dimensional visualization of the protein structure as well as abstract, two-dimensional representations, such as sequence diagrams. Both types of representations are enhanced by incorporating simulation values that are relevant for protein-ligand interactions, which are encoded, e.g., by mapping them via color onto the representation.

As protein-ligand interactions highly depend on surface geometry, presenting only an enhanced sequence diagram is not sufficient. To show the spatial relations between the protein and the ligands and to depict the conformation of the protein, the SES is used (R2). However, due to conformational changes during the simulation, some amino acids are not part of the surface at certain time steps. In order to incorporate them, a surface visualization alone is not sufficient. Furthermore, exact numeric values can be assessed better in a 2D plot than in a 3D visualization. Therefore, both representations are used, including the SES of the protein from a representative time step alongside an enhanced sequence diagram. The web-based visualization tool thus combines the advantages of both depictions and links them with suitable interaction techniques. A detailed description of the visualization capabilities, especially of the enhanced sequence diagram, is given in Section 4.3.4.

An overview of the proposed pipeline, including the data processing as well as the visualization, is shown in Figure 4.11. The pipeline starts with input data acquired from MD simulations or docking experiments. In the following, MD data is referred to as *simulation data* and docking data as *non-simulation data*. The data preprocessing application loads the data stepwise, i.e., the next simulation time step is only loaded if the current time step is processed. This results in a constant memory footprint that does not increase with input simulation length. As demanded by requirement R6, this allows commodity machines to perform the aggregation and especially to store and visualize the results. The main idea of the proposed preprocessing is that the values are temporally aggregated per amino acid, which makes the output independent of the number of input time steps. After the accumulation, three kinds of data are available. One includes the *aggregated data* per amino acid as aggregated raw data and newly derived data. The second is an *aggregated SES mesh* containing the colors for each value type, and the third is the *protein data*, which contains basic information about

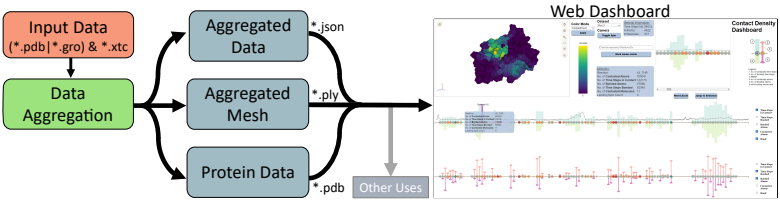


Figure 4.11 — Overview of the data pipeline. First, the *input data*, given as PDB or *GROMACS* files alongside an XTC trajectory, is analyzed and accumulated. The *data aggregation* results in three types of files: Human-readable JSON files that contain the measured results, a representative surface mesh containing color values for each of the accumulated quantities, and a PDB file providing atom positions corresponding to the surface mesh. The data is then visualized, as shown in the screenshot of the entire web-based visualization on the right side. - modified figure © EG & Wiley 2021 [Sch+21a]

the protein structure, including atom positions and types (elements). These three data parts can either be directly loaded by the web-based visualization application, enabling an exploratory analysis of the aggregated values, or they can be processed by other programs for further *other uses*. For example, the Polygon File Format (*.ply) of the meshes can be read by standard mesh viewers (e.g., MeshLab). A detailed description of the implementation details, including the preprocessing and the web-based visualization application, as well as the utilized file formats, is given in Section 4.3.5.

4.3.3 Data Processing

The first step of the pipeline consists of processing and aggregating the input data. The resulting data is small enough to be stored and transferred to the web-based visualization as described in Section 4.3.4. It comprises three sub-steps, with data preparation being the first, which places all time steps into the same reference frame. The second is to derive required quantities from each time step, and the third is to aggregate all read and derived data to produce a visualizable result. The main focus of the application lies on single or multiple MD simulation runs, but also supporting molecular docking results. Since simulation trajectories can be very large, most of the properties for aggregation are chosen in a way that allows continuous accumulation of values. That is, the statistics of a simulation always require the same amount of memory, regardless of the number of time steps.

Data Preparation During simulation, the receptor protein typically exhibits translational and rotational motion. Since this, before the aggregation, it is necessary to align the protein conformation of each time step or to use a *snapshot*-based approach, where the protein is aligned to the position and orientation of a reference snapshot. Therefore, a RMSD-minimization-based alignment is used (cf. Section 3.1.2) provided by the simulation framework *GROMACS* [Abr+15]. The collaborating domain scientist also uses this framework to conduct simulations. The algorithm tries to minimize the positional deviation of each atom to a reference position, where the reference here is the average position of the atom over time. In the case of docking data, each found conformation or docking pose of the ligand is taken as a time step.

Data Derivation After data preparation, all time steps are loaded incrementally. For each of them, several values are derived, which are discussed in the following.

Distances: The most important interactions typically happen at protein atoms in contact with ligands for a longer period, as they might block or hinder the ligand's path to the active site. Thus, the distances between protein and ligand atoms are calculated and used to derive the number of close ligand atoms for each protein atom. Two atoms are defined as close to each other if the distance between the vdW spheres (vdW radii based) is below a certain threshold value r_c. As default value 3.5 Å is used, as recommended by the domain scientist. This value was chosen since it approximately marks the distance at which attraction between atoms induced by vdW forces becomes relevant. Since the definition of closeness can depend on the use case, the user can adjust the threshold value. For example, if the user wants to detect only the stronger ionic or covalent bonds, a value of 1.0 Å or even lower would be feasible. In contrast, a higher value than the default would also include atoms that may be close but do not exert attractive forces.

Interactions: Atoms of the ligand and the protein that are close to each other may form hydrogen bonds. These bonds could be either responsible for holding back a ligand or for *pulling* it forward. Hydrogen bonds can be estimated based on the element, the distance, and the bonding angle between possible bonding atoms [Jef97]. However, nonpolar ligands do not tend to form hydrogen bonds. Thus, also carbon-carbon interactions are incorporated. They include, for instance, π-stacking, vdW, and hydrophobic interactions (see Section 2.3.2). This also satisfies requirement R4. These interactions are not as strong as hydrogen bonds but can still influence the movement of the ligand relative to the protein.

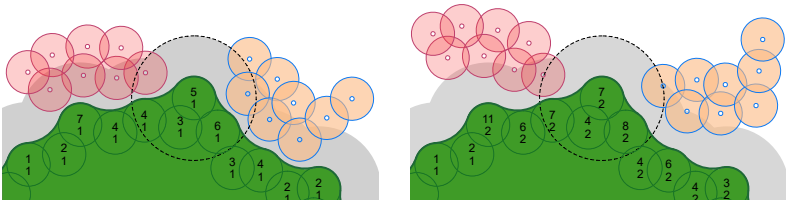
If the distance between two carbon atoms is below the threshold $r_b \approx 3.2$ Å, a carbon-carbon interaction is assumed [Alk+09].

Landing Spots: A current hypothesis by domain scientists is that a ligand could either move directly from the surrounding medium to the active site, or it could *land* on the protein surface and then crawl towards the active site. So-called *long-range-effects* of the protein surface have already been observed [LA95]. The source of those effects is unclear. May it be a protein side chain that blocks the movement of the ligand or a hydrogen bond holding it back. To provide more detailed insights into the behavior of the ligand and to satisfy requirement R5, the number of ligand landing spots on the protein surface is extracted. For this purpose, a counter is incremented for each ligand molecule in each snapshot or time step where the ligand is in contact with the protein surface. A time step without a contact resets the counter to zero. If one of the counters reaches the landing threshold l_t, a landing spot is registered for the currently viewed time step. The landing threshold is necessary because of the typical ligand behavior, which often consists of moving near the protein briefly and quickly drifting away again. Thus, only ligands that stay in contact for a certain time count as *landed*. The variable l_t is user-defined and depends on the time difference between the time steps. For the used data sets, a default value of 10 ps delivered the best results.

Further Values: For each protein atom, the number of different ligand molecules in contact is determined (n_{mol}). The Root Mean Square Fluctuation (RMSF) is also calculated for time-dependent simulation data. This value describes the internal per-atom movement of the protein during a simulation and is closely related to the aforementioned RMSD. It is computed by aggregating the difference between the position p_i of protein atoms i for each time step t and the corresponding atom position in a reference configuration at time step t_{ref} [Váz+18]:

$$v_{rmsf}(i) = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\Big(p_i(t) - p_i(t_{ref})\Big)^2} \tag{4.7}$$

Data Aggregation To avoid later re-calculations, all the derived per-time step data is continuously aggregated (cf. Figure 4.12 for more details). The data includes the values listed in Table 4.4, whereby the first four values ($n_{contact}$, n_{ctime}, n_{bond}, and n_{btime}), as well as the landing spot count n_{spots} can be directly derived by summing up the calculated contacts and interactions. For the number



(a) $t = 0$ **(b)** $t = 1$

Figure 4.12 — Aggregation of contacted atoms over two time steps. In time step (a) $t = 0$, two ligands (red and orange) are in the vicinity of the protein (green). For each protein atom, a neighbor search is performed (dashed circle). The overall searched area around the protein is encoded as a gray background. For example, five different ligand atoms are in the vicinity of the topmost protein atom, located in the center of the dashed circle. Thus, the time step counter is set to 1 (lower value), and the atom counter is set to 5 (upper value). Subsequently, the time simulation step (b) $t = 1$ is loaded, and the same calculation is performed, adding the new values to the previously calculated ones. Note that the atoms of the protein could also move (not depicted).- © EG & Wiley 2021 [Sch+21a]

of different contacted molecules n_{mol}, the IDs of all contacted molecules have to be stored and summed up after all time steps have been visited. Analogously, the continuous aggregation only calculates a part of the RMSF, namely the sum shown in Equation (4.7). The division by T and the taking of the root are performed after all time step information is known. In addition to the aforementioned values, the first and last time steps are determined for those periods where contacts happen or bonds are formed. The aggregated values are divided into two categories, where members of the first category can be calculated for docking and simulation data, and members of the second are only relevant for time-dependent simulation data. The members of the latter are the landing spot count, the RMSF, and the first/last time step.

4.3.4 Protein-Ligand Interaction Visualization

This section covers the design process of the visualization application, which was developed in close collaboration with the domain project partners in an iterative process. The described aggregated data is used and visualized using CLISD, which is discussed in the following paragraph. Furthermore, the visualization

Table 4.4 — Overview of all values accumulated during data processing. The upper five values can be calculated for simulation data and docking data. The lower ones only apply to time-dependent simulation data. The colored boxes represent the color coding used in the Figure 4.13

Value Description	Parameter Name
Number of Contacted Atoms( )	$n_{contact}$
Number of Time Steps in Contact ( )	n_{ctime}
Number of Bonded Atoms ( )	n_{bond}
Number of Time Steps Bonded ( )	n_{btime}
Number of Different Contacted Molecules	n_{mol}
Landing Spot Count	n_{spots}
Root Mean Square Fluctuation (RMSF) ( )	v_{rmsf}
First Contacted Time Step	t_{fc}
Last Contacted Time Step	t_{lc}
First Bonded Time Step	t_{fb}
Last Bonded Time Step	t_{lb}

contains a zoomed-in view (see *Sequence Detail View*), as well as a representation of the protein surface as SES (see *Molecular Surface Visualization*). A surface mesh and the protein data are only needed for the surface visualization. These files are small compared to the size of the actual data set and are fast to load, fulfilling requirement R7.

Compressed Ligand-Interaction Sequence Diagram (CLISD) The main goal of the 2D view is to give an *overview* of the protein-ligand interactions in line with requirement R1. However, the number of variables to represent poses a challenge. They include the time steps in contact, time steps bonded, contacted atoms, bonded atoms, and the RMSF (cf. Table 4.4). Besides, it is also intended to depict individual residues and to provide information on bonds that occurred during the simulation. This leads to a total of seven variables needed to be depicted in the CLISD.

During the visualization development, different approaches were considered. An initial take was using a circular representation, similar to the method of Vázquez et al. [Váz+18]. However, the high number of variables rapidly saturates the central part of the plot. As a result, a rectangular diagram was used instead, and different strategies were applied to save space. Several variables have only positive values since these are counts, which enabled it to use both

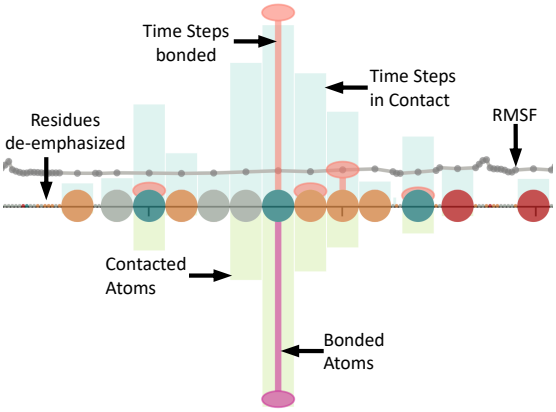


Figure 4.13 — A part of the CLISD. The x-axis denotes the amino acids of the protein as circles, and the upper bars show the number of time steps where a ligand is in contact (*time steps in contact*). The bars below encode the number of contacted ligand atoms (*contacted atoms*). The narrower bar below the x-axis represents detected bonds (*bonded atoms*), and the upper narrower bar shows the number of time steps with a bonded ligand (*time steps bonded*). The *RMSF* is displayed as a gray line. Amino acids with only a few contacts are de-emphasized, i.e., shrunk in the x-direction (*residues de-emphasized*). The amino acids are colored according to four physico-chemical properties, including polarity and hydrophilicity (( ) neutral hydrophobic, ( ) neutral polar, ( ) basic, ( ) acidic). - modified figure © EG & Wiley 2021 [Sch+21a]

the positive and negative y-axis to encode the *number of time steps in contact* and *contacted atoms* as bars. Second, visual representations of the residues were placed on the x-axis, similar to a classical sequence diagram. The residues are shown as small circles, color-coded by physico-chemical properties (cf. Figure 4.13). The circles can overlap with the bars, which is not a problem since the domain expert is mostly interested in regions with high values for $n_{contact}$ and n_{ctime}, where this overlap will not matter. In the second step, superposition is used to add more variables. Thinner bars with an elliptic cap encode the *bonded time steps* and *bonded atoms* in the positive and negative areas. Finally, the RMSF is drawn as a line chart in the positive area, leading to six additional variables visualized together with the sequence itself (see Figure 4.13).

Figure 4.14 — The screenshot shows the whole application. In the upper left, the protein surface visualization is located. In the top central, general information about the visualized protein is displayed alongside the selection fields for the data set and the currently displayed variable. Below, the exact calculated values for the currently selected amino acid can be inspected. The lower half shows the CLISD with a mouseover tooltip. In the upper right, a zoomed-in detail view of the CLISD is displayed, and besides the 3D view and the zoomed-in view, legends are displayed. - modified figure © EG & Wiley 2021 [Sch+21a]

Using larger thicknesses and less saturated colors for the bars in the background and more saturated colors and opacity for the elements in the foreground allowed the creation of this information-rich view. The presented values have vastly different ranges, and for that reason, the y-axis is shown without unit markers. Due to these range differences, bringing them on the same scale would make many bars unreadable small. Even when considering a logarithmic scaling of some bars, it would require at least two different logarithmic scales, making readability difficult. Therefore, the CLISD only gives a qualitative overview of the value distribution. However, if the user hovers over a residue, a tooltip box with the quantitative values appears, enabling a detailed analysis. If the user selects a residue, the values are also shown in the top center view (cf. Figure 4.14).

Since not all residues are equally important for researchers, e.g., residues that exhibit a high number of ligand contacts have a more prominent effect on the interaction, it is filtered based on this data. However, instead of completely removing non-interacting residues, they are shrunk and get assigned a smaller amount of space in the horizontal direction, shown on the left-hand side of Figure 4.13. In this way, a false impression of the position of the residues in the sequence diagram can be avoided. It enables showing only the important and saves space since most residues will never be in contact with ligands. That is, also making the CLISD more scalable than a normal equally-spaced distribution of all the elements. Although the CLISD is designed to be easily readable, there may be situations in which it can appear too cluttered. Therefore, multiple instances of the CLISD are stacked, enabling the user to switch off certain variables arbitrarily (cf. Figure 4.14).

Sequence Detail View In addition to the main view showing the CLISD, a supplementing detail view is added as exemplary shown in the top right of Figure 4.14. The detail view does not shrink residues and shows a cutout of only a few residues. As it can be seen in Figure 4.14, the slider in the scroll bar is small, indicating that a complete display of this diagram would exceed the available screen space. This zoomed-in view is added since the de-emphasized residues might directly influence the emphasized ones, for example, by performing large movements indicated by high RMSF values. The detail view is also linked to the main view. That is, if the user selects a residue, the zoomed-in detail view will center this residue, allowing for an in-depth analysis of this section of the protein even if the residues are de-emphasized in the main view.

Molecular Surface Visualization The SES of the protein received from the preprocessing step is stored as a triangle mesh. Each mesh vertex has multiple properties, e.g., colors and normals for lighting, required for rendering. Eight different coloring modes are available that either highlight the individual elements (amino acids) or encode the aggregated values of the protein-ligand interactions onto the surface. The coloring modes show the values $n_{contact}$, n_{ctime}, n_{bond}, n_{btime}, n_{mol}, n_{spots}, and n_{rmsf} and additionally, the eighth coloring mode depicts the hydrophobicity of the amino acids. The first seven modes use the Viridis sequential color map (). This reaches from violet for zero over green for the mid value to yellow for the maximum value (see Figure 4.10 and Figure 4.14). Liu and Heer [LH18] showed that this color map is superior to

several other sequential color maps in terms of error rates and response times. In contrast to the sequence diagram, the surface visualization uses the aggregated per-atom values instead of the values per amino acid. These values are more detailed and allow for a better perimeter of the relevant surface areas. The hydrophobicity coloring mode uses a diverging color map () depicting hydrophilic amino acids in blue, neutral in white, and hydrophobic ones in yellow.

Linked Interaction As mentioned, all views of the web-based visualization application are linked. For instance, when the user selects an atom in the 3D view, or more superficially, a point on the protein surface mesh, then the corresponding surface area of the entire residue is selected. This selection also leads to a marking of the same residue in the CLISD as well as in the zoomed view. The linking works for the opposite direction as well, i.e., selecting a residue in any of the diagrams leads to a selection and a highlighting in the other diagram and the protein mesh. Selections on the mesh are shown via a semi-transparent orange overlay. For selections in the CLISD, a vertical line is put as a highlight above the x-axis at the location of the corresponding residue. Hovering over the diagram or the surface has a similar effect, but another color is used here to highlight hovering interactions. These marks also propagated to the additional instances of CLISD diagrams. Additionally, as requested by the domain experts, the user can enter multiple amino acid IDs that will be marked separately, e.g., to highlight an active site. The IDs of an active site can be found by screening the mouse-over tooltips, but they are usually already known by domain experts beforehand.

4.3.5 Implementation Details

As mentioned earlier, data processing and the visualization of the proposed framework are split into two separate applications. This section briefly describes the implementation of them and which libraries and frameworks were used for their development.

Data Processing Application The application for value aggregation is written in C++ using the visualization framework *MegaMol* (see Section 2.6.1). For aggregating the values of the atoms a fixed-radius neighbor search is necessary, where the KD-tree implementation of *nanoflann* [BR14] is used. The KD-tree construction has to be performed for each time step, and the tree is used to store the ligand atoms since there are typically much fewer ligand atoms than

protein atoms. That is also beneficial for the parallelization of the aggregation process, as it enables a lock-free parallel computation for all protein atoms. Parallelization over all time steps would also be possible, but based on the design choices, the new approach is restricted to a low computational load that can run on commodity machines. Thus, the new approach is mostly IO-capped (cf. *Performance and Scalability*). This means that such a parallelization strategy could even harm the data throughput as the data is not read continuously.

The input data is given in the PDB file format and in the binary XTC trajectory file format (cf. Section 2.6.2). The aforementioned calculations are performed for each protein atom and amino acid. The amino acid values cannot be obtained by aggregating the already aggregated per-atom values, as this would lead to erroneous values due to multiple counting of contacts or interactions. The results of the aggregation process are written to a structured *JSON* file, which is later used for the visualization application. The *JSON* file contains an entry for each amino acid, listing the ID, the amino acid, and the aggregated values.

The 3D visualization requires the protein structure. While this is simple for docking data, Since there is only one conformation of the protein, a simulation of n time steps also offers n plausible choices for representative protein conformation. To retrieve the most representative conformation, the average structure of all atom positions is computed. Thereafter, it is searched for the time step, which has the smallest RMSD to this average. Additionally, the visualization requires a SES mesh, which is calculated by the MSMS tool [SOS96] and is stored as Polygon File Format (PLY) file that allows an arbitrary number of vertex attributes to be added. Thus, all aggregated values are stored alongside the atom IDs for each vertex. As the visualization also requires a PDB file, one is generated by copying the input PDB file and interchanging the atom positions with the positions of the representative time step. The file formats (PDB, JSON, PLY) are used because they are standardized formats, which makes it easy to read the data, even without the visualization application. This enables domain scientists to use the data more freely and increases reproducibility.

Web-based Visualization Application A screenshot of the fully web-based visualization application is depicted in Figure 4.14. It uses a server-client architecture and is written in TypeScript in combination with WebGL for the 3D visualization, whereas the CLISD is written in JavaScript using D3 [BOH11]. All views are linked with each other, allowing the user to select an amino acid in any of them. A selection is then propagated to all other views.

A drop-down menu is placed above the CLISD to load different data sets. If a new set is selected, the application will automatically update, and the corresponding PLY and PDB files are loaded, whereby the PDB file is used to assign the atom IDs from the mesh to the amino acids. In the drop-down menu to the left, the different aggregated variables can be selected to be presented on the protein surface by different coloring. For providing more orientation, a color legend as well as a legend for the CLISD are displayed.

To improve the usage of space, residues with a low number of contacts are de-emphasized by assigning them less space. Instead of setting a constant size for those, the space is dynamically calculated for each data set. This way, it can be ensured that most of the horizontal sequence space is devoted to important residues. Additionally, the parameter $W_{ER} \in [0, 1]$ determining the total width of all emphasized residues is set to a constant value, and the remaining space is dedicated to the remaining de-emphasized residues. This is achieved by counting the number of residues to emphasize (n_e). Given a user-defined contact threshold n_c, all amino acids with a $n_{ctime} < n_c$ are de-emphasized. Given the space to be devoted to the important residues W_{ER}, the size of each emphasized residue w_{er} is calculated as follows:

$$w_{er} = width \cdot W_{ER} / n_e \tag{4.8}$$

The empirically found value of W_{ER} was set to 0.7, as it is suitable in most cases. The results in the representation of the CLISD are exemplary shown in Figure 4.13. Larger values for W_{ER} do not visually filter the residues effectively, while much lower values would make them imperceptible.

For displaying the generated SES mesh, the application is built upon the *Mol** toolkit [3] [Seh+18], which is a collaborative project by the PDB in Europe and the RCSB PDB. It provides a technology stack for data delivery and analysis tools for macromolecules. Apart from file loading for many biomolecular file formats, it offers various common molecular visualizations, such as ball-and-stick. To properly incorporate the precalculated data, the toolkit was extended to support reading PLY files (see *Molecular Surface Visualization*), including all embedded vertex properties for coloring and grouping. This allows the protein surface visualization as depicted in the top left of Figure 4.14. Note that the probe radius for the shown SES is already set by the user during the precalculation step. It should typically correspond to the size of the ligands, e.g., the size of a water molecule. The rendered mesh supports selecting, also called picking,

[3] *Mol**. http://molstar.org (accessed 07/10/2024).

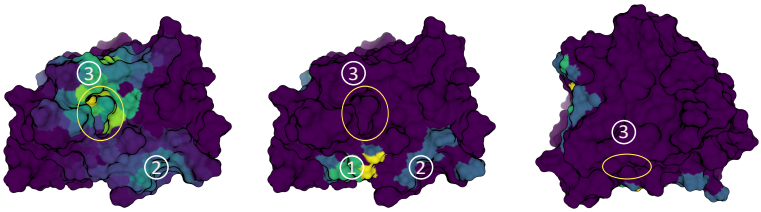
of vertex groups. This is used to identify the corresponding residues upon hovering over them or selecting them with the mouse. Due to the use of *Mol**, it is possible to interactively change the lighting conditions to add effects like Ambient Occlusion (AO) or to show contour lines.

4.3.6 Results and Discussion

To evaluate the aggregation and visualization system, this section presents two different application cases. The used data sets are received from different collaboration partners working in the fields of biochemistry and bioinformatics. The first data set is a MD simulation ensemble consisting of ten independent runs (see *Application Case I: MD-Simulation*). The whole ensemble, as well as all runs separately, are evaluated. The second data set is a smaller ligand docking data set (see *Application Case II: Molecular Docking*). The performance of the new approach is discussed in *Performance and Scalability*.

Application Case I: MD-Simulation One of the goals of the MD-simulation ensemble is to study the moving path of substrate or ligand molecules under realistic conditions. This especially includes a study of the ligand path along the surface of the receptor protein, as mentioned in Section 4.3.3. The researchers are interested in how ligands approach the active site. That is, whether a ligand has landed there directly from the solvent or after previous contact(s) on the surface, and if so, how the interactions happen. With current techniques, such as the filtering approach proposed by Vad et al. [Vad+17], some of this information can be obtained, but the spatial inspection is particularly difficult due to the size of the data set. Especially for the whole ensemble, millions of line segments would have to be rendered, even after filtering. This quickly becomes unfeasible performance-wise, as well as in terms of visual clutter. Therefore, it was decided not to use a direct visualization approach to satisfy requirement R3.

The simulation data set contains an ensemble simulation of a mutated variant of the enzyme *Candida antarctica lipase B* (CALB) [Upp+94], comprising ten independent runs of 160 ns each. With a time step size of 1 ps, resulting in a total number of 1.6 million time steps over all simulation runs. The simulated region is a cubic space with a side length of 32 nm, containing one CALB protein in water alongside 20 realistically distributed 4-paranitrophenol substrate molecules. The simulation was performed using *GROMACS* [Abr+15] and produced a final binary data set size of $\sim$2 TB, consisting mostly of water. The



(a) $n_{contact}$ Side View **(b)** n_{spots} Side View **(c)** n_{spots} Top View

Figure 4.15 — Surface visualization of the fourth run of the MD ensemble data set. The cavity containing the active site is marked with a yellow circle. Images (b) and (c) show the same surface coloring but from different angles. All the detected landing spots (① and ②) are not inside or at the border of the active site. The $n_{contact}$ in (a) reveals that there was indeed ligand movement towards and from the active site (③). Additionally, it can be seen that ligands tend to stay rather long in the *valley* at ②. - © EG & Wiley 2021 [Sch+21a]

water molecules in the input data are removed during the data preprocessing step as they were only required to create a realistic environment.

On a commodity machine (cf. R6), the pre-processing step takes approximately 4.2 hours for all ten runs (see *Performance and Scalability* for more details). The domain experts appreciated this since the actual simulation took several days. As their goal is to study general ligand paths, the pre-calculated landing spot values are of high importance to them as they indicate where the ligand molecules first come in contact with the protein. For this data set, the domain experts set the landing threshold value l_t to 10 ps (see Section 4.3.3). This value provides a good compromise between a stable contact and a good spatial correlation between measured and actual landing spots. When checking their hypothesis that ligand molecules do not directly access the active site but *crawl* along the surface, it quickly became apparent that this is mostly the case. One example of this behavior is shown in Figure 4.15. It depicts the molecular surface of the fourth ensemble run. Viewing the number of contacted atoms $n_{contact}$ shows that there appears to be a lot of movement in the cavity containing the active site, indicating that it has indeed been accessed (cf. Figure 4.15 (a)). The hypothesis that the molecules do not directly approach the active site from the surrounding medium can be tested by switching to the n_{spots} coloring scheme (R5). This affirms the hypothesis, as no landing spots were detected at the active site. The exact values can be checked in the linked CLISD.

