FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

# Relevanzorientierte Exploration von Molekulardynamik-Simulationen

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Visual Computing**

eingereicht von

**Thomas Trautner, BSc**
Matrikelnummer 01125421

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Univ.Ass. Dr.techn. Manuela Waldner, MSc
RNDr. Jan Byška, Ph.D.

Wien, 3. Oktober 2018

Thomas Trautner

Eduard Gröller

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

# Importance-Driven Exploration of Molecular Dynamics Simulations

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

**Diplom-Ingenieur**

in

**Visual Computing**

by

**Thomas Trautner, BSc**
Registration Number 01125421

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Assistance: Univ.Ass. Dr.techn. Manuela Waldner, MSc
RNDr. Jan Byška, Ph.D.

Vienna, 3rd October, 2018

Thomas Trautner

Eduard Gröller

# Erklärung zur Verfassung der Arbeit

Thomas Trautner, BSc
Erdbrustgasse 58/3+4, 1160 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 3. Oktober 2018

Thomas Trautner

# Acknowledgements

I would like to express my thanks especially to my supervisors Manuela Waldner and Jan Byška. Thank you for your motivation and restless support throughout the whole Master's thesis. This applied not only to theoretical questions in the area of visualization, but also practical software implementation tips that have continuously improved the quality of this work. Thanks to you, I have very much enjoyed researching internationally between Vienna, Brno, and Bergen.

Furthermore, I would like to thank Meister Eduard Gröller for his support during this work and throughout my studies, and for his joy of teaching visualization and computer graphics. Thank you for so many exciting insights that taught me how much there is still to learn.

I would also like to thank Barbora Kozlíková for her dedication and enthusiasm for molecular visualization. Because of her and her students, I discovered my excitement for visualizing complex molecular structures and animated MD-simulations.

Additionally, I would like to thank Stefan Bruckner and the entire visualization research group in Bergen not only for the great time I spent in Norway but also for the exciting future that lies ahead of me.

Of course, my greatest thanks go to my parents. Without your support throughout school and during my studies, I would not be able to make my small contribution to the world of research. Thank you for being who you are, and I am happy and proud to be your son!

# Kurzfassung

Ziel dieser Masterarbeit ist die Entwicklung einer neuen Echtzeit-Visualisierung, für die Erforschung von *Molekulardynamik (MD)-Simulationen*. Nachdem sich Computer-Hardware ständig verbessert und die Rechenleistung laufend zunimmt, werden MD-Simulationen immer detaillierter, sind leichter verfügbar und bestehen aus Hunderten, Tausenden oder sogar Millionen von einzelnen Simulationsschritten. Die Berechnung von solchen Simulationen ist nicht länger durch Algorithmen oder Hardware beschränkt, trotzdem ist es immer noch nicht möglich diese riesige Datenmenge, als animierte 3D Visualisierung, mit gängigen Visualisierungs-Werkzeugen zu erforschen. Zusätzlich dazu beansprucht die Nutzung von aktuellen Software-Werkzeugen, für die Erforschung solch langer Simulationen, zu viel Zeit. Molekulare Szenen sind äußerst dicht und detailreich, wodurch Visualisierungen unter einem visuellen Durcheinander leiden. Es ist daher leicht möglich, dass Benutzer wichtige Ereignisse verpassen.

Daher haben wir eine *Fokus & Kontext Visualisierung* entwickelt, die die Benutzer zu den relevantesten zeitlichen- und räumlichen Ereignissen geleitet und es ist nicht mehr notwendig, die Simulation linear zu explorieren. Unser Beitrag kann in folgende vier Themengebiete unterteilt werden:

1. Räumliche-Wichtigkeit durch *Levels of Detail*. Abhängig von Forschungsaufgaben können verschiedene geometrische Darstellungsformen, sowohl für Objekte im Fokus, als auch im Kontext ausgewählt werden.
2. Relevanzorientierte Verwaltung der Sichtbarkeit mit Hilfe von *Ghosting*. Dies soll verhindern, dass Kontextelemente Fokuselemente verdecken.
3. Zeitliche-Wichtigkeit mit Hilfe von *Adaptive Fast-Forward*. Dabei ist die Abspielgeschwindigkeit der Simulation abhängig von einzelnen-, oder einer Kombination aus mehreren Wichtigkeitsfunktionen.
4. Visuelle Verbesserung der abgespielten Simulation mit Hilfe von *Motion Blur*. Die Intensität der Bewegungsunschärfe wird zusätzlich dazu verwendet, die Stärke der aktuell gezeigten Beschleunigung zu illustrieren.

Diese Arbeit wurde von Beginn an in enger Zusammenarbeit mit Biochemikern des Loschmidt-Labors in Brünn, Tschechien entwickelt. Gemeinsam haben wir verschiedene Anwendungsfälle analysiert und die Flexibilität unserer neuartiger Fokus & Kontext Visualisierung demonstriert.

# Abstract

The aim of this thesis is a novel real-time visualization approach for exploring *molecular dynamics* (MD-)simulations. Through the constantly improving hardware and ever-increasing computing power, MD-simulations are more easily available. Additionally, they consist of hundreds, thousands or even millions of individual simulation frames and are getting more and more detailed. The calculation of such simulations is no longer limited by algorithms or hardware, nevertheless it is still not possible to efficiently explore this huge amount of simulation data, as animated 3D visualization, with ordinary and well established visualization tools. Using current software tools, the exploration of such long simulations takes too much time and due to the complexity of large molecular scenes, the visualizations highly suffer from visual clutter. It is therefore very likely that the user will miss important events.

Therefore, we designed a *focus & context approach* for MD-simulations that guides the user to the most relevant temporal and spatial events, and it is no longer necessary to explore the simulation in a linear fashion. Our contribution can be divided into the following four topics:

1. Spatial importance through different *levels of detail*. Depending on the type of research task, different geometrical representations can be selected for both, focus- and context elements.
2. Importance driven visibility management through *ghosting*, to prevent context elements from occluding focus elements.
3. Temporal importance through *adaptive fast-forward*. The playback speed of the simulation is thereby dependent on a single or a combination of multiple importance functions.
4. Visual declutter of accumulated frames through *motion blur*, which additionally illustrates the playback speed-up.

Since the very beginning, this work was developed in close cooperation with biochemists from the Loschmidt Laboratories in Brno, Czech Republic. Together, we analyzed different use cases demonstrating the flexibility of our novel focus & context approach.

# Contents

CHAPTER 1

# Introduction

*Molecular dynamic* (MD-)simulations play an important role in various scientific fields like chemistry, physics, biotechnology, agronomy, medicine, and pharmaceutical research. In this context, they are used to analyze the functionality, motion, mutation, and folding of proteins. Nowadays, biochemists use statistics and statistical tests in combination with static graphs, like line plots, to explore these dynamic interactions. To be able to generate new hypotheses during this exploratory analysis and to fully understand not only the focus, but also the context of such real-time simulations, currently used 3D visualizations are no longer sufficient.

Through the constantly improving hardware and ever-increasing computing power, MD-simulations are more easily available and therefore used more often. Additionally, they do not only get more detailed and more complex, but also much longer and consist of hundreds, thousands, or even millions of individual frames. Using current software tools and interfaces, the exploration of such long simulations, as animated 3D visualizations, takes too much time and it is no longer possible to explore the simulation in a linear fashion. Without precise guidance, the user has to repeatedly watch the whole MD-simulation, otherwise it is very likely that important events will be missed.

After some formative interviews with biochemists from the Loschmidt Laboratories, it was clearly visible that not only a spatial but also a temporal importance is necessary:

- A *spatial importance* is required because it is already challenging to explore individual not animated MD-simulation keyframes. Without spatial importance, scenes are too detailed, complex, and occluded to generate new hypotheses.
- A *temporal importance function* is needed since simply removing individual frames or sections of an MD-simulation, to reduce the overall length of the animation, is not enough to summarize a simulation or focus on certain parts of the animation.

To still be able to explore and gain insights into long MD-simulations that consist of an immense number of individual frames, our goal was to design and implement a combination of *spatial* and *temporal focus & context techniques*. Our novel, interactive, and animated 3D presentation technique gives important parts of the simulation more time and provides additional graphics resources. This should help guiding the user to the most relevant temporal and spatial events of the simulation without losing the context or general overview of the animation. It was achieved using various *adjustable importance functions* that allow the user to define individual molecules, like ligands or residues, as spatial focus elements and biochemical properties, such as speed or stuckness of molecules, as temporal importance for the playback speed of an animation.

Furthermore, we will introduce additional *declutter* algorithms such as *ghosting* and *motion blur*. Ghosting is thereby used to prevent context structures from occluding focus elements and motion blur serves as natural indicator for the playback speed and is simultaneously used to reduce visual flickering. The resulting visualization provides a guided exploration of a compact and more compressed MD-simulation that focuses on essential parts and relevant events of the animation.

A software tool for analyzing and visualizing protein structures, called *CAVER Analyst* [Ana], is currently developed at Masaryk University in Brno, Czech Republic. Using CAVER Analyst, biochemists can already load and process MD-simulations. Additionally, it allows the user to play, pause, forward, or skip parts of the animation using classic *video cassette recorder* (VCR) media controls. A screenshot of this molecular exploration tool is shown in Figure 1.1.

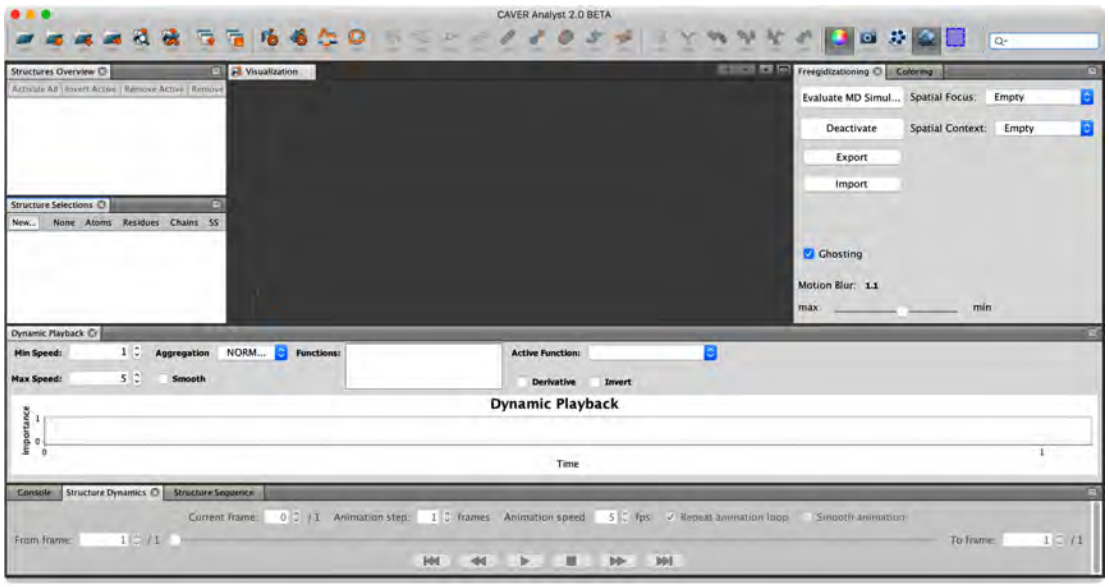


Figure 1.1: Graphical user interface of CAVER Analyst [Ana], which is a well established software tool for the exploration of MD-simulations, used by biochemists.

Naturally, there are many other 3D MD-simulation visualization tools with similar capabilities. Nevertheless, there is no tool that adjusts the visual presentation to what the user is most interested in, which would at the same time already reduce visual clutter.

In the following, we will provide two examples of such detailed, cluttered, and visually dense scenes including possible solutions using spatial importance. An example for challenging *ligand-residue* interaction is shown in Figure 1.2 and problematic *water-protein* interaction is shown in Figure 1.3.

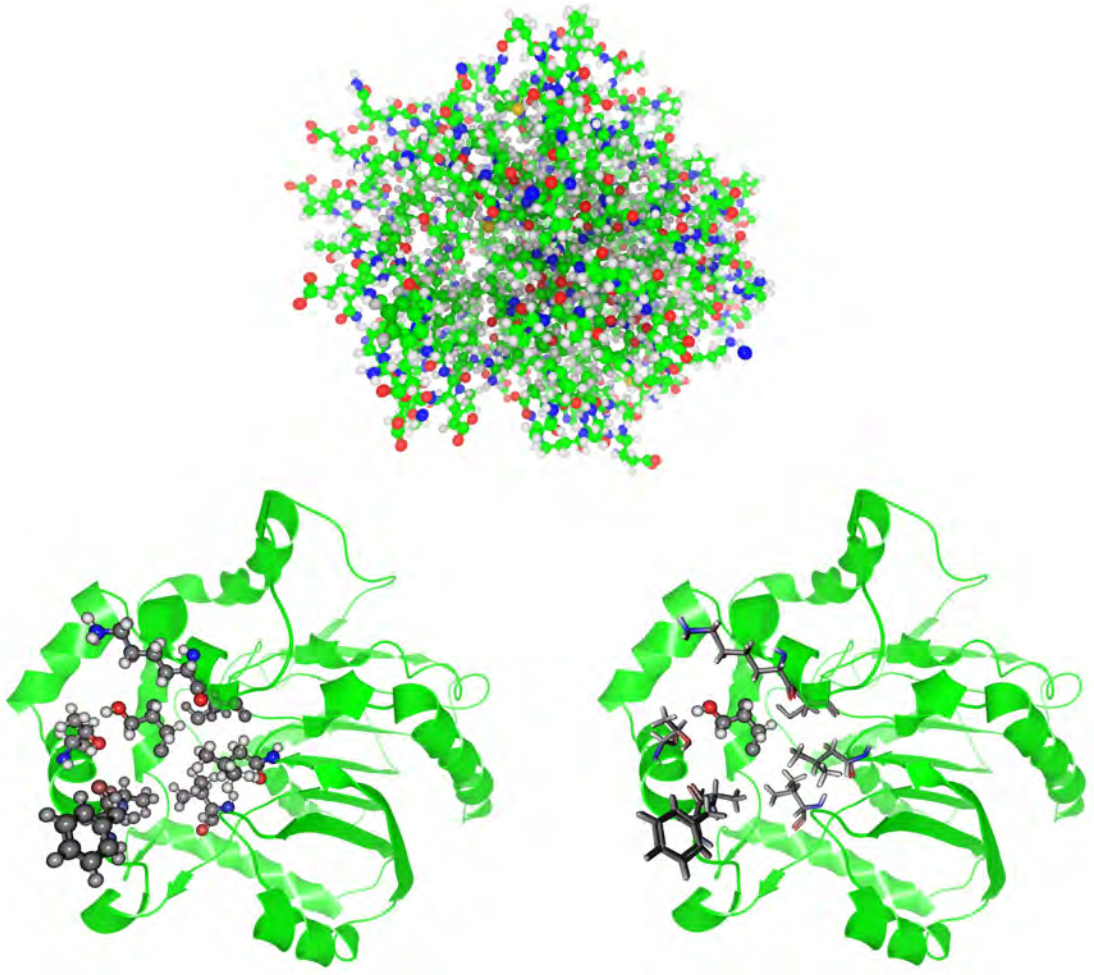

Figure 1.2: Comparison of: (top) Visualization of all atoms of a protein, without spatial importance, using a *balls & sticks* representation. (bottom left) Simplified visualization of the *secondary structure* of a protein, using a *cartoon* representation, and a detailed *balls & sticks* representation of the ligand and surrounding residues. (bottom right) Simplified visualization of the *secondary structure* using a *cartoon* representation, a simplified *sticks* representation of residues, and a detailed *balls & sticks* representation of the ligand. Renderings produced using CAVER Analyst [Ana].

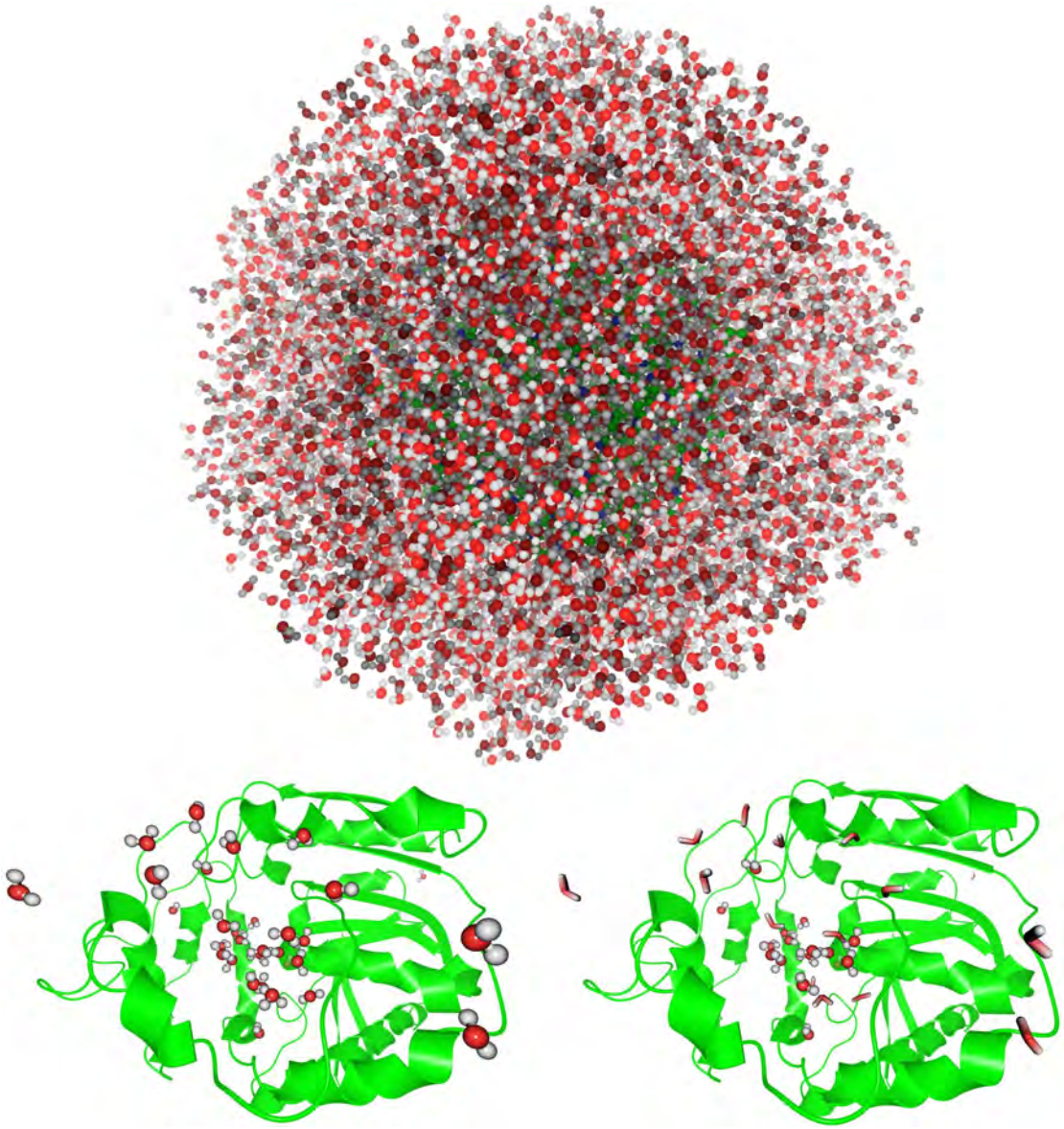

Figure 1.3: Comparison of: (top) Visualization of all atoms of a protein including surrounding water molecules, without spatial importance, using a *balls & sticks* representation. (bottom left) Simplified visualization of the *secondary structure* of a protein using a *cartoon* representation and a detailed *balls & sticks* representation of inside waters or waters that will enter the protein. (bottom right) Simplified visualization of the *secondary structure*, using a *cartoon* representation, a detailed *balls & sticks* representation of inside waters, and a simplified *sticks* representation of waters that are currently outside. Renderings produced using CAVER Analyst [Ana].

CHAPTER 2

# Background

This chapter provides definitions and explanations of important terms, from different scientific domains like *(bio)chemistry*, *physics*, *biology*, and other *natural sciences*, which are necessary to understand the following contributions and implementation details.

For a better overview, this chapter is divided into the following sections: Section 2.1 will focus on general definitions of atoms and molecules, including their representation types. Section 2.2 will explain molecular dynamics simulations in general, their historical development, how they are generated, and why their exploration is challenging.

## 2.1 Molecular Visualization

Kozlíková et al. [KKF+17] define *atoms* as smallest units that every matter consists of, independent of whether it is in a solid, liquid or gaseous aggregate state. If multiple atoms form chemical bonds, the resulting structure is then called a *molecule*. Molecules represent the smallest units of chemical compounds. If a molecule consists of thousands or millions of atoms, for example, like very large proteins, DNA or RNA, it is also referred to as a *macromolecule* [KKF+17]. A *ligand* represents an atom or molecule that tries to bind to different other chemical elements, to form a larger molecular structure [CWM99]. *Residues* represent chemical elements such as *amino acids* that are part of larger molecular structures, like proteins. They consist of single or multiple atoms that remain as a by-product of a chemical reaction. If biochemists want to analyze ligand-protein or ligand-residue interactions, they need visualization tools that allow them to study these processes in detail.

In the following section, we focus on publications from Olson [Ols18], who presents different types of structural molecular biology visualizations and Perkins [Per05a][Per05b] who highlights chronological representations of molecular structures.

Already in 1865, Hofmann [Hof65] introduced a so-called *glyptic formulæ*, which is sometimes also referred to as *croquet ball model*. It is the most basic type of a *balls & sticks* model and represents the first model using a planar arrangement of atoms in space. An example of three different structures is shown in Figure 2.1.