When inspecting the surface, it can be identified that two ligand paths go to or away from the active site, also implicitly satisfying requirement R3. Such phenomena are observed in most of the simulation runs, although the active site is not reached in all of them. The $n_{contact}$ and n_{ctime} view of the complete simulation ensemble shows that the path marked with number ③ in Figure 4.15 (a) is a common path across all runs. By checking the proportions between the $n_{contact}$ and n_{ctime} value and comparing them with the values of other amino acids, one can derive the specificity of the contacts. Where $n_{contact}$ is approximately the same as n_{ctime}, only slight contact is made, meaning that only a few ligand atoms are close to the protein surface. If $n_{contact}$ is much greater than n_{time}, a more specific or direct contact can be assumed, as more ligand atoms are closer to the viewed protein atom. Further inspection of other coloring modes leads to the insight that these movements occur for multiple ligand molecules.

In addition to the paths, a further observation is made. The domain experts specifically requested the calculation of carbon-carbon interactions, as the inspected ligand is nonpolar (cf. R4). When examining the bond values n_{bond} and n_{btime}, it becomes clear that most of the carbon-carbon interactions happen while the ligand resides in a *valley* of the surface. In conjunction with n_{ctime}, it can be concluded that some valleys hold the ligand molecules by forming interactions. Based on this knowledge, for example, another protein mutant can be designed that does not have such valleys.

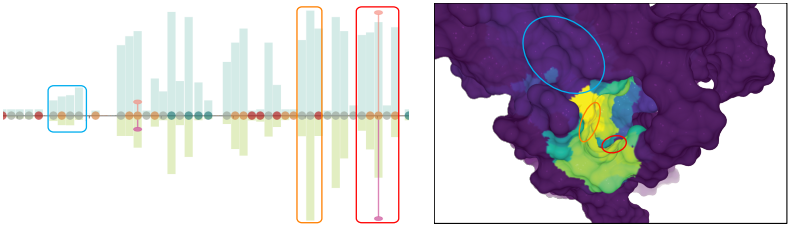
The domain scientists who conducted the simulation commended especially the interaction possibilities of the visualization framework. However, to understand the visualization fully, it was necessary to explain that the x-axis of the sequence diagram depicts the amino acid sequence in detail. After their feedback, the mentioned legends were added with the possibility of permanently marking certain amino acids, e.g., an active site. They also noted one minor drawback, which is that no area may be marked in the surface visualization when clicking on a certain residue in the CLISD. This can be the case for a residue that is not part of the protein surface over the whole simulation time. That behavior is mitigated by shrinking residues that have no contact. Additionally, the collaborating scientists liked that they did not need to use additional software and could share the visualization with colleagues by simply sending a link. They also appreciated that the rotation of the surface visualization remained the same when changing color modes, which helped them to maintain orientation.

Application Case II: Molecular Docking Apart from simulation data, the new tool can also be used to analyze docking results. Molecular docking experiments try to evaluate possible and energetically favorable ligand orientations and positions on the protein's surface (see Section 4.1.1). Another use case of molecular docking is determining possible active sites. For computationally modeled proteins, the active sites for given ligands are not always known beforehand or are not experimentally verified. Tendencies for certain surface locations can be determined by applying blind docking.

The data set contains multiple ligands docked with a given receptor protein. Several runs were performed for each ligand without specifying the target position on the protein surface. Each of the runs comprises 100 possible ligand conformations. Each run was handled separately in the preprocessing step. As mentioned in Section 4.3.3, not all calculated variables are applicable when using docking data. While the upper five values of Table 4.4 work as intended, none of the lower six is applicable. Since docking results have no temporal component, calculating time step-related variables is impossible. That is, due to the missing movement of the molecules, neither landing spots nor the RMSF can be calculated.

After the preprocessing, domain experts used the new tool to explore the docking results. Figure 4.16 (a) shows a cutout of the CLISD, as well as the most relevant part of the protein surface of one of the runs. In this case, hydrogen bonds were chosen as bond types. By combining all information, three specific areas of interest can be identified. The first is the area with the highest ligand presence probability, namely the residue with the highest n_{ctime} value. Most of the tested ligand molecules (82 out of 100) were docked at this location, i.e., certain surface properties of the protein are most likely favorable for the docking of ligands. One of these reasons could be a hydrogen bond formed at a specific surface location that is marked with a red border (cf. Figure 4.16). The presence of this bond can be investigated in the sequence view or the surface view by switching to one of the two bond-based coloring modes. The linked views allow for a direct investigation of this location in two different representations. The third area of interest is on the upper left of the surface view, marked in cyan. Only a few ligand molecules have a favorable position at this location, leading to a slightly lighter tone of purple. Investigating this position in the CLISD shows a lack of hydrogen bonds, which is one possible reason for the low presence of ligands.

For this task of protein-ligand docking analysis, the new tool allows for an easy and direct visual identification of relevant areas that could be prone to ligand



(a) CLISD Cutout **(b)** Predominant Docking Area

Figure 4.16 — The CLISD cutout in (a) and the protein surface visualization in (b), which shows the number of contacted time steps n_{ctime}, are both presenting the results of one single run of the molecular docking data set. The views have been cropped to the only area with a higher presence of ligands. The surface view shows a predominant docking area marked by green and yellow surface patches. The highest contact values ($n_{contact} = 1168$, $n_{ctime} = 82$) are highlighted by orange marks. The spot with the most often occurring H-bonds ($n_{bond} = 20$) is emphasized with red marks. Besides the yellow and green patches, an area with fewer contacts and without H-bonds can be identified (cyan marks). - modified figure (a) © EG & Wiley 2021 [Sch+21a]

binding. These specific input data sets are also a good example for the hinted *other uses* shown in Figure 4.11. In addition, using the standardized output data, the domain experts were able to work with the aggregated results even before the visualization system was usable. To generate surface renderings for an upfront analysis of the ligand density distribution, they used *MegaMol*, but other frameworks for protein surface visualization are also possible.

Performance and Scalability For a data aggregation-based approach, not only the calculation time but also the final data set size is essential. The smaller the final size is, the easier it is to transfer and visualize, but this is often accompanied by a greater loss of information. In this case, the omitted information is mainly the absolute atom position of each atom and the accountability of specific ligands for the observed effects. Examples for the original and the aggregated data sizes are listed in Table 4.5. It can be taken from the table that the resulting sizes heavily depend on the size of a single time step. Although the ligand docking set was much smaller than the others, the resulting data is approximately twice as large. The protein of the MD simulations consists of $\sim$ 300 amino acids, and the target protein of the molecular docking set

Table 4.5 — The original data set sizes of the two use cases and their sizes after the aggregation step. Cursive values mark binary data sets, the other are ASCII-encoded ones.

Data set	Original	Json	PDB	Mesh
Simulation (1 run)	*200 GB*	68 KB	357 KB	16 MB
Simulation ensemble	*2 TB*	68 KB	357 KB	16 MB
Docking (1 run)	65 MB	112 KB	623 KB	35 MB

comprises $\sim$ 500 amino acids. As the single MD simulation run is part of the whole ensemble and uses the same data, the size of the aggregated data is equal, although it contains different values. This led to compression by five orders of magnitude in the case of the MD simulation ensemble. Since the new approach is independent of the length of the simulation, petabytes of input data can be processed without increasing the memory footprint. Furthermore, the new approach uses the ASCII-encoded version of PLY file, as it makes it easier to write the mesh data. The size of the mesh data could be further reduced if the binary variant of the PLY file is used instead of the ASCII-encoded version.

As the aggregation times for the ligand docking data set are dominated by startup overhead, they were evaluated for the first run of the docking data. The initial data reading to retrieve the average atom positions for later calculating the RMSF took 699 s, and the second reading of the data for the actual value aggregation took 917 s. The timings do not include the writing time of the output data. The resulting data throughput was about 150 MB/s, which is close to the maximum reading speed of the used hard drive. The results for the other runs are approximately the same, differing only by a few seconds. The aggregation time of the whole ensemble behaved linearly to a single run and thus took ten times longer.

The final rendering of the aggregated values did not lead to any performance problems. The web-based tool was evaluated in four different browsers, including Google Chrome, Firefox, Opera, and Microsoft Edge. As it was not possible to turn off vertical synchronization in all browsers, it was left on during all measurements. It appeared that the visualization achieved almost constant 60 fps, even when the surface visualization was moved. Small drops down to roughly 30 fps were noticeable as the user changed the input data set or performed a brushing operation. Although the application also runs smoothly in modern smartphone browsers, it is recommended to use at least a tablet to have sufficient screen size to display the results easily readable.

As stated, the IO time is dominant in the presented use case, but with increasing sizes of single time steps, the neighbor search for ligand atoms could become dominant as it scales non-linearly with a complexity class $\mathcal{O}(m \log n)$ (m: no. of protein atoms, n: no. of ligand atoms). Although the actual visualization has no performance issues, available screen size could be a limiting factor for the CLISD. While the new approach scales better than the one of Vázquez et al. [Váz+18], there is a screen size drawback. For example, bar widths that are too small can make the visualization unreadable. This limit can appear when the width of individual bars falls below 1 mm and the gap between them values below 0.5 mm. In practice, this is the case, e.g., when the user has a screen width of 50 cm and at least 233 amino acids have to be rendered emphasized with $W_{ER} = 0.7$. That can only occur with either huge proteins or evenly distributed ligand movement over the whole protein surface. The latter was the case for the MD dataset. However, the screen space set for the emphasized amino acids was still more than twice the specified limit.

This section discussed an approach based on data aggregation to provide an overview of a MD simulation trajectory and to show relevant parts. A different approach utilizes dimensional reduction to summarize the entire simulation that is briefly described in the following Section 4.3.7.

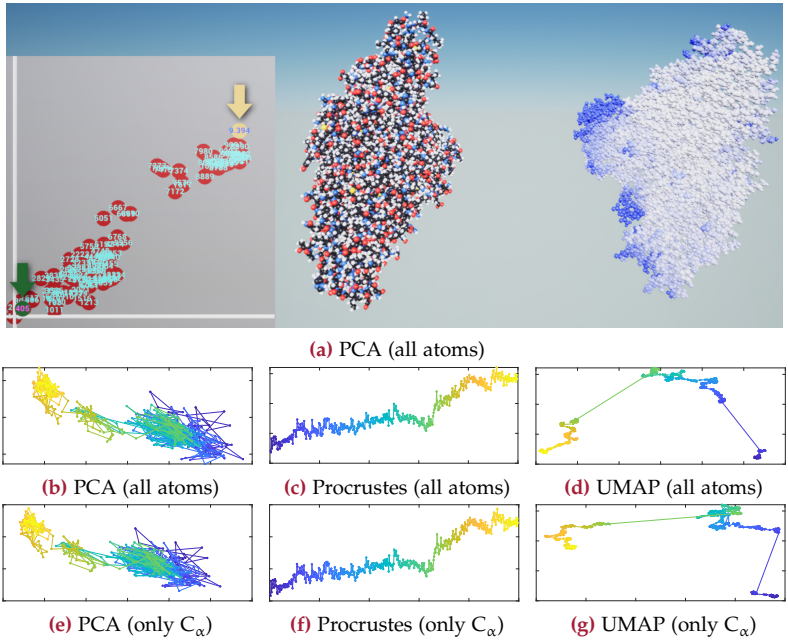
4.3.7 Comparative MD Simulation Analysis

An example of an alternative approach for MD simulation aggregation following a comparative-driven visualization method was developed in a co-supervised Master thesis. The result is depicted in Figure 4.17 and was presented at the Eurographics Workshop on Visual Computing for Biology and Medicine as a poster[4]. The application aims to give an overview of the simulation by showing conformation changes of a protein during the simulation. It allows the user to identify and visually compare interesting time points on demand.

The approach is based on dimensionality reduction, i.e., the protein is projected for each time step to a 2D space using its atomic positions[4]. Such an approach has been used before by utilizing widely used methods including PCA, t-SNE, or UMAP [How01; TG19; TWT21], but in current molecular visualization tools, analyzing protein simulations with this procedure is not established. As shown in Figure 4.17, three different dimensional reduction methods were compared. The scatter plots present the evolution of protein conformation during the simulation, and the distance in the embedding domain corresponds to the conformational difference. If points of the scatter plot form a cluster, they indicate similar conformations and maybe a stable conformation.

The simulation overview enables evaluating if there are semi-stable conformations[4]. That is, dimensional reduction has an advantage compared to methods like Procrustes analysis, which is part of RMSD offered by the majority of molecular visualization tools. Additionally, the user can select two time points for a detailed analysis. Therefore, the 3D structures of the selected protein conformations are presented in a side-by-side view. The user can interact with each 3D view individually or in a coordinated manner, whereby they can be zoomed and rotated. The exemplary Figure 4.17 (a) shows that the second protein has a white-to-blue gradient applied to present the conformation difference using the Euclidean distance between atom positions. White areas show parts of the protein with similar or equal conformation, and blue indicates a high difference.

[4] M. Bok, M. Schäfer, N. Brich, K. Schreiner, V. Fäßler, M. Keckeisen, O. Kohlbacher, and M. Krone. "Comparative Visual Analysis of Molecular Dynamics." *Eurographics VCBM Posters*. 2022



(a) PCA (all atoms)

(b) PCA (all atoms) **(c)** Procrustes (all atoms) **(d)** UMAP (all atoms)

(e) PCA (only C_α) **(f)** Procrustes (only C_α) **(g)** UMAP (only C_α)

Figure 4.17 — (a) shows the comparative visualization based on a MD simulation of the SARS-CoV-2 spike protein. On the left side of (a), a scatter plot shows a PCA-based dimensionality reduction for all time steps of the simulation, where each dot represents a time step. Two clusters are observable, reflecting the initially closed and open state at the simulation end. A time step is selected from each of these clusters (green and yellow dots/arrows). The corresponding protein conformations are visualized as space-filling models. The left one is colored by the chemical element, the right one by conformation difference, using a white-to-blue gradient (no difference to high difference). (b) - (g) are a comparison of different dimensionality reduction methods (columns: PCA, Procrustes (RMSD), UMAP) applied to all atoms (first row) and only to the C_α atoms (second row) colored by timestep: (1 n).[4]

4.4 Visual Analysis of Docking Data

Parts of this section have been published in:

- M. Schäfer, N. Brich, J. Byška, S. M. Marques, D. Bednář, P. Thiel, B. Kozlíková, and M. Krone. "InVADo: Interactive Visual Analysis of Molecular Docking Data." *IEEE Trans. Visual. Comput. Graphics*. 2024, pp. 1984–1997. DOI: 10.1109/TVCG.2023.3337642

In contrast to the sections before that discuss different approaches for analyzing and visualizing MD simulation results, this section takes up the investigating of molecular docking results. In comparison to MD simulations, precisely computing the behavior of a small system for a short period, molecular docking is a simplified version of it. It aims to predict the ligand's optimal conformation, orientation, and binding affinity onto the protein surface (see Section 4.1.1). This new work is a visual post-docking analysis tool further moving toward a very ligand-centered perspective of investigating protein-ligand interactions (cf. Figure 1.1). It focuses especially on spatial binding frequencies and protein-ligand interaction types, enabling the evaluation and decision-making by analyzing and enriching the results. Thus, it can support drug discovery, e.g., when it is utilized to analyze virtual screening results. As the approaches from the previous sections of computational chemistry contain one up to ∼hundreds of ligands, virtual screening results comprise 10s to 100,000s ligands. That enables the identification of possible binding sites and deriving over-represented functional groups and interaction types, which is of importance for protein engineering as well as for ligand designers.

The general docking procedure consists of target and ligand selection, their preparation, and the docking calculations, followed by the evaluation of the results [ML08]. There are many commonly used tools as *AutoDock Vina*, *Gold*, *DOCK*, *MOE*, *Glide*, *rDock* to perform the docking (see [Zha+22]), but these tools offer the user only very basic visualizations when evaluating the results. Therefore, often freely available molecular viewers, such as *PyMOL* [YCH17], *Chimera* [Pet+04], or *VMD* [HDS96], are used for visual inspection. However, they are also not specifically designed to support an efficient in-depth analysis of docking results. The typical analysis workflow includes processing the data in different tools and scripts, which is tedious and time-consuming. Hence, an application is needed that facilitates a comprehensive analysis of docking results by supporting and leading the user through the evaluation process.

Therefore, different visualization methods of proteins and ligands are necessary, such as detailed described in Section 2.5. Kozlíková et al. [Koz+17] gave a comprehensive overview of existing methods for visualizing molecules. Similarly, Osolodkin et al. [Oso+15] provided a review of visualization techniques for the exploration of chemical space. Besides the mentioned molecular viewers, although some other tools provide means to visualize ligands, e.g., *CaverAnalyst* [Jur+18] or *SAMSON Connect* [One]. They are not designed to visualize results of molecular docking and, thus, they lack specialized functionality and features. They are not interactive for large data sets and provide only basic interaction visualizations in some cases.

Visualization of Molecular Interactions Several research groups have focused specifically on the visual analysis of protein-ligand interactions. Furmanová et al. [Fur+17] proposed a method for the exploration of the geometric properties of the ligand transportation through the protein. Duran et al. [Dur+19] developed a system to explore long ligand trajectories via linked 2D and 3D views, offering enhanced charts that can be utilized for navigation to interesting parts of a simulation based on predefined properties. The methods proposed by Skånberg et al. [Skå+18] and Byška et al. [Byš+19] allow domain experts to define and extract their own properties. Schatz et al. [Sch+21a] aggregate ligand trajectories on the protein surface to study typical paths to the binding site (cf. Section 4.3). However, all these methods are tailored for the analysis of MD simulations and not molecular docking and, thus, cannot be easily extended to this case.

When analyzing docking data, it is crucial to understand the interactions between ligand atoms and protein amino acids. Hermosilla et al. [Her+17a] represented the interaction energies between a ligand and the surrounding amino acids with 2D and 3D arrows. As spatial representations often suffer from occlusion, alternative approaches provide such information with abstract representations. For example, *LigPlot+* [LS11] uses a structure diagram to represent a ligand in detail while protein amino acids are abstracted to spoked arcs. Furmanová et al. [Fur+20b] used similar structure diagrams, abstracting amino acids into stacked rectangles that provide information about their properties. In both cases, interactions are indicated by lines connecting the ligand atoms and the amino acids. However, these methods cannot be utilized in this case, as they are designed for exploring a single protein-ligand conformation.

Vázquez et al. [Váz+18] proposed a 2D visualization to explore multiple protein-ligand conformations over time. It abstracts and aggregates the amino acids into a circular layout around the ligand representation. *MolADI* [Bai+21] allows

for analyzing the evolution of the protein-ligand interactions over time using 2D plots. These visualizations can be used to analyze multiple interactions over time, but both approaches are limited to a single ligand. Jurčík et al. [Jur+19] used a matrix of small plots to explore large ensembles of ligand trajectories. However, this work is also not directly applicable to molecular docking results as the focus is on the comparison of multiple trajectories of the same ligand.

Large Ligand Ensembles The most typical examples for the visualization of multiple ligands are tools such as *ProteinPlus* [Fäh+17], which depicts the known ligands interacting with a selected protein directly in 3D or 2D using structure diagrams. *ProteinPlus* cannot be used in this case as it does not support the exploration of a large number of ligands or custom lists of docked ligands.

However, several tools have been proposed specifically for the exploration of large ligand ensembles. Janssen et al. [Jan+19] introduced a method that uses a scatter plot based on t-SNE embedding to compare the biological and chemical properties of a large number of ligands. A similar approach was presented by Sabando et al. [Sab+20], focusing on the comparison of different ligand groupings based on various descriptors. However, the scatter plots presented in these publications are not well suited for the exploration of the structural similarity of the ligands. To this end, Sabando et al. are using a separate 3D view for the spatial comparison of selected ligands. Gutlein et al. [GKK14] used actual 3D models of ligands instead of points within the embedding.

Enhanced tabular views are the most common approach for exploring various properties of individual ligands within an ensemble. For example, Sabando et al. [Sab+20] based their solution on Taggle [Fur+20a], while Data Warrior [San+15] heavily relies on the use of structure diagrams embedded directly in the tabular view to show the structural properties of individual ligands together with various quantitative values.

While all these tools mentioned above are well-suited for exploring large ligand ensembles, they are tailored to compare the ligands to each other. However, they do not consider the protein-ligand interactions that are crucial in the case of molecular docking data. To solve this drawback, additional research is required, which is the main focus of this section.

4.4.1 Ligand Properties and Interactions

Physico-Chemical Properties Ligands have various specific physico-chemical properties that can be used to infer their behavior in chemical reactions, sol-

ubility, stability, and other properties. This is crucial to support the users in judging the ligands regarding their drug-likeness, identifying possible chemical modifications to increase affinity towards a docking target, or inferring which and how environmental conditions could influence the docking.

The new-developed application is named InVADo (Interactive Visual Analysis of Molecular Docking Data). It uses information from ZINC 15 [SI15], a widely used ligand database containing more than 120 million drug-like compounds, for which it provides additional physico-chemical properties. A selection of three relevant physico-chemical properties is used that express the drug-likeness, which are partially based on the widely-used *"Lipinski's rule of five"* [Lip+01] that provides a likeliness of a drug to be effective when taken orally by a human. The first one is the *molecular weight*, a basic ligand property that allows inferring its size. The second one is the octanol-water partition coefficient *log P*, which is also part of this rule. It describes the distribution equilibrium of a molecule between the aqueous phase and n-octanol greasy phase, indicating its hydrophilicity (> 0 hydrophobic; < 0 hydrophilic) [San97]. The third one is called *fraction of* sp^3 (Fsp^3) and is a further drug-likeness score that is the coefficient of the number of sp^3 hybridized carbon atoms and the total number of carbon atoms. Sp^3 hybridized atoms have four single bonds to other atoms, which tend to be uniformly distributed around the carbon in a tetrahedral shape. Small molecules with more sp^3 hybridized carbons are thus less planar, have increased solubility, and can occupy more target space, which makes them better drug candidates [Wei+20].

Functional Groups and Interactions The aforementioned ligand physico-chemical properties are caused by functional groups influencing the interactions a ligand will form with other molecules. For docking data enrichment and to further support the decision-making of protein engineers and ligand designers, it is necessary to determine functional groups and protein-ligand interactions. Functional groups are sets of atoms that have specific physico-chemical properties [Mul94], for example, polar groups such as a carboxylic, ether, amine, or hydroxyl group. The tool *Checkmol* [Hai10] can be used to identify these groups. It distinguishes between 204 different types. From those functional groups, it is possible to derive reaction or interaction partners, interactions with other molecules, or possible modifications of a protein or ligand, making them crucial for molecular docking.

The new tool considers hydrogen bonds, halogen bonds, hydrophobic interactions, metal complexes, π-cation interactions, π-stacks, and salt bridges as

types of interaction. For more details on the types of protein-ligand interaction, see Section 2.3.2. In docking, these mentioned ones are among the most important properties for the chemical fit of a ligand to a given protein region. Together with their geometric fit, they lead to a lower or higher docking score (binding affinity). InVADo determines the mentioned interactions using the tool *Protein-Ligand Interaction Profiler (PLIP)* [Sal+15], which calculates non-covalent interactions at the atom level.

4.4.2 Tasks and Requirements

The goal was to create an interactive visual analysis application for molecular docking data. The main idea is to guide the users to interesting ligands or hotspots on the surface of the receptor molecule and to highlight over-represented physico-chemical properties within those areas. This enables the users to investigate and answer questions about the receptor molecule and particularly well-suited ligands. Consequently, information extracted and derived from docking data is, for example, used in computer-aided drug design (virtual screening) or by protein engineers working on biotechnological applications (raw material production, e.g., fermentation, etc.). It also helps to answer important scientific questions, for instance, biologists researching signal transduction in organisms in which possible ligands for receptors are investigated. There is no clear *"standard"* workflow, since it depends on the specific research question and data. In general, there is no coherent analysis environment. It is usually a combination of different tools and scripting, e.g., *MS Excel*, *PyMOL* [YCH17], *PLIP* [Sal+15], which leads to a complex and time-consuming workflow. Based on several discussions and semi-structured interviews with different domain experts working in pharmacology and biochemistry, common tasks and usage scenarios were identified to make InVADo as broadly applicable as possible.

When analyzing docking results, the first task is typically to get an impression of the distribution of different ligands on the molecular surface. That includes inspecting ligands based on their free binding energies, sizes, energy distributions, functional groups, and more. That is, users need to browse the results, inspect, and filter them before performing further analysis steps.

A second task is to identify points of interest, like clusters of chemical properties or specific ligands. This can indicate possible binding sites, which can be explored for their suitability as drug targets.

In addition to investigating clusters formed by ligands, users also need to determine the most frequent and highest-scoring ligands and explore their binding poses. These so-called *hits* are often possible drug candidates [Fer+15].

Besides finding the top-scoring hits, a more general and detailed analysis of all ligands is an often required task. That allows users to identify overall good binders, i.e., ligands with high docking scores in all found clusters, even if they are not the highest-scoring ones. The users are typically interested in identifying whether specific physico-chemical properties or functional groups (called pharmacophores) are over-represented in a specific region of the protein. From this information, the affinity for a certain ligand type can be derived. It also allows drawing conclusions about the specificity of a certain binding site, i.e., inferring preferred binders for this binding site.

Similarly, it is important to assess the poses of the ligands, i.e., their spatial embedding in the context of interactions and chemical properties. This supports lead optimization. A *lead* is a ligand that is a drug candidate that can be further improved with small structural modifications [Sho+02].

To gain a deeper understanding of the potential interplay of a receptor and ligands, it can be necessary to enrich the docking results using additional data or analysis. One such task is the localization of suitable hot-spots for protein engineering, for instance, suitable locations for a mutation that leads to increased catalytic rates of enzymes, which is important in biotechnology. Additional data can also help to execute some of the preceding tasks. This includes interactions like hydrogen bonds or hydrophobic contacts as well as the previously mentioned chemical properties and functional groups. As these additional data are not part of the docking results, their inclusion allows for a more detailed evaluation of the docking.

Based on the tasks described above, six requirements for the new visualization application were derived following the strategy by Brehmer and Munzner [BM13]:

R1 Provide a structured visual and textual access to the docking results, giving an overview that enables browsing and filtering as an entry point for further analysis.

R2 Identification of potential binding sites and other interesting hot-spots with high binding affinity.

R3 Hit identification by facilitating the determination of the most frequently docked and top-scoring ligands and binding poses.

R4 Support the identification of overall good binders and ligand types (not only top-scoring ones) for a comprehensive overview and specificity analysis.

R5 Visualization of docked ligands, interactions, and chemical properties to enable a detailed visual analysis.

R6 Enrichment of docking data to support evaluation of global docking results and decision-making.

4.4.3 Application Overview

The new application InVADo is a visual analysis tool specifically designed to support the exploratory analysis of molecular docking data satisfying the requirements listed in Section 4.4.2. Due to the amount and complexity of the data and, consequently, the large number of required views, it is intended to be used on dual or ultra-widescreen monitors.

To visualize the results, InVADo is separated into two main views consisting of a 3D and a 2D part (see Figure 4.18). The first one is a dashboard, which features plots and tables that give an overview of the docking results but also allows users to explore the data, e.g., to browse, search, and filter it, and to get additional details. The second one is a 3D view showing the structure of the receptor molecule and the docked ligands, the spatial location of the ligand clusters, and protein-ligand interactions, among others. InVADo filters the docking data by docking scores and spatially clusters the remaining ligands. These clusters are the entry point for users to explore, analyze, and evaluate the docking results. Clusters can be selected in the 3D view or via the mentioned dashboard with multiple linked plots and tables, each with an increasing level of granularity, ranging from a whole cluster to individual ligands and their binding poses. InVADo also incorporates data from post-docking analyses that determine protein-ligand interactions and functional groups, which are used for filtering and sorting, making the analysis more comprehensive.

The application offers user interactions such as selection, filtering, or changing the molecular representation. The views are tightly linked so that all selections and other changes are applied across all views. A demonstration of InVADo's feature is also given in a video[1] available in the supplementary material of the publication [Sch+24]. The provided views and their interactivity enable

[1] InVADo video: https://doi.org/10.1109/TVCG.2023.3337642/mm1

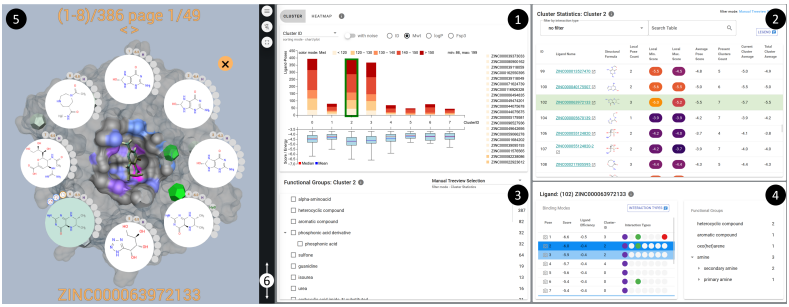


Figure 4.18 — InVADo Application Overview: ① ***Docking Overview***: Visualizes the results of the clustered docking as a stacked bar chart combined with a box plot, where the bars represent the clusters. ② ***Statistics View***: Gives cluster-specific information about ligands and their statistics in a tabular view with filter and sort options. ③ ***Functional Groups View***: Provides cluster-specific information about functional groups (chemical substructures) of the ligands presented as an expandable and selectable treeview. It also extends the filter options of the *Statistics View*. ④ ***Ligand View***: This table shows the individual binding poses offering ligand-specific details like the docking score and interactions. It is combined with a treeview presenting the functional groups of the current ligand. ⑤ ***3D Visualization***: Offers interactive visualization of the clustered docking results with a *Radial Menu* to browse them. ⑥ ***Sidebar Menu***: Collapsible control panel allowing users to adjust the appearance of the *3D Visualization* and to parametrize the clustering of the ligand binding poses and the functional groups. - © IEEE 2024 [Sch+24]

the users to get an overview as well as detailed information about individual ligands, which can lead to more profound insights into the properties of the receptor, its potential binding partners, and the interactions involved.

Algorithm Overview InVADo starts with a data processing step to get the visualization data for the views. That means a clustering or binding site estimation is performed, i.e., locations of high binding affinity are identified (R2), which are clusters of docked ligand binding poses. The goal is to identify hot-spots with a high density of docked ligands while discarding the remaining binding ligand poses that are not within one of these areas. Additionally, users can set a docking score threshold to discard poorly scored binding poses (R3/R4). Since the number of clusters is unknown priori, it was opted for

DBSCAN [Est+96]. The input parameters of DBSCAN are the maximum search radius and the minimum number of samples (ligand poses) in a cluster (see Section 2.4.2). As input for the clustering, the centroids of the ligand poses are used. Poses that are discarded either by the score threshold or by not being in one of the DBSCAN clusters are referred to as *noise*. Since the found clusters indicate possible binding sites, the part of the molecular surface is extracted that is close to the ligand atoms of a cluster, and is marked as a potential binding site.

Additionally, the clustering and analysis of the docking results is supplemented by enrichment steps. For a more comprehensive analysis, additional information on certain physico-chemical properties is added to the ligands (cf. Section 4.4.1). Further enrichment steps include the determination of functional groups of the ligands by using *Checkmol* [Hai10] and the calculation of the protein-ligand interactions with *PLIP* [Sal+15]. The functional groups found are, in turn, clustered using DBSCAN to identify over-represented functional groups in the ligand clusters.

These clustering steps and the information enrichment of the docking results form the basis for the visualizations of both main views, whose design and functionality are described in detail in the two following Sections 4.4.4 and 4.4.5.

4.4.4 Dashboard Functionalities

The design of the dashboard follows Keim's visual analytics mantra: *analyze first – show the important – zoom, filter and analyze further – details on demand* [Kei+08]. It consists of four views that feature an increasing amount of details about the docking results, the extracted clusters, the individual binding poses of a specific ligand, and the chemically relevant functional groups. This section describes these views and how they fulfill the requirements.

Docking Overview – Stacked Bar Chart & Box Plot The docking scores and the extracted clusters are used to provide an overview of the docking results (R1). The clustered docking results are further aggregated, and the information is presented as a stacked bar chart and a box plot (Figure 4.18 ① & Figure 4.19). Each cluster is represented by one vertically aligned bar and box plot, respectively. This combination allows the users to get an impression of how many ligands are within each cluster (height of the bars) and the distribution of docking scores within this cluster (box plot). The latter also contributes to fulfilling R3, as the plots can be used to identify extremes, e.g., the cluster with

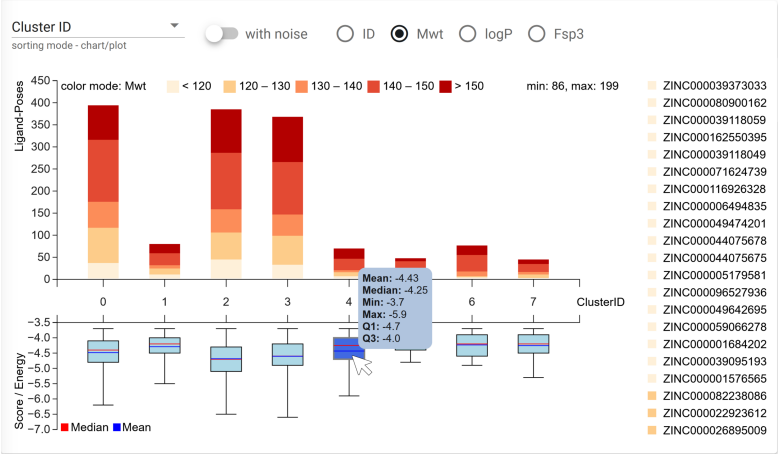


Figure 4.19 — Docking Overview – Clusters: This view is a combination of a stacked bar chart and a box plot showing the ligand pose clusters. The bar chart visualizes the number of clusters and binding poses within a cluster. Quantized color maps are used to encode either the molecular weight (*Mwt*), the octanol-water partition coefficient (*log P*), the fraction of sp^3 (Fsp^3), or the ligand ID (see legend to the right). The box plot gives an overview of the docking score distribution within each cluster. Tooltips with more detailed information are available for both visualizations. - © IEEE 2024 [Sch+24]

the highest amount of docked ligand poses or the highest mean or absolute docking score. Note that the *AutoDock Vina* scores, which correspond to the docking energies, are negative, with smaller values signifying better docking / higher binding affinity.

Users can sort the clusters either by their ID, the number of ligand binding poses or according to the mean docking score of the clusters. Furthermore, they can also choose to show information about the poses previously classified as noise (see Section 4.4.3). This will add a new bar and a new box plot containing the data of all noise poses.

Moreover, users can switch between four different coloring modes for the stacked bar chart (cf. Figure 4.19): each ligand is assigned either an individual color (*ID*) showing how many binding poses of the same ligand are in one stack, or a coloring depending on one of three physico-chemical properties is

used (molecular weight (*Mwt*), octanol-water partition coefficient (*log P*), or fraction of sp^3 (Fsp^3)). This enables the users to see the chemical composition of the clusters, which partially addresses R5, and it allows them to discover similarities between ligands or clusters. As mentioned in Section 4.4.1 the chemical properties are collected from the ZINC database or, if no ZINC name is present, are calculated with *PLIP* except Fsp^3 (R6). To clearly show the value distribution of the chemical properties, the data is quantized into five bins, to which a corresponding color is assigned (Mwt (), log P (), Fsp^3 ()). Note that only the value range between the quantile 5% and 95% is used, i.e., the first and last bin is extended to reduce the influence of outliers. In this range, equidistant bins are created. When the cursor hovers over a stack, all stacks of the same ligand in other clusters are highlighted, and a tooltip with detailed information about the pose, the ligand, and the values of the chemical properties appears. For the box plot, the tooltip shows the mean, median, box quantiles Q1/Q3, and the min/max values (cf. Figure 4.19).

Docking Overview – Heatmap As an alternative overview to the stacked bar chart and the box plot, a segmented heatmap is included (see Figure 4.20), which can be reached by switching to the HEATMAP tab (cf. Figure 4.18 ①). The heatmap not only gives an overview of the docking results and the clustering (R1) but also specifically addresses requirement R4, as it allows the users to easily identify overall good binders. The columns of the heatmap are the clusters or binding sites, and the rows are the ligands. A segmented heatmap is used, i.e., if a ligand has multiple poses that belong to the same cluster, the corresponding rectangle of the heatmap is divided horizontally into multiple segments. That is, each segment represents a binding pose of a ligand. The segments are colored according to the docking score of the respective pose using the established Inferno color map (), with black mapped to the maximum value (poor binding; binding free energy/score closer to zero) and light yellow mapped to the minimum value (best binding; free energy/score strongly negative). The plot is scrollable since the number of ligands usually exceeds the vertical screen space. A tooltip shows the docking score, the ZINC name of the ligand, and the different IDs, including the cluster, the ligand, and the specific binding pose. Similar to the bar charts and the box plot, the users can choose whether only the clusters are shown or if the noise is displayed in an additional column of the heatmap.

Each row can be seen as a docking profile that helps to identify overall good binders (R4). The heatmap directly shows if the binding poses of a specific

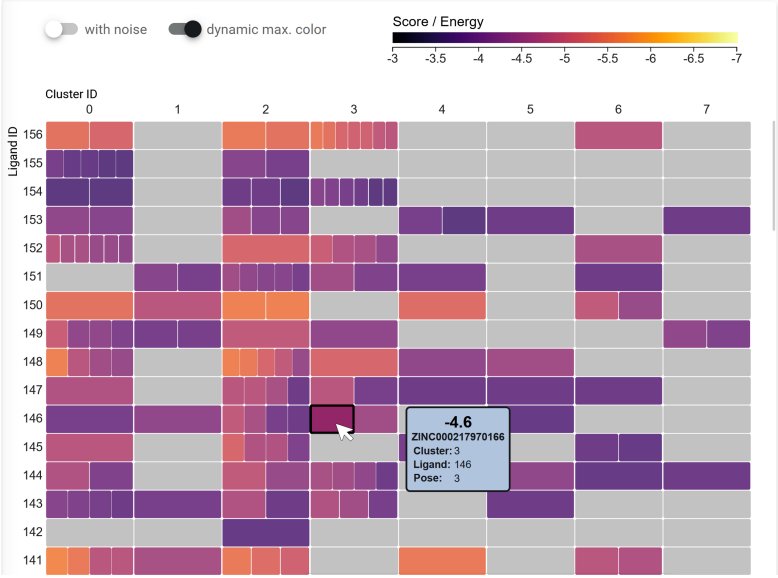


Figure 4.20 — Docking Overview – Heatmap: A segmented heatmap summarizes the docking and clustering results. Columns correspond to clusters and rows to individual ligands. Each cell of the heatmap where a cluster includes one or more poses of a ligand is drawn with one or multiple colored segments, depending on the number of binding poses present in this cluster. The segments are colored depending on their docking score using the Inferno color map shown at the top. A tooltip with additional information is available for each segment (i.e., ligand binding pose). - modified figure © IEEE 2024 [Sch+24]

ligand are docked in several different binding sites or mainly one, and which poses reached high scores — shown by a "hot" color on the Inferno color map — in any of the binding sites (R3). It also allows users to assess the identified binding sites (R2), e.g., the binding site specificity, by checking if a binding site has only a few but good binders or many low-scoring ones.

Statistics View The *Statistics View* (Figure 4.18 ② & Figure 4.21) is a table that provides detailed textual information and summarizing statistics about the ligands of a selected cluster or the whole docking data if no cluster is selected

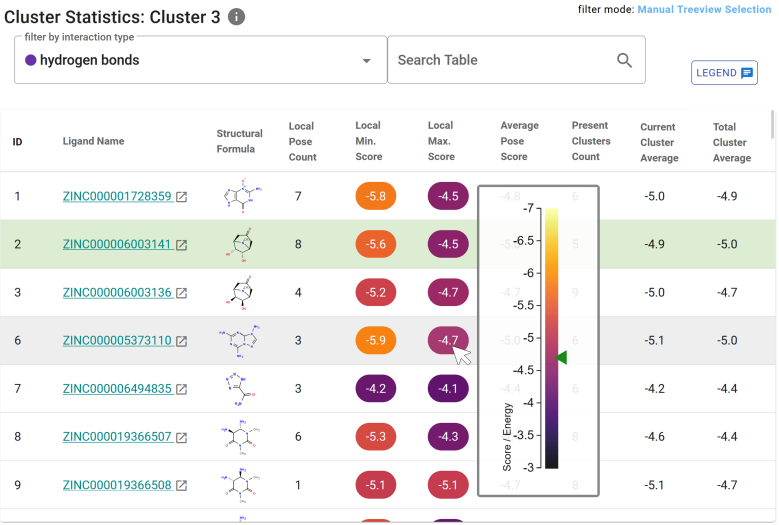


Figure 4.21 — Statistics View: An interactive table with cluster-specific information about the ligands. It provides the ligand name, the structural formula, min/max score values, and other statistics about each ligand in the currently selected cluster. The ligand in the table can be filtered by interaction types and searched for any given string or value (area above the table). Hovering over a ligand shows additional information, such as the tooltip presenting the binding score encoding by the Inferno color map. - © IEEE 2024 [Sch+24]

(R1). A cluster selection can be done by clicking on the corresponding stacked bar or box plot. The *Statistics View* includes the name, e.g., the ZINC ID, and an image of the structural formula of all ligands included in the selected cluster. Moreover, it also lists the number of docked poses of a ligand in the current cluster (*Local Pose Count*), the maximum and minimum scores of all poses in the cluster (*Local Min./Min. Score*), the average score of all poses of this ligand including non-clustered poses (*Avgerage Pose Score*), how many clusters contain binding poses of this ligand (*Present Clusters Count*), as well as the average score for the poses of the selected cluster (*Current Cluster Average*) and the average score of all binding poses present in any cluster (*Total Cluster Average*). A tooltip provides more information about the currently hovered value. This can be either a color-coded scale of all docking scores with a marking for the score of

the currently hovered pose (see Figure 4.21) or a magnification of the structural formula to make it easier to read. The table can be sorted by each column, which supports the user in identifying top-scoring ligands (R3). A ligand can be selected by clicking on the corresponding row highlighted in light green. Clicking on the name of a ligand in the second column opens the ZINC database webpage for this specific ligand, which provides more detailed information.

Since the minimum and maximum scores of a ligand are important to identify good binders, the background of these score entries is emphasized by coloring. Therefore, the Inferno color map () is used, with black mapped to the maximum value (poor binding) and light yellow mapped to the minimum value (best binding).

The *Statistics View* also offers search and filter capabilities (R1). Users can filter the ligands using a drop-down menu to show only ligands that exhibit a specific interaction (cf. Section 4.4.1). Furthermore, a text field is included to search for terms in all table columns, making it possible to search, e.g., for a specific ligand name or a certain score.

Functional Groups View As explained in Section 4.4.1, functional groups are chemical building blocks of ligands that influence their binding behavior. That is, analyzing the presence or absence of certain functional groups is a crucial task in identifying good binders (R4). To facilitate this, a treeview was integrated that shows which functional groups are available either in the whole data set or in the currently selected cluster (see Figure 4.18 ③). A further treeview was also added to the *Ligand View* showing the specific functional groups of a single ligand (Figure 4.22). Since docking data usually do not include explicit information about functional groups, the ligand data are enriched with this information derived by *Checkmol* (R6). A hierarchical view was chosen to present the functional groups, since similar functional groups can be assigned to super-groups. All groups listed for *Checkmol* were used to build their intrinsic hierarchy. The nodes of the treeview are expandable, and the number of each contained group is shown on the right. An exemplary hierarchy for the group *1,2-aminoalcohol* would be *hydroxy compound* → *alcohol* → *secondary alcohol* → *1,2-aminoalcohol*. The treeview enables users to browse the functional groups, compare their number, and identify over-represented groups. That is interesting information, especially for ligand designers or protein engineers, e.g., regarding free binding energy optimization.

The groups of the *Functional Groups View* in Figure 4.18 ③ can be selected using checkboxes, which act as a filter for the *Statistics View* table. A group selection

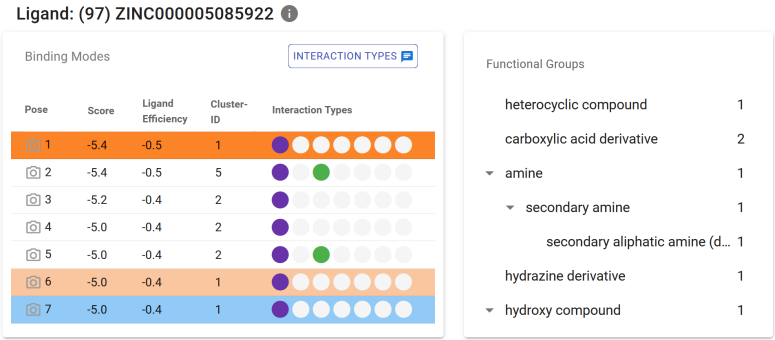


Figure 4.22 — Ligand View: This view provides information about the poses of a specific ligand using a table and a treeview. The table lists the pose ID, docking score, ligand efficiency, cluster affiliation, and information about the presence of various protein-ligand interactions. The treeview to the right shows the hierarchy of functional groups contained in the currently selected ligand. - © IEEE 2024 [Sch+24]

leads to collecting all ligands that contain one of the selected functional groups, which is followed by automatic insertion of their ligand names into the search field of the *Statistics View* table.

Ligand View After a cluster has been chosen from the *Docking Overview* and a ligand of interest is selected in the *Statistics View*, InVADo provides information about the ligand and its individual poses in the *Ligand View* (Figure 4.18 ④ & Figure 4.22). The data is presented in a tabular view, enabling the user to browse the pose properties and to compare and identify ligand poses with the highest ligand efficiency or certain interactions. Ligand efficiency is the binding free energy divided by the number of non-hydrogen atoms. The table of poses has the same sorting options as the *Statistics View* to support the identification of good binders (R3, R4). It lists the pose ID, the docking score, the aforementioned ligand efficiency, and the cluster assignment of each pose of the selected ligand. Furthermore, it shows which interaction types are present between the pose and the protein. The protein-ligand interactions are determined by *PLIP* and their presence is indicated by colored circles (R6). Based on discussions with the domain experts, certain interaction types were integrated, including (●) hydrogen bonds, (●) halogen bonds , (●) hydrophobic

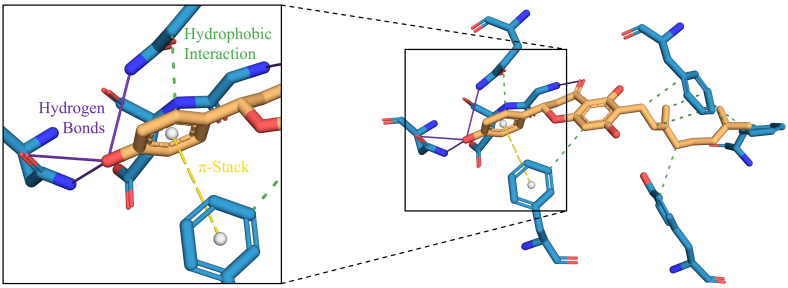


Figure 4.23 — Protein-Ligand Interactions: The flavonoid *Bonannione A* (orange) docked to *P-Gylcoprotein* (blue, only interacting protein residues are shown). Interactions are drawn as colored dashed or solid lines ((●) π-stack, (●) hydrophobic interaction, (●) H-bonds). Image created by *PLIP* [Sal+15]. - modified figure © IEEE 2024 [Sch+24]

interactions, (●) metal complexes, (●) π-cation interactions, (●) π-stacks, and (●) salt bridges. The colors that represent the interaction types follow the color conventions established in the domain and used by tools such as *PLIP*.

As shown in Figure 4.22, some binding poses are highlighted by a blue or orange background. Blue highlighted poses are in the selected cluster, and orange ones additionally contain at least one of the currently selected functional groups (cf. *Functional Groups View*). A click on a camera icon of a row will open a pre-rendered image of the protein generated by *PLIP*, showing a visualization of the selected pose of the ligand and the mentioned interactions, including the affected protein residues (see Figure 4.23). Additionally, the functional groups of the ligand are shown in a treeview to the right of the table of poses. It is similar to the larger treeview in the *Functional Groups View* with the difference that only ligand-specific functional groups are displayed.

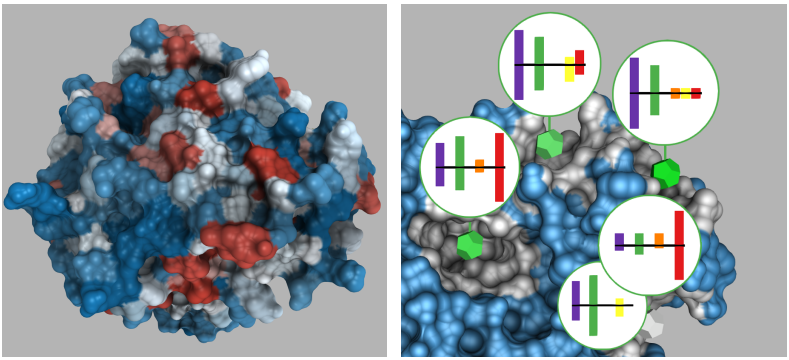
4.4.5 3D Visualization Features

The dashboard, as described in the previous Section 4.4.4, is complemented by a fully interactive 3D visualization of the clustered docking results (Figure 4.18 ⑤). It can show various representations of the receptor protein, the ligands, interactions, functional groups, and more.

Initially, only the protein and abstract representations of the clusters are visualized to give an overview. When selecting a cluster for detailed analysis, a *Radial Menu* appears that allows browsing the ligand binding poses within the cluster (Figure 4.25 ①). To enhance the analysis, additional information can be added to the 3D view, and multiple clustering, sorting, and filtering options can be adjusted and refined. Users can apply a clip plane to reduce clutter and improve the view of the binding sites. The 3D view and the 2D views provided by the dashboard are tightly linked, i.e., all selections and filters will be propagated to all other views, enabling a seamless analysis. The options of the 3D view can be adjusted via the sidebar menu of the dashboard (see Figure 4.18 ⑥ & Figure 4.25 ②).

Protein and Cluster Visualization By default, the receptor protein is rendered using the SES, a smooth molecular surface that shows the interface between the protein and a specific small molecule like a solvent or ligand [Koz+17; SK19] (see Section 2.5.2). It visualizes the 3D structure of the protein, which is an important aspect of docking. The spatial perception is enhanced using ambient occlusion [Gro+12], which particularly helps to get a better impression of the depth of cavities (i.e., possible binding sites; R2). The SES can be colored by the physico-chemical properties of the protein, which supports analyzing the binding capabilities (R5). For example, the hydrophobicity coloring uses a diverging blue-white-red color map, as shown in Figure 4.24 (a). This coloring can assist users in determining transmembrane proteins or hot-spots in a protein binding site that repel water and, thus, might be a suitable candidate for interacting with certain ligands. The coloring modes and the color maps are user-adjustable parameters, including, e.g., coloring by chemical element, B-factor, or protein chain.

The ligand clusters are visualized as dodecahedra placed at the centroid of the cluster (Figure 4.24 (b)) and colored using a linear color map from white to green (). The color expresses the quantitative difference in cluster size. It was opted for dodecahedra instead of simpler spheres to visually differentiate them from the atoms, which are often represented as spheres.



(a) Physico-Chemical Properties **(b)** Interaction Type Bar Charts

Figure 4.24 — (a) Hydrophobicity as a physico-chemical property is color-mapped onto the protein surface. (hydrophilic (blue), neutral (white), hydrophobic (red);). (b) The dodecahedrons mark cluster centers and encode by a linear color scale where the most ligand poses are located (few (white) to maximum (green);). The circles above them contain Interaction Type Bar Charts, summarizing the interactions in a cluster. The top bars show the relative number of interactions with respect to the binding site surface area and the bottom bars with respect to the total interaction count in the cluster. -

Selecting a cluster by clicking on the corresponding dodecahedron makes the dodecahedron disappear and reveals the *Radial menu*, as described in detail below.

Radial Menu The *Radial Menu* (Figure 4.25 ①) provides additional information about the respective cluster. The circular layout encloses the point of interest and enables an efficient usage of the screen space. It allows the users to browse all docked ligands in the selected cluster (R1). Each of the circular menu items shows the structural formula of an individual ligand pose. The *Radial Menu* items are sorted by the score in descending order. Optionally, sorting can be extended by a second condition: the ligand pose must exhibit a certain interaction, otherwise, it will get a rear position even if it has a high score. The number of menu items, the text size, and further parameters of the menu are user-adjustable. As typically not all results can be shown at once, the *Radial Menu* is divided into pages that can be browsed using the arrows above it.

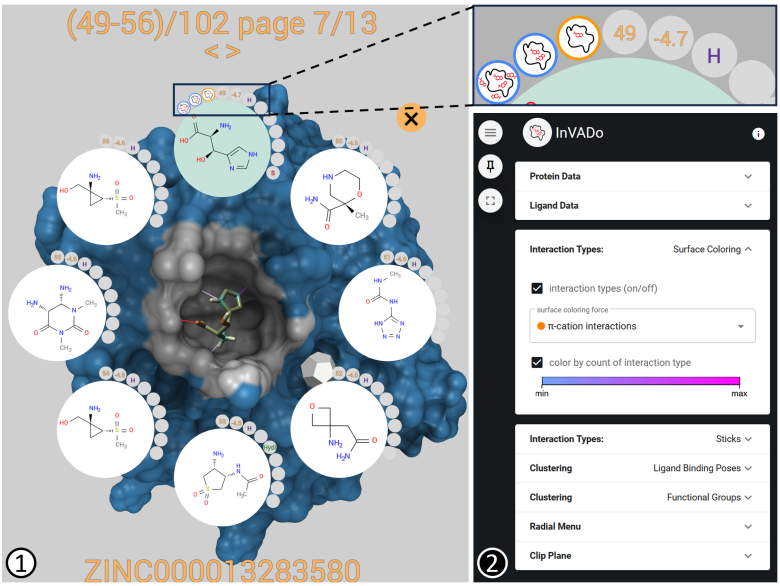
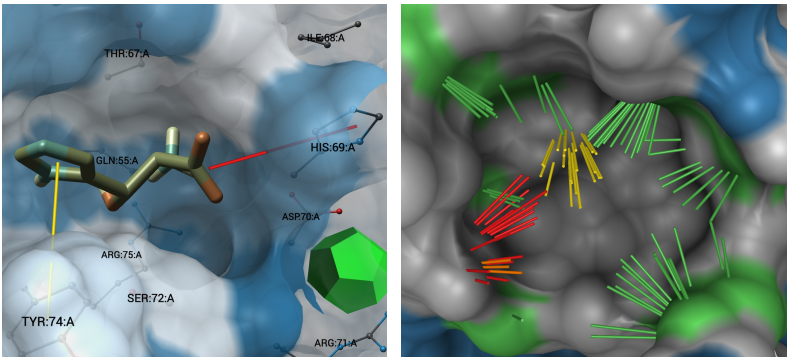


Figure 4.25 — ① The *Radial Menu* with a zoomed-view showing the sub-circles in detail. The center sub-circle shows the rank (or ID) of the ligand pose. To the right, the docking score and the presence of interactions are shown (here H-bonds). The outlined sub-circles to the left are control elements for different rendering modes of ligand-binding poses: only the selected pose, all ligand poses limited to the selected binding site, or all poses. ② The expanded *Sidebar Menu* as indicated in Figure 4.18 ⑥ showing different control panels for steering the 3D visualization.