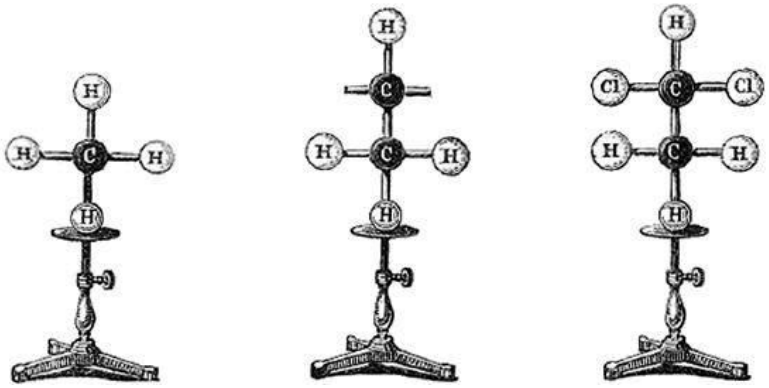


Figure 2.1: Three examples of the historically first planar balls & sticks model, introduced by Hofmann [Hof65] in 1865. (left) Methane. (middle) Ethylene. (right) Dichloroethylene. Image from Hofmann [Hof65].

Soon after that, Kekulé [Kek65] adapted this model so that it was no longer limited to planar models and further added the possibility to create double- and triple bonds. It corresponds to the balls & sticks model, which is still used nowadays in three-dimensional computer generated visualizations. A photograph of such a physically modeled structure is shown in Figure 2.2 and a digital rendering of a more complex molecule using nowadays visualization tools is shown in Figure 2.3.

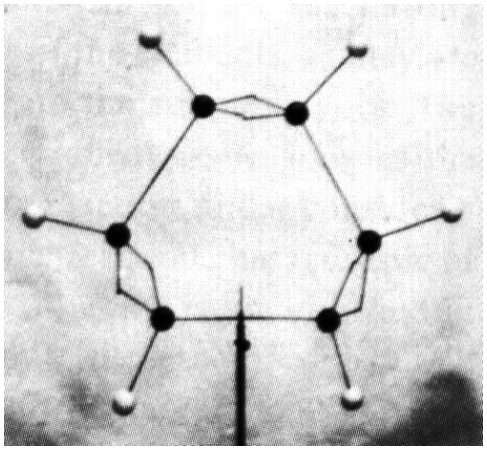

Figure 2.2: Photograph of a physical balls & sticks model that can show circular structures including double- and triple bonds, introduced by Kekulé [Kek65] in 1865. Image from Kekulé [Kek65].

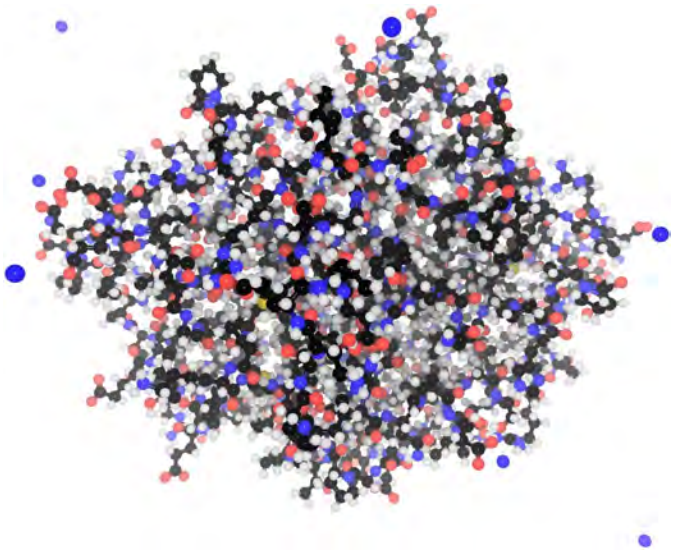

Figure 2.3: Visualization of a *balls & sticks* representation, similar to the physical model introduced by Kekulé [Kek65]. Rendering produced using CAVER Analyst [Ana].

A simplified version of the balls & sticks model is the *sticks model*. Instead of showing atoms and their bonds, it only visualizes the bonds between atoms. The advantage of this representation type is that spheres do not occlude underlying bonds. This is especially relevant for larger models. An example is shown in Figure 2.4.

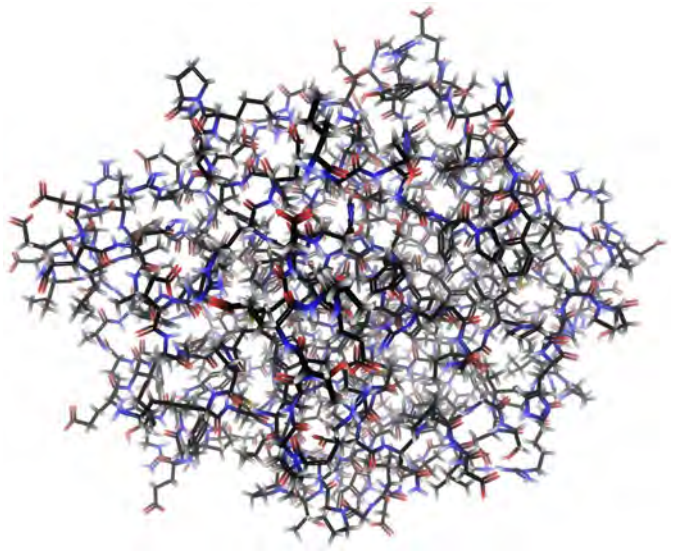

Figure 2.4: Visualization of a *sticks* representation, which is a simplified version of the balls & sticks model, showing only bonds between atoms. Rendering produced using CAVER Analyst [Ana].

In 1965, Koltun [Kol65] introduced a *space-filling* approach, which resulted in the *Corey Pauling Koltun* (CPK) representation model. This representation uses the so-called *van der Waals radius*, introduced by van der Waals in 1873 [Waa73], as sphere radius of individual atoms. The radius represents half the distance between two unbound atoms and is therefore dependent on the type of atom. An example is shown in Figure 2.5.

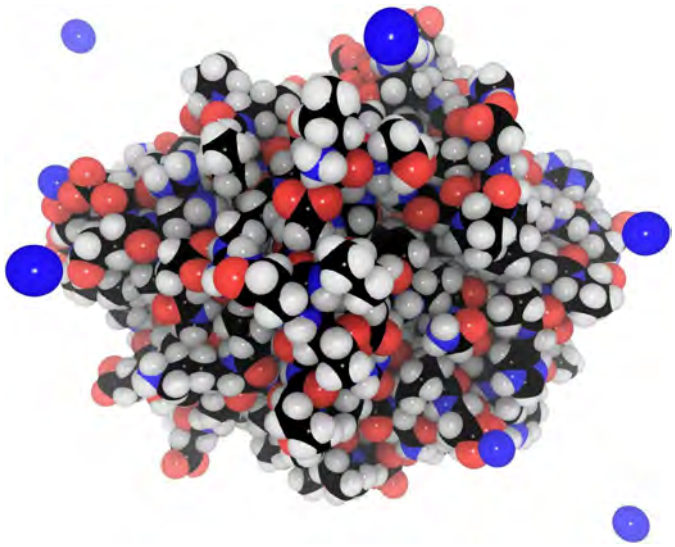

Figure 2.5: Visualization of a *space-filling* representation using the *van der Waals radii* [Waa73] and an atom coloring scheme introduced by Koltun [Kol65]. Rendering produced using CAVER Analyst [Ana].

An advantage of this representation, using radii of spheres that are dependent on atomic types, is that it immediately shows the shape and surface of the original molecule and furthermore their actual relation and relative distances to each other. Additionally, biochemical properties or other information can be encoded as colors of the spheres [KKF+17]. The default color scheme, depending on atom types, was introduced by Koltun [Kol65]. An overview of the colors, of the CPK model, is shown in Table 2.1. Typical disadvantages of the space-filling approach are that bindings and therefore also the type of bindings (single, double, triple) are not shown. In addition, the spheres very likely occlude underlying structures of atoms or hide encapsulated interior atoms. To be able to analyze the whole surface, the user has to rotate the object.

An important milestone in the history of space-filling representations that solved these occlusion problems was presented by Lee and Richards [LR71] in 1971. They visualized molecular structures as spheres, using their van der Waals radii. But instead of exploring just the surface of a molecule, this innovation allowed scientists to additionally analyze and study the interior of a molecule. The visualization was done using a stack of computer generated prints on plastic sheets, placed on top of each other. An example of their results is shown in Figure 2.6.

| | |
|---|---|
| hydrogen (**H**) | white |
| carbon (**C**) | black |
| nitrogen (**N**) | dark blue |
| oxygen (**O**) | red |
| fluorine (**F**), chlorine (**Cl**) | green |
| bromine (**Br**) | dark red |
| iodine (**I**) | dark violet |
| noble gases (**He**, **Ne**, **Ar**, **Xe**, **Kr**) | cyan |
| phosphorus (**P**) | orange |
| sulfur (**S**) | yellow |
| boron (B) | peach |
| alkali metals (**Li**, **Na**, **K**, **Rb**, **Cs**, **Fr**) | violet |
| alkaline earth metals (**Be**, **Mg**, **Ca**, **Sr**, **Ba**, **Ra**) | dark green |
| titanium (**Ti**) | gray |
| iron (**Fe**) | dark orange |
| other elements | pink |

Table 2.1: List showing default colors used by the CPK model, which was introduced by Koltun [Kol65] in 1965.

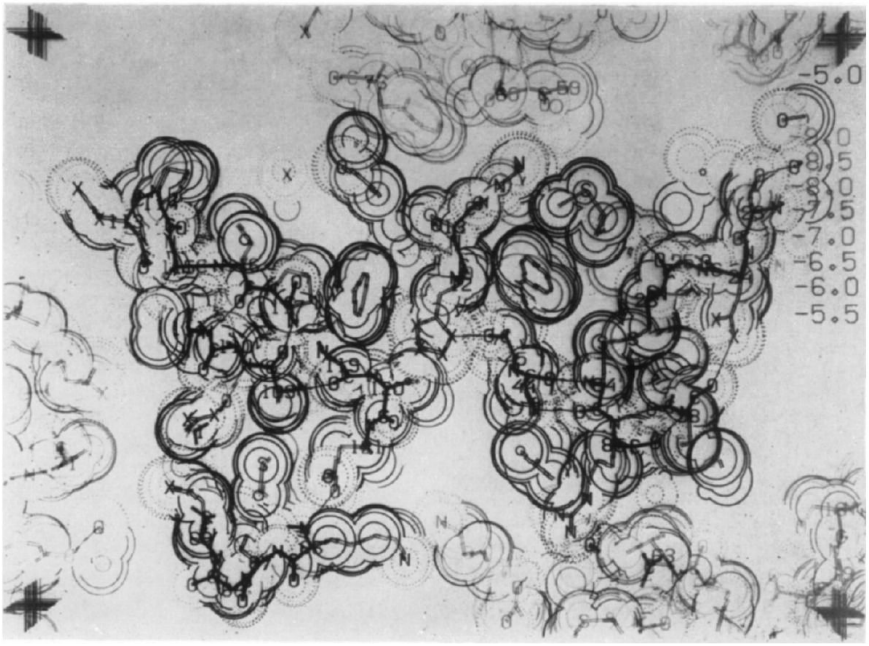


Figure 2.6: Visualization of a molecule, using van der Waals radii, that allows the exploration of interior molecular structures and prevents occlusion problems. Image from Lee and Richards [LR71].

A simplified version of the space-filling model is a representation, where spheres are reduced to single points. Instead of rendering whole spheres using their van der Waals radii, only the center of an atom is marked. On the one hand, this prevents occlusion problems, but on the other hand, this leads to problems in correctly perceiving the hierarchical spatial order of individual atoms. An example of such a simplified version, using crosses to indicate the center of atoms, is shown in Figure 2.7.

Figure 2.7: Visualization of a simplified space-filling representation using little crosses that indicate the center of atoms, instead of rendering individual spheres. Rendering produced using CAVER Analyst [Ana].

If a molecule is too large and even the bonds are too dense for a simplified stick representation, a so called *cartoon-* or *ribbon-representation* can be used. Richardson [Ric81] introduced this representation model in 1981. Instead of the surface, it represents the *secondary structure* of a molecule and typically consists of arrows, ribbons, and cylinders or spirals. According to Kozlíková et al. [KKF+17], the most common elements are the $\alpha$-helix (spiral) and the $\beta$-sheet (broad band with arrow). An example is shown in Figure 2.8.

Next, we will briefly mention currently used illumination models in molecular visualizations. In 1975, Phong [Pho75] developed a local illumination model, which was 1977 further adapted by Blinn [Bli77]. Both represent the most basic but still frequently used local illumination models that consider ambient, diffuse, and specular illumination. Such simple approaches are sufficient for smaller molecular models. But larger models that may also contain cavities, require more elaborate techniques. An example is *ambient occlusion*, which was developed based on the work of Miller [Mil94] and Zhukov et al. [ZIK98]. An example of a locally illuminated molecule, in comparison to a molecule that is lit using ambient occlusion, is shown in Figure 2.9

Figure 2.8: Example of a simplified *cartoon* representation. It visualizes the secondary structure of a molecule, instead of rendering detailed molecular structures like atoms and bonds. Rendering produced using CAVER Analyst [Ana].

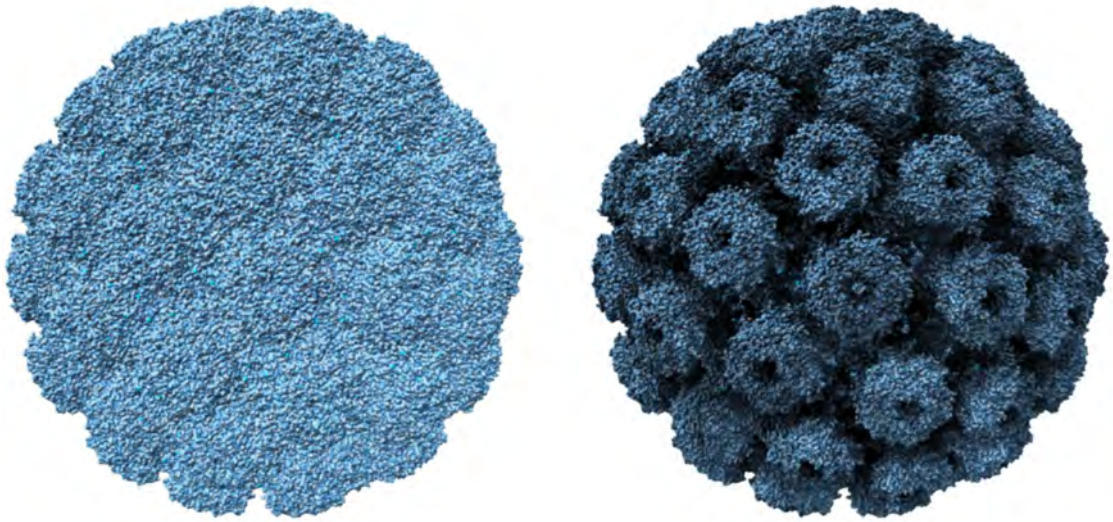

Figure 2.9: Comparison of different illumination models: (left) Local illumination that does not provide a well-visible general structure of the molecule and (right) ambient occlusion that does provide such a detailed overview. Image from Kozlíková et al. [KKF+17].

The most popular and widely used data source for molecular structure data is the *Research Collaboratory for Structural Bioinformatics* (RCSB) *Protein Data Bank* (PDB) [BWF+]. It is an online data base for three-dimensional datasets of large molecules. The data are mostly uploaded by biologists and chemists and are used for education or research. It is managed by *Worldwide Protein Data Bank* and all their data is freely and publicly available. In August 2018, the data base contained 143.392 different datasets.

## 2.2 Molecular Dynamics Simulations

The following definitions about *molecular dynamic* (MD) simulations are based on standard literature from Frenkel and Smit [FS01], Andrew R. Leach [Lea09], and a recently published state of the art report by Kozlíková et al. [KKF+17] about the visualization of biomolecular structures.

Frenkel and Smit [FS01] describe MD-simulations as many-body systems where particles obey the *law of mechanics*. The result of such a simulation is a collection of hundreds, thousands, or millions of individual keyframes. Each keyframe contains the spatial information of the molecule, trajectories of individual elements, and additional biochemical properties. According to Hansson et al. [HOvG02], MD-simulations are used to better understand natural processes, simulated on the basis of different timescales. Additionally, they can be used to study molecular interactions like the ligand binding or to analyze the conformational space during ligand docking. Hansson et al. [HOvG02] furthermore mention important properties of MD-simulations like realistic representations of molecules and their surrounding, the size of a molecular structure, and the duration of the simulation.

Historically, the first MD-simulation was computed by Alder and Wainwright [AW57], in 1957. It was based on a simplified system consisting of hard spheres. Within such systems, all objects follow *Newton's law of motion* [Lea09], because they either remain still, or move constantly in straight lines until another force acts on them. If two spheres collide, they bounce off and move constantly in straight lines again. The goal of such simulations is reaching an equilibrium for a chosen set of parameters. In addition, it can be observed which spheres collide, the point in time they collide, and how the collision affects their behavior. An example of these early simulations is shown in Figure 2.10.

Unfortunately, this model provides only most basic insights into molecular interactions. Rahman [Rah64] introduced a more complex simulation system that additionally considers changes of forces, if a particle moves or collides with a different particle. According to Leach [Lea09], the resulting complexity can no longer be solved analytically, but requires *finite difference methods*.

In the following, we will describe how initializing and running an MD-simulation can be done. According to Leach [Lea09] and Frenkel et al. [FS01], the first step is finding an initial setting based on real experiments or abstract thought experiments. This includes defining the number of elements that need to be simulated. In addition, elements should be placed according to the corresponding structure of interest and so that they do not

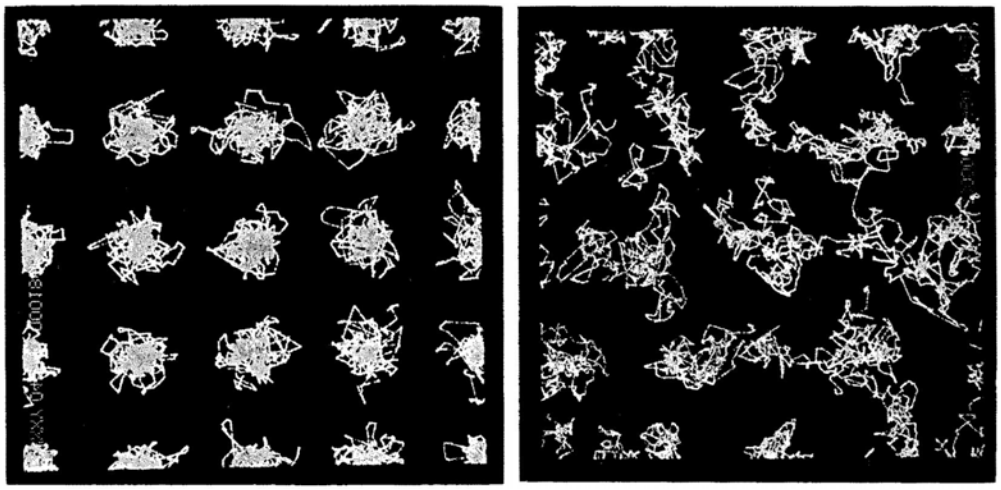

Figure 2.10: Comparison of two different simulations showing pathways of 32 hard spheres after 3000 collisions. The particles were placed in (left) a solid material or (right) a fluid. Image from Alder and Wainwright [AW59].

overlap. In the next step, starting velocities are assigned to individual atoms, depending on the chosen temperature of the system. During the next step, the actual computation is done depending on all kinds of forces that can affect atoms regardless of whether they are bound or not. This is done using Newton's law of motion until the system becomes stable or until the simulated system reaches an initially defined equilibrium, which can be dependent on factors such as temperature, pressure, kinetic or total energy. If these parameters were chosen correctly, the simulation terminates. Otherwise, the computation needs to be restarted using different starting values or a changed equilibrium. If an equilibrium is reached, the position data, but also additional parameters such as energies or velocities of the individual atoms, are stored per keyframe.

According to Leach [Lea09], the biggest challenge in MD-simulations is not just the simulation of individual atoms, but the computation of collisions of entire molecules. This is a challenge because collisions between molecules cannot be represented by single rigid spheres like it was done with atoms. The consequences of such complex collisions are dependent on the position and orientation of the molecules, because of their different shapes and compositions. If a molecular structure is not rigid but flexible, further intramolecular interactions must be considered.

CHAPTER 3

# Related Work

The following chapter contains related work from a wide variety of scientific domains like *visualization*, *computer graphics*, *computer vision*, *illustration*, *multimedia*, *human computer interaction*, *bioinformatics*, and *biochemistry*. Since, to the best of our knowledge, there is no comparable related work that deals with *spatio-temporal real-time visualizations of MD-simulations*, this work was inspired by similar problem formulations from related scientific domains. Examples include fundamental challenges of rendering large animated MD-simulations in 3D while interacting with the visualization in real-time, typical *motion-based focus & context techniques* that are mainly used in the arts or video domain, and the adaption of playback speed of MD-simulations, which is closely related to the *dynamic playback adaption* of surveillance camera footage or highlighting sections of sport videos.

Section 3.1 will focus on different 2D and 3D visualization types of MD-simulations that allow domain experts to explore and analyze previously unknown properties of the simulations. Section 3.2 describes different *focus & context visualization strategies* for static visualizations, animated visualizations, and in connection with molecular visualizations. Section 3.3 will focus on *non-linear* and *adaptive video playbacks*, *different fast-forward strategies*, various possibilities for the user to interact with time-varying data, and examples of *visual feedback* about the currently used playback speed of a video.

## 3.1 Visualizations of MD-Simulations

In the following, we will introduce the current state of the art in visualizing MD-simulations in 3D as well as in 2D, that allows domain experts to explore the dataset, gain novel insights, and confirm or disprove assumptions. Therefore, we have chosen two different approaches as representatives: In 3D, we will consider a multi-scale saliency function to extract meaningful time steps of MD-simulations and in 2D, we will focus on state-transition graphs.

Patro et al. [PIV10] highlight the importance of meaningful visualizations of MD-simulations. This is necessary, since computers are nowadays capable of computing MD-simulations with a length and complexity like never before. Visualizing such simulations is not only restricted by hardware devices and limited screen resolutions, but also overwhelming to the user and human absorbing capacities. The authors therefore recommend reconsidering the visual representation of MD-simulation and also the selected range of represented elements. Similar to our approach, the main interest of Patro et al. [PIV10] is extracting most relevant events within an MD-simulation, to be able to summarize an MD-simulation, provide meaningful fast-forwards, introduce an indexing, and improve further explorations. In their approach, the importance of every keyframe is computed based on how much the per-atom saliency measure differs from the previous and subsequent keyframe. This computation is then performed over multiple scales of the simulation. A visualization of this process flow is shown in Figure 3.1.