Above the arrows, a text label shows which ranks of ligand poses are currently presented (see Figure 4.25 ①: ranks 49–56 out of 102 poses are shown, i.e., the user is on page 7 of 13).

Besides the rank and score of the current ligand pose, additional information is shown in sub-circles (cf. Figure 4.25). The small circles around each item present the rank (or ID) and the docking score of the pose, allowing users to compare binding poses and identify, e.g., the best-scoring ligands and poses (R3). To the right of the score are multiple sub-circles with fixed positions for each of the seven interaction types. Colored characters are drawn into the corresponding



(a) Ligand Pose **(b)** Interaction Types

Figure 4.26 — (a) A ligand pose is visualized as stick representation located in a binding site. Surrounding protein residues are labeled and are visible through the semitransparent SES. Interactions are shown as red and yellow sticks. (b) The interactions in a binding site across all ligand poses are visualized as sticks and colored according to interaction type. Surface patches affected by hydrophobic interactions are highlighted by color (green). - © IEEE 2024 [Sch+24]

sub-circle, indicating the interaction if it is present for the pose (R6). If a menu item (ligand pose) is selected, three further sub-circles appear to the left of the rank. Clicking on them allows the user to control which 3D models of that ligand are rendered (R1). By default, only the currently selected ligand pose is rendered (see Figure 4.25 ①). That is indicated by a symbol illustrating the protein binding site as a black line enclosing one small red ligand, while other ligands are depicted in a lighter shade of gray. The second option is to show all poses of the currently selected ligand located in the current cluster, which is indicated by the icon showing all ligands within the black outline in red. The last option is to display all ligand poses regardless of cluster affiliation, indicated by red ligands inside and outside the binding site.

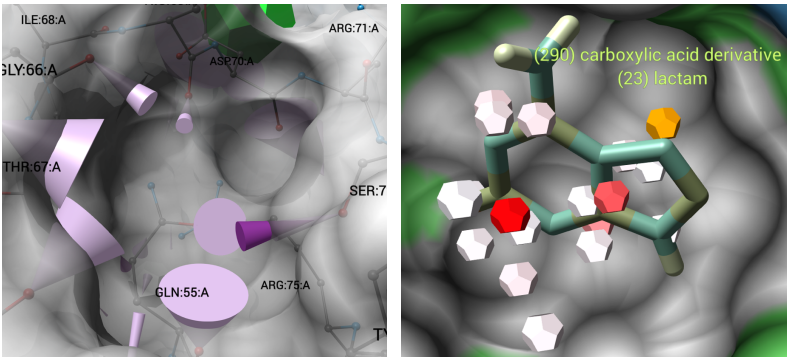
Ligand & Interaction Visualizations When a ligand is selected, for instance, via the *Radial Menu* or the *Docking Overview*, it will be visualized as stick model colored according to the chemical element (Figure 4.26 (a)). In addition, the interactions between the ligand and the protein can be visualized as thin, colored sticks, complementing and corresponding to the *Ligand View* of the

dashboard. This 3D visualization allows for a detailed spatial assessment of the docking pose and the interactions with respect to the protein (R5). If a pose is selected, the interactions shown are limited to the interactions of the current ligand pose. If no pose is selected, the user can opt to see all interactions of all binding poses. Depicting all interactions gives an overview of potential favored interactions and their spatial and directional distribution (Figure 4.26 (b)).

Hydrogen bonds are usually the most prevalent interactions. Visualizing them as sticks can, thus, introduce visual clutter. To reduce clutter while retaining information about the lengths, angles, and the spatial distribution of H-bonds that interact with the same protein atom, they are aggregated to *H-bond cones* (Figure 4.27 (a)). Each cone is oriented towards the average direction of all H-bonds, and its length shows the average length of the H-bonds. The opening angle of the cone encodes the mean spatial distribution of the H-bonds. The cones are colored according to their type, i.e., whether the protein atom is the acceptor (■) or donor (■).

Surface patches corresponding to protein atoms affected by a certain interaction can be highlighted in the interaction-specific color (Figure 4.26 (b)). That allows users to locate preferred regions for certain interactions (R2). Furthermore, the surface coloring can be switched from a mode solely showing the presence of an interaction type to a surface coloring that also reflects the number of interactions counted for the individual protein atoms. The interaction type counts are encoded using a linear color scale (▬). This points to hot-spots with a potential high binding affinity (R2), which is valuable information for protein engineers and ligand designers.

Binding Site and Functional Group Visualization Besides the aforementioned surface coloring by interaction type or the number of interactions, further options were added, allowing users to explore spatial aspects of the docking data. To visualize the estimated binding sites based on the clusters, a transparent rendering mode is offered, whereby the parts of the protein surface that are not close to the currently selected cluster are rendered transparently. That reduces clutter and helps to focus on the binding site (R1/R2). To achieve this, the binding site patches have to be derived from the ligands in the clusters. This is done by a neighbor search for each atom of all ligands using a distance criterion of 1.0 Å to find nearby protein atoms. On this basis, the parts of the surface corresponding to the protein atoms that fulfill this criterion can be determined (gray surface parts in Figure 4.24 (b)).



(a) H-bond Cones **(b)** Functional Groups

Figure 4.27 — (a) The H-bond cones summarize multiple hydrogen bonds between a protein atom and the docked ligands. The cone shows the average direction of the H-bonds and the opening angle distribution. The color indicates whether the protein is the acceptor (light purple) or the donor (dark purple). (b) Clusters of functional groups are visualized as dodecahedra colored by the number of groups, where white corresponds to a few and red to the maximum (). On hover, a text label is shown that indicates the groups contained as well as their number. - © IEEE 2024 [Sch+24]

The ligands within a cluster are also used to find clusters of functional groups previously determined by *Checkmol*. The functional groups are represented as smaller dodecahedrons when compared to the dodecahedrons representing ligand clusters. The dodecahedrons of the functional group cluster are colored by size using a linear color scale (), as shown in Figure 4.27 (b). The clustering has user-adjustable parameters for the minimum number of cluster members and the search radius. The clustering is performed individually for each super-group of functional groups. In the second step, the found functional group clusters are aggregated to avoid intersections of the dodecahedrons if they are too close to each other. They can be accessed by a click, which will show a label with the amounts and functional group types. This will also select the corresponding functional groups in the *Functional Groups View* (Figure 4.18 ③). Additionally, the *Statistics View* is filtered accordingly. While hovering a dodecahedron, it is highlighted in orange and the remaining dodecahedrons are hidden until a selection is made or the hovering ends.

Interaction Type Bar Chart An *Interaction Type Bar Chart* can be displayed for each cluster/binding site. It summarizes the interactions by type. The presented and aggregated information provides the user with an additional cluster-specific overview (R1), supporting the comparison of clusters/binding sites, and allows for identifying binding sites with an interesting composition or profile of interactions (R2). The glyph-like visualization shown in Figure 4.24 (b) consists of two bar charts. The top bars show the number of residues that belong to the binding site and are affected by an interaction. The bottom part shows the area of the binding site that is affected by each interaction. This information is derived from the surface areas of the interacting residues. The space-efficient bidirectional layout allows for a qualitative comparison of the absolute number of interactions versus the relative interaction surface area.

4.4.6 Architecture and Implementation Details

This section provides more details on the implementation of InVADo. As mentioned in the application overview, it is intended to be used in a dual or ultra-widescreen monitor setup. This is necessary to provide sufficient screen space for the large amount and complexity of the data and, consequently, the large number of required views. InVADo was implemented as a two-window application consisting of a dashboard and a 3D visualization. Its client-server architecture is presented in Figure 4.28. The source code of InVADo is publicly available in a GitHub repository.[2]

For the *3D Visualization*, the open-source visualization framework *MegaMol* is used as described in Section 2.6.1. The framework is tailored to visualize large scientific data sets, particularly molecular dynamics data. Due to its modular architecture, it supports the prototyping of new visualizations with low overhead. Together with its various built-in biomolecular visualization features, it forms the basis of InVADo.

The dashboard was implemented as a web app using the D3 and the *Vuetify* framework (see Section 2.6.1 *Web Development Environment*), which is a Vue UI library reducing the effort of web development by providing many pre-build components [Lei+23]. The interactive visualizations are generated using D3, which binds data to DOM/SVG elements and manipulates them based on the visualization data [BOH11].

[2] InVADo source code: `https://github.com/MarcoSchaeferT/InVADo_setup`

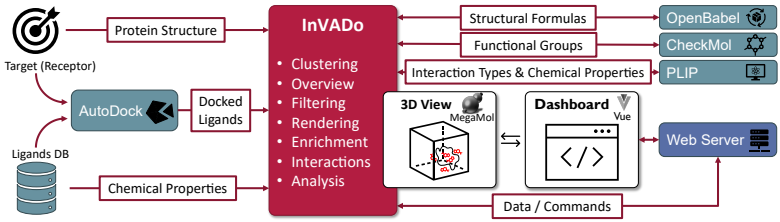


Figure 4.28 — InVADo Architecture: The red rectangle at the center shows InVADo, which has some of its key features listed. Teal rectangles show external tools and data sources. To the left, the input data is listed, which consists primarily of the docking results, as computed by *AutoDock Vina*, the ligands from a database, and the target (or receptor protein) against which the ligands were docked. Additional data to enrich the docking results for a comprehensive visual analysis are provided by *PLIP*, *Checkmol*, and *OpenBabel*. InVADo controls all these external programs and automatically collects their output. All the data is processed, analyzed, and integrated by InVADo to produce an interactive 3D view of the clustered docking results, which is implemented using the *MegaMol* framework. This view is complemented by a highly linked dashboard that is implemented using the Vue library. Communication and data exchange between the two components is established via an additional Python web server. - modified figure © IEEE 2024 [Sch+24]

Data, Preprocessing & External Tools The input data are Protein Data Bank (PDB), Partial Charge (Q), Atom Type (T) (PDBQT) files as they are, for instance, available for ligands from the ZINC database [SI15]. PDBQT is also the output format of *AutoDockTools* utilized for ligand/protein preparation before docking. The format is used to store a set of ligand conformations consisting of atom positions and a docking score (an example is given in the supplemental material [Sch+24]). A more detailed description of the molecule preparation for docking, including *AutoDockTools*, *AutoDock* and the PDBQT format can be found in Section 2.6.2. InVADo generates additional data by calling the external tools shown as turquoise boxes on the right side of Figure 4.28. *Open-Babel* [OBo+11] creates the structural formulas of the ligands used in both the 3D and 2D views and converts the input data into the file format used by *Checkmol*. *Checkmol* [Hai10] determines the functional groups of the ligands, and *PLIP* [Sal+15] calculates the interactions for the protein-ligand complexes that are created by InVADo for each ligand pose.

To achieve high performance, InVADo is mainly written in C++. For smaller tasks related to data download, conversion, and communication with external tools, parallelized Python scripts are used, which are controlled by the C++ application.

GPU-Accelerated Tasks & Rendering InVADo requires neighbor searches for multiple tasks such as different clusterings, determining binding site patches or interacting residues of the protein surface, and the identification of residues adjacent to functional group clusters. Therefore, a CUDA implementation of the fast fixed-radius nearest neighbor search by Hoetzlein [Hoe14] is used. It significantly accelerated the self-implemented version of DBSCAN [Est+96], which is utilized to cluster the ligand poses and functional groups.

For fast rendering, GLSL shaders are used for ray casting spheres, cylinders, and cones [RE05; Gum03]. Additionally, a modified version of the approximate ambient occlusion by Grottel et al. [Gro+12] is applied to improve the spatial perception of the protein structure. Furthermore, a certain CUDA implementation of the Thrust library is used to perform the view-dependent sorting of triangles necessary for a transparent rendering of the protein surface mesh.

4.4.7 Evaluation and Discussion

To evaluate InVADo, a case study was conducted with a docking data set consisting of SARS-CoV-2 spike protein docked with FDA drugs [PKS21]. Additionally, expert feedback was collected from biochemists in structured feedback sessions. In these sessions, the domain experts worked with an academically published data set collected for the analysis of P-glycoprotein inhibitors [Mar+21].

Case Study: SARS-CoV-2 Spike Protein Targeting To create the data for this case study, it was followed the approach of Pande et al. [PKS21], who docked FDA drugs against the SARS-CoV-2 spike protein. The FDA drugs retrieved from ZINC did not include all ligands named in the paper, but six of the nine top-scored were present (Ergotamine, Ponatinib, Yaz, Naldemedine, Conivaptan, Orap). The protein was prepared, and in contrast to Pande et al., the drug data was used without further calculation steps like adding hydrogens or assigning new partial charges (charges already included in ZINC ligands). The drug data were docked with *AutoDock Vina* with a docking setup similar to Pande et al. [PKS21]. Ten different ligand poses were calculated for each of the 2,215 ligands (paper: 1,565). InVADo was set up to cluster the docking results

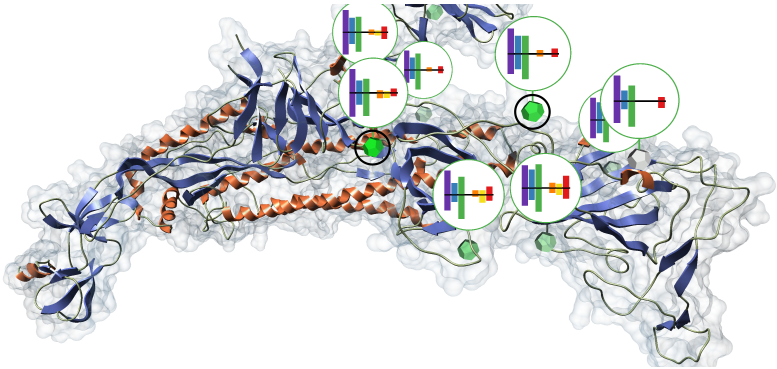


Figure 4.29 — The structure of the SARS-CoV-2 spike protein (chain A only; PDB ID: 7DDN) is presented together with its ligand-docking clusters determined by InVADo (dodecahedrons colored by ligand number). The entire protein is rendered with semi-transparently SES in combination with its cartoon representation. The Interaction Type Bar Charts above the clusters show the interaction type composition of the individual clusters.

of 22,150 ligand binding poses with a minimum cluster size of 200, a search distance of 4.0 Å, and a docking score of -6.0 kcal/mol.

The results, as visualized by the 3D view and the *Docking Overview*, showed that nine clusters with two similar-sized main clusters were found (cf. black circles in Figure 4.29 and cluster IDs 2 and 0 in Figure 4.30). These two possible binding sites, with 1,378 and 1,156 poses docked, are located in the Receptor Binding Domain (RBD) (amino acids: 319-541) as reported by the paper. This result shows the validity of the cluster approach of InVADo.

Using the ZINC-IDs of top-scoring ligands from the publication as a search query for the *Statistics View* revealed that their poses are in an energy range from -8.6 to -7.8 kcal/mol, deviating from the reported range of -8.2 to -6.5 kcal/mol. This deviation can be explained by the missing ligand preparation and the fact that the data used contain only a subset of six of the nine top-scoring ligands found in the publication. That is, InVADo helped to verify that the differences are rather small, showing that the produced results are in line with those of the publication.

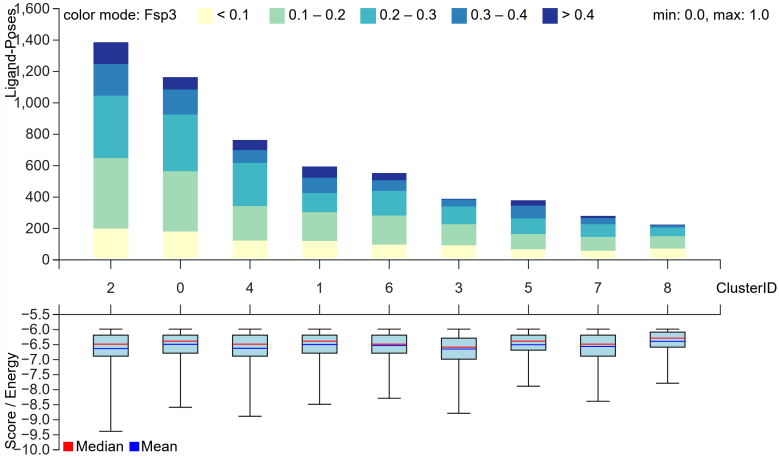


Figure 4.30 — Showing nine clusters found by InVADo in the SARS-CoV-2 data set. The clustering was applied at 22,150 ligand binding poses with a minimum cluster size of 200, a search distance of 4.0 Å, and a minimum docking score of -6.0 kcal/mol. The stacked bar chart is colored according to the drug score fsp^3. The box plot at the bottom shows the docking score distribution of the individual clusters. - modified figure © supplemental material IEEE 2024 [Sch+24]

Using the *Ligand View* table, it was possible to determine that all six mentioned ligands have their top-scoring poses in the same binding site, i.e., the cluster coinciding with the RBD. This also matches the findings of Pande et al. [PKS21].

InVADo offers many additional possibilities for detailed analyses of the docking results. The general summary of interactions presented by the *Interaction Type Bar Chart* showed that H-bonds and hydrophobic interactions are the most prevalent interaction types (cf. Figure 4.29). This is in line with the findings of Pande et al. [PKS21], who described that H-bonds *"play a crucial role in binding affinity, selectivity, and the stability"* of a protein-ligand complex and that the *"hydrophobic interactions also stabilize the complex"*. This can also be observed in the 3D view using the stick representation of the interactions, or by switching to the interaction type-based surface coloring. In addition, the 3D view reveals detailed information about the locations of H-bonds and hydrophilic interactions in the binding site. The analysis of functional group clusters using the *Functional Groups View* showed that heterocyclic compounds

and aromatic compounds are highly over-represented, both of which can be responsible for hydrophobic interactions. Amine- and hydroxyl-compounds that can form H-bonds are also over-represented, although to a lesser degree. That further supports the findings of the *Interaction Type Bar Chart* and indicates a stronger interaction in that binding site, which can be interesting for designing more specific drugs against SARS-CoV-2. To summarize, InVADo guides the users to the known RBD using the detected main clusters, and moreover, it supports an in-depth analysis of the binding conditions, for instance, based on the aggregated interaction types and the functional group clusters.

Expert Feedback: P-Glycoprotein Inhibitors InVADo was also evaluated with five domain experts (E1–5), who were asked to solve four different example tasks in the structured feedback sessions (see supplementary material for detailed task description and questionnaire [Sch+24]). The domain experts are biochemists, and the mentioned tasks were created in consultation with one of them and one additional biochemist. One of the experts (E5), who had helped to define the tasks and requirements, was specifically included to ensure that these were properly implemented in the final application. Due to their help in defining the user tasks, providing the screening data from their paper for the user sessions, and their feedback on the initial design of InVADo, they are coauthors of the publication on which this section is based.

The data used for the study was obtained as part of a previous project on the *Screening of Natural Compounds as P-Glycoprotein Inhibitors against Multidrug Resistance* in the context of cancer treatment [Mar+21]. P-glycoprotein is a pump that ejects foreign substances from cells (see Figure 4.31). It is co-responsible for multidrug resistance in the context of using chemotherapeutics to fight cancer. The aim was to find inhibitors of the P-glycoprotein for the treatment of multidrug resistance so that the chemotherapeutics can work as intended before their efflux.

Five biochemists were asked to test InVADo and try to solve the four tasks. As mentioned before, four of them had not seen InVADo before. A short video[3] was provided, briefly explaining all the features of InVADo and a questionnaire was prepared, including the tasks as well (see supplementary material [Sch+24]). Furthermore, the screen and audio of the test sessions were captured. One of the experts was familiar with the data set and all of them work with molecular docking. Their self-assigned experience level ranges from beginner to proficient.

[3] InVADo video: https://doi.org/10.1109/TVCG.2023.3337642/mm1

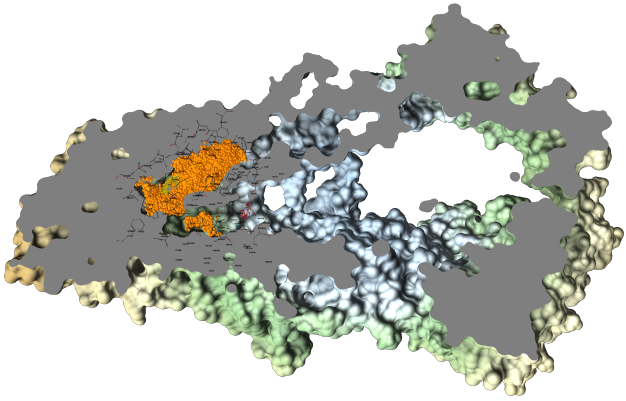


Figure 4.31 — Human P-glycoprotein, a molecular pump for the transport of foreign substances out of the cell [Mar+21] (PDB ID: 4M1M), rendered in InVADo with a clip plane applied (gray). The orange spheres are ligand atoms of one of the clusters found by InVADo. The atoms are surrounded by the ball-and-stick representation of the interacting protein residues. - © IEEE 2024 [Sch+24]

The chosen data showcases the strengths of InVADo, as it contains a large protein, for which all the preferred binding areas are located in the inner part of the protein. Thus, this data is more difficult to handle, and the domain experts have to use the clip plane to see any binding pose or interaction (cf. Figure 4.31).

The data preparation and analysis workflow described in the original article [Mar+21] used a variant of *AutoDock Vina*, followed by the analysis of results by *MS Excel* and custom scripts. To visualize the results, the authors used *PyMOL*, for which they also needed to write further scripts. In contrast to this involved and time-consuming pipeline, the experts noted that InVADo is a well-designed, comprehensive tool with a nice, user-friendly interface (E2, E3, E4, E5). This confirms the claim that InVADo is an intuitive, easily accessible, and comprehensive tool for post-docking analysis.

Overall, all testers were able to solve the four tasks and the feedback was very positive. The collected feedback also contained helpful critiques and avenues for future development. The experts were asked to rate the following points on a five-point Likert scale ranging from *strongly dislike / disagree* to *strongly like / agree*: appearance, user interface design, application structure, offered visualizations, and interactivity. All categories reached an average of *strongly*

	User				
	1	2	3	4	5
general appearance	5	5	4	5	5
user interface design	5	5	4	5	4
application structure	4	5	5	4	4
offered visualizations	4	4	5	5	5
interactivity	5	4	5	4	5

	User				
	1	2	3	4	5
intuitive & user-friendly	3	4	5	5	4

1	strongly dislike
2	dislike
3	neutral
4	like
5	strongly like

1	strongly disagree
2	disagree
3	undecided
4	agree
5	strongly agree

Table 4.6 — Expert Feedback: The heatmap-like colored tables display the questionnaire results of the structured feedback sessions, which were conducted with five domain experts who analyzed the P-glycoprotein inhibitors data [Mar+21]. The legend shows the color encoding, ranging from 1 to 5: strongly dislike/disagree to strongly like/agree . Questionnaire available in supplementary material [Sch+24]. - modified table © supplemental material IEEE 2024 [Sch+24]

like except *application structure*, which received an overall rating of *like* (see Table 4.6). The experts were also asked if they would see a benefit of using InVADo compared to their current workflow and tools. On average, all of them *strongly agreed* that they would use and recommend InVADo for their future work. Only E1 noted that InVADo's application profile does not fit in with their currently used docking pipeline but added that they reckon it would be a *"wonderful tool for [...] virtual screening"*.

A general agreement among the experts was that the tool is initially overwhelming, as it offers *"a huge amount of options"* (E1). Specifically, they explained that the tool size makes it a bit hard to navigate if the options were not studied before, but *"there are other tools that are harder to learn"*. Furthermore, the expert stated that they realized that it is a general limitation of analysis programs to get all the information you want while simultaneously keeping it simple. They added that they needed a bit of explanation to solve all tasks in the intended way. The testing time was not limited, and on average, it took ∼70 minutes to explain the tool in more detail, explore any feature of InVADo, and solve the tasks. Nevertheless, the tool was rated as intuitive and "very user-friendly" (E4) with an average of *agree*. The domain experts said that the tool is very well-designed from the ligand perspective (E1, E3, E5).

As a future extension, they suggested also adding further features from the protein perspective, e.g., the possibility to mark unwanted residues and automatically discard ligands interacting with these residues (E2). In addition, they were also interested in seeing the individual contribution of the interaction types (E5). This fits with the observation that the biochemists not only used the various overview features but also often started to make a detailed analysis of a single ligand binding pose. They wanted to comprehend the presented interactions from the shown molecular structure of the ligand and the protein. This indicates that InVADo is also suitable for the analysis of single ligand docking (R5). The fact that the domain experts reported that they would use InVADo also for pre- and post-optimization analysis further underpins it.

The experts liked the deep integration between the 3D view and the various panels. They described it as a convenient and comprehensive 3D view of ligands, clusters, and their functional groups, providing all the valuable information. They agreed that it allows getting a quick overview of how the binding works and mentioned that InVADo can be used to summarize the characteristics of binding with a "*very nice interface*" (E2), which allows arranging the different compounds. In this context, they also rated the offered tabular views as extremely useful and highlighted that they like the high number of features. Regarding the question of how InVADo fits into the existing pipeline, they mentioned that it would replace some steps of their pipeline, such as *MS Excel* and *PyMOL* (E3). Moreover, they usually have to do a lot of manual scripting to extract the important information in *PyMOL*, and they added that the layout of InVADo is very simple compared to this process. The biochemists also mentioned that they liked the surface coloring by interaction count, the segmented heatmap, and the cone representation of H-bonds, which was a completely new representation to them. It can help to see whether a ligand binds more precisely to the binding site than others.

They especially liked that the tool is also intuitive from a chemistry perspective, thus fitting their mental model. The functional group clusters were specifically mentioned because this feature allows for *"pharmacophore mapping"* (E5), i.e., "*mapping the protein region based on the functional groups*", which they rated as very useful for drug design. Among other things, the possibility of rendering multiple binding poses of the same ligand was rated as very good, as it allows for directly seeing the preferential binding poses of a ligand.

4.5 Summary and Conclusion

Parts of this section have been published in:

- M. Schäfer and M. Krone. "A Massively Parallel CUDA Algorithm to Compute and Visualize the Solvent Excluded Surface for Dynamic Molecular Data." EG Workshop on Molecular Graphics and Visual Analysis of Molecular Data, 2019. DOI: 10.2312/molva.20191094
- K. Schatz, J. J. Franco-Moreno, M. Schäfer, A. S. Rose, V. Ferrario, J. Pleiss, P.-P. Vázquez, T. Ertl, and M. Krone. "Visual Analysis of Large-Scale Protein-Ligand Interaction Data." *Comput. Graph. Forum*. 2021, pp. 394–408. DOI: 10.1111/cgf.14386
- M. Schäfer, N. Brich, J. Byška, S. M. Marques, D. Bednář, P. Thiel, B. Kozlíková, and M. Krone. "InVADo: Interactive Visual Analysis of Molecular Docking Data." *IEEE Trans. Visual. Comput. Graphics*. 2024, pp. 1984–1997. DOI: 10.1109/TVCG.2023.3337642

This chapter focused on the visualization and analysis of data from the field of computational chemistry. Therefore, different types of data were used, including molecular trajectories obtained from MD simulations and virtual screening results produced by molecular docking. The three discussed approaches contribute to the investigation of protein-ligand interactions, whereby it was further moved towards a more ligand-centered perspective. This continues the perspective change from the preceding Chapter 3 (cf. Figure 1.1), which started with a protein-centered perspective about a deep learning-based approach for studying all structural levels of proteins. In contrast, this chapter has concentrated more on ligand paths along the protein surface and the interaction types involved in such transport processes or during non-covalent binding in the context of molecular docking.

The first two approaches discussed in this chapter deal with dynamic MD simulation data using different strategies. The first one uses a massively parallelized CUDA algorithm for an interactive and direct visualization of the MD data. The second method follows the idea of deriving new data and properties by data aggregation, which are mapped onto the protein surface that is coupled with a special and interactive sequence diagram (CLISD). Finally, InVADo, as a post-docking analysis tool, is the most ligand-centered approach with the possibility of providing a comprehensive overview and opportunities for a

detailed analysis of molecular docking results. This section summarizes the three approaches and presents conclusions within the context of visualizing and investigating protein-ligand interactions using computational chemistry data.

The first presented approach is about a complex molecular surface computation method. It has been shown that the new approach represents a fast algorithm to compute the SES by utilizing the tremendous parallelization potential of a GPU (cf. Section 4.2). It became apparent that the calculation time depends on the spatial extents and structural features, such as cavities and tunnels that create void spaces, and, consequently, on the number of atoms contributing to the SES. However, on average, the new algorithm has overall linear behavior in computation time and memory consumption in contrast to previous approaches and can render large molecular data sets interactively using a single consumer GPU. Thereby, the visualization of the SES is analytically correct and pixel-precise. This is paired with an adjustable probe radius to simulate different ligands and a coloring of the protein surface that illustrates physico-chemical properties. The adjustable probe allows identifying tunnels and cavities, and the surface coloring helps to understand interactions, since physico-chemical properties, critically influence protein-ligand interactions, besides spatial features. To conclude, the approach supports the user in understanding the dynamic behavior of molecules and their interactions in MD simulations, e.g., contributing to elucidate transport processes and docking events, making it valuable for biochemical and pharmaceutical research.

The second approach discussed in Section 4.3 also shows a direct visualization of MD data, but instead of visualizing the raw data, it follows a data aggregation approach. In summary, a visualization system was presented, facilitating the visual analysis of large protein-ligand interaction data from entire MD simulation trajectory ensembles. Since visualization tools typically concentrate on the movement of single ligands, which can merely provide a hint of the overall system behavior, this tool provides a holistic approach. It uses a preprocessing pipeline to take an ensemble of ligands into account focusing on the general system behavior. The challenge of dealing with millions of simulation time steps was addressed by entirely decoupling the aggregation and preprocessing from the visualization. That allows for a constant low visualization memory footprint independent of the underlying data of up to several terabytes.

The final visualization data is presented by a web-based visual analysis application consisting of multiple linked 2D and 3D views. Via brushing and linking, it is possible to identify amino acids that interact with ligands in the novel abstract sequence diagram (CLISD). A SES representation of the protein

enables users to analyze the spatial structure of the depicted protein, its cavities, or landing spots. The interactive visualization is designed explicitly for the exploratory analysis of protein-ligand interactions from MD simulation data, but as shown, it can also be used to analyze docking results. The system was developed in tight collaboration with domain experts from biology. It was evaluated, and its usability was shown in two case studies, including a docking and a MD simulation data set. The domain experts especially liked the fact that they did not need to use additional tools for the analysis of the data and could share the visualization with other colleagues by simply sharing a web link. The visual analysis application provides a meaningful overview of the entire simulation and has a good scalability through its preprocessing pipeline. It draws attention to the relevant parts, allowing scientists to get new insights into dynamic molecule behavior and protein-ligand interactions. The system can contribute to a comprehensive understanding of ligand paths and involved interactions, playing a crucial role in the investigation of the processes of where and how a ligand approaches an active site. This makes it useful for researchers in the field of drug discovery and structural biology.

As described before, the work in Section 4.3 that deals with MD data and uses a data aggregation approach is also partly suitable for analyzing docking data. However, the subsequent work in this chapter as discussed in Section 4.4 deals more comprehensively and intensively with the analysis of docking data. To summarize, InVADo was developed since commonly used docking software provides no or very basic evaluation possibilities, need much scripting and external molecular viewers, which are not designed for an efficient docking data analysis. InVADo is a visual analysis application that facilitates a comprehensive analysis of docking results by supporting and leading the users through the evaluation process. As a post-docking analysis tool, it structures and visualizes the data in multiple ways, e.g., by filtering and spatial clustering. It also enriches the data with post-docking results, including physico-chemical properties, functional groups, and protein-ligand interactions (e.g., H-bonds, π-stacks). The combined data is visualized by various interactive tables and multiple highly linked 3D and 2D visualizations. This means interactions and selections in one view affect all other views, including the 3D visualization.

The dashboard-like visualization application is structured into views of various detail levels, starting from a cluster overview to more granular, detailed views. More precisely, an entry point is the *Docking Overview* as a summary of the found clusters. This is complemented by the *Statistics View*, which also can present cluster-specific information together with the *Functional Groups View*. The most

granular hierarchy level is the *Ligand View* providing specific information about the individual binding poses. The *Segmented Heatmap* bridges all different hierarchy detail levels. In this way, InVADo enables the user to make well-founded decisions on the further processed docking results.

The application was designed in close collaboration with domain experts by first collecting common tasks and inferring requirements, followed by a refinement of the application in an iterative process. The system is designed to be user-friendly and to provide a comprehensive overview of the docking results.

In an exemplary case study with structured feedback sessions, domain experts confirmed that InVADo facilitates and accelerates the analysis workflow. They rated it as a convenient, comprehensive, and feature-rich tool, especially useful for virtual screening and in the context of pre- and post-optimization analysis. They indicated that they want to use it in the future to replace the tedious steps in their analysis pipeline.

CHAPTER 5

Discussion and Outlook

This chapter evaluates the methods presented in the previous chapters regarding their suitability and implications for scientific research and industry. As mentioned in the introduction, the most important concept for the methods developed for this thesis was the interactive and user-adjustable visual exploration and analysis of complex biomolecular data for studying protein-ligand interactions. The aim was to support forming a comprehensive understanding of protein-ligand interactions by illuminating the topic from different perspectives. Therefore, diverse methods were used to process, enrich, and visualize various protein-ligand data types (cf. Figure 1.1). The developed approaches allow for an extensive, detailed, and interactive visualization of protein-ligand interactions. They depict the ligand's and protein's structure, dynamics, and various interaction types, thus contributing to gaining new insights into the complex interplay of proteins and ligands. This can support successful developments in biotechnology, e.g., improved biofuel production, and in systems biology, finding new signal transduction ways as well as gene or enzyme regulation possibilities. Another important field of application is assisting in solving crucial and challenging problems in drug discovery, for instance, overcoming the growing problem of antibiotic resistance, finding new immune therapeutics or personalized vaccines against cancer, and further developments.

As protein-ligand interactions are essential for critical physico-chemical processes of life, it is crucial to understand them and to evaluate the results of

tools that serve the necessary data for studying those complex biomolecular relationships. To achieve this, the thesis first approached protein-ligand interactions from a protein-centered perspective, focusing on protein classification and analysis. In this context, the used data include protein structure files (PDB) and abstract planar protein surface representations. These data were processed by methods comprising CNNs and projection methods to create Molecular Surface Maps for image-based clustering. The evaluation of these works showed that they are able to classify proteins according to their fold class or enzyme reaction class and that they can find functional similar proteins. Both methods, discussed in this context, are based on deriving and learning protein features and can be used to compare and cluster proteins. They contribute to comprehending protein-ligand interactions by building a solid foundation for deriving common features of interactions and investigating which conditions must be fulfilled from the protein perspective to form them.

The methods discussed in Chapter 3 are more protein-centered and allow for the analysis and study of protein structure, surface area interactions, and related effects, before it was moved to a ligand-centered perspective in the second major chapter of the thesis. Chapter 4 focused increasingly on a detailed analysis of interactions, molecular dynamics, docking, and specific ligand properties. In this context, data from the field of computational chemistry were used, including MD simulation trajectories and molecular docking results (virtual screening). The new approaches range from the interactive protein surface visualization (SES), using GPU-accelerated algorithms, over the analysis of MD simulations, applying visual analysis enabled by dashboard-based visualization and aggregation approaches. That was followed by the visualization of virtual screening results using a dashboard-like presentation and spatial clustering. The evaluation of these approaches showed that they are suitable for the visual analysis of protein-ligand simulation results, providing a comprehensive overview as well as enabling a detailed inspection of the results. This is useful for evaluating and verifying the results' correctness and can support discovering unexpected features that may lead to new hypotheses for further research. These visual approaches reveal characteristics of the simulated biomolecules that are not readily apparent to analysts when examining the raw data.

As mentioned before, the first method of the thesis is a solely protein-centered approach. The method presented in Section 3.2 *Structure Learning Using Neural Networks* aims to learn on 3D protein structures using a neural network architecture and connects this knowledge with protein function and fold classes. The CNN-based approach introduced new polling operations and a new convolu-

tional operator to capture all structural levels of proteins, including primary, secondary, tertiary, as well as quaternary structures of entire protein complexes. The network was trained on a scale of 1,000s to 10,000s proteins, depending on the classification task. It was shown that the approach outperformed state-of-the-art methods in classifying proteins according to their fold classes and enzyme reaction classes.

Learning interrelations between protein structure and function can help to infer how interactions between protein and ligands work and may go beyond the known interaction types or identify unknown interaction patterns. An example is to find sets of interaction types that often can be observed, e.g., for a specific class of enzymes. It can also lead to findings about common structural elements with typical physico-chemical arrangements, such as protein domains. The tremendous pace in the evolution of machine learning and the fast development of new network architectures thus may enable a precise de novo prediction of protein functions. An example of this rapid progress is the recently published tool *AlphaFold* 3 [Abr+24], which was already able to predict protein structures with high accuracy in its previous versions and now is even capable of predicting protein-ligand interactions with an accuracy that outperforms state-of-the-art physical-based molecular docking tools such as *AutoDock*.

A method using a CNN as well was presented in Section 3.3 *Protein Surface Map Similarity Clustering*. It focuses on an image-based clustering of Molecular Surface Maps and continues approaching protein-ligand interactions from a protein-centered perspective. This approach is based on a scale from tens to thousands of proteins and concentrates entirely on the protein surface by creating Molecular Surface Maps colored based on topology or physico-chemical properties. These 2D surface maps enable a hierarchical classification based on surface similarity. That is, unlike the previously discussed network architecture, this method considers not only structural but also physico-chemical attributes of the protein surface, enhancing its ability to discern similarities among proteins regarding their function. The evaluation has demonstrated that the maps can be used to compare and cluster proteins successfully. The comparison is based on descriptive feature vectors computed using a neural network called *MobilenetV2*.

For analyzing complete protein ensembles, the application provides 3D views of the surface and corresponding surface maps linked to a dendrogram representing the clustering results. Thus, it enables a detailed analysis and assessment of the clustering quality of individual protein pairs. The user can choose between different map types and can adjust the clustering threshold on the fly for tuning the results. Since surface similarity often correlates with functional

similarity, the new approach can be used to identify functionally analogous proteins. That supports the inference of common features or physico-chemical properties of protein-ligand interactions responsible for certain protein functions. The method can also help researchers gain insights into ligand transport processes and associated interactions, which is especially of interest to structural biologists and pharmaceutical researchers. This can be achieved by utilizing it to investigate the protein-ligand interactions of a single protein by mapping interactions of a ligand onto the occlusion-free as well as view-independent surface representation. These properties can be the aggregated results of a MD simulation, showing general ligand paths, e.g., how a substrate approaches the active site of an enzyme, or presenting where and which interactions were formed during the simulation. Moreover, this method has the potential to analyze previously uncharacterized proteins, facilitating hypothesis generation regarding their functions.

Biological life is primarily determined by a constantly ongoing process of exchange of molecules, digesting, synthesis, and conformational changes. Everything in a cell depends on dynamic equilibrium. This means it is essential to investigate the dynamic behavior of protein-ligand interaction to enable a holistic understanding and to get new insights, e.g., into transport processes or the forming of binding sites. The objective is also to analyze the temporal behavior of proteins and ligands, assess their binding affinity, identify the types of interactions involved, and comprehend the transport mechanisms occurring across the surface.

To this end, the first method of Chapter 4 is a fast and highly parallel algorithm exploiting the massive computing power of a GPU. It presents a new approach to interactively visualize the complex and analytical correct molecular surface named SES. In contrast to the previous method, the scale is a single protein whose dynamic behavior over time was predicted using Molecular Dynamics simulation. The method followed a new approach of calculating all intersection points of intersecting protein atom sphere triplets, which were used to derive the SES surface patches (cf. Figure 4.1). The comparison against a similar algorithm showed that the VRAM consumption is much lower, allowing for visualization and investigation of larger molecules than before, which is of interest as it becomes feasible to simulate ever larger molecules. However, while the new method is not the fastest available, it still achieves interactive frame rates for proteins on single consumer graphic cards, e.g., up to $\sim$ 20k atoms (1080 GTX).

Moreover, the method allows the mapping of physico-chemical properties (e.g., flexibility, charge, hydrophobicity) onto the surface, as it is also possible with various other tools [SD20; Pet+04; Seh+21]. That makes it an even more valuable visualization tool for the visual analysis and exploration of complex, dynamic molecular data, as it is important for biochemists and structural biologists. Illustrating physico-chemical properties that influence protein-ligand interactions can significantly enhance comprehension and, consequently, facilitate knowledge discovery. The user can also choose between coloring schemes dependent on the chemical element, amino acid type, or secondary structure. Additionally, it is possible to adjust the probe radius interactively since the SES is determined in real time. That makes it instrumental in investigating protein tunnels for ligands of different sizes or to get an impression of the spatial accessibility of specific surface areas, such as binding sites.

However, as mentioned, the method is not the fastest method available compared to Krone et al. [KGE11]. Thus, a necessary action is to improve performance in the future. A possible approach is determining all atoms contributing to the SES during the simulation and to discard all not-involved atoms. That would lead to significantly lower computational costs, especially for globular proteins. An additional computational improvement consists of a hybrid surface approach. Areas of interest, such as tunnels and known binding sites are represented using the exact and analytical correct SES, and the remaining surface parts are approximated using, for instance, the Gaussian molecular surface [Kro+12]. That is similar to the approach of Parulek et al. [Par+14], leading to accelerated computation times [SK19]. Besides this, an additional idea is to test whether the rendering speed would benefit from defining the SES using a signed distance field [OF03] in combination with ray marching or sphere tracing [Har96] for rendering.

A further promising direction to increase the render performance is to exploit new hardware and software features of GPUs, for example, Ray Tracing (RT) cores or Deep Learning Super Sampling (DLSS) 3. The rendering performance could be improved using the RT cores. The elemental operation of glyph ray casting shaders, such as used for the described rendering of the SES, is finding the intersection of a ray with an object in the scene (SES surface patches), where RT cores are exactly designed for. A fast RT core approach can be combined with efficient image-denoising algorithms that allow an efficient rendering. A ray is not cast for each pixel of an image, but only the pixels required to derive the remaining pixels are computed, i.e., the final image. Accelerated glyph ray casting by simultaneously avoiding casting a ray for every pixel can

significantly improve the rendering speed, giving more scientific computing and visualization possibilities. Another optimization would be the usage of modern image techniques for interactive 3D environments similar to DLSS 3, which utilizes a neural network executed on specialized GPU cores called tensor cores [Men23]. Besides other key features, DLSS 3 allows generating intermediate images with much lower computational cost than rendering the entire image, thus achieving significantly higher frame rates leading to more interactive visualizations. Motion vectors are used to generate intermediate images. Because of this, the dynamic data of MD simulation would be suitable for this approach. MD trajectories already store the direction vectors, not as motion vectors in image space, but they provide the future position of each atom for each time step. That may allow skipping each second time step when rendering the MD trajectories and replacing it by inserting the skipped time step via frame generation of techniques like DLSS 3. This approach could also insert new time steps, increasing the time resolution of the trajectory. It has to be proven if using such a frame generation technique would be beneficial for the visualization of a dynamic SES also evaluating the sufficient analytical correctness of such generated images.

The fast SES computation method was followed by an approach also dealing with the dynamic molecular data of MD simulations. In contrast, the new approach is not visualizing the raw data directly but uses an aggregation approach. The method described in Section 4.3 *Visual Analysis of Large-Scale Protein-Ligand Interactions* summarizes the simulation data and represents a visualization system, facilitating the visual analysis of protein-ligand interactions from entire MD simulation trajectory ensembles. It focuses on the overall system behavior by deriving the movement of multiple ligands along the protein surface, collecting certain aggregated values, comprising different interactions formed during the simulation. The challenge of dealing with millions of simulation time steps was addressed by entirely decoupling the aggregation and preprocessing from the visualization. That allows for a constant low visualization memory footprint independent of the underlying data of up to several terabytes.

The data is represented using an interactive web dashboard consisting of multiple instances of a novel abstract sequence diagram CLISD that is linked with a 3D representation of the protein surface (SES). The 3D representation allows for an analysis of cavities or landing spots. Via brushing and linking the user can identify amino acids that interact with ligands, whereby the application draws attention to most interacting amino acids by emphasizing them in the CLISD or highlighting them on the protein surface. Additionally, the user can choose

different protein coloring modes depending on aggregated features, such as the number of contacts or bonded atoms. This information is important for scientists of systems biology and protein engineers, who are especially interested in those values providing information about contacts between proteins and ligands. The system allows understanding the dynamic behavior of a small molecular system consisting of a protein and one to hundreds of ligands. It can contribute to a comprehensive understanding of general ligand paths and the protein-ligand interactions, which is crucial in investigating how a ligand approaches an active site. Furthermore, the application makes the comprehension of simulations more transparent and accessible, which can also help to improve physico-chemical properties of the enzymes. That can include substrate affinity, accessibility, and catalytic rates that can be changed by getting insights about certain amino acids, which can be targeted for mutation.

Based on the previously discussed methods, a future direction would be a comprehensive analysis tool for MD simulation trajectories, which is built on the two methods mentioned above (Sections 4.2 and 4.3) and the work of Byška et al. [Byš+19]. This combination would allow addressing Keim's well-known visual analytics mantra: *analyze first – show the important – zoom, filter and analyze further – details on demand* [Kei+08]. The analysis and visualization of the aggregation approach for MD simulations would provide an overview of the whole simulation, focusing on essential interactions and the ligand overall movements (cf. Section 4.3). This overview would be complemented by the essential event detection method of Byška et al. [Byš+19] that represents a further *analysis and filtering*. The spatial occurrences of important events could also be encoded on the protein surface. The application can play parts of the simulation using the cartoon representation of the protein, which can be supplemented by the fast, exact, and analytically correct SES representation method (cf. Section 4.2). This would allow studying and analyzing both, on the one hand, the overall system behavior of ligands interacting with a protein and, on the other hand, a detailed analysis of relevant periods of time from the simulation results. This enables the user to directly identify areas of interest in the spatial and temporal dimensions of MD simulation trajectories. The application would also be able to zoom and filter and provide the possibility for a detailed analysis. The analysis could be even further improved by enriching the detailed analysis of specific time steps of interest with possibly missing protein-ligand interactions calculated by a tool like *PLIP* developed by Adasme et al. [Ada+21].

The interactive visualization of the mentioned aggregation approach is specifically designed for the exploratory analysis of protein-ligand interactions from MD simulation data, but it was also shown that it can be used to analyze docking results as well. However, the discussed approach has some limitations and is unsuitable for a comprehensive analysis of virtual screening results, which is solved by the visual analysis tool InVADo. With this application, the thesis moved further to the most ligand-centered perspective, taking into account-specific interaction types as well as certain physico-chemical properties of ligands. InVADo, as discussed in Section 4.4 *Visual Analysis of Docking Data*, also belongs to the field of computational chemistry and deals with the analysis of 10s to ∼100,000s ligands docked against one target protein. It is designed as a dashboard-like two-screen application to address the lag of free available comprehensive evaluation software for docking results. To achieve this, InVADo aggregates the docking results by spatial clustering the ligand poses and enriches the data further by post-docking analysis, including the determination of protein-ligand interactions and functional groups.

The final data is visualized by multiple strongly linked 2D and 3D views, enabling users to explore and analyze the docking data interactively. That is provided by an overview of all possible binding sites and further interdependent views. The views have an increasing level of detail, where the most detailed one presents certain physico-chemical properties and functional groups of one specific ligand binding pose. The users can adjust various clustering and visualization parameters, allowing them to tune the results to their needs. The possibility of using multiple molecular representations enables the user to analyze different levels of protein structure and ligand properties. The representations comprise the simple ball-and-stick model, the complex molecular surface SES, and the opportunity to visualize the protein structure using the cartoon representation. The application was evaluated by domain experts who rated the enriched data as helpful for drug development and that they were able to derive insights from it for protein or ligand engineering [Sch+24]. InVADo was described as a convenient, comprehensive, and feature-rich tool that is especially useful for virtual screening, especially in the context of pre- and post-optimization analysis. The application contributes to gaining insights into docking results, investigating protein-ligand interaction, enabling the derivation of common features of binding sites or preferred ligands, and enabling the extraction of drug candidates (lead compounds). It is also designed to lead the user to interesting hot-spots such as areas with high ligand numbers (potential binding sites), which serve as an entry point for further analysis. That is

complemented by highlighting functional group clusters that are areas with a high density of certain functional groups, allowing to derive protein and ligand design possibilities. Furthermore, InVADo enables the evaluation of overall good binding ligands or ligand specificity, enabled by an interactive segmented heatmap. To summarize the evaluation with domain scientists, the tool was found to be valuable for researchers in the field of drug discovery and structural biology [Sch+24]. InVADo serves a comprehensive docking overview and contributes to understanding general protein-ligand interactions. This is also enabled by its many analysis possibilities and a detailed investigation of single ligands that interact with the target protein and functional groups, causing specific interaction types. InVADo allows the investigation of conditions that must be fulfilled to form interactions and also enables the study of specific interaction types, which offers the possibility to derive new hypotheses for further research.

A recently published work shows the importance of molecular docking in actual research by describing an exponential increase in publications using the keyword *molecular docking* [AC24], reaching almost 13,000 publications in 2023. They also discussed that docking has been shown to be a crucial tool in the field of drug discovery. Thus, it is important to spend more research on protein-ligand visualization, especially in the direction of molecular docking, to enable a comprehensive evaluation and analysis of virtual screening results. This will contribute to new findings and a more profound understanding of protein-ligand interactions, may lead to accelerated drug discovery, and may help address the problem of antibiotic resistance, as mentioned at the beginning of the thesis. The actual inflationary use in livestock farming is increasingly leading to issues in managing infections since the amount of working antibiotics continues to fall. A well-known example of this phenomenon is Methicillin-resistant *Staphylococcus aureus* (MRSA), often causing post-operational complications in hospitals [Sil+23]. It is, therefore, all the more important that tools are available to contribute to a comprehensive understanding of protein-ligand interactions and to facilitate the analysis and evaluation of molecular docking results.

Based on this context, a further direction to enhance InVADo is to add the possibility to perform a re-scoring for a specific ligand or a set of ligand binding poses [Sch+24]. This feature is desired by domain scientists and would improve evaluation and informed decision-making. The re-scoring would be part of a more complex integration for guided lead optimization. This would be further complemented by methods allowing better focusing on the part of interest. This can be done by setting a cluster center as look-at-point of the virtual camera and

implementing depth of field combined with smooth camera transition when switching to other binding sites or clusters. Moreover, adding more shapes or glyphs for different interaction forces, such as the presented H-bond cones, e.g., pillars, would be a further option. This would ease the differentiation between various interaction types and could even increase the rendering performance when numerous interactions of the same type are encoded via a summarizing glyph instead of thousands of colored sticks. Thus, a current limitation is that the interactivity of the prototypical application can decrease if many ligand binding poses are contained in a docking experiment ($\sim$5,000). This means that the frame rate can be low if the overview is displayed and all interaction types are selected for visualization.

Furthermore, to have broader coverage of the drug development process and to make InVADo an even more comprehensive application while additionally lowering the amount of manual scripting, the complete integration of the docking itself would be a possible next step. In order to implement this, the tool *Dockey* of Du et al. [Du+23] could be utilized. It provides a GUI for performing molecular docking and virtual screening based on variants of *AutoDock*. The integration of *Dockey* would add molecular sanitization, molecular preparation, and docking execution, after which the results are further processed by InVADo to perform the post-docking analysis. That would make InVADo a complete tool for virtual screening and post-docking analysis, thus further facilitating the accessibility of molecular docking and the evaluation of the results.

Visual analysis tools such as InVADo are designed to give a comprehensive and interactive overview of the data. They also provide search and filter options and often enable a detailed analysis. In contrast, machine learning methods are designed to learn patterns and to recognize data relationships. They classify or predict data based on the learned patterns. The development of machine learning methods has reached a fast pace. They are powerful for certain tasks and have enabled several breakthroughs. A thesis-related example is the recently published version 3 of *AlphaFold* [Abr+24] solving the protein folding problem by accurately predicting the protein structure better as its predecessors and now also allowing to predict protein-ligand complexes by even outperforming state-of-the-art docking tools as *AutoDock*. However, machine learning methods are not always suitable for tasks that require a deeper understanding of the data and the ability to explore and analyze it interactively. An essential point of visual analysis tools is that they are designed to be used by domain experts who may not have a background in machine learning. This means, they also can provide scientists with a facilitated access to machine learning-based results.

The data produced, and the patterns learned by machine learning methods, such as *AlphaFold*, need to be visualized and interpreted by researchers and domain experts to gain insights and validate the results. This is crucial since machine learning methods are incapable of developing new scientific approaches. They are not designed for exploratory analysis, thus not allowing the identification of unknown phenomena. Typically, human expertise is required for interpretation and hypothesis generation, which is facilitated by visual analysis. For example, machine learning models can suggest amino acids for a targeted mutation to improve the binding affinity between a protein and ligand. However, it cannot automatically determine if this modification will increase the efficacy of a drug. More complex tasks, such as increasing the catalytic rate of an enzyme or improving its stability, are performed by targeted mutations by protein engineers and ligand designers. One form of stability improvement could involve enabling an enzyme's catalytic activity for a broader pH range, which is valuable for biotechnology. Another example is the identification of allosteric sites distinct from the enzyme's active site, which can modulate its activity and serve as a drug target [HNZ17]. While machine learning can support finding an allosteric site, human analysis is required for its evaluation and the impact of a predicted inhibitor on the pathway or the disease mechanism. These tasks require a more profound understanding of the data and the ability to explore and analyze it interactively, making visual analysis crucial for those processes.