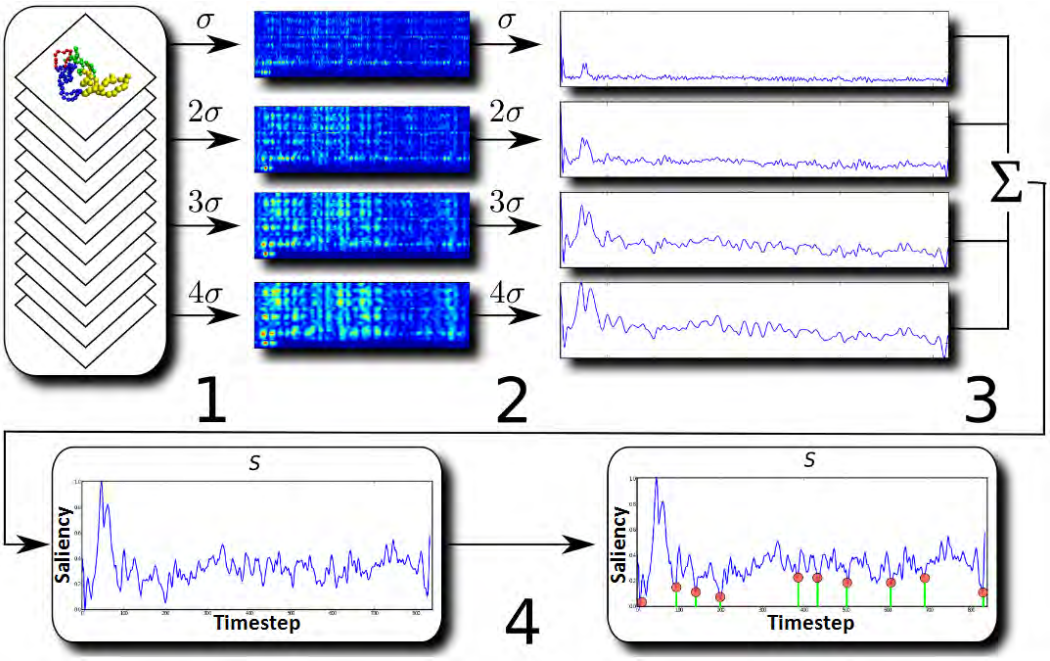


Figure 3.1: Keyframe extraction process: (Step 1) Computation of saliency values for every atom, at different scales. (Step 2) For every scale, merger of individual saliency values into representative functions. (Step 3) Combining saliency functions into a single multi-scale saliency function. (Step 4) Selection of meaningful keyframes that represent the MD-simulation depending on *local minima* of the function. Image from Patro et al. [PIV10].

The final result is a collection of significant keyframes that represent the MD-simulation. A time step represents a meaningful keyframe, if the multi-scale saliency function reaches a local extremum. A local minimum, for example, indicates a keyframe that is highly representative. An example is shown in Figure 3.2.

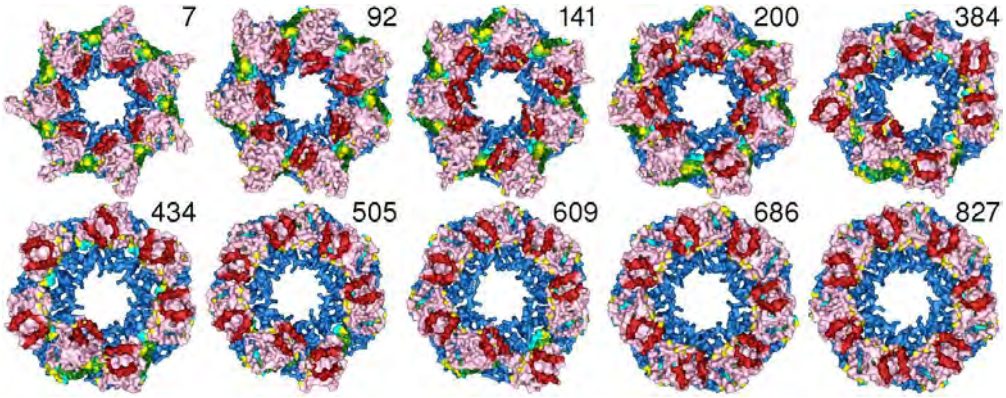


Figure 3.2: Visualization of 10 meaningful keyframes, of a complex MD-simulation consisting of 3 668 atoms and 834 keyframes. The function for extracting these keyframes is shown in Figure 3.1. Image from Patro et al. [PIV10].

Patro et al. [PIB+11] present a different approach of summarizing MD-simulations as state-transition graphs. The transitions and the number of transitions, which can be encoded as thickness of an edge, represent the flow of a biomolecule. Major challenges here are the extraction of states, that are used as vertices in the graph, and then the identification of transitions between these states. As soon as this computation is finished, different graph layout algorithms can be applied. An example of two different results of such a visualization is shown in Figure 3.3.

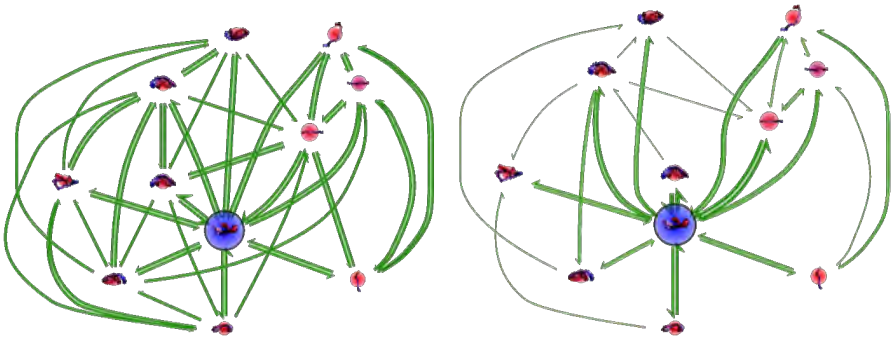

Figure 3.3: Visualization of an MD-simulation as state-transition graphs, consisting of 11 different states. (left) A graph with uniform edge width. (right) A graph where the number of transitions is encoded using the edge width. Image from Patro et al. [PIB+11].

In comparison to their results, our approach was not only finding representative visualizations and meaningful keyframes. We wanted to keep the simulation interactive and display the entire simulation as animation, using a computed importance function for adaptive fast-forward playback. In our case, the importance function is not dependent on individual representative states but on other meaningful values that can be selected according to the focus of interest, such as the speed or stuckness of a ligand.

## 3.2 Focus & Context Visualizations

In this section, we want to highlight rendering- and visualization strategies that provide an intuitive categorization of important elements (that are in focus and therefore displayed with a greater level of detail) and unimportant objects (that are in context and therefore displayed in a simplified way). Hauser [Hau04] introduced a generalized definition of focus & context visualization:

> *"focus+context visualization is the uneven use of graphics resources (space, opacity, color, etc.) for visualization with the purpose of visually discriminating those parts of the data which are in focus from the respective context, i.e., the rest of the data"* [Hau04].

MD-simulations consist of hundreds, thousands, or millions of atoms and therefore, tend to be visually cluttered. According to Rosenholtz et al. [RLN07] *visual clutter* negatively influences decision-making, reduces the performance of object recognition, prevents the user from being able to visually separate objects in the scene, and worsens the short-term memory. While visualizing MD-simulations, it is therefore important to reduce clutter and apply focus & context approaches similar to the ones explained in the following.

For a better overview, this section is divided into three parts: Subsection 3.2.1 describes focus & context approaches of static visualizations from domains like *volume rendering*, *stylized rendering*, and *medical illustrations*. Subsection 3.2.2 highlights focus & context visualizations for animations including examples from *video processing* and the analysis of video records. Subsection 3.2.3 finally describes focus & context approaches in the context of molecular visualizations.

### 3.2.1 Focus & Context in Static Visualizations

Viola et al. [VFSG06] highlight the importance of focus & context visualizations based on volume rendering. Using their visualization, the user selects an object of interest and the tool itself automatically recommends a possible point of view, to explore the object in focus. To prevent occlusion problems, the authors additionally introduce cut-away views and different visual representations of highlighted objects. Thereby, it is always possible to distinguish between context information and visually emphasized focus objects. To further de-emphasis context information, the authors additionally mention *(semantic) depth of field effects*, introduced by Kosara et al. [KMH01]. An example of different highlighted organs within a human torso is shown in Figure 3.4.

This confirms our initial assumption that particularly dense molecular structures and occluded scenes could highly benefit from focus & context visualizations. In MD-simulations, the user can select individual molecular focus structures, instead of organs, bones, or blood vessels. The focus is then visually emphasized and surrounding structures are abstracted. Similar to their approach, objects can further be highlighted by changing their appearance and color, depending on chemical properties or user preferences.

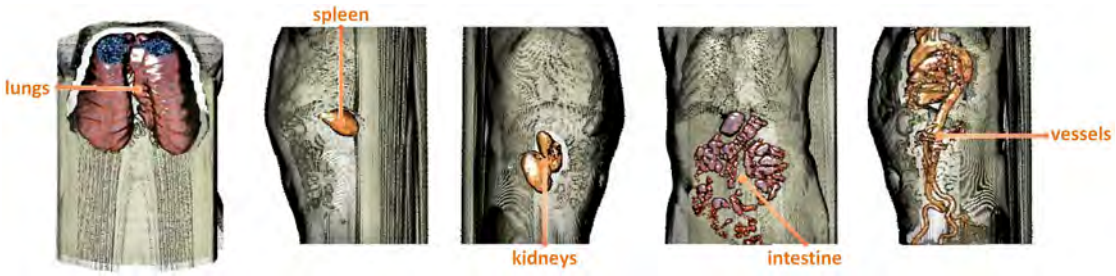


Figure 3.4: Different volume rendering examples with focus on lungs, spleen, kidneys, intestines, or vessels. The human torso additionally provides context information. Image from Viola et al. [VFSG06].

Bruckner et al. [BGKG06] introduce *context-preserving volume rendering*, sometimes also referred to as *ghosting*. Initially, this is a technique mostly used by illustrators, to show functionality or inner life of technical devices or organic objects that would not be visible otherwise. They introduced this, since simple clipping of occluding objects mostly removes context preserving elements and often even parts of elements that are in focus. According to Bruckner et al. [BGKG06], this is done as follows: The transparency of an object is dependent on the importance of the object itself, and the complexity and shape of a surface. Especially less detailed surfaces become transparent and only necessary edges, that are needed to mentally complete the visible object, are still displayed. An example of a rendered hand, focusing on inner structures like blood vessels, muscles, and bones, is shown in Figure 3.5.

Figure 3.5: Rendering of a hand focusing on inner structures like blood vessels, muscles, and bones. Image from Bruckner et al. [BGKG06].

We have noticed that the use of simplified cartoon representations, to provide context information, tends to occlude visually more important elements in the inside of a molecular structure such as atoms, ligands, or residues represented as balls & sticks. Although the approach of Bruckner et al. [BGKG06] was initially introduced for volume rendering, we came to the conclusion that the exploration of MD-simulations would also benefit from semi-transparent and context preserving rendering strategies. During our research, this assumption has been repeatedly confirmed by biochemists.

A related approach is presented by Viola et al. [VKG04]. The authors introduce a view-dependent *importance-driven volume rendering* visualization. It represents an importance driven focus & context approach dependent on object importance, levels of sparseness, and an importance composition. The object importance represents a positive numerical value describing the visual importance of every structure. The level of sparseness provides different levels of abstraction using opacity and color saturation modulation, screen-door transparency, and volume thinning. Examples of the three different levels of abstraction are shown in Figure 3.6. According to Viola et al. [VKG04], *maximum importance projection* (MImP) and *average importance-compositing* are two possible approaches for importance composition. Using MImp, for example, only the most important objects are displayed and occluding structures are fully transparent.

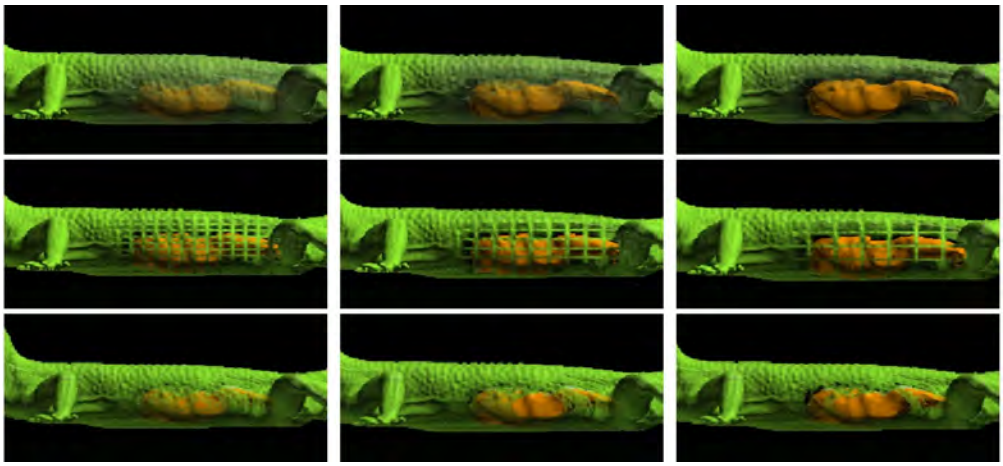

Figure 3.6: Visualization of three different levels of abstraction using variable factors of sparseness. (Top), opacity and color saturation modulation, (middle) screen-door transparency, and (bottom) volume thinning. Image from Viola et al. [VKG04].

Cole et al. [CDF+06] introduce *stylized rendering* used for 3D models, in the domain of non-photorealistic rendering. Initially, the user selects the camera position, a region of interest, and a visual style. The system then creates a rendering that intuitively draws the user's attention to certain parts of the visualization, which the authors call *"stylized focus"*. In addition, the authors mention that this approach can not only be used for static scenes, but also for dynamic scenes in which the user's attention is guided from one region of interest to the next, which they then call *"stylized focus pull"*. This is done

using different shading effects, color variations, various textures, etc. Their approach originates from the arts domain where artists use different techniques to highlight parts of drawings, paintings, or illustrations. A collection of renderings of an architectural model, were the focus changes from buildings in the front, to skyscrapers in the back, is shown in Figure 3.7.

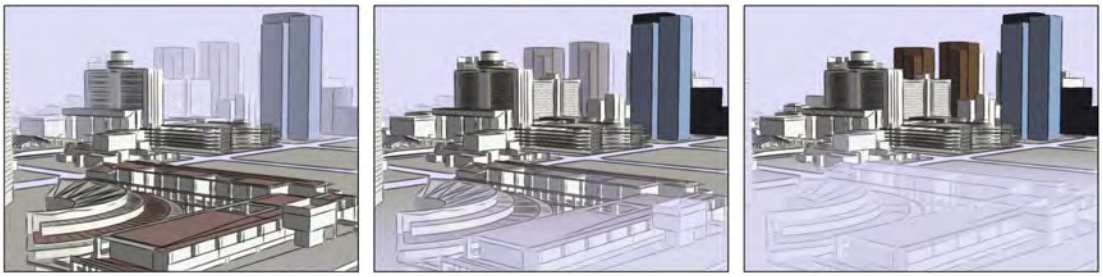

Figure 3.7: Different illustrations were the focus is (left) on buildings in the front, (middle) changes from front to back, and is (right) on the skyscrapers in the background. Image from Cole et al. [CDF+06].

Similar to the approach from Cole et al. [CDF+06], we use colors and color transitions, transparency of surfaces, diverse representation types of atoms, and different levels of detail of molecular structures, to direct the user's attention to specific elements.

A different approach for medical illustrations, instead of architectural renderings, is presented by Baer et al. [BALP09]. In recent years, many novel representation types of objects in focus have been developed and in their work, they compare three different representation types of lymph nodes: cutaways, stippling representations, and different colorings. According to Baer et al. [BALP09], the challenge today is to find the right visualization for every use case. A user study has shown that all participants preferred additional guidance, like focus & context approaches, in complex medical visualizations. The resulting user study had the following order of popularity: cutaways, stippling, and red coloring. Cutaways was thereby the most accurate method including the shortest reaction time. Additionally, there was no significant difference between stippling and red coloring, concerning reaction time and accuracy. The authors additionally emphasize the fact that most users achieved the best results with cutaways, but still preferred different technique. Therefore, it was important for us to not only implement the most efficient visualization, but in addition allow the user to choose between different focus & context approaches. An example of all three lymph node representations is shown in Figure 3.8.

Although most of these approaches are also applicable in molecular visualizations, MD-simulations are still challenging especially since scenes are no longer static, but animated simulations consisting of thousands of keyframes and millions of atoms that can move freely in space. In the next subsection, we will therefore explain different focus & context approaches for animations.

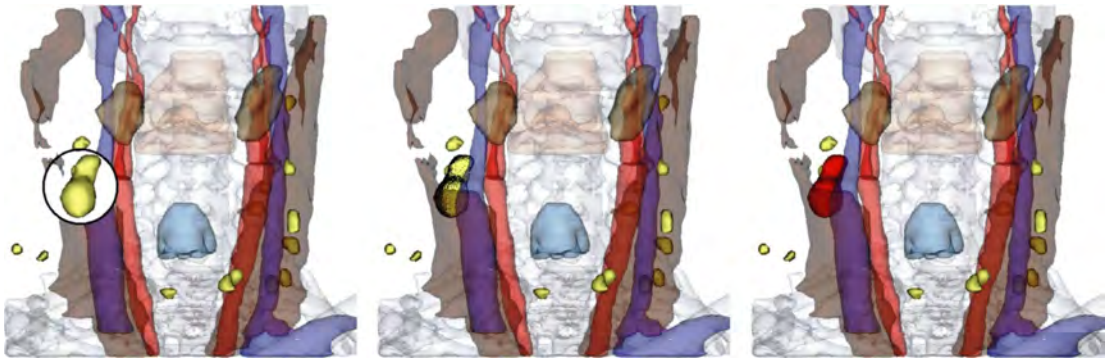

Figure 3.8: Collection of different focus & context approaches that are used to highlight a lymph node: (left) Using a cutaway approach, (middle) highlighting the object using stipplings, and (right) changing the color to red to further emphasize the object of interest. Image from Baer et al. [BALP09].

### 3.2.2 Focus & Context for Animations

In the following, we will describe a general time model for time-varying visualizations and then, present motion-based focus & context approaches from evolutions of building constructions and the field of video processing that are also applicable in MD-simulations.

Wolter et al. [WAH+09] present a fundamental time model used for interactions and visualizations of time-varying data, independent of the exact time scale of the underlying data. Depending on the type of a visualization, time spans from seconds, minutes, days, months, years, and centuries or milliseconds, microseconds, and nanoseconds need to be considered. Independent of the exact time interval, the underlying data needs to be mapped to a different format, so that the user can explore the visualization within seconds or minutes. The authors define the following fundamental time model:

> **User Time** represents the linear, acyclic, and continuous real-time we live in. It is the time within which a user can interact with the visualization or react to certain events.
>
> **Simulation Time** represents the mapping between continuous user time and discrete time steps of a simulation. In our case, it represents the duration of an atom being displayed on screen.
>
> **Time Steps** represent discrete frames that are produced by a simulation. In our case, it represents individual *keyframes* of an MD-simulation. It is the short duration where a single keyframe of the simulation is valid.

After defining a general time model that can be used for visualizations from various scientific domains, in the following, we will focus on representative algorithms that apply focus & context approaches to time-varying data using such fundamental time models.

Carvalho et al. [CDSRC08] introduce a focus & context approach based on a *temporal degree of interest* (TDOI), which can be used for discrete time-dependent data. Furthermore, TDOI defines visual properties such as color or transparency, but also other non-photorealistic approaches such as edge variations or sketch representations. Typical application areas are *spatio-temporal* evolutions of building constructions. An example is shown in Figure 3.9.

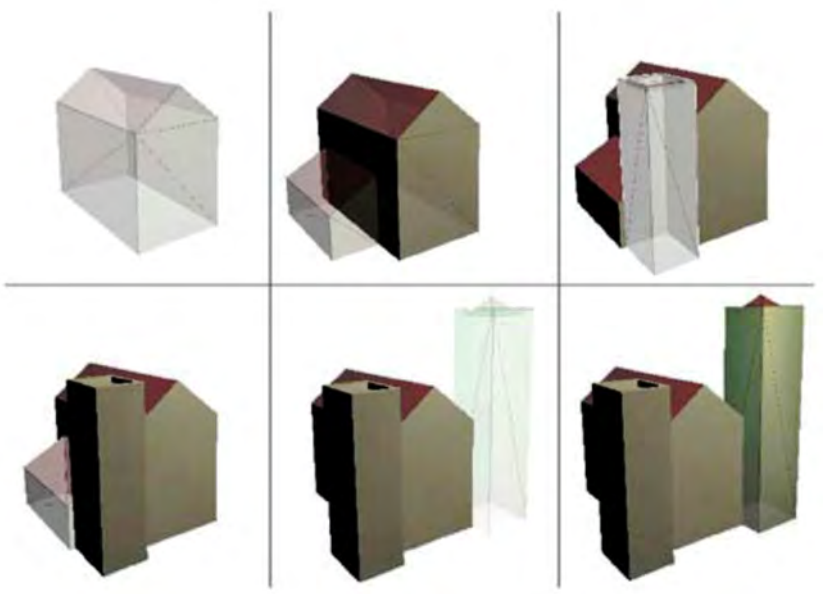

Figure 3.9: Six examples of a spatio-temporal evolution of a building construction. Image from Carvalho et al. [CDSRC08].

Carvalho et al. [CDSRC08] additionally mention challenges, especially if continuous data, instead of discrete time steps, are used. Nevertheless, we tried to adapt their approach and used a similar TDOI functions for our molecular visualization of MD-simulations.