A visual inspection is necessary for evaluating the results of machine learning methods. For *AlphaFold 3* the question arises, which features and relations were learned, and can they be translated to interaction types or conditions that must be met for their formation? This could improve existing definitions of interaction types, or maybe enable discovering new forms of protein-ligand interactions. The visual analysis of interactions will allow gaining new insights into their complex interplay, contributing to answering current research questions. That includes elucidating incompletely comprehended physico-chemical processes of life. An example is the investigation of the human gut microbiome, which has been an emerging field of research for years. It becomes increasingly clear what a significant influence the microbiome has on the human body. The metabolites (ligands) produced by the gut bacteria can interact with human proteins. For instance, they can have a substantial impact on the gene expression [ND21], can influence immune responses [Yoo+20], and are involved in metabolic diseases [San+19]. Combining machine learning methods with visual analysis approaches will be essential in the future, enabling researchers to gain a comprehensive understanding of the data and enable more informed decisions.

Moreover, it would be beneficial to facilitate the creation of 3D visualizations of protein-ligand interactions and to increase spatial or depth perception. An example where a more realistic rendering has been proven to increase image comprehension is the work of Lindemann et al. [LR11]. This publication evaluated the influence of more advanced illumination models, which go far beyond traditional Phong shading. It investigated the impact on the comprehension of direct volume renderings of medical data. In the case of InVADo, increasing the depth perception would allow scientists to get a more refined impression of how a ligand is located in a binding site and how and where interactions are formed [Sch+24]. To accomplish that, a possibility is using actual 3D rendering software like *Blender 4* or game engines like *Unreal Engine 5* (e.g., see [Bok+22]) already providing complex and realistic illumination models. This would enable the creation of even more immersive visualizations of protein-ligand interactions, for instance, by using ray tracing for realistic lighting and shadowing effects. It would be beneficial for creating detailed images of protein-ligand binding poses, which can be used for scientific communication, storytelling, or educational purposes. An example of this is the work by Brady Johnston [Joh+24], where a Python-based plugin for Blender called *Molecular Nodes* was created, allowing to load and visualize data of proteins, DNA, RNA and MD trajectories as well. The images and videos produced by it are impressive and can be used to illustrate the results of molecular docking studies or, in general, of protein-ligand interaction in a visually appealing way. It can help to communicate scientific research results to a broader audience and make them more accessible to non-experts as well.

Following on from the point of scientific communication, future developments in visualizing protein-ligand interactions should ease the sharing of scientific results. An example that underpins this is a comment of a domain scientist who participated in the case study of the MD simulation aggregation approach (see Section 4.3). It was especially liked that the scientists can share the visualization by simply sending a link, without needing additional software. InVADo uses an easily shareable web-based dashboard as well, but this is also linked with 3D desktop application. This means the 3D visualizations of protein-ligand interactions should be moved to the web-based side to allow an easy sharing of scientific results. That is especially important for interdisciplinary research, where researchers from different fields need to collaborate and share their results. Everyone can easily access an entirely web-based visualization, which would help them to share their findings. Nowadays, web browsers make it possible to explore interactive and even computationally complex visualizations.

Years ago, only WebGL was available, which has several limitations and does not provide all shader stages of traditional OpenGL (see Section 2.1.2). For instance, geometry and compute shaders are not available, where the latter enables to use GPUs for high parallelization of custom algorithms (cf. Section 2.2). But with the upcoming WebGPU API, a new standard for web-based graphics rendering, these limitations are overcome [Ken22]. WebGPU is designed to be a low-level API that provides more direct access to the graphics hardware, similar to Vulkan or DirectX 12, thus achieving high performance. That will allow creating GPU-accelerated web applications, executing algorithms in compute shaders, and running complex visualizations entirely in the web browser. Additionally, a web browser has increasingly developed into a kind of independent operating system that offers mostly the same requirements and features on different platforms. That is, web-based approaches are making scientific visualization tools more platform-independent and will also facilitate the setup of such tools, which are crucial points when evaluating the accessibility and usability of scientific software. These facts contribute to a more facilitated sharing of research and enable scientists to collaborate more effectively.

Another essential point for future work in the interactive visualization of protein-ligand interactions is to consider uncertainty and to raise a general awareness of uncertainty in protein-ligand data. Imaging techniques as crystallographic methods for investigating protein structure have inaccuracies, e.g., crystal imperfections and limitations in resolution, as factors of uncertainty. In addition, computational methods use specific models to produce a significant amount of protein-ligand data. These models are always based on approximations and simplifications of the real world, which implies certain amounts of uncertainty in the results of these methods. This uncertainty can be caused by the approximations of the force fields used in MD simulations, for instance, or by approximations of the docking algorithms and simplification in machine learning models. A future task is to investigate which types and sources of uncertainty are present for protein-ligand data, and to develop new methods for visualizing them while simultaneously avoiding visual clutter.

As presented, there are many possible optimizations and future directions. Still, as an overall conclusion, it can be stated that the discussed research contributions provide a solid basis and a broad set of tools for conducting further research from different points of view. In particular, the change of perspective from protein-centered to ligand-centered in this thesis has proven to be advantageous. This change of perspective made it possible first to provide a type of overview of protein-ligand interactions and then to approach a detailed

analysis of the interactions. That is, seen from an atomistic level first *macroscopic effects* and features were observed and extracted. Then, the protein-centered perspective was followed by a subsequent continuous change to a ligand-centered view. That means a detailed analysis was increasingly carried out, considering the dynamic molecular behavior and specific local interaction types from which the macroscopic effects result. In this context, applied visual and exploratory data analysis is essential for achieving a more holistic understanding of protein-ligand interactions. The developed approaches enable a visually comprehensive and interactive investigation of interactions. They will contribute to gaining new insights into how the molecular machinery of life works and get a more sophisticated understanding of the underlying mechanisms crucial for systems biology, medical chemistry, and pharmaceutical research.

Bibliography

[Aba+16] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. "TensorFlow: A System for Large-Scale Machine Learning." *12th USENIX Symp. Oper. Syst. Des. Implement. OSDI 16*. 2016, pp. 265–283. (Accessed 08/17/2023) (see p. 29)

[Abr+15] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess, and E. Lindahl. "GROMACS: High Performance Molecular Simulations through Multi-Level Parallelism from Laptops to Supercomputers." *SoftwareX*. 2015, pp. 19–25 (see pp. 43, 104, 128, 132, 142)

[Abr+24] J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A. J. Ballard, J. Bambrick, et al. "Accurate Structure Prediction of Biomolecular Interactions with AlphaFold 3." *Nature*. 2024 (see pp. 189, 196)

[AC24] C. Aguiar and I. Camps. *Molecular Docking in Drug Discovery: Techniques, Applications, and Advancements*. 2024. Pre-published (see p. 195)

[Ada+21] M. F. Adasme, K. L. Linnemann, S. N. Bolz, F. Kaiser, S. Salentin, V. J. Haupt, and M. Schroeder. "PLIP 2021: Expanding the Scope of the Protein-Ligand Interaction Profiler to DNA and RNA." *Nucleic Acids Res.* 2021, W530–W534 (see p. 193)

[AK01] V. Agrawal and R. K. Kishan. "Functional Evolution of Two Subtly Different (Similar) Folds." *BMC Struct. Biol.* 2001 (see p. 48)

[Alb+22] B. Alberts, R. Heald, A. Johnson, D. Morgan, M. Raff, K. Roberts, P. Walter, J. H. Wilson, and T. Hunt. *Molecular Biology of the Cell*. Seventh edition, international student edition. W.W. Norton & Company, 2022 (see p. 22)

[Alb18] C. Albon. *Machine Learning with Python Cookbook: Practical Solutions from Preprocessing to Deep Learning*. First edition. O'Reilly Media, 2018 (see p. 28)

[ALC16] N. Alharbi, R. S. Laramee, and M. Chavent. "MolPathFinder: Interactive Multi-Dimensional Path Filtering of Molecular Dynamics Simulation Data." *Conf Comput. Graph. Vis. Comput.* 2016, pp. 9–16 (see p. 126)

[Ale+09] P. A. Alexander, Y. He, Y. Chen, J. Orban, and P. N. Bryan. "A Minimal Sequence Code for Switching Protein Structure and Function." *Proc. Natl. Acad. Sci.* 2009 (see pp. 48, 49)

[Alh+19] N. Alharbi, M. Krone, M. Chavent, and R. S. Laramee. "Hybrid Visualization of Protein-Lipid and Protein-Protein Interaction." *Proc. EG VCBM*. The Eurographics Association, 2019 (see p. 126)

[Alk+09] I. Alkorta, F. Blanco, J. Elguero, J. A. Dobado, S. M. Ferrer, and I. Vidal. "Carbon ··· Carbon Weak Interactions." *J. Phys. Chem. A*. 2009, pp. 8387–8393 (see p. 133)

[All+15] W. J. Allen, T. E. Balius, S. Mukherjee, S. R. Brozell, D. T. Moustakas, P. T. Lang, D. A. Case, I. D. Kuntz, and R. C. Rizzo. "DOCK 6: Impact of New Features and Current Docking Performance." *J. Comput. Chem.* 2015, p. 1132 (see p. 102)

[All+19] E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi, and G. M. Church. "Unified Rational Protein Engineering with Sequence-Based Deep Representation Learning." *Nat. Methods*. 2019 (see pp. 53, 66, 67)

[Alt+90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. "Basic Local Alignment Search Tool." *Journal of Molecular Biology*. 1990, pp. 403–410 (see pp. 46, 47, 53)

[Alt+97] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. "Gapped BLAST and PSI-BLAST: A New Generation of Protein Database Search Programs." *Nucleic Acids Res.* 1997, pp. 3389–3402 (see pp. 46, 47)

[AM15] E. Asgari and M. Mofrad. "Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics." *PLoS ONE*. 2015 (see p. 53)

[Ami+18] A. Amidi, S. Amidi, D. Vlachakis, V. Megalooikonomou, N. Paragios, and E. I. Zacharaki. "EnzyNet: Enzyme Classification Using 3D Convolutional Neural Networks on Spatial Representation." *PeerJ*. 2018, e4750 (see p. 53)

[Aze19] W. F. Azevedo, ed. *Docking Screens for Drug Discovery*. Methods in Molecular Biology. Springer New York, 2019 (see pp. 101, 103)

[Bai+18] A. Baierl, A. Theorell, U. Mackfeld, P. Marquardt, F. Hoffmann, S. Moers, K. Nöh, P. C. F. Buchholz, J. Pleiss, and M. Pohl. "Towards a Mechanistic Understanding of Factors Controlling the Stereoselectivity of Transketolase." *ChemCatChem*. 2018, pp. 2601–2611 (see p. 91)

[Bai+21] B. Bai, R. Zou, H. C. S. Chan, H. Li, and S. Yuan. "MolADI: A Web Server for Automatic Analysis of Protein–Small Molecule Dynamic Interactions." *Molecules*. 2021, p. 4625 (see p. 152)

[Bal+20] F. Baldassarre, D. M. Hurtado, A. Elofsson, and H. Azizpour. "GraphQA: Protein Model Quality Assessment Using Graph Convolutional Networks." *Bioinformatics*. 2020 (see pp. 54, 66, 67)

[Ban+19] B. Banaganapalli, F. A. Morad, M. Khan, C. S. Kumar, R. Elango, Z. Awan, and N. A. Shaik. "Molecular Docking." *Essentials of Bioinformatics, Volume I: Understanding Bioinformatics: Genes to Proteins*. Ed. by N. A. Shaik, K. R. Hakeem, B. Banaganapalli, and R. Elango. Springer International Publishing, 2019, pp. 335–353 (see pp. 102, 103)

[BB19] T. Bepler and B. Berger. "Learning Protein Sequence Embeddings Using Information from Structure." *Int. Conf. Learn. Represent.* 2019 (see pp. 53, 66, 67)

[Bel+24] H. Belghit, M. Spivak, M. Dauchez, M. Baaden, and J. Jonquet-Prevoteau. "From Complex Data to Clear Insights: Visualizing Molecular Dynamics Trajectories." *Front. Bioinform.* 2024, p. 1356659 (see p. 100)

[Ber+00] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. "The Protein Data Bank." *Nucleic Acids Res.* 2000, pp. 235–242 (see pp. 1, 16, 32, 40, 46, 66, 116)

[Ber+15] J. M. Berg, J. L. Tymoczko, G. J. Gatto, and L. Stryer. *Biochemistry*. Eighth edition. W.H. Freeman & Company, a Macmillan Education Imprint, 2015 (see pp. 5, 22, 26, 35)

[Bid+08] K. Bidmon, S. Grottel, F. Bös, J. Pleiss, and T. Ertl. "Visual Abstractions of Solvent Pathlines near Protein Cavities." *Comput. Graph. Forum* 27.3. 2008, pp. 935–942 (see p. 126)

[BJ09] N. J. Burgoyne and R. M. Jackson. "Predicting Protein Function from Surface Properties." *From Protein Structure to Function with Bioinformatics*. Ed. by D. J. Rigden. Springer Netherlands, 2009, pp. 167–186 (see p. 70)

[BKS10] C. Bissantz, B. Kuhn, and M. Stahl. "A Medicinal Chemist's Guide to Molecular Interactions." *J. Med. Chem.* 2010, pp. 5061–5084 (see p. 26)

[BM13] M. Brehmer and T. Munzner. "A Multi-Level Typology of Abstract Visualization Tasks." *IEEE Trans. Vis. Comput. Graph.* 2013, pp. 2376–2385 (see p. 156)

[BOH11] M. Bostock, V. Ogievetsky, and J. Heer. "D^3 Data-Driven Documents." *IEEE Trans. Vis. Comput. Graph.* 2011, pp. 2301–2309 (see pp. 40, 140, 173)

[Bok+22] M. Bok, M. Schäfer, N. Brich, K. Schreiner, V. Fäßler, M. Keckeisen, O. Kohlbacher, and M. Krone. "Comparative Visual Analysis of Molecular Dynamics." *Eurographics VCBM Posters*. 2022 (see pp. 149, 198)

[BP66] L. E. Baum and T. Petrie. "Statistical Inference for Probabilistic Functions of Finite State Markov Chains." *Ann. Math. Stat.* 37.6. 1966, pp. 1554–1563. (Accessed 08/22/2023) (see pp. 46, 48)

[BR14] J. L. Blanco and P. K. Rai. *Nanoflann: A C++ Header-Only Fork of FLANN, a Library for Nearest Neighbor (NN) with KD-trees*. 2014 (see p. 139)

[Bre01] L. Breiman. "Random Forests." *Machine Learning*. 2001, pp. 5–32 (see p. 29)

[Bro+09] B. R. Brooks, C. L. Brooks III, A. D. Mackerell Jr., L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, et al. "CHARMM: The Biomolecular Simulation Program." *J. Comput. Chem.* 2009, pp. 1545–1614 (see p. 104)

[Bro+92] K. W. Brodlie, J. R. Gallop, C. D. Osland, L. A. Carpenter, R. J. Hubbold, P. Quarendon, R. A. Earnshaw, and A. M. Mumford, eds. *Scientific Visualization*. Springer Berlin Heidelberg, 1992 (see p. 15)

[BS16] F. Biedermann and H.-J. Schneider. "Experimental Binding Energies in Supramolecular Complexes." *Chem. Rev.* 2016, pp. 5216–5300 (see pp. 5, 26)

[Bur18] S. V. Burger. *Introduction to Machine Learning with R: Rigorous Mathematical Analysis*. First edition. O'Reilly Media, 2018 (see pp. 26–29)

[Byš+15] J. Byška, A. Jurčik, M. E. Gröller, I. Viola, and B. Kozlíková. "MoleCollar and Tunnel Heat Map Visualizations for Conveying Spatio-Temporo-Chemical Properties across and along Protein Voids." *Comput. Graph. Forum* 3.34. 2015, pp. 1–10 (see p. 125)

[Byš+16] J. Byška, M. Le Muzic, M. E. Gröller, I. Viola, and B. Kozlíková. "AnimoAminoMiner: Exploration of Protein Tunnels and Their Properties in Molecular Dynamics." *IEEE Trans. Vis. Comput. Graphics* 22.1. 2016, pp. 747–756 (see p. 125)

[Byš+19] J. Byška, T. Trautner, S. Marques, J. Damborský, B. Kozlíková, and M. Waldner. "Analysis of Long Molecular Dynamics Simulations Using Interactive Focus+Context Visualization." *Comput. Graph. Forum*. 2019, pp. 441–453 (see pp. 126, 152, 193)

[Cas+05] D. A. Case, T. E. Cheatham, T. Darden, H. Gohlke, R. Luo, K. M. Merz, A. Onufriev, C. Simmerling, B. Wang, and R. J. Woods. "The Amber Biomolecular Simulation Programs." *J Comput Chem*. 2005, pp. 1668–1688 (see p. 104)

[Cha+17a] M. Chaker-Margot, J. Barandun, M. Hunziker, and S. Klinge. "Architecture of the Yeast Small Subunit Processome." *Science*. 2017 (see p. 120)

[Cha+17b] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017, pp. 77–85 (see p. 56)

[Che+18] J. Chen, M. Guo, X. Wang, and B. Liu. "A Comprehensive Review and Comparison of Different Computational Methods for Protein Remote Homology Detection." *Brief. Bioinform.* 2018, pp. 231–244 (see pp. 46–48, 52)

[Che24] C. C. G. Chemical Computing Group. *Molecular Operating Environment (MOE)*. 2024. `https://www.chemcomp.com/Research-Citing_MOE.htm` (accessed 01/15/2024) (see p. 102)

[CM21] N. A. Church and J. L. McKillip. "Antibiotic Resistance Crisis: Challenges and Imperatives." *Biologia*. 2021, pp. 1535–1550 (see p. 2)

[Con83] M. L. Connolly. "Analytical Molecular Surface Calculation." *J. Appl. Crystallogr.* 1983, pp. 548–558 (see pp. 36, 109, 110, 116)

[CP06] T. K. Chaudhuri and S. Paul. "Protein-Misfolding Diseases and Chaperone-Based Therapeutic Approaches." *FEBS Journal*. 2006, pp. 1331–1349 (see p. 24)

[CS10] R. G. Coleman and K. A. Sharp. "Protein Pockets: Inventory, Shape, and Comparison." *J. Chem. Inf. Model*. 2010, pp. 589–603 (see p. 25)

[CW20] C. Cai and Y. Wang. "A Note on Over-Smoothing for Graph Neural Networks." *ICML 2020 Graph Represent. Learn. Workshop*. 2020 (see p. 54)

[CWD07] S. J. Costelloe, J. M. Ward, and P. A. Dalby. "Evolutionary Analysis of the TPP-dependent Enzyme Family." *J. Mol. Evol.* 2007, pp. 36–49 (see p. 91)

[Dal+92] A. Dalby, J. G. Nourse, W. D. Hounshell, A. K. I. Gushurst, D. L. Grier, B. A. Leland, and J. Laufer. "Description of Several Chemical Structure File Formats Used by Computer Programs Developed at Molecular Design Limited." *J. Chem. Inf. Comput. Sci.* 1992, pp. 244–255 (see p. 41)

[Dan+19] J. M. Dana, A. Gutmanas, N. Tyagi, G. Qi, C. O'Donovan, M. Martin, and S. Velankar. "SIFTS: Updated Structure Integration with Function, Taxonomy and Sequences Resource Allows 40-Fold Increase in Coverage of Structure-Based Annotations for Proteins." *Nucleic Acids Res.* 2019, pp. D482–D489 (see p. 66)

[DB02] Daan Frenkel and Berend Smit. *Understanding Molecular Simulation : From Algorithms to Applications*. Vol. 2nd ed. Computational Science Series. Academic Press, 2002 (see pp. 101, 104)

[Der+18] G. Derevyanko, S. Grudinin, Y. Bengio, and G. Lamoureux. "Deep Convolutional Networks for Quality Assessment of Protein Folds." *Bioinformatics* 34.23. 2018, pp. 4046–53 (see pp. 53, 66, 67)

[Des+15] J. Desaphy, G. Bret, D. Rognan, and E. Kellenberger. "Sc-PDB: A 3D-database of Ligandable Binding Sites—10 Years On." *Nucleic Acids Res* 43.D1. 2015, pp. 399–404 (see pp. 46, 62)

[DeV+09] R. DeVane, W. Shinoda, P. B. Moore, and M. L. Klein. "Transferable Coarse Grain Nonbonded Interaction Model for Amino Acids." *J. Chem. Theory Comput.* 2009 (see p. 59)

[dGRO23] documentation GROMACS development team. *XTC File Format*. 2023. https://manual.gromacs.org/documentation/current/reference-manual/file-formats.html#xtc (accessed 06/29/2023) (see p. 43)

[Die19] F. Diehl. "Edge Contraction Pooling for Graph Neural Networks." Version 1. *arXiv*. 2019 (see pp. 66, 67)

[Dos+05] Z. Dosztanyi, V. Csizmok, P. Tompa, and I. Simon. "IUPred: Web Server for the Prediction of Intrinsically Unstructured Regions of Proteins Based on Estimated Energy Content." *Bioinformatics*. 2005, pp. 3433–3434 (see p. 32)

[DR00] A. Drummond and A. G. Rodrigo. "Reconstructing Genealogies of Serial Samples Under the Assumption of a Molecular Clock Using Serial-Sample UPGMA." *Mol. Biol. Evol.* 2000, pp. 1807–1815 (see p. 29)

[dslea23] documentation scikit-learn. *2.3 Clustering*. 2023. https://scikit-learn.org/stable/modules/clustering.html (accessed 05/31/2023) (see p. 31)

[Du+23] L. Du, C. Geng, Q. Zeng, T. Huang, J. Tang, Y. Chu, and K. Zhao. "Dockey: A Modern Integrated Tool for Large-Scale Molecular Docking and Virtual Screening." *Brief. Bioinform.* 2023, bbad047 (see p. 196)

[Dug06] R. G. Duggleby. "Domain Relationships in Thiamine Diphosphate - Dependent Enzymes." *Acc. Chem. Res.* 2006, pp. 550–557 (see p. 91)

[Dur+19] D. Duran, P. Hermosilla, T. Ropinski, B. Kozlikova, A. Vinacua, and P.-P. Vazquez. "Visualization of Large Molecular Trajectories." *IEEE Trans. Visual. Comput. Graphics*. 2019, pp. 987–996 (see pp. 125, 126, 152)

[Dur+98] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. 1st ed. Cambridge University Press, 1998 (see p. 30)

[Eas+17] P. Eastman, J. Swails, J. D. Chodera, R. T. McGibbon, Y. Zhao, K. A. Beauchamp, L.-P. Wang, A. C. Simmonett, M. P. Harrigan, C. D. Stern, et al. "OpenMM 7: Rapid Development of High Performance Algorithms for Molecular Dynamics." *PLOS Computational Biology*. 2017, e1005659 (see p. 104)

[EB06] R. C. Edgar and S. Batzoglou. "Multiple Sequence Alignment." *Current Opinion in Structural Biology*. 2006, pp. 368–373 (see pp. 32, 46)

[Ebe+21] J. Eberhardt, D. Santos-Martins, A. F. Tillack, and S. Forli. "AutoDock Vina 1.2.0: New Docking Methods, Expanded Force Field, and Python Bindings." *J. Chem. Inf. Model.* 2021, pp. 3891–3898 (see pp. 41, 102, 103)

[EG18] R. Egan and F. Gibou. "Fast and Scalable Algorithms for Constructing Solvent-Excluded Surfaces of Large Biomolecules." *J. Comput. Phys.* 2018, pp. 91–120 (see p. 107)

[El-+19] S. El-Gebali, J. Mistry, A. Bateman, S. R. Eddy, A. Luciani, S. C. Potter, M. Qureshi, L. J. Richardson, G. A. Salazar, A. Smart, et al. "The Pfam Protein Families Database in 2019." *Nucleic Acids Res.* 2019 (see p. 53)

[Eln+21] A. Elnaggar, M. Heinzinger, C. Dallago, G. Rehawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger, et al. "ProtTrans: Towards Cracking the Language of Life's Code through Self-Supervised Learning." *bioRxiv*. 2021 (see pp. 66, 67)

[Est+96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." *Proc. Second Int. Conf. Knowl. Discov. Data Min.* KDD'96. AAAI Press, 1996, pp. 226–231 (see pp. 29, 63, 159, 175)

[Fäh+17] R. Fährrolfes, S. Bietz, F. Flachsenberg, A. Meyder, E. Nittinger, T. Otto, A. Volkamer, and M. Rarey. "ProteinsPlus: A Web Portal for Structure Analysis of Macromolecules." *Nucleic Acids Res.* 2017, W337–W343 (see pp. 3, 153)

[FCE11] R. D. Finn, J. Clements, and S. R. Eddy. "HMMER Web Server: Interactive Sequence Similarity Searching." *Nucleic Acids Res*. 2011, W29–37 (see pp. 46, 48)

[FDB16] S. Farkona, E. P. Diamandis, and I. M. Blasutig. "Cancer Immunotherapy: The Beginning of the End of Cancer?" *BMC Med*. 2016, p. 73 (see p. 2)

[Fer+15] L. G. Ferreira, R. N. Dos Santos, G. Oliva, and A. D. Andricopulo. "Molecular Docking and Structure-Based Drug Design Strategies." *Molecules*. 2015, pp. 13384–13421 (see p. 156)

[Flu00] J. Flusser. "On the Independence of Rotation Moment Invariants." *Pattern Recognit.* 33.9. 2000, pp. 1405–1410 (see pp. 74, 79)

[Fly72] M. J. Flynn. "Some Computer Organizations and Their Effectiveness." *IEEE Trans. Comput.* 1972, pp. 948–960 (see p. 108)

[FM67] W. M. Fitch and E. Margoliash. "Construction of Phylogenetic Trees." *Science* 155.3760. 1967, pp. 279–284 (see pp. 29, 74)

[For+16] S. Forli, R. Huey, M. E. Pique, M. F. Sanner, D. S. Goodsell, and A. J. Olson. "Computational Protein–Ligand Docking and Virtual Drug Screening with the AutoDock Suite." *Nat Protoc*. 2016, pp. 905–919 (see p. 42)

[Fou+17] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur. "Protein Interface Prediction Using Graph Convolutional Networks." *NIPS*. 2017 (see p. 54)

[FT20] H. Fukuda and K. Tomii. "DeepECA: An End-to-End Learning Framework for Protein Contact Prediction from a Multiple Sequence Alignment." *BMC Bioinformatics*. 2020, p. 10 (see p. 49)

[Fur+17] K. Furmanová, M. Jarešová, J. Byška, A. Jurčík, J. Parulek, H. Hauser, and B. Kozlíková. "Interactive Exploration of Ligand Transportation through Protein Tunnels." *BMC Bioinformatics*. 2017, p. 22 (see pp. 125, 152)

[Fur+19] K. Furmanová, A. Jurčík, B. Kozlíková, H. Hauser, and J. Byška. "Multiscale Visual Drilldown for the Analysis of Large Ensembles of Multi-Body Protein Complexes." *IEEE Trans. Vis. Comput. Graphics* 26. 2019, pp. 843–852 (see p. 127)

[Fur+20a] K. Furmanova, S. Gratzl, H. Stitz, T. Zichner, M. Jaresova, A. Lex, and M. Streit. "Taggle: Combining Overview and Details in Tabular Data Visualizations." *Information Visualization*. 2020, pp. 114–136 (see p. 153)

[Fur+20b] K. Furmanová, O. Vávra, B. Kozlíková, J. Damborský, V. Vonásek, D. Bednář, and J. Byška. "DockVis: Visual Analysis of Molecular Docking Trajectories." *Comput. Graph. Forum*. 2020, pp. 452–464 (see p. 152)

[GB78] J. Greer and B. L. Bush. "Macromolecular Shape and Surface Maps by Solvent Exclusion." *Proc. Natl. Acad. Sci.* 75. 1978, pp. 303–307 (see p. 36)

[Gen+17] S. Genheden, A. Reymer, P. Saenz-Méndez, and L. A. Eriksson. "Computational Chemistry and Molecular Modelling Basics. "2017 (see p. 104)

[GJ19] H. Gao and S. Ji. "Graph U-Nets." *ICML*. 2019, pp. 2083–2092 (see p. 54)

[GKJ19] J. G. Greener, S. M. Kandathil, and D. T. Jones. "Deep Learning Extends de Novo Protein Modelling Coverage of Genomes Using Iteratively Predicted Structural Constraints." *Nat. Commun.* 2019, p. 3977 (see p. 49)

[GKK14] M. Gütlein, A. Karwath, and S. Kramer. "CheS-Mapper 2.0 for Visual Validation of (Q)SAR Models." *J. Cheminformatics*. 2014, p. 41 (see p. 153)

[Gli+19] V. Gligorijevic, P. D. Renfrew, T. Kosciolek, J. K. Leman, K. Cho, T. Vatanen, D. Berenberg, B. Taylor, I. M. Fisk, R. J. Xavier, et al. "Structure-Based Function Prediction Using Graph Convolutional Networks." *bioRxiv*. 2019 (see pp. 53, 54, 66, 67)

[GM78] J. Gasteiger and M. Marsili. "A New Model for Calculating Atomic Charges in Molecules." *Tetrahedron Letters*. 1978, pp. 3181–3184 (see p. 42)

[Goo+21] D. S. Goodsell, M. F. Sanner, A. J. Olson, and S. Forli. "The AutoDock Suite at 30." *Protein Sci.* 2021, pp. 31–43 (see p. 103)

[Gow+16] R. Gowers, M. Linke, J. Barnoud, T. Reddy, M. Melo, S. Seyler, J. Domański, D. Dotson, S. Buchoux, I. Kenney, et al. "MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations." Python in Science Conference. 2016, pp. 98–105 (see pp. 41, 128)

[Gra+19] P. Gralka, M. Becher, M. Braun, F. Frieß, C. Müller, T. Rau, K. Schatz, C. Schulz, M. Krone, G. Reina, et al. "MegaMol - a Comprehensive Prototyping Framework for Visualizations." *Eur. Phys. J. Spec. Top.* 2019, pp. 1817–1829 (see pp. 7, 37–39)

[Gro+12] S. Grottel, M. Krone, K. Scharnowski, and T. Ertl. "Object-Space Ambient Occlusion for Molecular Dynamics." *Proc. 2012 IEEE Pac. Vis. Symp.* PACIFICVIS '12. IEEE Computer Society, 2012, pp. 209–216 (see pp. 167, 175)

[Gro+15] S. Grottel, M. Krone, C. Muller, G. Reina, and T. Ertl. "MegaMol—A Prototyping Framework for Particle-Based Visualization." *IEEE Trans. Visual. Comput. Graphics*. 2015, pp. 201–214 (see pp. 7, 33, 37–39, 121, 125)

[Gum03] S. Gumhold. "Splatting Illuminated Ellipsoids with Depth Correction." *Int. Symp. Vis. Model. Vis.* 2003 (see pp. 34, 106, 175)

[GWH18] F. Groh, P. Wieschollek, and Hendrik P. A. Lensch. "Flex - Convolution (Million-Scale Point-Cloud Learning beyond Grid-Worlds)." *Asian Conf. Comput. Vis. ACCV*. 2018 (see p. 57)

[HAC18] J. Hou, B. Adhikari, and J. Cheng. "DeepSF: Deep Convolutional Neural Network for Mapping Protein Sequences to Folds." *Proc. 2018 ACM Int. Conf. Bioinforma. Comput. Biol. Health Inform.* 2018 (see pp. 53, 54, 65, 67)

[Hai10] N. Haider. "Functionality Pattern Matching as an Efficient Complementary Structure/Reaction Search Tool: An Open-Source Approach." *Molecules*. 2010, pp. 5079–5092 (see pp. 154, 159, 174)

[Hal+04] T. A. Halgren, R. B. Murphy, R. A. Friesner, H. S. Beard, L. L. Frye, W. T. Pollard, and J. L. Banks. "Glide: A New Approach for Rapid, Accurate Docking and Scoring. 2. Enrichment Factors in Database Screening." *J. Med. Chem.* 2004, pp. 1750–1759 (see p. 102)

[Har96] J. C. Hart. "Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces." *Vis. Comput.* 1996, pp. 527–545 (see p. 191)

[HD18] S. A. Hollingsworth and R. O. Dror. "Molecular Dynamics Simulation for All." *Neuron*. 2018, pp. 1129–1143 (see pp. 101, 103, 104)

[HDS96] W. Humphrey, A. Dalke, and K. Schulten. "VMD: Visual Molecular Dynamics." *J Mol Graph*. 1996, pp. 33–38, 27–28 (see pp. 3, 37, 128, 151)

[He+16] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition." *2016 IEEE Conf. Comput. Vis. Pattern Recognit. CVPR*. 2016, pp. 770–778 (see pp. 60, 61)

[Hea+98] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf. "Support Vector Machines." *IEEE Intell. Syst. Their Appl.* 1998, pp. 18–28 (see p. 29)

[Her+17a] P. Hermosilla, J. Estrada, V. Guallar, T. Ropinski, A. Vinacua, and P.-P. Vazquez. "Physics-Based Visual Characterization of Molecular Interaction Forces." *IEEE Trans. Vis. Comput. Graph.* 2017, pp. 731–740 (see pp. 126, 152)

[Her+17b] P. Hermosilla, M. Krone, V. Guallar, P.-P. Vázquez, À. Vinacua, and T. Ropinski. "Interactive GPU-based Generation of Solvent-Excluded Surfaces." *Vis. Comput.* 2017, pp. 869–881 (see p. 107)

[Her+18] P. Hermosilla, T. Ritschel, P.-P. Vazquez, A. Vinacua, and T. Ropinski. "Monte Carlo Convolution for Learning on Non-Uniformly Sampled Point Clouds." *ACM Trans Graph Proc SIGGRAPH Asia* 37.6. 2018 (see pp. 57, 58)

[Her+21] P. Hermosilla, M. Schäfer, M. Lang, G. Fackelmann, P.-P. Vázquez, B. Kozlikova, M. Krone, T. Ritschel, and T. Ropinski. "Intrinsic-Extrinsic Convolution and Pooling for Learning on 3D Protein Structures." International Conference on Learning Representations. 2021 (see pp. 7, 11, 48, 52, 55, 57, 58, 61, 66–68, 95)

[Hes+08] B. Hess, C. Kutzner, D. Van Der Spoel, and E. Lindahl. "GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation." *J. Chem. Theory Comput.* 2008, pp. 435–447 (see p. 43)

[HF12] K. Hasegawa and K. Funatsu. "New Description of Protein–Ligand Interactions Using a Spherical Self-Organizing Map." *Bioorg. Med. Chem.* 20.18. 2012, pp. 5410–5415 (see p. 72)

[HH92] S. Henikoff and J. G. Henikoff. "Amino Acid Substitution Matrices from Protein Blocks." *Proc. Natl. Acad. Sci.* 1992, pp. 10915–10919 (see p. 47)

[HK14] J. Hass and P. Koehl. "How Round Is a Protein? Exploring Protein Structures for Globularity Using Conformal Mapping." *Front. Biosci.* 2014, p. 26 (see p. 72)

[HM90] R. B. Haber and D. A. McNabb. "Visualization Idioms: A Conceptual Model for Scientific Visualization Systems." *Vis. Sci. Comput.* 74. 1990, p. 93 (see p. 16)

[HNZ17] W. Huang, R. Nussinov, and J. Zhang. "Computational Tools for Allosteric Drug Discovery: Site Identification and Focus Library Design." *Computational Protein Design*. Ed. by I. Samish. Springer, 2017, pp. 439–446 (see p. 197)

[Hoe14] R. C. Hoetzlein. *Fast Fixed-Radius Nearest Neighbors: Interactive Million-Particle Fluids*. 2014 (see pp. 111, 112, 175)

[Hol+23] L. Holm, A. Laiho, P. Törönen, and M. Salgado. "DALI Shines a Light on Remote Homologs: One Hundred Discoveries." *Protein Sci.* 2023, e4519 (see pp. 46, 49)

[Hol20] L. Holm. "Using Dali for Protein Structure Comparison." *Structural Bioinformatics: Methods and Protocols*. Ed. by Z. Gáspári. Methods in Molecular Biology. Springer US, 2020, pp. 29–42 (see p. 49)

[How01] P. W. Howe. "Principal Components Analysis of Protein Structure Ensembles Calculated Using NMR Data." *J. Biomol. NMR*. 2001, pp. 61–70 (see p. 149)

[HS12] J. Heer and B. Shneiderman. "Interactive Dynamics for Visual Analysis." *ACM Queue* 10.2. 2012, pp. 30–55 (see p. 125)

[Hu62] M.-K. Hu. "Visual Pattern Recognition by Moment Invariants." *IRE Trans. Inf. Theory* 8.2. 1962, pp. 179–187 (see pp. 74, 79)

[Hue+07] R. Huey, G. M. Morris, A. J. Olson, and D. S. Goodsell. "A Semiempirical Free Energy Force Field with Charge-Based Desolvation." *J. Comput. Chem.* 2007, pp. 1145–1152 (see p. 41)

[HYL17] W. Hamilton, Z. Ying, and J. Leskovec. "Inductive Representation Learning on Large Graphs." *NIPS*. 2017, pp. 1024–34 (see p. 54)

[Ibr+18] M. Ibrahim, P. Wickenhäuser, P. Rautek, G. Reina, and M. Hadwiger. "Screen-Space Normal Distribution Function Caching for Consistent Multi-Resolution Rendering of Large Particle Data." *IEEE Trans. Vis. Comput. Graphics* 24.1. 2018, pp. 944–953 (see p. 125)

[Ing+19] J. Ingraham, V. Garg, R. Barzilay, and T. Jaakkola. "Generative Models for Graph-Based Protein Design." *Adv. Neural Inf. Process. Syst.* Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019 (see p. 54)

[Irw+20] J. J. Irwin, K. G. Tang, J. Young, C. Dandarchuluun, B. R. Wong, M. Khurelbaatar, Y. S. Moroz, J. Mayfield, and R. A. Sayle. "ZINC20—A Free Ultralarge-Scale Chemical Database for Ligand Discovery." *J. Chem. Inf. Model.* 2020, pp. 6065–6073 (see pp. 1, 46)

[Jan+19] A. P. A. Janssen, S. H. Grimm, R. H. M. Wijdeven, E. B. Lenselink, J. Neefjes, C. A. A. van Boeckel, G. J. P. van Westen, and M. van der Stelt. "Drug Discovery Maps, a Machine Learning Model That Visualizes and Predicts Kinome–Inhibitor Interaction Landscapes." *J. Chem. Inf. Model.* 2019, pp. 1221–1229 (see p. 153)

[Jef97] G. A. Jeffrey. *An Introduction to Hydrogen Bonding*. Topics in Physical Chemistry. Oxford University Press, 1997 (see p. 132)

[Jim+17] J. Jiménez, S. Doerr, G. Martínez-Rosell, and A. S. Rose. "DeepSite: Protein-Binding Site Predictor Using 3D-convolutional Neural Networks." *Bioinformatics* 33.19. 2017, pp. 3036–3042 (see p. 54)

[Jin+09] J. Jin, X. Xie, C. Chen, J. G. Park, C. Stark, D. A. James, M. Olhovsky, R. Linding, Y. Mao, and T. Pawson. "Eukaryotic Protein Domains as Functional Units of Cellular Evolution." *Sci. Signal.* 2009, ra76–ra76 (see p. 45)

[Joh+24] B. Johnston, J. Elferich, R. B. Davidson, Y. Zhuang, Y. Yao, T. Tubiana, P. Kunzmann, O. Laprevote, Rich, TheJeran, et al. *BradyAJohnston/MolecularNodes: V4.2.7 for Blender 4.2*. Version v4.2.7. Zenodo, 2024 (see p. 198)

[Jon+97] G. Jones, P. Willett, R. C. Glen, A. R. Leach, and R. Taylor. "Development and Validation of a Genetic Algorithm for Flexible Docking 1 1Edited by F. E. Cohen." *Journal of Molecular Biology*. 1997, pp. 727–748 (see p. 102)

[Jum+21] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. "Highly Accurate Protein Structure Prediction with AlphaFold." *Nature*. 2021, pp. 583–589 (see pp. 26, 52)

[Jur+16] A. Jurcik, J. Parulek, J. Sochor, and B. Kozlikova. "Accelerated Visualization of Transparent Molecular Surfaces in Molecular Dynamics." *IEEE Pac. Vis. Symp.* 2016, pp. 112–119 (see p. 107)

[Jur+18] A. Jurč ík, D. Bednář, J. Byška, S. M. Marques, K. Furmanová, L. Daniel, P. Kokkonen, J. Brezovský, O. Strnad, J. Štourač, et al. "CAVER Analyst 2.0: Analysis and Visualization of Channels and Tunnels in Protein Structures and Molecular Dynamics Trajectories." *Bioinformatics*. 2018, pp. 3586–3588 (see p. 152)

[Jur+19] A. Jurčík, K. Furmanová, J. Byška, V. Vonásek, O. Vávra, P. Ulbrich, H. Hauser, and B. Kozlíková. "Visual Analysis of Ligand Trajectories in Molecular Dynamics." *2019 IEEE Pac. Vis. Symp. PacificVis*. 2019 IEEE Pacific Visualization Symposium (PacificVis). 2019, pp. 212–221 (see p. 153)

[KA12] I. Kufareva and R. Abagyan. "Methods of Protein Structure Comparison." *Methods Mol Biol*. 2012, pp. 231–257 (see pp. 46, 49, 52)

[Kab76] W. Kabsch. "A Solution for the Best Rotation to Relate Two Sets of Vectors." *Acta Cryst A*. 1976, pp. 922–923 (see p. 49)

[Kar+13] P. Kar, S. M. Gopal, Y.-M. Cheng, A. Predeus, and M. Feig. "PRIMO: A Transferable Coarse-Grained Force Field for Proteins." *J. Chem. Theory Comput.* 2013 (see p. 59)

[KBE09] M. Krone, K. Bidmon, and T. Ertl. "Interactive Visualization of Molecular Surface Dynamics." *IEEE Trans. Vis. Comput. Graphics* 15.6. 2009, pp. 1391–1398 (see pp. 106, 107, 115)

[KDE10] M. Krone, C. Dachsbacher, and T. Ertl. "Parallel Computation and Interactive Visualization of Time-Varying Solvent Excluded Surfaces." *Proc. First ACM Int. Conf. Bioinforma. Comput. Biol.* BCB'10: ACM International Conference on Bioinformatics and Computational Biology. ACM, 2010, pp. 402–405 (see p. 110)

[Kei+08] D. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler. "Visual Analytics: Scope and Challenges." *First Publ Lect. Notes Comput. Sci. No 4404 2008 Pp 76-90*. 2008 (see pp. 159, 193)

[Ken22] B. Kenwright. "Introduction to the WebGPU API." *ACM SIGGRAPH 2022 Courses*. SIGGRAPH '22: Special Interest Group on Computer Graphics and Interactive Techniques Conference. ACM, 2022, pp. 1–184 (see p. 199)

[KGE11] M. Krone, S. Grottel, and T. Ertl. "Parallel Contour-Buildup Algorithm for the Molecular Surface." *IEEE Symp. Biol. Data Vis. BioVis*. 2011, pp. 17–22 (see pp. 107, 121, 122, 191)

[KH20] M. Kulmanov and R. Hoehndorf. "DeepGOPlus: Improved Protein Function Prediction from Sequence." *Bioinformatics*. 2020, pp. 422–429 (see p. 53)

[Kir12] A. Kirk. *Data Visualization: A Successful Design Process ; a Structured Design Approach to Equip You with the Knowledge of How to Successfully Accomplish Any Data Visualization Challenge Efficiently and Effectively*. Online-Ausg. Packt Pub, 2012 (see p. 1)

[KKH18] M. Kulmanov, M. A. Khan, and R. Hoehndorf. "DeepGO: Predicting Protein Functions from Sequence and Interactions Using a Deep Ontology-Aware Classifier." *Bioinformatics*. 2018, pp. 660–668 (see p. 53)

[Kno+14] A. Knoll, I. Wald, P. Navratil, A. Bowen, K. Reda, M. E. Papka, and K. Gaither. "RBF Volume Ray Casting on Multicore and Manycore CPUs." *Comput. Graph. Forum* 33.3. 2014, pp. 71–80 (see p. 125)

[Koe01] P. Koehl. "Protein Structure Similarities." *Curr. Opin. Struct. Biol.* 2001, pp. 348–353 (see p. 48)

[Kol+16] I. Kolesár, J. Byška, J. Parulek, H. Hauser, and B. Kozlíková. "Unfolding and Interactive Exploration of Protein Tunnels and Their Dynamics." *Eurographics Workshop Vis. Comput. Biol. Med.* 2016 (see pp. 72, 88)

[Kon+16] D. G. Kontopoulos, D. Vlachakis, G. Tsiliki, and S. Kossida. "Structuprint: A Scalable and Extensible Tool for Two-Dimensional Representation of Protein Surfaces." *BMC Struct. Biol.* 2016, p. 4 (see p. 72)

[Koz+17] B. Kozlíková, M. Krone, M. Falk, N. Lindow, M. Baaden, D. Baum, I. Viola, J. Parulek, and H.-C. Hege. "Visualization of Biomolecular Structures: State of the Art Revisited." *Computer Graphics Forum*. 2017, pp. 178–204 (see pp. 3, 33–35, 152, 167)

[Kro+12] M. Krone, J. E. Stone, T. Ertl, and K. Schulten. "Fast Visualization of Gaussian Density Surfaces for Molecular Dynamics and Particle System Trajectories." *EuroVis - Short Pap.* 2012, pp. 67–71 (see pp. 115, 191)

[Kro+13] M. Krone, G. Reina, C. Schulz, T. Kulschewski, J. Pleiss, and T. Ertl. "Interactive Extraction and Tracking of Biomolecular Surface Features." *CGF*. 2013, pp. 331–340 (see p. 75)

[Kro+17] M. Krone, F. Frieß, K. Scharnowski, G. Reina, S. Fademrecht, T. Kulschewski, J. Pleiss, and T. Ertl. "Molecular Surface Maps." *IEEE Trans. Vis. Comput. Graph.* 2017, pp. 701–710 (see pp. 7, 70–78, 96)

[KS14] G. A. Kochetov and O. N. Solovjeva. "Structure and Functioning Mechanism of Transketolase." *Biochim. Biophys. Acta BBA - Proteins Proteomics*. 2014, pp. 1608–1618 (see p. 91)

[KS83] W. Kabsch and C. Sander. "Dictionary of Protein Secondary Structure: Pattern Recognition of Hydrogen-Bonded and Geometrical Features." *Biopolymers*. 1983, pp. 2577–2637 (see p. 60)

[KSH12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." *Adv. Neural Inf. Process. Syst.* Ed. by F. Pereira, C. Burges, L. Bottou, and K. Weinberger. Vol. 25. Curran Associates, Inc., 2012 (see pp. 56, 58)

[KW05] M. A. Koch and H. Waldmann. "Protein Structure Similarity Clustering and Natural Product Structure as Guiding Principles in Drug Discovery." *Drug Discov. Today*. 2005, pp. 471–483 (see p. 48)

[KW17] T. N. Kipf and M. Welling. "Semi-Supervised Classification with Graph Convolutional Networks." *ICML*. 2017 (see pp. 54, 57, 66, 67)

[KYB03] I. Korf, M. Yandell, and J. Bedell. *BLAST*. 1st ed. O'Reilly & Associates, 2003 (see p. 47)

[KZS15] G. Koch, R. Zemel, and R. Salakhutdinov. "Siamese Neural Networks for One-Shot Image Recognition." *ICML Deep Learn. Workshop*. Vol. 2. 2015 (see p. 68)

[LA95] V. J. LiCata and G. K. Ackers. "Long-Range, Small Magnitude Nonadditivity of Mutational Effects in Proteins." *Biochemistry* 34.10. 1995, pp. 3133–3139 (see p. 133)

[Lar+07] M. A. Larkin, G. Blackshields, N. P. Brown, R. Chenna, P. A. McGettigan, H. McWilliam, F. Valentin, I. M. Wallace, A. Wilm, R. Lopez, et al. "Clustal w and Clustal x Version 2.0." *Bioinformatics* 23.21. 2007, pp. 2947–2948 (see p. 94)

[Le +15] M. Le Muzic, L. Autin, J. Parulek, and I. Viola. "cellVIEW: A Tool for Illustrative and Multi-Scale Rendering of Large Biomolecular Datasets." *Proc. EG VCBM*. The Eurographics Association, 2015, pp. 61–70 (see p. 125)

[Lea01] A. R. Leach. *Molecular Modelling: Principles and Applications*. 2nd ed. Prentice Hall, 2001 (see pp. 99–101, 104)

[Lei+23] J. Leider, H. Leider, K. Watts-Deuchar, A. Kaaman, A. Henry, and community. *Vuetify — A Material Design Framework for Vue.Js*. 2023. `https://vuetifyjs.com/en/` (accessed 06/21/2023) (see pp. 40, 173)

[Lew11] E. Lewars. *Computational Chemistry: Introduction to the Theory and Applications of Molecular and Quantum Mechanics*. 2nd ed. Springer, 2011 (see pp. 99, 100)

[LH18] Y. Liu and J. Heer. "Somewhere over the Rainbow: An Empirical Assessment of Quantitative Colormaps." *Proc CHI Conf. Hum. Factors Comput. Syst.* 2018, 598:1–598:12 (see p. 138)

[Lic+18] N. Lichtenberg, R. Menges, V. Ageev, A. George, P. Heimer, D. Imhof, and K. Lawonn. "Analyzing Residue Surface Proximity to Interpret Molecular Dynamics." *Comput. Graph. Forum* 37.3. 2018, pp. 379–390 (see p. 126)

[Lin+10] N. Lindow, D. Baum, S. Prohaska, and H.-C. Hege. "Accelerated Visualization of Dynamic Molecular Surfaces." *Comput. Graph. Forum*. 2010, pp. 943–952 (see p. 107)

[Lip+01] C. A. Lipinski, F. Lombardo, B. W. Dominy, and P. J. Feeney. "Experimental and Computational Approaches to Estimate Solubility and Permeability in Drug Discovery and Development Settings." *Adv. Drug Deliv. Rev.* 2001, pp. 3–26 (see p. 154)

[Lo +00] L. Lo Conte, B. Ailey, T. J. P. Hubbard, S. E. Brenner, A. G. Murzin, and C. Chothia. "SCOP: A Structural Classification of Proteins Database." *Nucleic Acids Research*. 2000, pp. 257–259 (see p. 28)

[LR11] F. Lindemann and T. Ropinski. "About the Influence of Illumination Models on Image Comprehension in Direct Volume Rendering." *IEEE Trans. Vis. Comput. Graph.* 2011, pp. 1922–1931 (see p. 198)

[LR71] B. Lee and F. Richards. "The Interpretation of Protein Structures: Estimation of Static Accessibility." *Journal of Molecular Biology*. 1971, pp. 379–400 (see p. 35)

[LS11] R. A. Laskowski and M. B. Swindells. "LigPlot+: Multiple Ligand–Protein Interaction Diagrams for Drug Discovery." *J. Chem. Inf. Model.* 2011, pp. 2778–2786 (see pp. 3, 152)

[Mah15] E. Mahieu. *Trilateration and the Intersection of Three Spheres - Wolfram Demonstrations Project*. 2015. https://demonstrations.wolfram.com/TrilaterationAndTheIntersectionOfThreeSpheres (see pp. 109, 114)

[Mar+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, G. S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. *TensorFlow: Large-scale Machine Learning on Heterogeneous Systems*. 2015 (see p. 64)

[Mar+18] S. Markidis, S. W. D. Chien, E. Laure, I. B. Peng, and J. S. Vetter. "NVIDIA Tensor Core Programmability, Performance & Precision." *2018 IEEE Int. Parallel Distrib. Process. Symp. Workshop IPDPSW*. 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). 2018, pp. 522–531 (see p. 52)

[Mar+21] S. M. Marques, L. Šupolíková, L. Molčanová, K. Šmejkal, D. Bednář, and I. Slaninová. "Screening of Natural Compounds as P-Glycoprotein Inhibitors against Multidrug Resistance." *Biomedicines*. 2021, p. 357 (see pp. 175, 178–180)

[MDB87] B. McCormick, T. DeFanti, and M. Brown, eds. *Visualization in Scientific Computing*. Vol. 21. 6. ACM SIGGRAPH, Computer Graphics, 1987 (see p. 15)

[Mea80] D. Meagher. *Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer*. 1980 (see p. 107)

[Men23] B. Mengistu. "Deep-Learning Realtime Upsampling Techniques in Video Games." *Scholarly Horizons: University of Minnesota, Morris Undergraduate Journal*. 2023 (see p. 192)

[MHO08] G. M. Morris, R. Huey, and A. J. Olson. "Using AutoDock for Ligand-Receptor Docking." *Curr. Protoc. Bioinforma.* 2008, pp. 8.14.1–8.14.40 (see p. 42)

[Mia+18] H. Miao, E. De Llano, T. Isenberg, M. E. Gröller, I. Barisic, and I. Viola. "DimSUM: Dimension and Scale Unifying Maps for Visual Abstraction of DNA Origami Structures." *Comput. Graph. Forum* 37.3. 2018, pp. 403–413 (see p. 125)

[Mic+11] N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein. "MDAnalysis: A Toolkit for the Analysis of Molecular Dynamics Simulations." *J. Comput. Chem.* 32.10. 2011, pp. 2319–2327 (see p. 128)

[Mik+13] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient Estimation of Word Representations in Vector Space." *arXiv*. 2013 (see p. 53)

[Min+20] S. Min, S. Park, S. Kim, H.-S. Choi, and S. Yoon. "Pre-Training of Deep Bidirectional Protein Sequence Representations with Structural Information." *Bioinformatics*. 2020 (see p. 53)

[ML08] G. M. Morris and M. Lim-Wilby. "Molecular Docking." *Molecular Modeling of Proteins*. Ed. by A. Kukol. Methods Molecular Biology™. Humana Press, 2008, pp. 365–382 (see pp. 102, 151)

[Mor+09] G. M. Morris, R. Huey, W. Lindstrom, M. F. Sanner, R. K. Belew, D. S. Goodsell, and A. J. Olson. "Auto-Dock4 and AutoDockTools4: Automated Docking with Selective Receptor Flexibility." *J Comput Chem*. 2009, pp. 2785–2791 (see pp. 42, 102, 103)