Bai et al. [BAAR12] introduce a *de-animation algorithm* that makes it easier for the user to observe and focus on certain movements that may be otherwise hard to see, such as the precise finger movements while someone plays a guitar. It is challenging, since not only the fingers are moving, but also the guitar itself. An example of a musician playing the guitar is shown in Figure 3.10.

In MD-simulations, a similar approach that highly reduces the motion of the surrounding molecule could be used. Afterwards, only trajectories of individual focus elements, such as residues or ligands, are visible. Using such an approach, especially atoms and their bonds are challenging since bonds and interactions between atoms would be distorted and can no longer be displayed realistically. For biochemists, it would be impossible to draw meaningful conclusions from such visualizations.

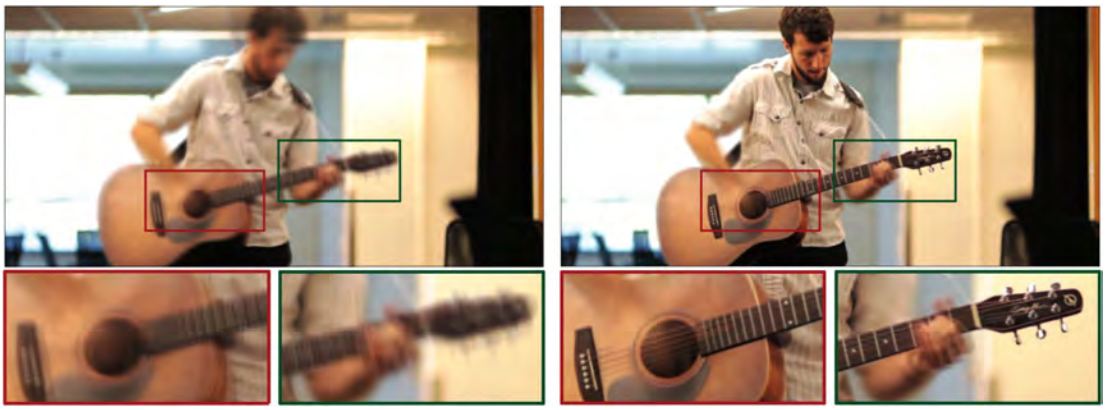

Figure 3.10: Comparison of (left) a video of a musician playing the guitar resulting in a totally blurred image and (right) a de-animated video of the same scene resulting in a sharp background and precise finger movements. Image from Bai et al. [BAAR12].

Additionally, the Bai et al. [BAAR12] refer to *motion magnification*, to emphasize small details or movements. It was introduced by Liu et al. [LTF+05] and deals with the exact opposite. Their approach allows the user to focus on possible important elements that are otherwise invisible. A video of a girl sitting on a swing can be processed in a way that afterwards, even the beam on which the swing was mounted, starts visibly resonating. An example of this video is shown in Figure 3.11.

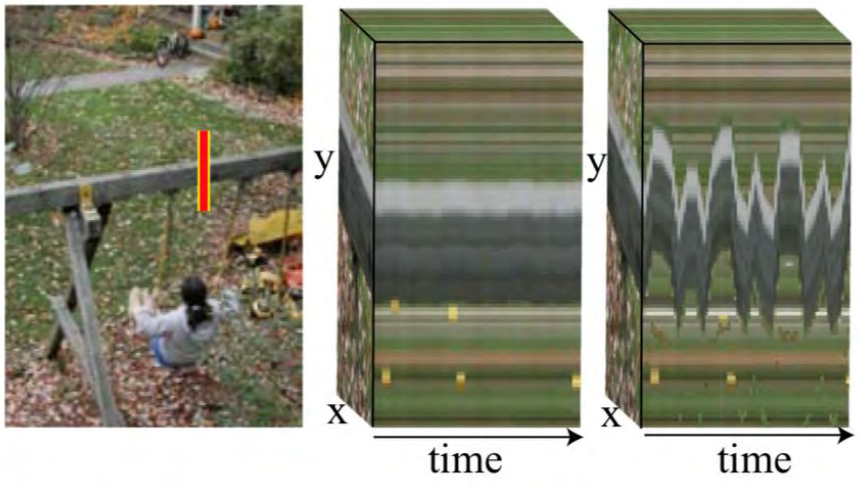


Figure 3.11: Composition of a video of a girl sitting on a swing. (left) A single screenshot including the highlighted region that is magnified. (middle) The sequence before the magnification and (right) the sequence after the magnification. Image from Liu et al. [LTF+05].

MD-simulations can generate similar effects to motion magnification, by rendering a geometric simplification of the simulation. Less important elements are displayed simplified and objects of higher importance are displayed in full detail. Therefore, movements of important objects are easier to observe and even smallest movements are emphasized while movements of unimportant objects are smoothed out since they are rendered in a more simplified way.

Since MD-simulations are not videos, further molecular properties can be used for their visualization. For example, the exact spatial position of each element, when and where context elements occlude or intersect focus structures, the periodic table of elements, colors, van der Waals radii, and many other properties are known. Furthermore, all this data can be updated in real-time, for example, if the user interacts with the scene, modifies rendering types or colors of molecular structures, or simply changes the viewing perspective. All these possibilities and similar approaches that can be used for MD-simulations will be discussed in detail in the next subsection.

### 3.2.3 Focus & Context for Molecular Visualizations

In this subsection, we will discuss related work from focus & context visualizations of biomolecular structures using a state of the art report published by Kozlíková et al. [KKF+17]. Afterwards, we will present an example by Cerqueira et al. [CBFR08], showing the importance of visualizing ligand-protein interactions using various rendering strategies and different levels of abstraction.

Kozlíková et al. [KKF+17] provide insights into currently used visual representations of molecules. This can already be challenging since computers can simulate highly complex and constantly increasing molecular processes. It becomes even more challenging, if such simulations need to be visualized in real-time, while domain experts can still change parameters and interact with the resulting focus & context visualization. Therefore, the authors [KKF+17] distinguish between two different types of representations:

> **Static geometry** which is similar to visualizing still images. According to Kozlíková et al. [KKF+17], static images can still be used to express dynamic interactions or results that originate from interaction processes.
>
> **Animations** which do not simply represent pre-rendered videos, but a visualization that changes simultaneously to user interactions.

A common way of rendering balls & sticks models that represent atoms as spheres (using their van der Waals radii) and bonds between atoms as cylinders, is typically done through triangulation. That is, because GPU hardware is optimized for such data structures. Kozlíková et al. [KKF+17] highlight the importance of additionally improving this rendering processes, for example, by using impostors, which were presented by Montani et al. [MTC06]. Instead of rendering highly tessellated geometry of atoms and bonds that consists of hundreds, thousands, or even millions of vertices, impostor rendering requires much less vertices that need to preprocessed. This or similar speed-up

approaches are necessary, to still be able to interact with a large, detailed, and animated MD-simulation in real-time. An example of how a complex molecular structure can be rendered using impostors is shown in Figure 3.12.

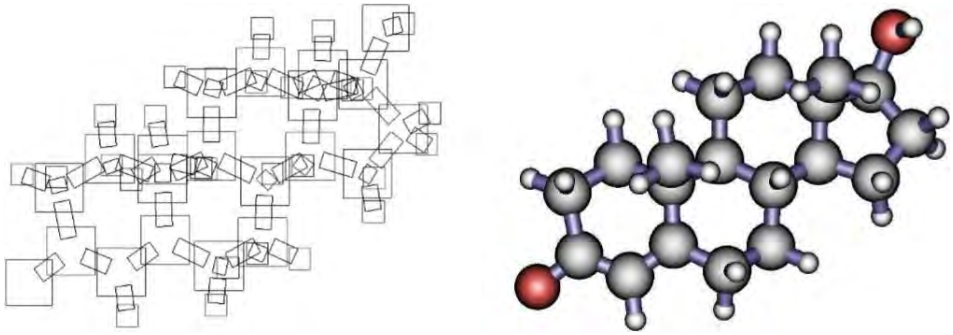

Figure 3.12: Example of a molecular visualization using sphere- and cylinder impostors: (left) wire-frame rendering of impostor quads defined by 4 vertices each and (right) the final rendering result. Image from Montani et al. [MTC06].

Moritz and Meyer [MM04] present a virtual reality case study that shows how simultaneously presenting different levels of abstraction supports the exploration of macromolecules. A simplified way of visualizing molecules is the *cartoon representation*. It especially highlights the general structure of a protein instead of the exact atom positions. Additionally, such simplified rendering strategies reduce occlusions and make it easier to perceive the overall structure of a molecule. A focus & context example of two different cartoon representations, in combination with detailed surface models, is shown in Figure 3.13.

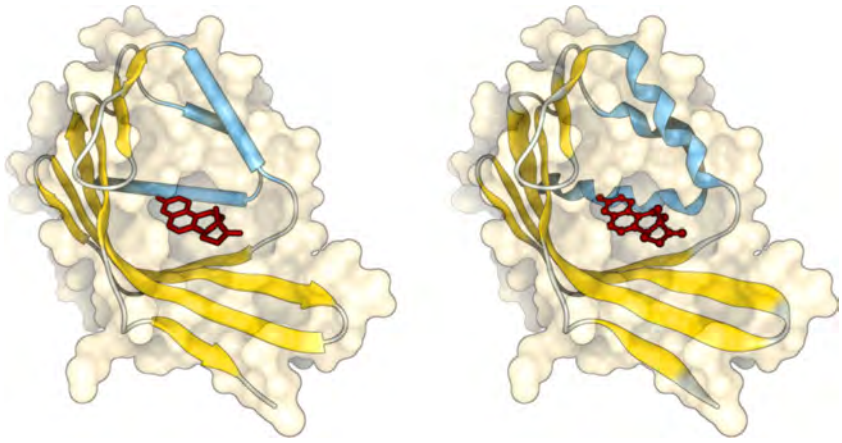

Figure 3.13: Visualization of two different cartoon representations surrounded by the semi-transparent surface of a protein and a red ligand in the center. The cartoon representation consists of yellow arrows and (left) blue cylinders or (right) blue spirals, representing helices. The ligand in the middle is rendered using (left) sticks or (right) balls & sticks. Image from Kozlíková et al. [KKF+17].

Additionally, O'Donoghue et al. [OGF+10] present an overview of different visualization- and rendering strategies that are used in the field of three-dimensional molecular visualization. Especially in the beginning of an exploration, they highlight the importance of showing the user a first impression of the general molecular structure. This can be done by displaying simplified structures, like the already mentioned cartoon representation. For the following exploration of individual elements, like ligands, the authors recommend using either space-filling strategies or balls & sticks representation types.

Our visualization of MD-simulations highly benefits of these findings by O'Donoghue et al. [OGF+10] and therefore uses these rendering types as default setting. In our visualization, the entire structure of the molecule is initially simplified as cartoon representation, which provides a meaningful first impression of the general structure. If the ligand is selected as focus element, it will be displayed using balls & sticks. Inspired by Cerqueira et al. [CBFR08], it is also possible to select different rendering strategies within our visualization. Initially, focus elements are rendered using balls & sticks representations and context elements are rendered using a sticks representation. Nevertheless, it is up to the user to overrule this initial setting using personal preferences or rendering strategies that are more suitable, depending on the current research questions. An example that requires such changes could be that interactions of the ligand with surrounding residues are in focus and not the ligand itself. Thus, the ligand could be visualized more simplified, for example, by using sticks. The otherwise simplified surrounding structures are then in focus and could be rendered using balls & sticks or space-filling strategies.

Such ligand interactions play an essential role in many research areas. Cerqueira et al. [CBFR08] especially highlight application fields like *medicine*, *agronomy*, and the *pharmaceutical industry*. The authors present a novel protocol that allows the user to predict possible ligand binding poses. They emphasize the importance of considering the flexibility of the protein, instead of rigid proteins, during this docking procedure. In their resulting visualizations, they focus on ligands and residues that are located close to the ligand, including their orientation. The surrounding protein only provides context information. Figure 3.14 shows two possible visualizations of ligand binding sites.

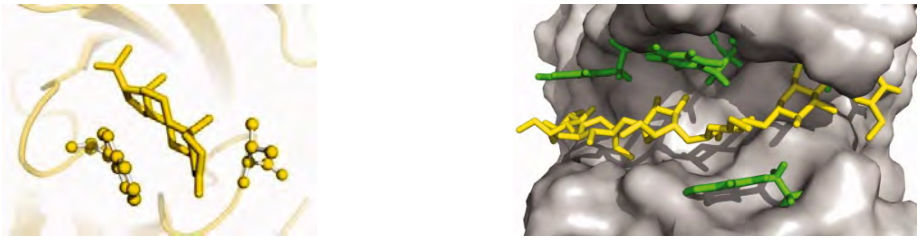

Figure 3.14: Comparison of possible ligand binding sites, including surrounding residues, within a protein. Visualization of the ligand using yellow sticks (left) surrounded by yellow residues as balls & sticks, and the context protein using a cartoon representation, or (right) surrounded by green residues as sticks, and the context protein using a surface rendering strategy. Image from Cerqueira et al. [CBFR08].

In addition, the authors [OGF+10] mention, that it can be helpful to superimpose different visual representations of the same structures. Using our approach, this is done as follows: The entire structure of the protein is represented as cartoon structure. Focus elements are then rendered using more detailed rendering strategies. Since the rendering is done in 3D, it can easily happen that context elements occlude focus elements. In case of occlusions, we apply *ghosting* and render context structures semi-transparently, to prevent and weaken negative side effects. This way, context will never occlude focus elements and therefore the context is always visually less emphasized than focus elements.

Inspiration for our spatial importance approach, using different levels of abstractions, also comes from van der Zwan et al. [vdZLBI11]. In their work, they draw attention to potential difficulties researchers have, trying to understand the overall structure and function of complex molecular data, such as proteins. According to the authors, the resulting interactive illustration improves the spatial perception and provides a *visualization of continuous transitions* between different stages of structural abstraction, which at the same time enhances the "illustrativeness". In their work, they especially focus on seamless transitions between different levels of abstraction. The following structural abstraction stages are used: *space fill*, *balls & sticks*, *licorice*, *backbone*, and *ribbon*. An example of all types is shown in Figure 3.15 and an example of compositions of different levels of abstraction, to highlight parts within a protein, is shown in Figure 3.16.

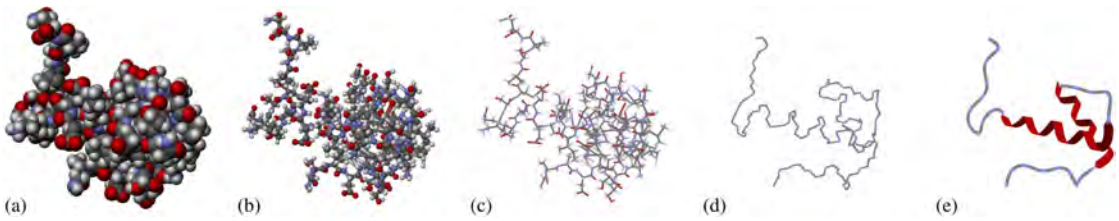


Figure 3.15: Visualization of different molecular abstractions. (a) Space fill using van der Waals radii, (b) balls & sticks, (c) licorice, (d) backbone, and (e) ribbon. Image from van der Zwan et al. [vdZLBI11].

## 3.3 Time-Varying Data Visualization

In this section we will present related work from the domain of *animated visualizations* and *fast-forward video visualizations*, whose focus is the generation of *time-compressed videos*. Additionally, we will discuss possibilities for the user to interact with the visualization in real-time and different approaches of visual feedback, representing the adaptive playback speed, without distracting or confusing the user.

Höferlin et al. [HHWH11] describe three different types of video analysis that can be used to summarize and shorten a raw input video:

> **Video abstraction**, like *still images* or *video skimming*. *Still images* are used to represent a video clip as single or multiple still images. These can be used in

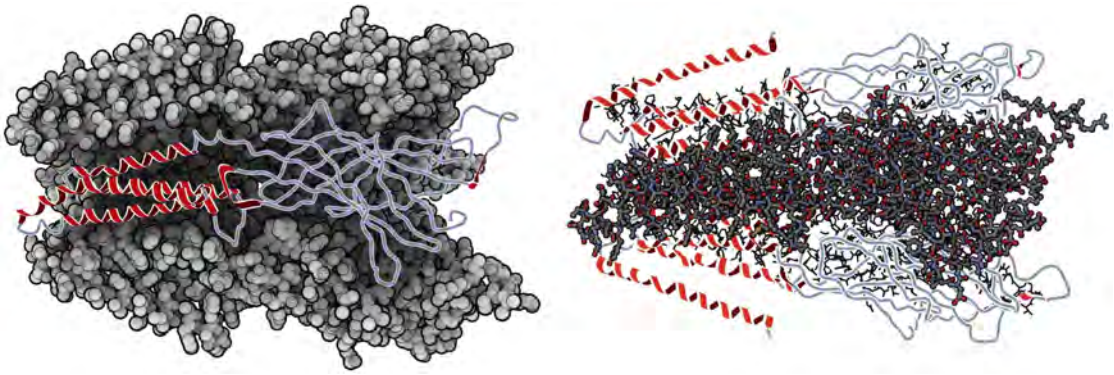

Figure 3.16: Composition of different rendering strategies: (left) Inner structures are highlighted using simplified rendering strategies like ribbons and backbone, whereas the background is rendered using the detailed space-filling rendering strategy. (right) On the contrary, a detailed rendering of the inner structures using balls & sticks and a simplified rendering of the outer structures using ribbons and backbone rendering strategies is shown. Image from van der Zwan et al. [vdZLBI11].

slide shows or like it was presented by Liu et al. [LMH+07] as image mosaics. An example of such a mosaic is shown in Figure 3.17.

*Video skimming* is used to create much shorter and more meaningful video clips than the original raw video. It uses a temporal importance value that excludes unnecessary parts of the video. Therefore, the same playback speed is used but less important sections of the video are skipped. Typical examples are movie trailers. In MD-simulations, this could be done by visualizing the exact point in time the ligand performs and finishes the binding instead of showing the whole simulation.

Figure 3.17: Example of a video collage that immediately visualizes highlights of a video within a single composition. Image from Liu et al. [LMH+07].

**Video browsing** is a video analysis approach that requires user interaction and is mostly used if the goal is to find a specific event within the video. It is implemented as a slider, which is already integrated into most video players. The slider can be moved by the user, to find interesting events within the video. This approach is very error-prone, especially if the user is not aware of where or how many interesting events occur in the video. Since MD-simulations are even more complex, it was important for us that this process is not only dependent on the user interaction, but also on additional importance functions that guide the user to interesting events. This additionally reduces the chance of overlooking interesting events.

**Adaptive video fast-forward** uses a playback speed that is dependent on additional measurements or relevance values whereas classic *fast-forward* simply plays the video at a new constant speed. Höferlin et al. [HKH+12] define adaptive video fast-forward as *non-linear time mapping*, of unedited video material, to a different animated visualization time. A possible relevance value could be the present motion or visual complexity of a scene. For example, if a scene is dense, the video will slow down and if the scene is sparse, the video will speed up. Figure 3.18 shows a comparison of both approaches.

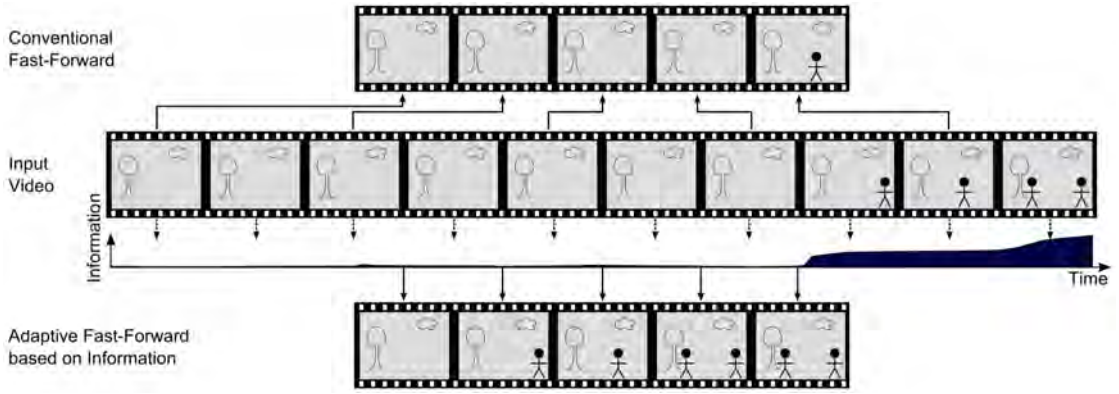


Figure 3.18: Comparison of (top) conventional fast-forward and (bottom) adaptive fast-forward. Both result in an output video that contains 5 frames. Using fast-forward, every second frame of the original raw video is removed. Using adaptive fast-forward, only frames with low visual complexity are removed. Image from Höferlin et al. [HHWH11].

In our case, this raw material is represented by the MD-simulations, which consists of hundreds or even thousands of individual simulation keyframes. These frames are then mapped to an importance driven visualization that mostly displays important or most relevant keyframes. The importance of every keyframe is thereby specified by a temporal importance function. Since we want to skip unimportant frames and emphasize important keyframes, this importance function is most likely not linear.

The authors additionally state that adaptive fast-forward is especially useful in scenarios where *unedited materials*, like in our case computed MD-simulations, are used. Since we are using keyframes from an MD-simulation, instead of an input video, we further benefit from the fact that we know exactly which molecular elements are visible at each point in time and what their parameters like speed, movement direction, or other properties are. In our case, the scene complexity alone is independent of relevant events, but we can easily use parameters that are more meaningful for exploring MD-simulations.

Next, we want to highlight the work from Wildemuth et al. [WMY+03], recommending a uniform fast-forward playback speed of 1:64 of the original video, using a default playback speed of 30 *frames per second* (FPS). Thereby, every $64^{th}$ frame is displayed and all other frames in between are skipped. The authors performed a user study, which has shown that a video cannot be played arbitrarily fast. It was examined whether textual and graphical elements were detected within a video, if subsections of a video are recognized, and if users were able to summarize the video in writing. If a video is played faster, it can be viewed quicker since the overall duration is shorter. Unfortunately, the faster a video is played, the more difficult it is for the viewer to understand and perceive the content and correlations. In addition, the authors mentioned that it is important to allow the user to manually adjust the playback speed.

In contrast to uniform fast-forward, adaptive fast-forward approaches require an importance function that defines the playback speed. It is essential but also challenging to create such a significant and meaningful function. The type and category of a video can already help creating such functions. Divakaran et al. [DO07] divide videos into two categories: *scripted videos* and *unscripted videos*.

> **Scripted videos** as feature films or news: They can easily be subdivided into scenes and chapters. If the user is watching a recording of news on TV and is only interested in the weather forecast, all reports from before can be easily skipped since the weather forecast is usually always at the end.
>
> **Unscripted videos** do not have these precise scenes or chapters and therefore, it is much harder to subdivide them and skip unimportant scenes. If the user is watching a soccer game and is only interested in goals, there is no specific point in time the application can always fasten forward to, as there are different numbers of goals scored during every game and at random points in time.

In their approach, Divakaran et al. [DO07] use the voice of the commentator to detect highlights of unscripted videos shown on TV. The user can select a certain level of importance and the video player will automatically guide the user through all the important highlights of that video. Therefore, they introduce a *highlight search* where the player automatically skips to the start position of a located highlight or *summarize playback* where the player only shows highlights of the selected importance layer. An example is shown in Figure 3.19.

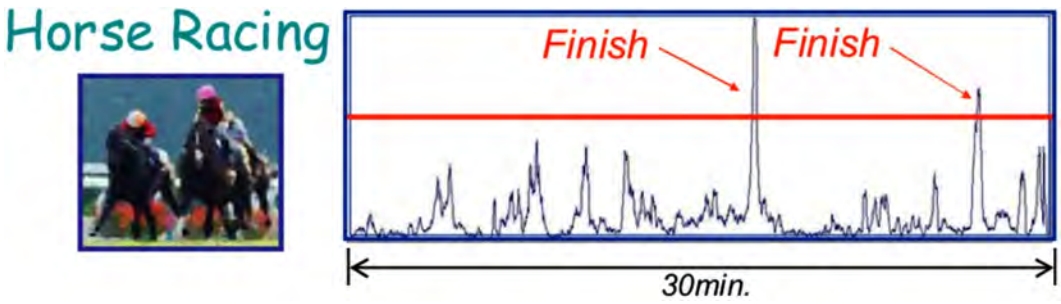


Figure 3.19: Composition of a horse race including the voice of a commentator, which immediately leads to highlights within the video. Image from Divakaran et al. [DO07].

Instead of the voice of a moderator, MD-simulations can be explored using temporal importance functions that represent possible important parts of the simulation. Instead of watching the whole simulation over and over again, the user is directly guided to important events. This highly reduces the chance of overlooking interesting events.

Additionally, Höferlin et al. [HKH+12] highlight the importance of combining *frame skipping* and *temporal blending of successive frames*:

> **Frame skipping** like (adaptive) fast-forward, removes as many unimportant frames as possible. This is advantageous since it does not change the appearance of objects within the video, it is easy to compute, and the acceleration is not limited by any factors beside the number of keyframes [HKH+12]. On the contrary, this removal of individual frames or sections disrupts the motion perception and increases *motion blindness*, like it was shown by Scott-Brown and Cronin [SBC07].
>
> **Temporal blending of successive frames** can be used for *temporal anti-aliasing*. Therefore, it is used to reduce disruption that were introduced by frame skipping and it is then often referred to as *motion blurring* [War04]. This approach focuses on the human visual system, which also blurs fast moving objects. The resulting motion strikes are a natural way to help our brain trace a movement.

For our approach of visualizing MD-simulations, it was also necessary to combine these two approaches. We used adaptive fast-forward to skip less important parts of the simulation and to reduce the overall length of the animation. This allows the user to explore simulations that consist of an immense number of keyframes in a more compact manner and in much less time, since the resulting simulation is shorter and mostly displays important keyframes. Unfortunately, skipping individual frames also results in visual flickering of elements, because their spatial positions changes more from one frame to the next. Therefore, we used motion blur to reduce visual flickering of elements, which is a natural result and an indicator the human brain is used to. Motion blur is additionally used to indicate our current playback speed and the pace of individual molecular elements within the simulation.

Rav-Acha et al. [RAPP06] introduce *video synopsis*, a space-time volume that simultaneously shows multiple events within the same video frame, although they originally appeared at different times. This highly reduces the length of surveillance camera footage but requires a segmentation of the scene into individual objects. It is done by creating a so-called *space-time volume*. Another possibility is a *stroboscopic image* where multiple dynamic appearances of a single object are displayed within the same frame. This obviously requires additional training for the user, since the same object can appear at multiple different places within the same frame. Therefore, the length of the video is reduced by increasing the visual density of the frames. An example of this space-time volume is shown in Figure 3.20.

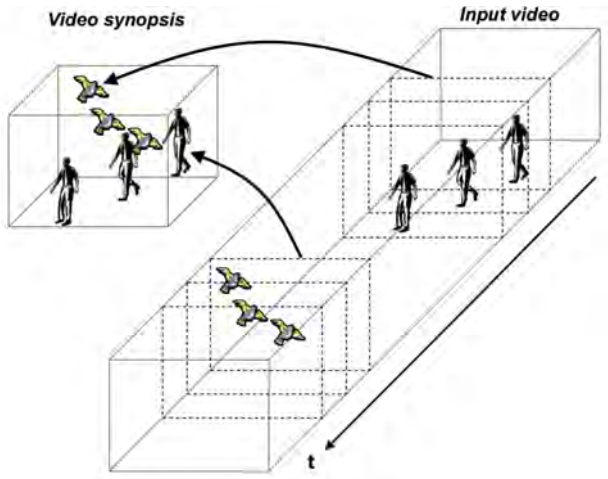


Figure 3.20: Example of a space-time volume that combines different objects within the same frame, although they originally occur at different points in time. Image from Rav-Acha et al. [RAPP06].

In MD-simulations, this is highly distracting since the exact atom positions are falsified and visible atoms and their interactions would no longer correspond to the actual simulation. Furthermore, the screen would be completely over-plotted since MD-simulations usually consist of several thousands of molecular elements and the visual density is then even increased.

Balabanian et al. [BVMG08] introduce *temporal style transfer functions* for volumetric data. The underlying dataset consists of individual time steps that together represent a development over time. Using their [BVMG08] approach, a collection of multiple time steps can be visualized within a single image. Figure 3.21 shows two possible visualizations of a bouncing super-ellipsoid using a temporal transfer function for a dataset, consisting of 10 time steps. In both visualizations, especially the first and the last time step are emphasized. By changing the temporal importance function, an intermediate time step can be highlighted. Otherwise, only the contours of the individual overlapping super-ellipsoids are shown.

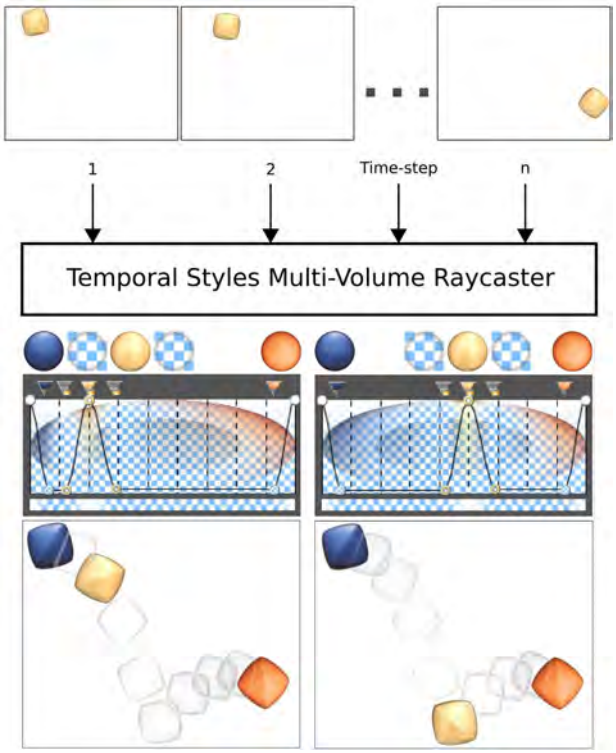


Figure 3.21: Visualization of a time-varying dataset consisting of 10 time steps using a temporal style transfer function. The first time step is colored using a blue lit sphere and the last time step using an orange lit sphere. Depending on the temporal transfer function, an intermediate time step is either colored using a yellow lit sphere or only the contours of the super-ellipsoids are displayed. Image from Balabanian et al. [BVMG08].

In the following, we will present possibilities for the user to interact with time-varying data. Examples are: First, non-linear video playback presented by Dragicevic et al. [DRB+08]. Second, direct dragging of visualized objects along trajectories, presented by Wolter et al. [WHTPK09] and finally, interactions with animated visualizations using different *graphical user interface* (GUI) elements, presented by Kondo and Collins [KC14], Hurst et al. [HJ05], and Willett et al. [WHA07].

Dragicevic et al. [DRB+08] introduce a novel interaction method called *relative flow dragging*. Using their approach, the user directly clicks on an object within the video and moves it along its visual trajectories, instead of pre-computing any importance information or playback speed. Hereby, the mouse movement can help to find a precise moment or position of an object within the video. The advantage of this method is the close match between user input and video player output. Instead of using a seeker bar, which is commonly used for *time centric browsing*, they introduce this direct manipulation for *space centric browsing*. The speed-up of the video is therefore only dependent on the mouse movement of the user. An example is shown in Figure 3.22.

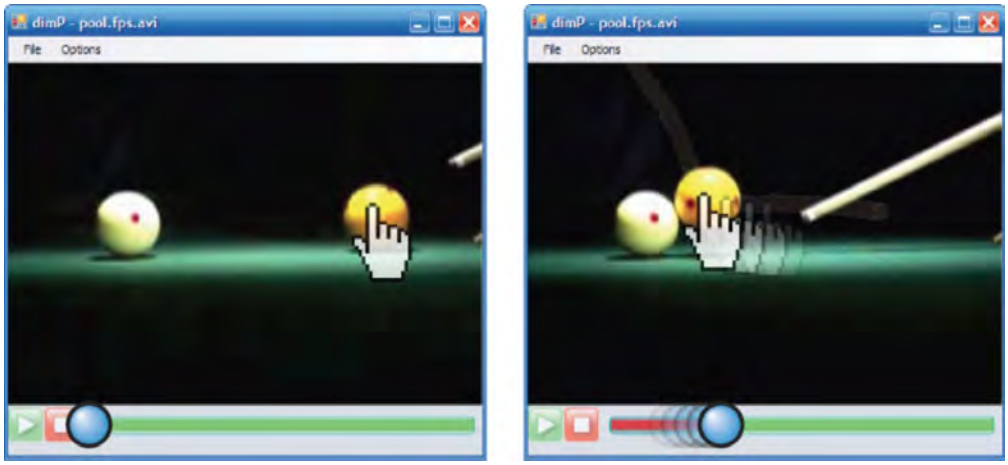


Figure 3.22: Example of relative flow dragging, where the user directly clicks on a billiard ball and moves it along its trajectory instead of simply playing the video. Image from Dragicevic et al. [DRB+08].

This frame navigation by mouse movement could also be used for visualizing MD-simulations, for example, by clicking on the ligand and moving it, to identify the point in time the animation should jump to. Currently this functionality is not implemented within our framework, but it might be advantageous for future publications. Unfortunately, it is problematic at the same time, since the user must know the exact movement of the ligand beforehand. This could be difficult because MD-simulations are mostly far too long to be learned by heart and the motion of atoms is not necessarily uniform in a particular direction, but may seem almost random to inexperienced users. A solution could be to display the simplified movement of the ligand as overlay.

Instead of dragging objects from a video, Wolter et al. [WHTPK09] present an interaction method that allows the user to directly drag an object along its trajectory, within a *virtual reality* (VR) based visualization system. The authors additionally highlight the importance of cutaway techniques, to prevent occlusion problems. An example of such a visualization is shown in Figure 3.23.

Kondo and Collins [KC14] present *DimpVis*, an object-centric temporal interaction technique for time-varying data, which can be used on bar charts, scatter plots, heat maps, and pie charts. DimpVis allows the user to directly interact with the visualization itself, instead of using a time slider. This can be done using mouse movements or touch gestures. The visualization is enhanced using additional visual paths that indicate how the dataset changes over time. Exploring a time-varying dataset, the user can then focus on the visualization itself and does not have to constantly check the time slider. An example of an interactive bar chart visualization, using touch gestures, is shown in Figure 3.24.

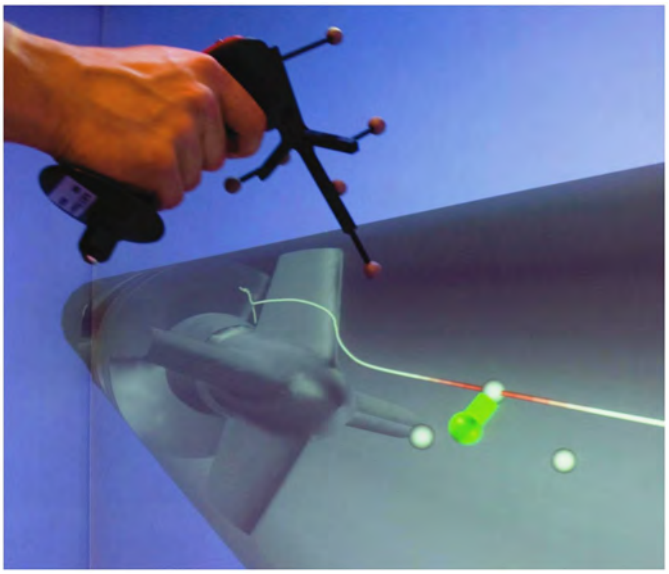

Figure 3.23: Example of a user dragging and tracing a particle within a VR based visualization system. Image from Wolter et al. [WHTPK09].


Figure 3.24: Example of DimpVis used on bar charts. The user can vertically drag individual bars along their hint paths, to find out about changes over time. Image from Kondo and Collins [KC14].

Hurst et al. [HJ05] introduce a different approach called *ZoomSlider*. It is a graphical interface used to browse a video or access an animated visualization. Using sliders, the length of the slider is unfortunately limited by the screen resolution or window size. This can be problematic, especially if the playback of a video takes multiple hours or days instead of a few seconds. Hurst et al. [HJ05] present an approach in which the temporal position within the video can be selected horizontally using a classical GUI slider, but in addition to this, the zoom level can be changed simultaneously by moving the mouse vertically. The accuracy of vertical scaling is only limited by the height of the window. This allows the user to find an almost arbitrarily precise point in time within the video independent of the duration and without the use of additional interaction hardware or other GUI elements. An example of such a ZoomSlider interface is shown in Figure 3.25.

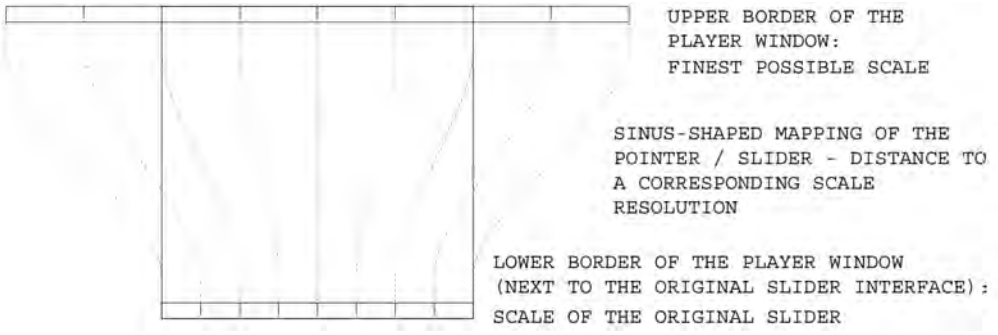


Figure 3.25: Example of the ZoomSlider interface including explanations of the differences between horizontal and vertical mouse moves. Image from Hurst et al. [HJ05].

Using approaches like the ZoomSlider, the user can access an exact point in time within a simulation. Nevertheless, it requires prior knowledge because the user needs to know which parts of the simulation he or she is interested in. This is a challenge for MD-simulations because the exact timing of an atom binding may not be known or because the pre-phase or post-phase of a binding may additionally provide significant insight about the simulation. Therefore, it is important to not only find a single point in time but to be able to explore the entire video in a more compact manner.

A different approach, called *scented widgets*, is presented by Willett et al. [WHA07]. The authors present a wide variety of commonly used GUI widgets such as sliders, check boxes, radio buttons, trees, list boxes, or combo boxes that are extended with embedded visualizations. An example of four different widgets is shown in Figure 3.26.


Figure 3.26: Examples of different widgets containing embedded visualizations (from left to right): A slider including a color coded bar chart, ranked check boxes with additional star icons, a color coded list box with check mark icons, and a tree including authors and the number of edits. Image from Willett et al. [WHA07].

Willett et al. [WHA07] mention that especially in the beginning of an analysis, or if the user is not familiar with the underlying dataset, scented widgets can improve the exploration. Widgets will be used less, the more familiar a user is with the dataset. In our visualization of MD-simulations, the slider corresponds to the playback function and instead of the bar chart, an importance function is shown. This allows the user to skip unimportant events, jump to specific points within the simulation, or modify the function depending on other molecular properties.

In the following, we want to discuss the current state of the art approaches for providing the user with visual feedback representing the adaptive playback speed. Cheng et al. [CLCC09] introduce the so called *SmartPlayer*, which represents an adaptive video fast-forward approach and uses the metaphor of *scenic car driving*. If something visually pleasing or interesting is in sight, the driver usually slows down and on the visually monotonous highway, the car will speed up. A *tachometer* indicates the current speed-up and especially important events are additionally highlighted. The player changes the playback speed of a video depending on the visual complexity of a scene and learns user preferences from previous videos. The complexity of a scene can, for example, be derived using automatic analysis methods similar to those used in video skimming. A disadvantage of this approach is that it requires lots of attention by the user who has to continuously look down at the tachometer and up at the video again to see the playback of the video as well as the actual speed-up factor located at the lower end of the screen. Cheng et al. [CLCC09] additionally mention that the playback speed should not change frame by frame, but allow the user to adapt to the speed-up by incrementally increasing the speed. An example of this visual representation is shown in Figure 3.27 (left).

In addition to the Smart-Player, Höferlin et al. [HKH+12] introduce two additional approaches: First, a so called *Color-Frame*. This approach adds a border to the video that is then colors using a heat-map, indicating the current speed-up factor. The used coloring is similar to the speedometer coloring introduced by Cheng et al. [CLCC09]. Second, *Analog Video Cassette Recorder (VCR) Fast Forward*. This approach creates additional horizontal streaks that indicate a speed-up of the playback. If those distorted lines are not present, the object in the video moves with original speed. This is disadvantageous while exploring MD-simulations, since the object might be of high relevance and can no longer be observed by the user, since its shape and movement is horizontally distorted. Both types of visual feedback are shown in Figure 3.27 (middle, right).

While exploring MD-simulations, we found out that motion blur is perceived much more natural and the blur does not negatively affect the user-experience, since the basic geometric shape of the object is not changed and both the speed of an object and the fast-forward speed-up are directly encoded using the strength of the blur. Since the user can additionally specify a temporal importance function, there is no inconsistent or conflicting visual feedback between object speed and fast-forward-speed-up. Nevertheless, users already indicated that they additionally want some numerical vales and more precise feedback about the exact speed-up.

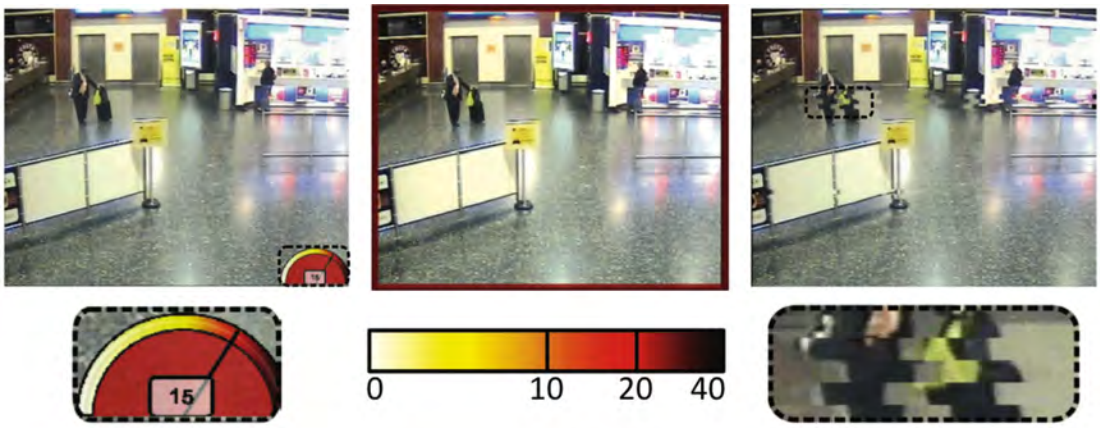


Figure 3.27: Comparison of three different approaches for visual speed-up feedback for the user: (left) *Smart-Player* [CLCC09] uses a speedometer. (middle) *Color-Frame* [HKH+12] colors the border of the video according to the currently used speed-up. (right) *Analog VCR Fast Forward* [HKH+12] uses horizontally distorted lines to indicate a speed-up. Image from Höferlin et al. [HKH+12].

CHAPTER 4

# Spatio-Temporal Focus & Context for MD-Simulations

The following chapter describes the already provided *graphical user interface* (GUI) of CAVER Analyst [Ana] and the theoretical functionality of our visualization in detail. We will explain already supported features (which are shown in Figure 4.1, left) and our contribution, describing how we have augmented this functionality (which is shown in Figure 4.1, right). Section 4.1 describes the GUI and the explanation of theoretical details is divided into Section 4.2 *Spatial Importance* and Section 4.3 *Temporal Importance*.