[Mor+14] D. S. Morris, Goodsell, M. E. Pique, and W. Lindstrom. *AutoDock4*. AutoDock Suite. 2014. `https://autodocksuite.scripps.edu/autodock4/` (accessed 07/05/2023) (see pp. 41, 42)

[Mor+98] G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Belew, and A. J. Olson. "Automated Docking Using a Lamarckian Genetic Algorithm and an Empirical Binding Free Energy Function." *J. Comput. Chem.* 1998, pp. 1639–1662 (see pp. 41, 42)

[MSM09] M. Maheshwari, S. Silakari, and M. Motwani. "Image Clustering Using Color and Texture." *First Int. Conf. Comput. Intell. Commun. Syst. Netw.* 2009, pp. 403–408 (see pp. 74, 79)

[Mul94] P. Muller. "Glossary of Terms Used in Physical Organic Chemistry (IUPAC Recommendations 1994)." *Pure Appl. Chem.* 1994, pp. 1077–1184 (see p. 154)

[Mun15] T. Munzner. *Visualization Analysis and Design*. A.K. Peters Visualization Series. CRC Press, Taylor & Francis Group, CRC Press is an imprint of the Taylor & Francis Group, an informa business, 2015 (see p. 16)

[Mur+95] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. "SCOP: A Structural Classification of Proteins Database for the Investigation of Sequences and Structures." *J. Mol. Biol.* 1995, pp. 536–540 (see pp. 51, 62, 65, 67)

[MZ09] S. Mukherjee and Y. Zhang. "MM-align: A Quick Algorithm for Aligning Multiple-Chain Protein Complex Structures Using Iterative Dynamic Programming." *Nucleic Acids Res.* 2009, e83 (see pp. 46, 49–51, 62, 70, 85, 94)

[ND21] R. G. Nichols and E. R. Davenport. "The Relationship between the Gut Microbiome and Host Gene Expression: A Review." *Hum Genet*. 2021, pp. 747–760 (see p. 197)

[NW70] S. B. Needleman and C. D. Wunsch. "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins." *Journal of Molecular Biology*. 1970, pp. 443–453 (see p. 47)

[OBo+11] N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch, and G. R. Hutchison. "Open Babel: An Open Chemical Toolbox." *J Cheminform*. 2011, p. 33 (see p. 174)

[OF03] S. Osher and R. Fedkiw. "Signed Distance Functions." *Level Set Methods and Dynamic Implicit Surfaces*. Ed. by S. Osher and R. Fedkiw. Springer, 2003, pp. 17–22 (see p. 191)

[One] OneAngstrom. *SAMSON*. `https://www.samson-connect.net/` (accessed 05/12/2023) (see p. 152)

[Ore+97] C. A. Orengo, A. D. Michie, S. Jones, D. T. Jones, M. B. Swindells, and J. M. Thornton. "CATH – a Hierarchic Classification of Protein Domain Structures." *Structure*. 1997, pp. 1093–1109 (see pp. 51, 62, 67)

[Oso+15] D. I. Osolodkin, E. V. Radchenko, A. A. Orlov, A. E. Voronkov, V. A. Palyulin, and N. S. Zefirov. "Progress in Visual Representations of Chemical Space." *Expert Opin Drug Discov*. 2015, pp. 959–973 (see p. 152)

[Par+14] J. Parulek, D. Jönsson, T. Ropinski, S. Bruckner, A. Ynnerman, and I. Viola. "Continuous Levels-of-Detail and Visual Abstraction for Seamless Molecular Visualization." *Comput. Graph. Forum* 33.6. 2014, pp. 276–287 (see pp. 126, 191)

[Ped+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. "Scikit-Learn: Machine Learning in Python." *J. Mach. Learn. Res.* 12.85. 2011, pp. 2825–2830 (see p. 31)

[Pet+04] E. F. Pettersen, T. D. Goddard, C. C. Huang, G. S. Couch, D. M. Greenblatt, E. C. Meng, and T. E. Ferrin. "UCSF Chimera–a Visualization System for Exploratory Research and Analysis." *J Comput Chem*. 2004, pp. 1605–1612 (see pp. 3, 37, 151, 191)

[PKS21] M. Pande, D. Kundu, and R. Srivastava. "Drugs Repurposing against SARS-CoV2 and the New Variant B.1.1.7 (Alpha Strain) Targeting the Spike Protein: Molecular Docking and Simulation Studies." *Heliyon*. 2021, e07803 (see pp. 175, 177)

[Pog73] A. Pogorelov. *Extrinsic Geometry of Convex Surfaces*. Trans. by Israel Program For Scientific Translations. Translations of Mathematical Monographs. American Mathematical Society, 1973 (see p. 55)

[PRV13] J. Parulek, T. Ropinski, and I. Viola. "Seamless Visual Abstraction of Molecular Surfaces." *Proc. 29th Spring Conf. Comput. Graph.* ACM. 2013, pp. 107–114 (see p. 126)

[Qi+17] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space." *NIPS*. 2017 (see p. 58)

[Rag+17] M. Ragoza, J. Hochuli, E. Idrobo, J. Sunseri, and D. R. Koes. "Protein–Ligand Scoring with Convolutional Neural Networks." *J Chem. Inf. Model.* 57.4. 2017, pp. 942–957 (see p. 53)

[Rao+19] R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, X. Chen, J. Canny, P. Abbeel, and Y. S. Song. "Evaluating Protein Transfer Learning with TAPE." *Adv. Neural Inf. Process. Syst.* 2019 (see pp. 53, 66, 67)

[RE05] G. Reina and T. Ertl. "Hardware-Accelerated Glyphs for Mono- and Dipoles in Molecular Dynamics Visualization." *EUROVIS 2005 Eurographics IEEE VGTC Symp. Vis.* 2005, 6 pages (see p. 175)

[Ric77] F. M. Richards. "Areas, Volumes, Packing, and Protein Structure." *Annu. Rev. Biophys. Bioeng.* 1977, pp. 151–176 (see pp. 34, 35)

[Ric78] J. S. Richardson. "The Anatomy & Taxonomy of Protein Structure." *Proc. Nat. Acad. Sci. USA* 75. 1978, pp. 2674–2678 (see p. 3)

[Riz+15] S. Rizzi, M. Hereld, J. Insley, M. E. Papka, T. Uram, and V. Vishwanath. "Large-Scale Parallel Visualization of Particle-Based Simulations Using Point Sprites and Level-of-Detail." *Proc. EGPGV*. The Eurographics Association, 2015, pp. 1–10 (see p. 125)

[RS07] S. J. Rahi and K. Sharp. "Mapping Complicated Surfaces onto a Sphere." *Int. J. Comput. Geom. Appl.* 2007, pp. 305–329 (see pp. 72, 76)

[Rui+14] S. Ruiz-Carmona, D. Alvarez-Garcia, N. Foloppe, A. B. Garmendia-Doval, S. Juhos, P. Schmidtke, X. Barril, R. E. Hubbard, and S. D. Morley. "rDock: A Fast, Versatile and Open Source Program for Docking Ligands to Proteins and Nucleic Acids." *PLoS Comput. Biol.* 2014 (see p. 102)

[SA22] M. Segal and K. Akeley. *The OpenGL® Graphics System: A Specification (Version 4.6 (Core Profile))*. Khronos Group Inc., 2022 (see pp. 18, 19)

[Sab+20] M. V. Sabando, P. Ulbrich, M. Selzer, J. Byška, J. Mičan, I. Ponzoni, A. J. Soto, M. L. Ganuza, and B. Kozlíková. "ChemVA: Interactive Visual Analysis of Chemical Compound Similarity in Virtual Screening." *arXiv*. 2020 (see p. 153)

[Sal+15] S. Salentin, S. Schreiber, V. J. Haupt, M. F. Adasme, and M. Schroeder. "PLIP: Fully Automated Protein–Ligand Interaction Profiler." *Nucleic Acids Res.* 2015, W443–W447 (see pp. 155, 159, 166, 174)

[Sán+10] B. Sánchez, M. Zúñiga, F. González-Candelas, C. G. de los Reyes-Gavilán, and A. Margolles. "Bacterial and Eukaryotic Phosphoketolases: Phylogeny, Distribution and Evolution." *J. Mol. Microbiol. Biotechnol.* 2010, pp. 37–51 (see p. 91)

[San+15] T. Sander, J. Freyss, M. von Korff, and C. Rufener. "DataWarrior: An Open-Source Program For Chemistry Aware Data Visualization And Analysis." *J. Chem. Inf. Model.* 2015, pp. 460–473 (see p. 153)

[San+18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. "MobileNetV2: Inverted Residuals and Linear Bottlenecks." *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*. 2018, pp. 4510–4520 (see pp. 74, 80)

[San+19] S. Sanna, N. R. Van Zuydam, A. Mahajan, A. Kurilshikov, A. Vich Vila, U. Võsa, Z. Mujagic, A. A. M. Masclee, D. M. A. E. Jonkers, M. Oosting, et al. "Causal Relationships among the Gut Microbiome, Short-Chain Fatty Acids and Metabolic Diseases." *Nat Genet*. 2019, pp. 600–605 (see p. 197)

[San97] J. M. Sangster. *Octanol-Water Partition Coefficients: Fundamentals and Physical Chemistry*. John Wiley & Sons, 1997 (see p. 154)

[Sch+14] K. Scharnowski, M. Krone, G. Reina, T. Kulschewski, J. Pleiss, and T. Ertl. "Comparative Visualization of Molecular Surfaces Using Deformable Models." *Comput. Graph. Forum*. 2014, pp. 191–200 (see p. 76)

[Sch+17] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu. "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN." *ACM Trans. Database Syst.* 2017, pp. 1–21 (see p. 63)

[Sch+19] K. Schatz, M. Krone, T. L. Bauer, V. Ferrario, J. Pleiss, and T. Ertl. "Molecular Sombreros: Abstract Visualization of Binding Sites within Proteins." *Eurographics Workshop Vis. Comput. Biol. Med.* 2019 (see p. 72)

[Sch+20a] M. Schäfer, A. Cremer, M. Aktürk, and M. Krone. "Towards an Enhanced Interactive Protein Sequence Diagram." *Eurographics VCBM Posters*. 2020 (see p. 32)

[Sch+20b] K. Schatz, F. Frieß, M. Schäfer, T. Ertl, and M. Krone. "Analyzing Protein Similarity by Clustering Molecular Surface Maps." *EG Workshop Vis. Comput. Biol. Med.* 2020, pp. 103–114 (see pp. 7, 11, 50, 70, 85, 86)

[Sch+21a] K. Schatz, J. J. Franco-Moreno, M. Schäfer, A. S. Rose, V. Ferrario, J. Pleiss, P.-P. Vázquez, T. Ertl, and M. Krone. "Visual Analysis of Large-Scale Protein-Ligand Interaction Data." *Comput. Graph. Forum*. 2021, pp. 394–408 (see pp. 8, 13, 123, 124, 131, 134, 136, 137, 143, 146, 152, 182)

[Sch+21b] K. Schatz, F. Frieß, M. Schäfer, P. C. F. Buchholz, J. Pleiss, T. Ertl, and M. Krone. "Analyzing the Similarity of Protein Domains by Clustering Molecular Surface Maps." *Computers & Graphics*. 2021, pp. 114–127 (see pp. 7, 12, 30, 49, 51, 70, 71, 73, 75, 83, 87, 90, 93–95)

[Sch+24] M. Schäfer, N. Brich, J. Byška, S. M. Marques, D. Bednář, P. Thiel, B. Kozlíková, and M. Krone. "InVADo: Interactive Visual Analysis of Molecular Docking Data." *IEEE Trans. Visual. Comput. Graphics*. 2024, pp. 1984–1997 (see pp. 8, 13, 101, 151, 157, 158, 160, 162, 163, 165, 166, 168, 170, 172, 174, 177–180, 182, 194, 195, 198)

[SCS02] I. Schomburg, A. Chang, and D. Schomburg. "BRENDA, Enzyme Data and Metabolic Information." *Nucleic Acids Res.* 30.1. 2002, pp. 47–49 (see pp. 70, 84)

[SD20] L. L. C. Schrödinger and W. DeLano. *PyMOL*. Version 2.4.0. 2020. `http://www.pymol.org/pymol` (see pp. 3, 37, 128, 191)

[Seh+18] D. Sehnal, A. Rose, J. Koca, S. Burley, and S. Velankar. "Mol*: Towards a Common Library and Tools for Web Molecular Graphics." *Workshop Mol. Graph. Vis. Anal. Mol. Data*. EG, 2018, pp. 29–33 (see p. 141)

[Seh+21] D. Sehnal, S. Bittrich, M. Deshpande, R. Svobodová, K. Berka, V. Bazgier, S. Velankar, S. K. Burley, J. Koča, and A. S. Rose. "Mol* Viewer: Modern Web App for 3D Visualization and Analysis of Large Biomolecular Structures." *Nucleic Acids Res.* 2021, W431–W437 (see pp. 3, 33, 40, 191)

[SH14] F. Sievers and D. G. Higgins. "Clustal Omega." *Curr. Protoc. Bioinforma.* 2014, pp. 3.13.1–3.13.16 (see pp. 46, 48)

[Sho+02] B. K. Shoichet, S. L. McGovern, B. Wei, and J. J. Irwin. "Lead Discovery Using Molecular Docking." *Current Opinion in Chemical Biology*. 2002, pp. 439–446 (see p. 156)

[SI15] T. Sterling and J. J. Irwin. "ZINC 15 – Ligand Discovery for Everyone." *J. Chem. Inf. Model.* 2015, pp. 2324–2337 (see pp. 154, 174)

[Sil+23] V. Silva, S. Araújo, A. Monteiro, J. Eira, J. E. Pereira, L. Maltez, G. Igrejas, T. S. Lemsaddek, and P. Poeta. "Staphylococcus Aureus and MRSA in Livestock: Antimicrobial Resistance and Genetic Lineages." *Microorganisms*. 2023, p. 124 (see p. 195)

[Sim+97] K. T. Simons, C. Kooperberg, E. Huang, and D. Baker. "Assembly of Protein Tertiary Structures from Fragments with Similar Local Sequences Using Simulated Annealing and Bayesian Scoring Functions." *J Mol. Biol.* 1997 (see p. 59)

[SK19] M. Schäfer and M. Krone. "A Massively Parallel CUDA Algorithm to Compute and Visualize the Solvent Excluded Surface for Dynamic Molecular Data." EG Workshop on Molecular Graphics and Visual Analysis of Molecular Data, 2019 (see pp. 7, 12, 34, 105, 106, 108, 109, 114, 117–120, 167, 182, 191)

[Skå+18] R. Skånberg, M. Linares, C. König, P. Norman, D. Jönsson, I. Hotz, and A. Ynnerman. "VIA-MD: Visual Interactive Analysis of Molecular Dynamics." *Workshop Mol. Graph. Vis. Anal. Mol. Data*. Ed. by J. Byska, M. Krone, and B. Sommer. The Eurographics Association, 2018 (see pp. 126, 152)

[SM18] V. Salmaso and S. Moro. "Bridging Molecular Docking to Molecular Dynamics in Exploring Ligand-Protein Recognition Process: An Overview." *Front. Pharmacol.* 2018, p. 923 (see p. 101)

[SM58] R. R. Sokal and C. D. Michener. "A Statistical Method for Evaluating Systematic Relationships." *Univ. Kans. Sci. Bull.* 2.38. 1958, pp. 1409–1438 (see pp. 29, 30, 71)

[SN87] N. Saitou and M. Nei. "The Neighbor-Joining Method: A New Method for Reconstructing Phylogenetic Trees." *Mol. Biol. Evol.* 4.4. 1987, pp. 406–425 (see p. 29)

[SOS96] M. F. Sanner, A. J. Olson, and J. C. Spehner. "Reduced Surface: An Efficient Way to Compute Molecular Surfaces." *Biopolymers*. 1996, pp. 305–320 (see pp. 36, 75, 106, 140)

[SR10] K. Stierand and M. Rarey. "Drawing the PDB: Protein-Ligand Complexes in Two Dimensions." *ACS Med. Chem. Lett.* 2010, pp. 540–545 (see p. 3)

[Ste56] H. Steinhaus. "Sur La Division Des Corp Materiéls En Parties." *Bull. Acad. Polon. Sci* 1.804. 1956, p. 801 (see p. 29)

[Sto+16] J. E. Stone, M. J. Hallock, J. C. Phillips, J. R. Peterson, Z. Luthey-Schulten, and K. Schulten. "Evaluation of Emerging Energy-Efficient Heterogeneous Computing Platforms for Biomolecular and Cellular Simulation Workloads." *2016 IEEE Int. Parallel Distrib. Process. Symp. Workshop IPDPSW*. 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). 2016, pp. 89–100 (see p. 103)

[Str+20] N. Strodthoff, P. Wagner, M. Wenzel, and W. Samek. "UDSMProt: Universal Deep Sequence Models for Protein Classification." *Bioinformatics*. 2020 (see pp. 53, 66, 67)

[SW81] T. F. Smith and M. S. Waterman. "Identification of Common Molecular Subsequences." *Journal of Molecular Biology*. 1981, pp. 195–197 (see p. 47)

[SY16] D. Storti and M. Yurtoglu. *CUDA for Engineers: An Introduction to High-Performance Parallel Computing*. Addison-Wesley, 2016 (see p. 21)

[Syk24] V. J. Sykora. "Automated Virtual Screening." *High Performance Computing for Drug Discovery and Biomedicine*. Ed. by A. Heifetz. Methods in Molecular Biology. Springer US, 2024, pp. 137–152 (see p. 102)

[SZS20] M. M. Stepniewska-Dziubinska, P. Zielenkiewicz, and P. Siedlecki. "Improving Detection of Protein-Ligand Binding Sites with 3D Segmentation." *Sci Rep*. 2020, p. 5035 (see p. 64)

[TA95] M. Totrov and R. Abagyan. "The Contour-Buildup Algorithm to Calculate the Analytical Molecular Surface." *J. Struct. Biol.* 1995, pp. 138–143 (see p. 107)

[TG19] G. A. Tribello and P. Gasparotto. "Using Dimensionality Reduction to Analyze Protein Trajectories." *Front. Mol. Biosci.* 2019 (see p. 149)

[Tho+19] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. "Kpconv: Flexible and Deformable Convolution for Point Clouds." *ICCV*. 2019, pp. 6411–20 (see p. 57)

[TL12] Y. Y. Tseng and W.-H. Li. "Classification of Protein Functional Surfaces Using Structural Characteristics." *PNAS*. 2012, pp. 1170–1175 (see p. 70)

[TMB02] B. Tversky, J. B. Morrison, and M. Betrancourt. "Animation: Can It Facilitate?" *Int. J. Hum-Comput. St.* 57.4. 2002, pp. 247–262 (see pp. 123, 125)

[TO10] O. Trott and A. J. Olson. "AutoDock Vina: Improving the Speed and Accuracy of Docking with a New Scoring Function, Efficient Optimization, and Multithreading." *J. Comput. Chem.* 2010, pp. 455–461 (see p. 103)

[Tor+19] P. H. M. Torres, A. C. R. Sodero, P. Jofily, and F. P. Silva-Jr. "Key Topics in Molecular Docking for Drug Design." *Int. J. Mol. Sci.* 2019, p. 4574 (see p. 101)

[Tow+19] R. J. L. Townshend, R. Bedi, P. A. Suriana, and R. O. Dror. "End-to-End Learning on 3D Protein Structure for Interface Prediction." *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2019, pp. 15642–15651 (see pp. 53, 54)

[TWT21] F. Trozzi, X. Wang, and P. Tao. "UMAP as Dimensionality Reduction Tool for Molecular Dynamics Simulations of Biomacromolecules: A Comparison Study." *J. Phys. Chem. B*. 2021, pp. 5022–5034 (see p. 149)

[Upp+94] J. Uppenberg, M. T. Hansen, S. Patkar, and T. A. Jones. "The Sequence, Crystal Structure Determination and Refinement of Two Crystal Forms of Lipase B from Candida Antarctica." *Structure* 2.4. 1994, pp. 293–308 (see p. 142)

[Vad+17] V. Vad, J. Byška, A. Jurcík, I. Viola, E. Gröller, H. Hauser, S. M. Marques, J. Damborskỳ, and B. Kozlíková. "Watergate: Visual Exploration of Water Trajectories in Protein Dynamics." *Proc. EG VCBM*. The Eurographics Association, 2017 (see pp. 125, 142)

[Váz+18] P. Vázquez, P. Hermosilla, V. Guallar, J. Estrada, and A. Vinacua. "Visual Analysis of Protein-Ligand Interactions." *Comput. Graph. Forum*. 2018, pp. 391–402 (see pp. 123, 126, 133, 135, 148, 152)

[vLux07] U. von Luxburg. "A Tutorial on Spectral Clustering." *Stat Comput*. 2007, pp. 395–416 (see p. 59)

[VP14] C. Vogel and J. Pleiss. "The Modular Structure of ThDP-dependent Enzymes." *Proteins Struct. Funct. Bioinforma.* 2014, pp. 2523–2537 (see pp. 91, 92)

[Wan+04] R. Wang, X. Fang, Y. Lu, and S. Wang. "The PDBbind Database: Collection of Binding Affinities for Protein-Ligand Complexes with Known Three-Dimensional Structures." *J. Med. Chem.* 2004, pp. 2977–2980 (see pp. 46, 62)

[Wan+19] J. Wang, S. Hazarika, C. Li, and H. Shen. "Visualization and Visual Analysis of Ensemble Data: A Survey." *IEEE Trans. Vis. Comput. Graphics* 25.9. 2019, pp. 2853–2872 (see p. 125)

[Wan+23] L. Wang, I. Alibay, B. Feddersen, and F. Langenfeld. *MDAnalysis User Guide Documentation*. 2023. `https://userguide.mdanalysis.org/stable/formats/format_reference.html` (accessed 06/29/2023) (see p. 41)

[Web92] E. Webb, ed. *Enzyme Nomenclature 1992. Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology and the Nomenclature Classification of Enzymes*. Academic Press, 1992 (see pp. 66, 68, 84)

[Wei+20] W. Wei, S. Cherukupalli, L. Jing, X. Liu, and P. Zhan. "Fsp3: A New Parameter for Drug-Likeness." *Drug Discovery Today*. 2020, pp. 1839–1845 (see p. 154)

[Wei07] D. Weiskopf. *GPU-based Interactive Visualization Techniques*. Mathematics & Visualization. Springer, 2007 (see p. 16)

[Wen20] P. Wennker. *Künstliche Intelligenz in der Praxis: Anwendungen in Unternehmen und Branchen: KI wettbewerbs- und zukunftsorientiert einsetzen*. Springer Gabler, 2020 (see pp. 27, 29)

[Wes+06] J. D. Westbrook, H. Yang, Z. Feng, and H. M. Berman. "The Use of mmCIF Architecture for PDB Data Management." *International Tables for Crystallography*. Ed. by S. R. Hall and B. McMahon. Red. by H. Fuess, Th. Hahn, H. Wondratschek, U. Müller, U. Shmueli, E. Prince, A. Authier, V. Kopský, D. B. Litvin, M. G. Rossmann, et al. 1st ed. International Union of Crystallography, 2006, pp. 539–543 (see p. 41)

[WQF19] W. Wu, Z. Qi, and L. Fuxin. "Pointconv: Deep Convolutional Networks on 3d Point Clouds." *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* 2019, pp. 9621–30 (see p. 57)

[WZ16] G. Wang and W. Zhu. "Molecular Docking for Drug Discovery and Development: A Widely Used Approach but Far from Perfect." *Future Med. Chem.* 2016, pp. 1707–1710 (see pp. 101, 103)

[Xio+14] Y. Xiong, J. Esquivel-Rodriguez, L. Sael, and D. Kihara. "3D-SURFER 2.0: Web Platform for Real-Time Search and Characterization of Protein Surfaces." *Protein Struct. Predict.* 2014, pp. 105–117 (see p. 46)

[Xio+19] Y. Xiong, M. Ren, R. Liao, K. Wong, and R. Urtasun. "Deformable Filter Convolution for Point Cloud Reasoning." *arXiv*. 2019 (see p. 57)

[XP98] C. Xu and J. Prince. “Snakes, Shapes, and Gradient Vector Flow.” *IEEE Trans. Image Process.* 1998, pp. 359–369 (see p. 76)

[XW05] R. Xu and D. WunschII. “Survey of Clustering Algorithms.” *IEEE Trans. Neural Netw.* 2005, pp. 645–678 (see p. 31)

[XZ09] D. Xu and Y. Zhang. “Generating Triangulated Macromolecular Surfaces by Euclidean Distance Transform.” *PLOS ONE*. 2009, e8140 (see p. 107)

[XZ10] J. Xu and Y. Zhang. “How Significant Is a Protein Structure Similarity with TM-score = 0.5?” *Bioinformatics*. 2010, pp. 889–895 (see pp. 51, 62, 63)

[Yan+18] K. K. Yang, Z. Wu, C. N. Bedbrook, and F. H. Arnold. “Learned Protein Embeddings for Machine Learning.” *Bioinformatics*. 2018 (see p. 53)

[YCH17] S. Yuan, H. S. Chan, and Z. Hu. “Using PyMOL as a Platform for Computational Drug Design.” *WIREs Comput. Mol. Sci.* 2017, e1298 (see pp. 151, 155)

[YHR15] E. Yuriev, J. Holien, and P. A. Ramsland. “Improvements, Trends, and New Ideas in Molecular Docking: 2012-2013 in Review: Improvements, Trends, and New Ideas in Molecular Docking.” *J. Mol. Recognit.* 2015, pp. 581–604 (see pp. 101, 103)

[Yin+18] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec. “Hierarchical Graph Representation Learning with Differentiable Pooling.” *NIPS*. 2018, pp. 4800–4810 (see p. 54)

[Yoo+20] J. Yoo, M. Groer, S. Dutra, A. Sarkar, and D. McSkimming. “Gut Microbiota and Immune System Interactions.” *Microorganisms*. 2020, p. 1587 (see p. 197)

[Zha+22] B. Zhang, H. Li, K. Yu, and Z. Jin. “Molecular Docking-Based Computational Platform for High-Throughput Virtual Screening.” *CCF Trans. HPC*. 2022 (see pp. 102, 151)

[Zha05] Y. Zhang. “TM-align: A Protein Structure Alignment Algorithm Based on the TM-score.” *Nucleic Acids Research*. 2005, pp. 2302–2309 (see pp. 46, 51)

[Zho21] Z.-H. Zhou. *Machine Learning*. Trans. by S. Liu. Springer, 2021 (see pp. 26, 29, 31)

[Zhu+22] L.-T. Zhu, X.-Z. Chen, B. Ouyang, W.-C. Yan, H. Lei, Z. Chen, and Z.-H. Luo. "Review of Machine Learning for Hydrodynamics, Transport, and Reactions in Multiphase Flows and Reactors." *Ind. Eng. Chem. Res.* 2022, pp. 9901–9949 (see pp. 27, 29)

[ZS04] Y. Zhang and J. Skolnick. "Scoring Function for Automated Assessment of Protein Structure Template Quality." *Proteins Struct. Funct. Bioinforma.* 2004, pp. 702–710 (see pp. 49, 51)

Dissertation

Eberhard Karls Universität Tübingen

2024