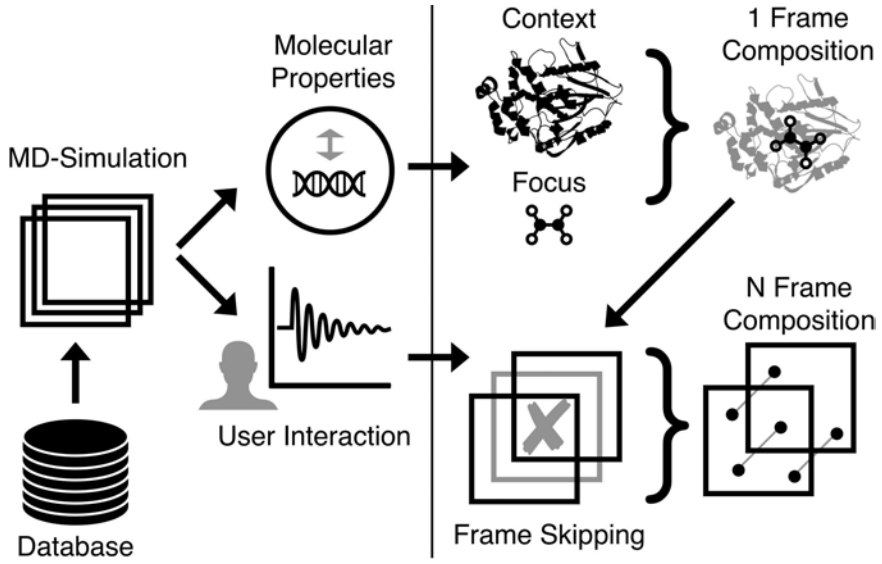


Figure 4.1: Overview of CAVER Analyst [Ana] including (left) already implemented functionality for MD-simulations and (right) our contribution to enhance the exploration.

## 4.1 Graphical User Interface

This section describes how the user can interact with our real-time MD-simulation using an already provided GUI. Additionally, it provides an explanation of the already implemented functionality of CAVER Analyst [Ana] (Figure 4.1, left).

Input for our visualizations are individual keyframes of MD-simulations. These frames are pre-computed on (super)computers and then stored in online or local databases. This large number of simulated keyframes are loaded by the user and serve as input for our interactive 3D visualization. Next, the user can interact with the visualization using a graphical user interface. It allows the user to play, pause, forward, or skip parts of the animation using classic VCR media controls, which is shown in Figure 4.2.

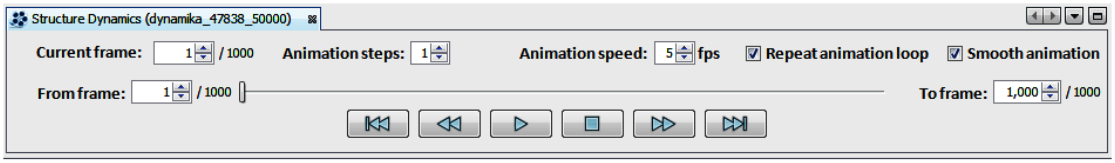


Figure 4.2: CAVER Analyst [Ana] GUI that allows the user to play, pause, forward, or skip parts of the animation.

Keyframes of MD-simulations consist of a wide variety of file formats, such as topology data, trajectory data, binary files, etc. These datasets already provide certain geometric molecular properties such as the Euclidean distance between atoms, the speed at which molecules move, the stuckness of ligands, and many others. After importing the keyframes, the user can choose from many different temporal importance functions, combine them, smooth them, change their minimum- and maximum thresholds, or compute and observe the derivative of such functions. The GUI for these modifications is shown in Figure 4.3.

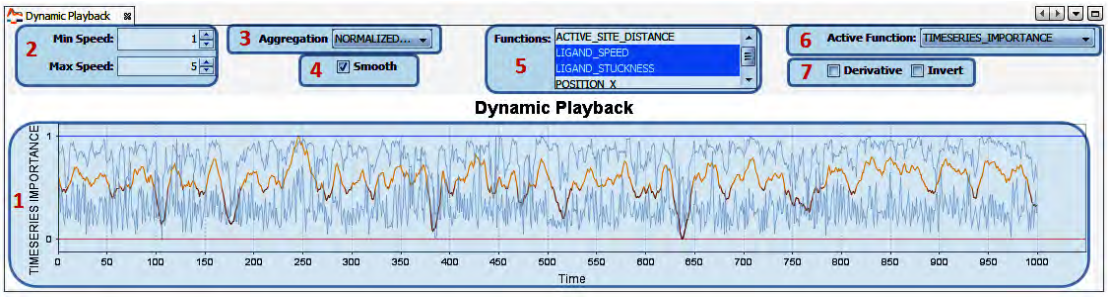


Figure 4.3: CAVER Analyst [Ana] GUI for modifications of the temporal importance function. (1) 2D plot of the importance function. If multiple functions are selected, the resulting function is colored and the individual functions are grayed out. (2) Settings for minimum and maximum playback speed. (3) Selection of how functions are aggregated. Options are: normalized sum, mean, maximum, or minimum values. (4) Possibility to smooth the function. (5) List selection of integrated importance functions. (6) Selection of the active function. (7) Check boxes to invert a function or calculate its derivative.

Every importance function represents a normalized 2D function, which is why function values are always in the range from 0 to 1. The basis of their computation are underlying keyframes. If these importance functions are then used for temporal explorations, the individual function values of the corresponding keyframes determine the playback speed of the simulation. The closer an importance value is to 0, the faster the simulation will be played. If the importance value is exactly 1, the original simulated playback speed will be used. An example of an importance function that depends on the speed of a ligand is shown in Figure 4.4. The higher a function value is, the faster the ligand moved.

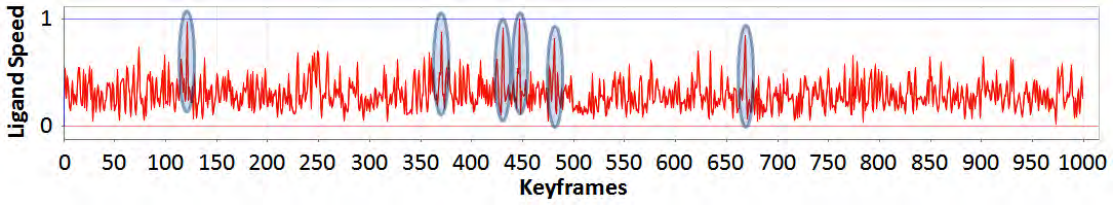


Figure 4.4: Example of an importance function depending on the speed of a ligand. The function is normalized and has an overall length of 1 000 keyframes. In addition, six points over timer were marked, at which the ligand moved particularly fast but only for a very short duration. Importance function provided by CAVER Analyst [Ana].

In the following, we will present examples of how temporal importance functions can be used for the exploration of different research questions. Considering the speed of a ligand, MD-simulation can be analyzed in three different ways:

1. Biochemists are interested in time ranges in which the ligand moves especially fast, i.e., the faster a ligand moves the more relevant a keyframe is. In this case, the function, as shown in Figure 4.4, can be used. The slower the ligand moves, the faster the simulation will be played.
2. Biochemists are interested in time ranges in which the ligand moves very slow. This is interesting for users who want to verify whether surrounding residues are responsible for this slowdown. In this case, the importance function must be inverted. The faster the ligand moves, the faster the simulation will be played. The inverted function from Figure 4.4 is shown in Figure 4.5.

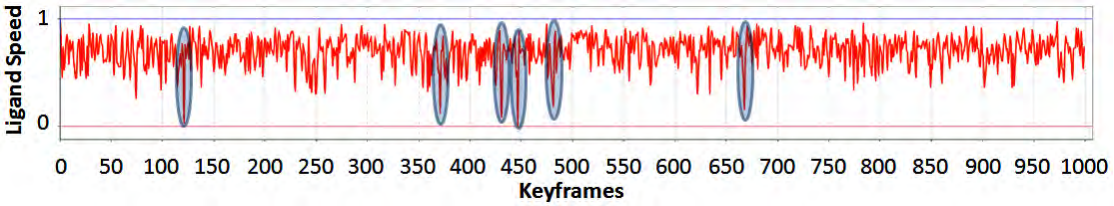


Figure 4.5: Example of an importance function, which represents the inverse of the function shown in Figure 4.4. Importance function provided by CAVER Analyst [Ana].

3. Biochemists are interested in abrupt changes of the ligand speed. Therefore, the derivative of a function can be computed. This is especially interesting for users who want to spend more time specifically at points in time with strong changes in the ligand speed. The derived function from Figure 4.4 is shown in Figure 4.6.

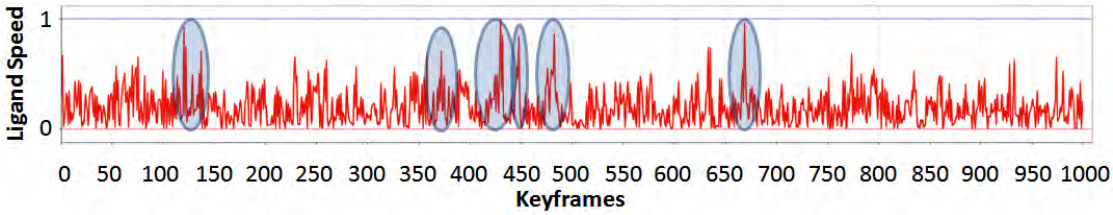


Figure 4.6: Example of a derivative, which represents the derivation of the importance function shown in Figure 4.4. Derivative provided by CAVER Analyst [Ana].

Furthermore, CAVER Analyst [Ana] provides a GUI to choose between different rendering strategies. This can be done in real-time and no pre-computation is needed. To do so, we added two different selections to the GUI. The first one is for focus elements and the second one is for context element. The GUI, including an example of how the currently used context rendering strategy could be changed, is shown in Figure 4.7.

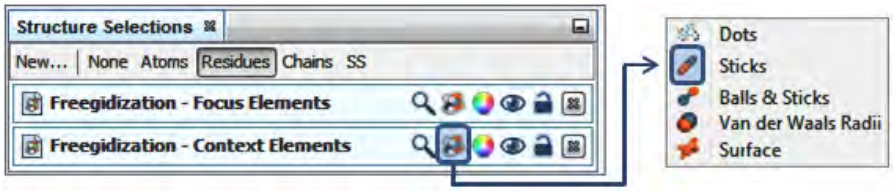


Figure 4.7: CAVER Analyst [Ana] GUI to change rendering strategies for focus and context selections. Additionally, it can be used to change the color model of both selections or hide/show individual selections.

In order to be able to assign individual elements, such as atoms, molecules, residues, etc. to focus and context groups, or even discard them if they neither belong to focus nor context, the whole MD-simulation has to be processed keyframe by keyframe. For the selection of spatial focus & context, the provided GUI is shown in Figure 4.8.

In this case, the user chose the ligand as spatial focus and ligand contacts, i.e., residues surrounding the ligand, as spatial context. By pushing the "*Evaluate MD Simulation*" button, the distances of all residues to the ligand are calculated in every keyframe. If their van der Waals radii overlap is greater or equal to -0.4 Å, as recommended by *UCSF Resource for Biocomputing, Visualization, and Informatics* [Chi], the residue is assigned to the spatial context group and the ligand is stored in the focus group.

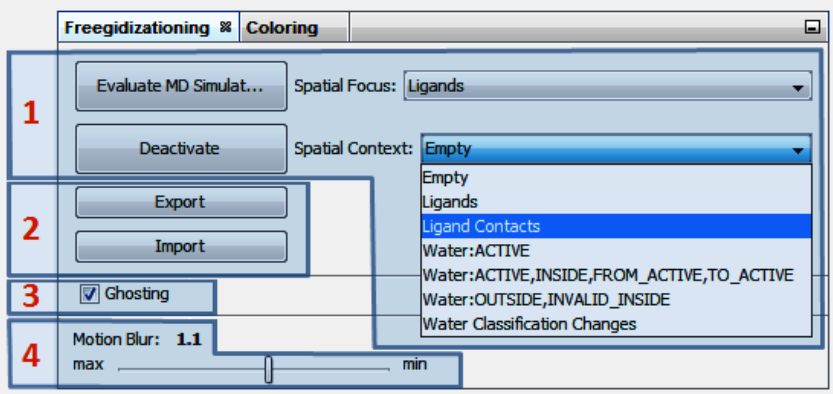


Figure 4.8: CAVER Analyst [Ana] GUI for the selection of the spatial importance function. (1) List selections for spatial focus and context elements. In this case, the ligand is in focus and ligand contacts are in context. (2) Possibility to export and import already computed spatial importance functions. (3) Check box to enable or deactivate the ghosting procedure. (4) Slider to manually change the intensity of motion blur.

In addition, the GUI shown in Figure 4.7 can be used to show or hide individual selections, for example, if context residues are no longer required and should be hidden. Furthermore, the user can change the used color model, which is responsible for coloring individual atoms or entire residues. By default, the CPK model is used. For example, in practice it is common to assign different custom colors to *carbon* atoms. A comparison between the CPK model and orange colored carbon atoms is shown in Figure 4.9.

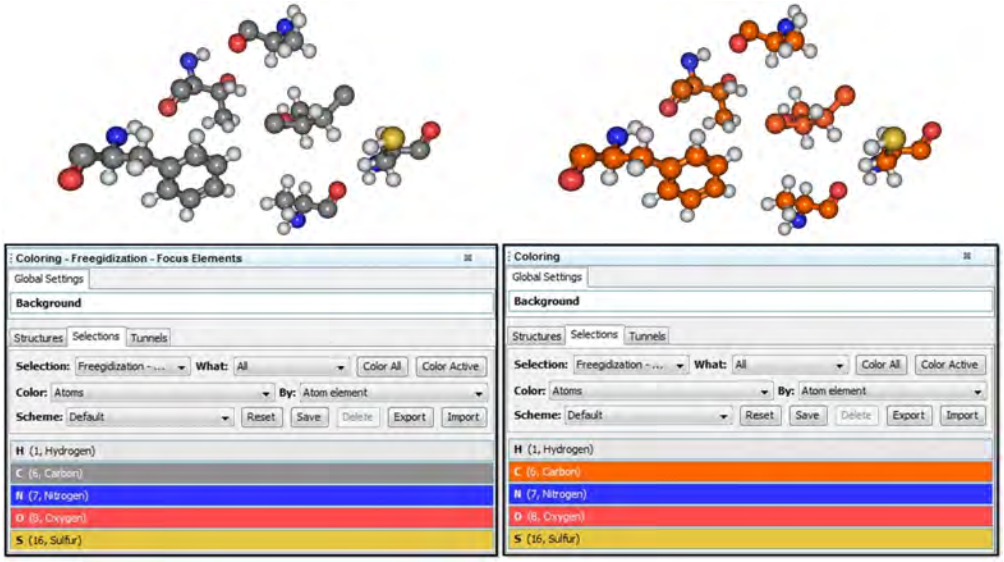


Figure 4.9: Comparison of two coloring strategies: (left) Default CPK color model and (right) orange colored carbon atoms. Renderings produced using CAVER Analyst [Ana].

Instead of only changing colors per atom, we additionally give the user the possibility to select a categorical color coding. Thereby, all context elements are uniformly colored and all focus elements are uniformly colored. This should help to distinguish between focus and context, especially if the same rendering strategy is selected for both, focus and context elements. An example that uses balls & sticks for focus and context elements is shown in Figure 4.10.

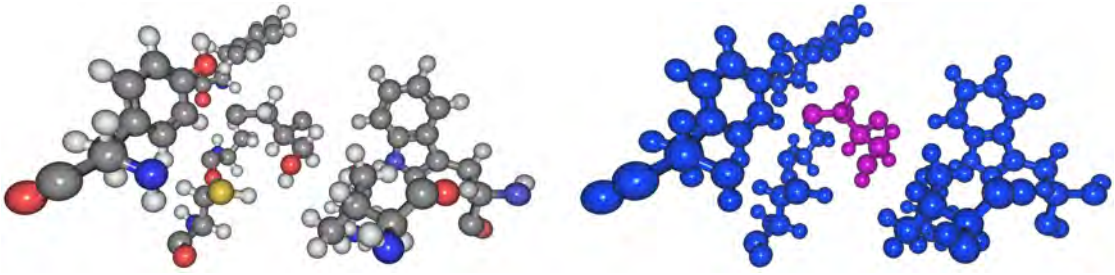

Figure 4.10: Comparison of: (left) The CPK color model used per atom and (right) a uniform purple color for focus elements and a uniform blue color for context elements. Renderings produced using CAVER Analyst [Ana].

## 4.2 Spatial Importance

To provide a spatial importance, we combine different rendering strategies. This includes a simplified representation of entire proteins to provide the user with context information. An example of such a simplified rendering strategy is the cartoon representation. In addition, the user can select the focus of the exploration using the already described GUI from Figure 4.8. For example, the ligand and surrounding residues can be analyzed. Additionally, different geometric *levels of detail* and colors can be selected for all elements within the scene. The ligand, for example, can be visualized using a detailed balls & sticks representation and surrounding residues using a sticks representation. The resulting composition of all focus & context elements represents an enhanced simulation keyframe. Unfortunately, this naive composition does not prevent context elements from occluding focus elements. Therefore, we apply an additional *ghosting* procedure, which renders context elements semi-transparently and simultaneously highlights edges of context structures. This is necessary since the focus is visually always more important than the context.

Spatial importance can be divided into two main components: First, the selection of suitable rendering strategies depending on current research questions or hypotheses. Second, visibility management through ghosting procedures, which prevents occlusions and ensures that focus elements are visually emphasized. Therefore, the following section is divided into Subsection 4.2.1 *Rendering Strategies for Focus & Context* and Subsection 4.2.2 *Visibility Management.*

### 4.2.1 Rendering Strategies for Focus & Context

The most important part of spatial importance is the use and combination of different rendering strategies. Examples, which are also shown in Figure 4.11, are: dots, sticks, balls & sticks, space-filling, and surface representations. Additionally, complex molecules or proteins, consisting of an immense number of atoms, can be simplified using a cartoon representation, which is ideal for visualizing the context.

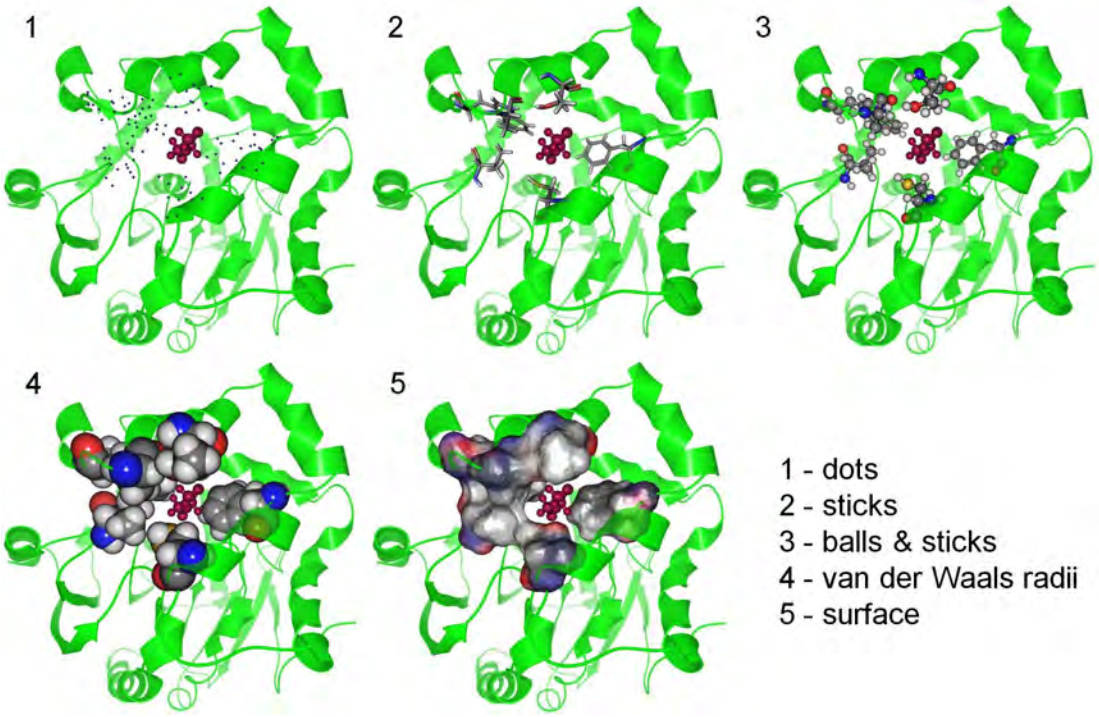


Figure 4.11: Comparison of five different rendering strategies for residues surrounding a violet ligand in focus. The ligand is rendered using a detailed balls & sticks representation and the protein is rendered using a simplified green cartoon representation. Renderings produced using CAVER Analyst [Ana].

Depending on the chosen form of representation, either the interactions of residues are emphasized, or the movements of the ligand. If residues are in focus, they could be rendered using sticks or space-filling representations, which is shown in Figure 4.12 (top), (bottom left), or using a balls & sticks and surface representation. These rendering strategies especially emphasize their movements. If the user is, for example, interested in the interaction of the ligand with surrounding residues, our CAVER Analyst [Ana] extension recommends the following default rendering strategies: A balls & sticks representation for the ligand in focus and a sticks representation for surrounding residues. This can be changed or arbitrarily combined depending on user preferences or research questions.

If residues are in the context, they can be represented in a more simplified form using dots, which is shown in Figure 4.12 (bottom right). Thereby, their movement is deemphasized and only provides context information, similar to the cartoon structure for the whole protein. The same approach applies to the ligand. If the ligand is in focus, it could be rendered using a surface or balls & sticks representation, which is shown in Figure 4.12 (top), (bottom right), and if it represents just the context, it could be rendered using a sticks representation, which is shown in Figure 4.12 (bottom left), or dots.

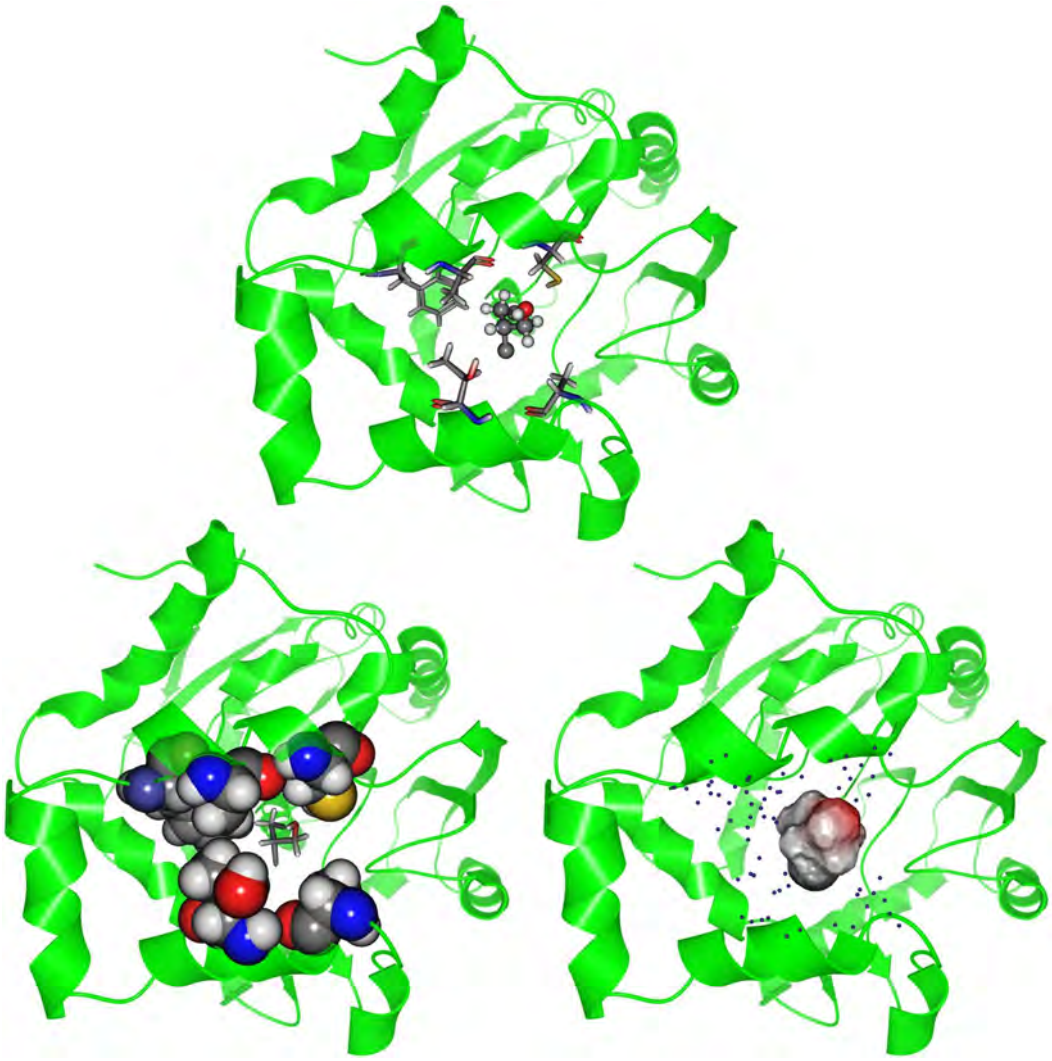

Figure 4.12: Three different combinations of rendering strategies: Residues are rendered using (top) a sticks representation, (bottom left) a space-filling representation, and (bottom right) dots. The ligand is rendered using (top) a balls & sticks representation, (bottom left) a sticks representation, and (bottom right) a surface representation. Renderings produced using CAVER Analyst [Ana].

### 4.2.2 Visibility Management

Although the spatial importance function already reduces the density of highly complex MD-simulations, resulting compositions of focus and context elements can still lead to occlusions. To prevent this, we have analyzed a wider variety of rendering strategies for visibility management. In the following, we will discuss and present a comparison of five different strategies:

1. *No visual preference*: This strategy does not differentiate between focus and context elements and therefore leads to the already described occlusion problems. An example is shown in Figure 4.13 (1).
2. *Clip planes*: Using a single or multiple planes, this strategy clips occluding elements. Especially problematic here is that both context and focus elements are removed. Additionally, by already partially removing occluding objects, the whole context information is lost. An example is shown in Figure 4.13 (2).
3. *Rendering order*: This strategy renders context elements first and subsequently, independent of their depth values, focus elements. On the one hand, this prevents occlusions and on the other hand, the spatial impression and depth perception is completely destroyed. An example is shown in Figure 4.13 (3).
4. *(Semi)-transparent rendering*: In order to preserve a correct depth perception, occluding structures are rendered semi-transparently. This shows the user that a context element is currently in front of a focus structure without totally occluding the focus element. An example is shown in Figure 4.13 (4).
5. *Ghosting*: This procedure represents an extension of semi-transparent rendering. It allows us to check per-pixel, whether a context element was rendered in front of a focus element. If this happened, the context is not simply replaced by the focus, as this would destroy the spatial perception and the depth of the visualization. Instead, the context pixel is rendered semi-transparently. This preserves the spatial order and focus elements will always be visible. The gradient of every occluding context pixel is thereby used to determine its transparency, which preserves the visibility of the focus and additionally highlights edges and outlines of context structures. An example is shown in Figure 4.13 (5).

An overview of the *gosting* procedure, including all three steps necessary to perform ghosting, are listed below and shown in Figure 4.14:

1. **Edge detection:** The first step is rendering the scene containing focus as well as context elements. This image is subsequently used to compute a *grayscale intensity image*, which is then used for horizontal- and vertical edged detection using *Sobel convolution kernels*.
2. **Depth textures:** Two depth textures are rendered, one containing focus and context elements and one containing only focus elements. These textures are then used to find out if context pixels occlude focus pixels.

3. **Composition:** Depending on a per-pixel occlusion check, based on the two depth textures, colors of the focus and focus & context image are linearly interpolated using the computed edge image, which especially highlights edges.

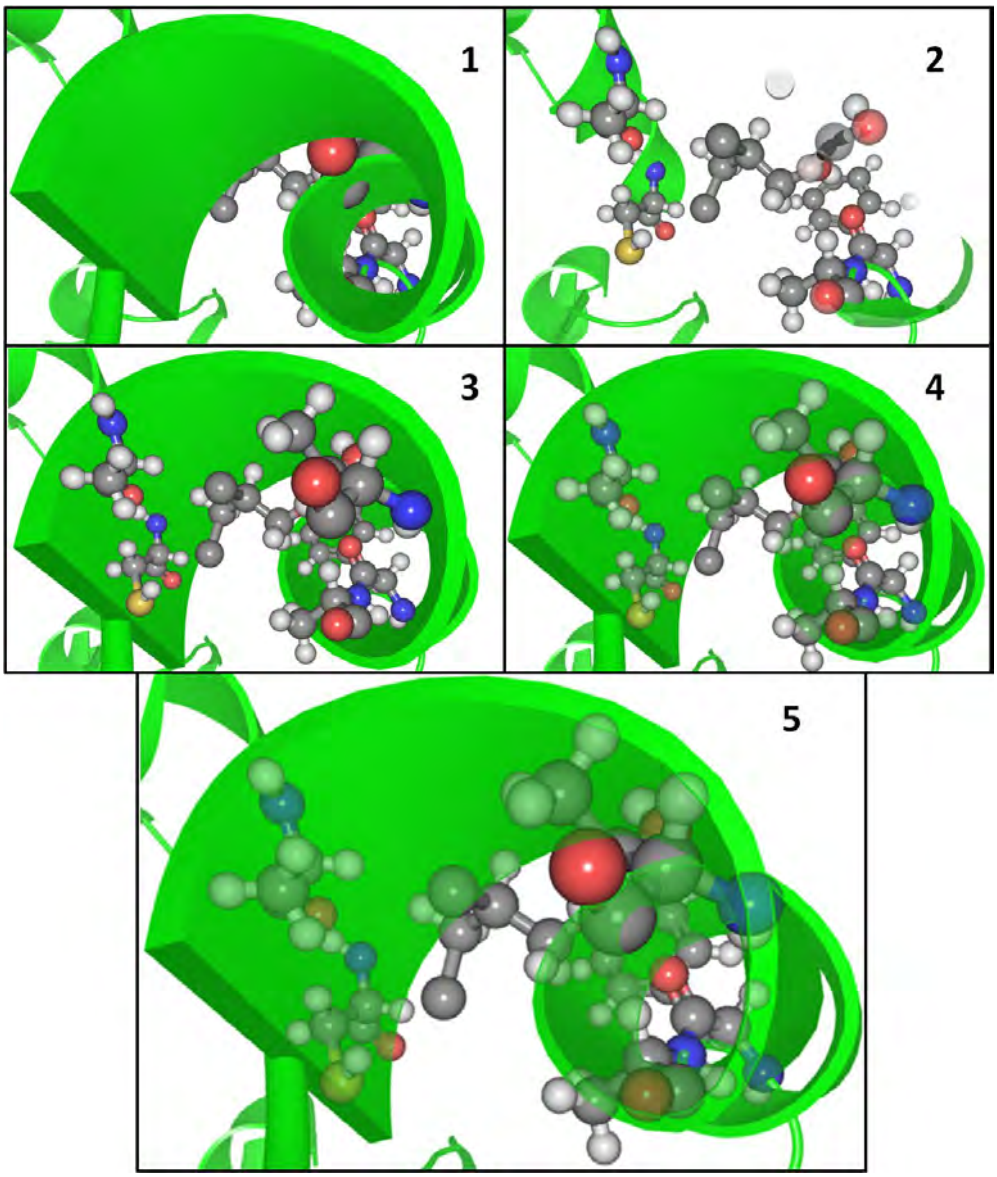


Figure 4.13: Comparison of different types of visibility management: (1) No visual preference for focus elements. (2) Visibility management through clip planes, which simultaneously remove focus elements. (3) Rendering focus elements always on top of context structures. (4) Rendering occluding context structures semi-transparently. (5) Ghosting, which renders occluding elements semi-transparently and additionally highlights edges of context structures. Renderings produced using CAVER Analyst [Ana].

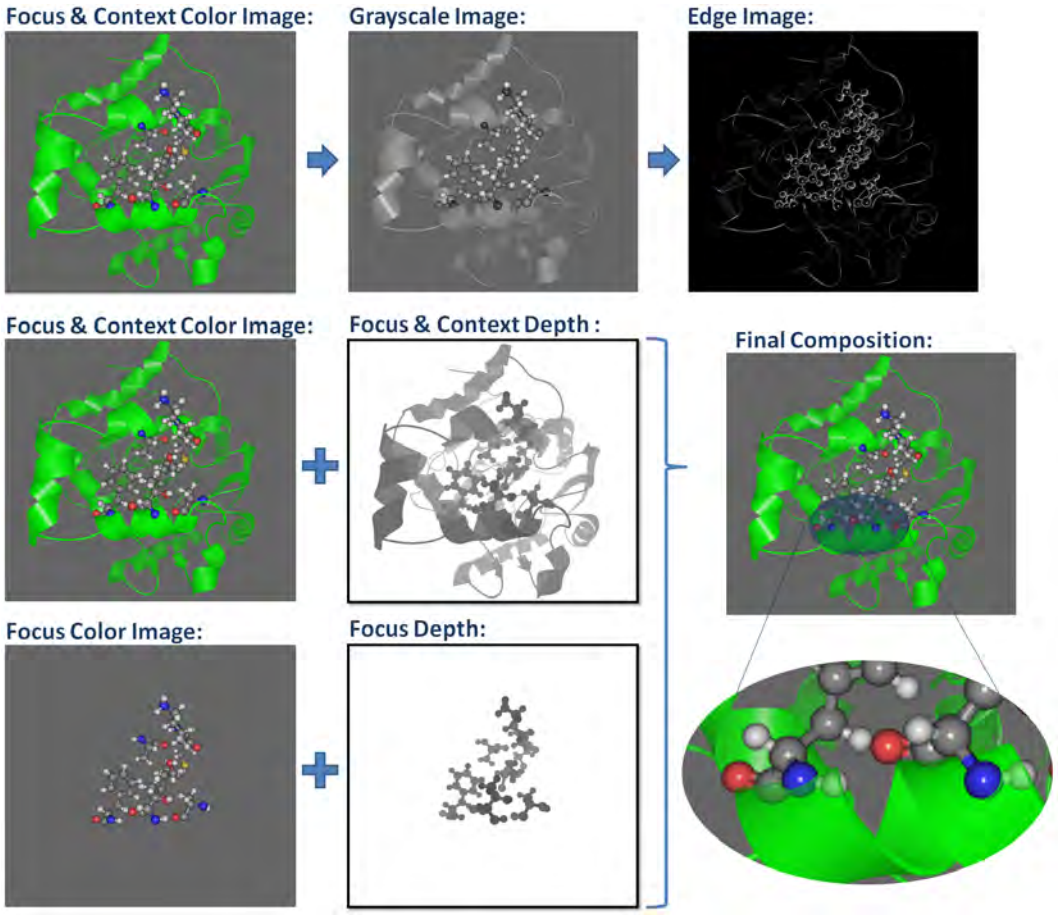


Figure 4.14: Overview of the ghosting procedure consisting of the following three steps: Edge detection, computation of depth textures, and the final image composition. Renderings produced using CAVER Analyst [Ana].

As already mentioned, the first step is rendering the scene containing focus as well as context elements. A possible result of such a rendering is shown in Figure 4.15 (left).

Next, based on this color image, a grayscale intensity image is created. This was done similar to the *rgb2gray()* function provided by MATLAB 2016b [Mat16]. The exact calculation of a grayscale intensity pixel $P_{Gray}$ is shown in Equation 4.1 (where $P_R$ denotes the red-, $P_G$ the green-, and $P_B$ the blue color value of a pixel).

$$P_{gray} = 0.2989 \cdot P_R + 0.5870 \cdot P_G + 0.1140 \cdot P_B \tag{4.1}$$

If this calculation is performed on every pixel, the result is a computed grayscale intensity image, which is shown in Figure 4.15 (right).

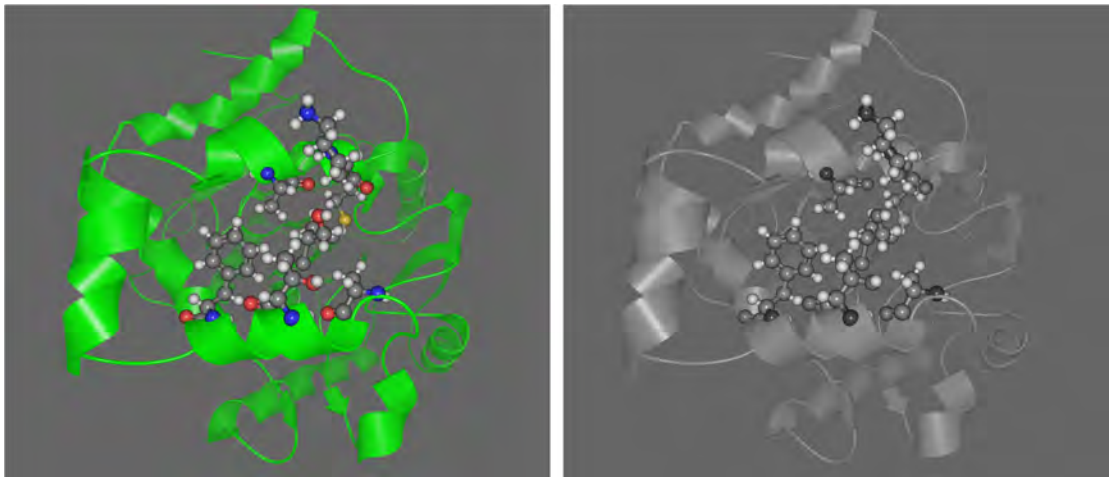

Figure 4.15: (left) Rendering of a scene containing focus and context elements, which is subsequently used for the computation of (right) the grayscale intensity image. Renderings produced using CAVER Analyst [Ana].

During the next step, a convolution of the grayscale intensity image and a horizontal *Sobel kernel*, to detect horizontal edges, and a vertical *Sobel kernel*, to highlight vertical edges, is performed. The 3x3 matrix $M_{Horizontal}$ represents a horizontal kernel:

$$M_{Horizontal} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

The transpose of the above matrix represents the 3x3 matrix $M_{Vertical}$ of a vertical kernel:

$$M_{Vertical} = M'_{Horizontal} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

The result of a convolution with $M_{Horizontal}$ is shown in Figure 4.16 (left) and the result of a convolution with $M_{Vertical}$ is shown in Figure 4.16 (right).

If a convolution was performed using a horizontal- and a vertical kernel, both results can be combined on a per-pixel basis ($P_{Horizontal}$ and $P_{Vertical}$), with the help of the following Equation 4.2. The resulting pixel is called $P_{HV}$.

$$P_{HV} = \sqrt{P^2_{Horizontal} + P^2_{Vertical}} \tag{4.2}$$

The resulting image of this composition is shown in Figure 4.17.

In the next step, two depth images are rendered. The first image consists of the scene including all focus and context elements, which is shown in Figure 4.18 (left) and the second depth image contains only focus elements, which is shown in Figure 4.18 (right).

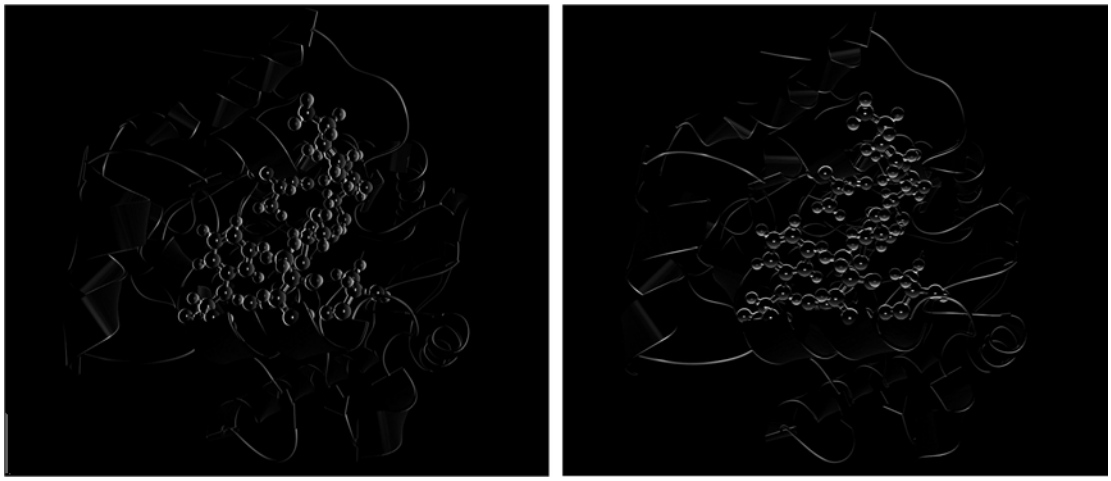

Figure 4.16: Convolution of a grayscale intensity image and (left) a horizontal Sobel kernel, which emphasizes horizontal edges, and (right) a vertical Sobel kernel, which highlights vertical edges. Images produced using MATLAB 2016b [Mat16].

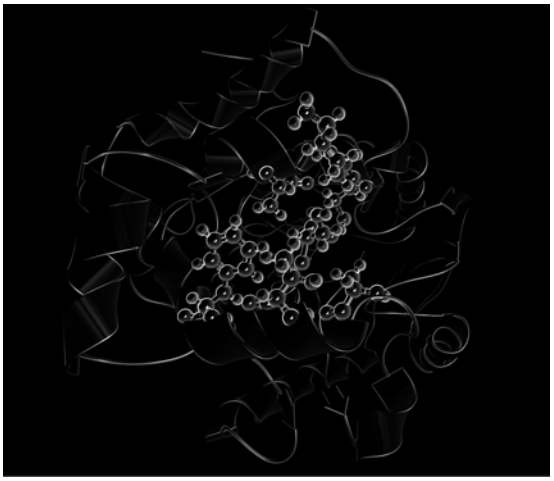

Figure 4.17: Combination of the horizontal and vertical Sobel convolution. Image produced using MATLAB 2016b [Mat16].

Finally, the already mentioned per-pixel occlusion check is performed, depending on individual depth texture texels. If a context texel occludes a focus texel, the color of the resulting pixel is linearly interpolated between focus and context image. The interpolation value is defined by the result of the combination of the horizontal and vertical edge detection. The result of the ghosting procedure is shown in Figure 4.19.

This ghosting procedure especially highlights edges, which emphasizes the context structure on the one hand, and on the other hand, allows the user the analysis of focus elements without any occlusions. Figure 4.20 shows a juxtaposition of classical visualizations and renderings produced using our ghosting approach.

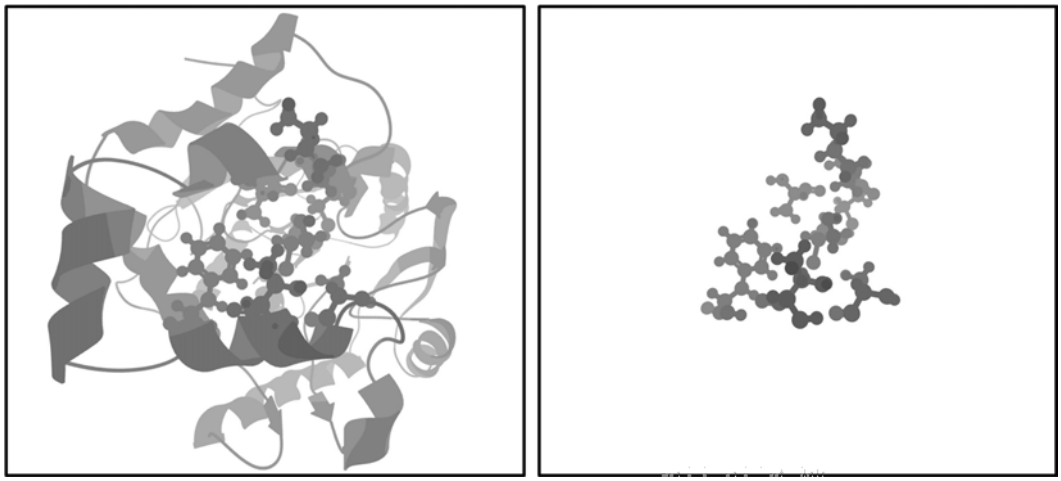

Figure 4.18: (left) Depth image consisting of all focus and context elements and (right) depth image consisting of only focus elements. Renderings produced using CAVER Analyst [Ana].

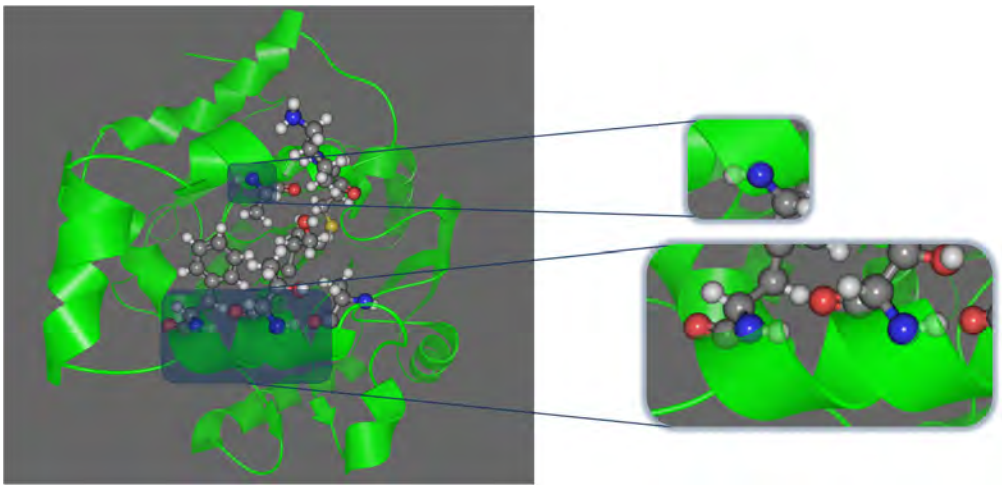

Figure 4.19: Final result of the ghosting procedure including highlighted regions where ghosting is especially visible. Renderings produced using CAVER Analyst [Ana].

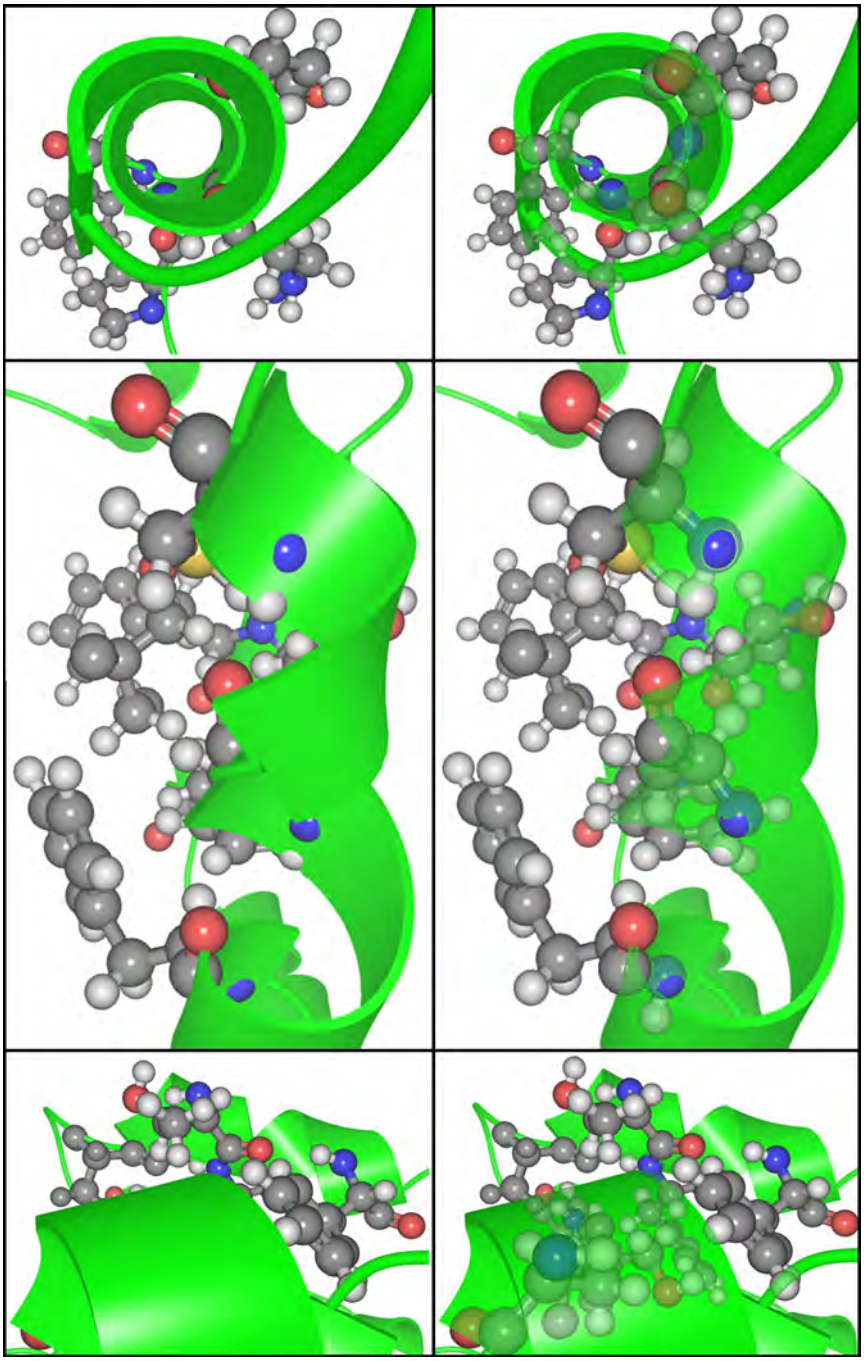

Figure 4.20: Three different examples visualizing the effect of (left) classical rendering without occlusion management and (right) ghosting. Renderings produced using CAVER Analyst [Ana].

## 4.3 Temporal Importance

After the spatial composition of individual frames, the temporal importance is considered. Our main goal was to decrease the duration of the MD-simulation and provide a more compact playback of the animation. Again, the user can interact with the visualization in real-time using a single or a combination of multiple importance functions. Thereby, individual frames are skipped and the playback speed is adapted, depending on the importance values of individual frames. If the temporal importance function is 1, the simulation is played using the original playback speed. If the function is 0, the playback speed is highly increased and if the importance value is between 1 and 0, the speed is increased depending on interpolated speed-up values. Unfortunately, the removal of individual frames disrupts the motion perception and increases *motion blindness*. Therefore, we additionally apply *motion blur* that reduces this disruption and simultaneously visualizes the speed of elements in a natural way.

The most important part of temporal importance is the use of different importance functions that influence the adaptive fast-forward playback speed of an animated MD-simulation. Inventing and implementing such functions itself was not part of this work. Loading and combining such functions was an already implemented feature of the CAVER Analyst [Ana] visualization toolkit. Part of our contribution was to extend this manipulation of importance functions and provide an implementation that allows the user to compute derivatives of such functions. Domain experts requested this functionality during one of our presentations, because they were particularly interested in sections where drastic changes in the functions occur.

Until now, we have always considered keyframes independently form each other. The first step was determining which molecular elements are visible, per keyframe, using a spatial importance function. Subsequently, we described a ghosting procedure, which is necessary to prevent context structures from occluding focus elements. In the next step, individual keyframes are no longer considered to be independent of each other. Instead, they are combined to an animated visualization using a temporal importance function, which introduces new challenges, which we will discuss in detail in the following.

The user can choose whether a single or a combination of multiple temporal importance functions should be used. Depending on the chosen spatial importance function, we analyze different molecular properties of either water molecules or ligands that can be used as temporal guidance. All these features are calculated per keyframe and determine the temporal importance value of this particular keyframe. Typical properties are:

> **Water molecules:** According to Vad et al. [VBJ+17], the number of *active*, *from active*, *to active*, and *inside* waters, or the number of waters that change between states (Section 6.2 provides a detailed explanation of different types of water molecules). Thereby, water molecules are categorized depending on whether they reach the active site within a protein or not. Another molecular property, which is suitable for temporal guidance is the *root-mean-square deviation* (RMSD) of atomic positions from the protein.

**Ligands:** The distance of a ligand to the active site, the speed or stuckness of the ligand, the change of the spatial position in X, Y, or Z direction, the distance of the ligand to the surface of the protein, or the RMSD.

An example of an importance function is shown in Figure 4.21. In this particular case, the importance function is influenced by the number of *inside water molecules* (colored yellow). If no such molecules are present, the temporal importance value of this particular keyframe is 0, if four molecules are visible, the function value is 0.5, and for the highest number of eight molecules, the function value is 1. During the computation of such functions, all individual keyframes of the MD-simulation are considered. The highest number of inside waters is then assigned to an importance value of 1. If fewer molecules are visible, the percentage is computed and assigned as importance value to this particular keyframe. As already mentioned, other geometric properties could be used instead of the number of individual inside water molecules.

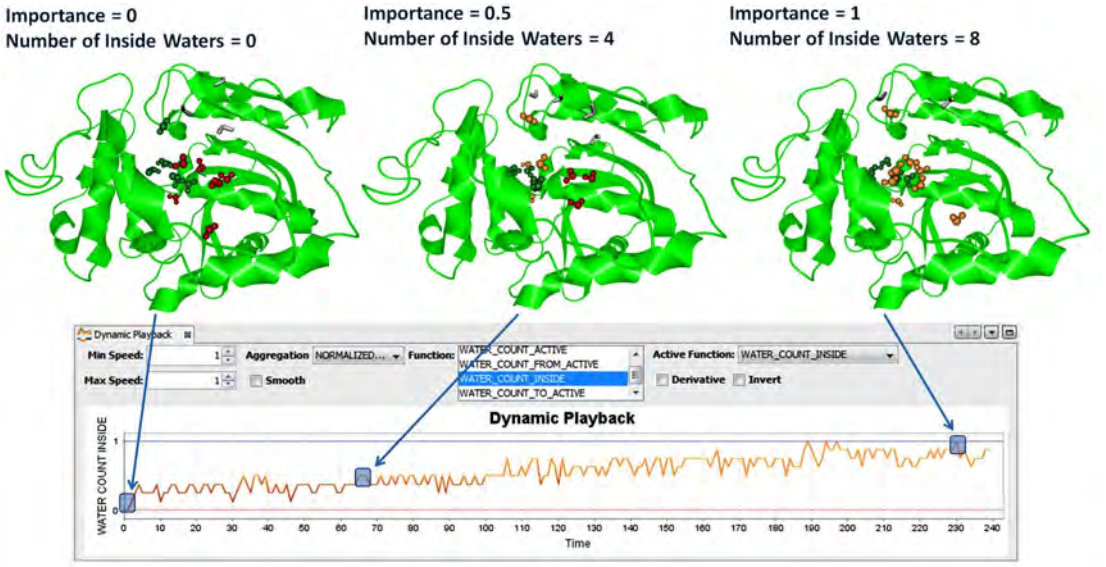


Figure 4.21: Visualization of three different keyframes of an MD-simulation with varying importance values depending on the number of visible *inside waters (colored yellow)*. (left) Keyframe 1: Contains no visible inside waters and has an importance value of 0. (middle) Keyframe 65: Four visible inside waters and an importance value of 0.5. (right) Keyframe 230: The highest number of eight visible inside waters and an importance value of 1. Renderings produced using CAVER Analyst [Ana].

If the importance values of the three keyframes, shown in Figure 4.21, are then applied to the adaptive playback speed of MD-simulations, this results in the following average rendering and visibility durations:

**Keyframe 1**, with an importance value of 0, was rendered 9 times with an average rendering time of 20.7 ms. This keyframe was visible to the user for the shortest period of time. In total only 187 ms.

**Keyframe 65** has an importance value of 0.5 and was rendered 18 times with an average rendering time of 16.8 ms. Keyframe 65 was therefore visible to the user for about 304 ms.

**Keyframe 230**, with an importance value of 1 was considered as highly relevant and therefore rendered 87 times with an average rendering time of 10.7 ms. As a result, keyframe 230 was visible for the longest period of time, namely 936 ms.

Adaptive playback speeds result in variable time savings, since unimportant keyframes are visible for a shorter period of time and the user has more time to explore the most relevant keyframes. This is, of course, highly dependent on the chosen importance function. Figure 4.22 shows a comparison of total playback times using six different temporal importance functions for exploring an underlying MD-simulation consisting of 1 000 keyframes. These results show an average time saving of 50–60 %.

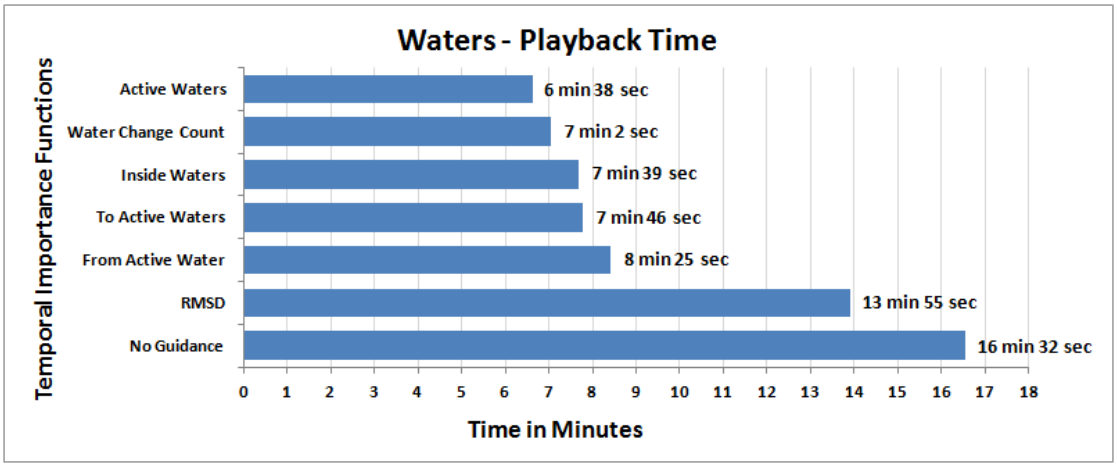


Figure 4.22: Visualization of playback times using six different temporal importance functions, which are necessary to explore an MD-simulation for water and protein interaction consisting of 1 000 keyframes. *No Guidance* represents a non-adaptive playback speed and assumes an importance of 1 for every keyframe of the simulation.

This temporal exploration can be used for any other arbitrary research task, for example, ligand-residue interactions instead of water-protein interactions. Figure 4.23 shows an example of the time savings for a longer MD-simulation consisting of 2 162 keyframes. Using a temporal importance function, to explore this dataset, results in an average time saving of 45 %.

### 4.3.1 Visibility Smoothing

Evaluating MD-simulations using a spatial importance function results in subsets of visible focus and context elements per keyframe. These subsets contain all molecular elements that are either in focus or context within this particular keyframe. This assignment is performed independently of the previous or next keyframe. If a temporal

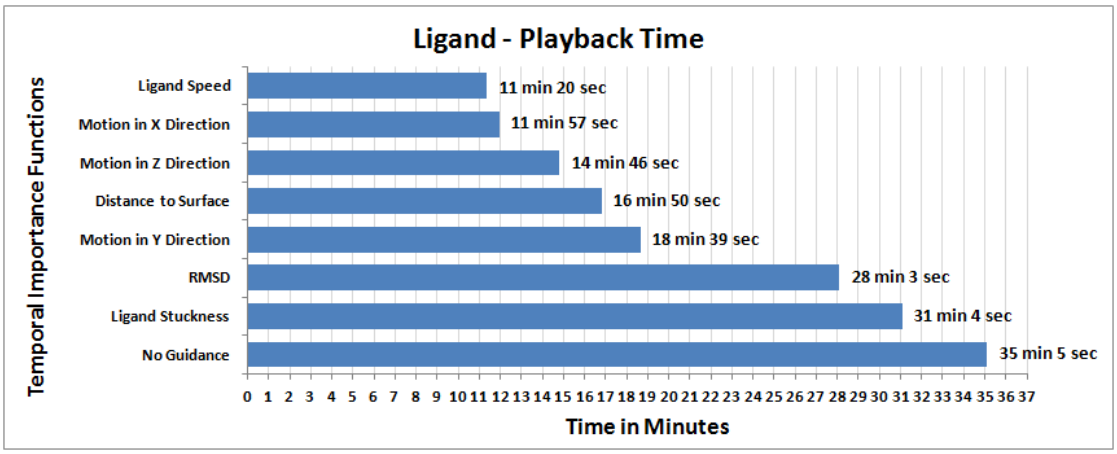


Figure 4.23: Comparison of seven different temporal importance functions used to explore an MD-simulation for ligand and residue interaction consisting of 2 162 keyframes. Again, *No Guidance* uses an equal playback speed for every keyframe.

importance function is then used for the adaptive playback of these keyframes, it is important to prevent visual flickering, for example, if molecular structures only pop up for a single keyframe. Therefore, this initially independent assignment of focus and context elements needs to be modified depending on previous and future keyframes. In the following, we will describe some special cases that explain why this additional modification also referred to as *smoothing* of subsets of visible elements per keyframe, is needed. The following examples refer to ligand-residue interactions, but are valid for arbitrary molecular scenarios:

> **Case 1:** A single residue approaches the ligand and in keyframe 2 and 7, the distance to it is below a predefined threshold. If a residue is rendered only for a single keyframe, it would pop up for a fraction of a second and immediately disappear again. The user would perceive this as distracting visual interruption. Therefore, the residue is smoothed-out and it will not be rendered. An example of this case is shown in Figure 4.24. In this example, only the residue buto not the ligand is displayed per keyframe.

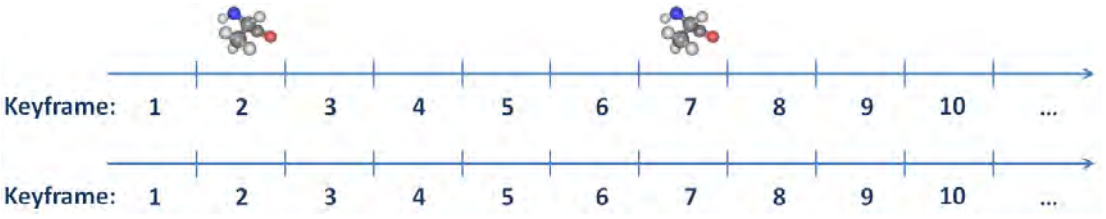


Figure 4.24: Visualization of a smoothing operation where an individual residue would unnecessarily attract the user's attention, since it is only rendered for a fraction of a second. (top) Visible residues before the smoothing. (bottom) Visible residues after the smoothing. Renderings produced using CAVER Analyst [Ana].

**Case 2:** A single residue is close to the ligand and is therefore in focus. For two keyframes (4, 5), the distance to the ligand increases and therefore the residue is no longer in focus. Not rendering a residue for two keyframes means that the residue will not be visible for a fraction of a second. This again causes a visual flicker and distracts the user. Although the distance of a residue to the ligand is greater than the defined threshold, we will still render the residue during these two keyframes to prevent any distractions. An example of this case is shown in Figure 4.25.

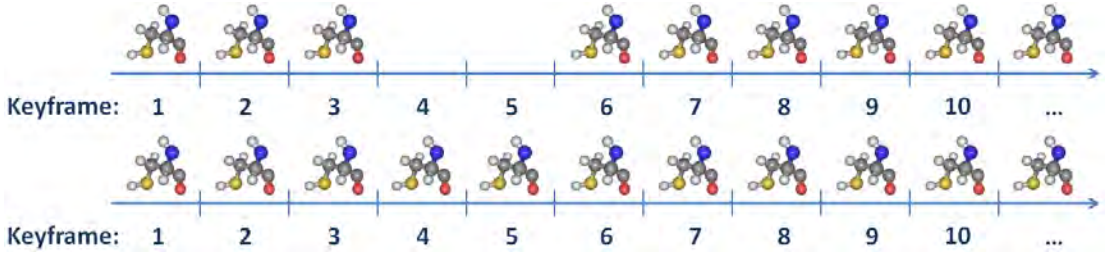


Figure 4.25: Visualization of an example where a residue would not be rendered for a fraction of a second. This would again unnecessarily attract the user's attention. (top) Visible residues before the smoothing. (bottom) Visible residues after the smoothing. Renderings produced using CAVER Analyst [Ana].

**Case 3:** Starting from the beginning of the simulation, a single residue is located close to the ligand. Then, the distance to the ligand increases (keyframe 4-8). Within that time, the distance to the ligand is decreased once (keyframe 6) and therefore it would be visible for a single keyframe. This means that the residue will slowly fade-out and then suddenly becomes visible, before it becomes invisible again. In order to prevent this flicker, we will not render the residue (in keyframe 6), although the distance to the ligand is below a chosen threshold. An example of this case is shown in Figure 4.26.

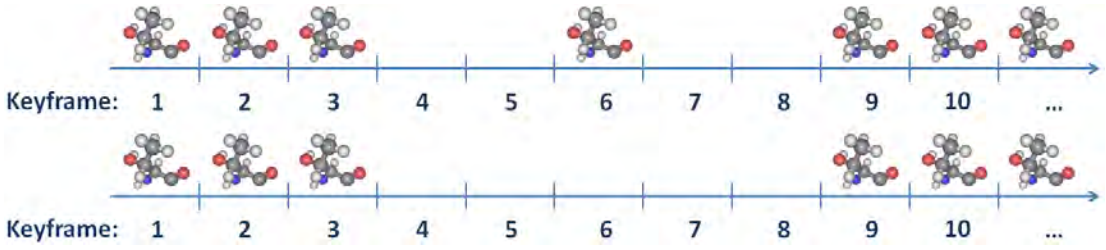


Figure 4.26: Visualization of a smoothing operation where a residue becomes invisible, suddenly pops up (keyframe 6), and then becomes invisible again. To prevent a visual flicker, the residue will be smoothed out. (top) Visible residues before the smoothing. (bottom) Visible residues after the smoothing. Renderings produced using CAVER Analyst [Ana].

**Case 4:** This case represents a typical setting that does not require any smoothing. From the beginning on, a residue is close to the ligand. Over time, the distance increases until it is too far away from the ligand. As soon as the distance to the ligand is greater than a chosen threshold, atom colors will fade-out to the context color. An example of this case, without any smoothing, is shown in Figure 4.27.

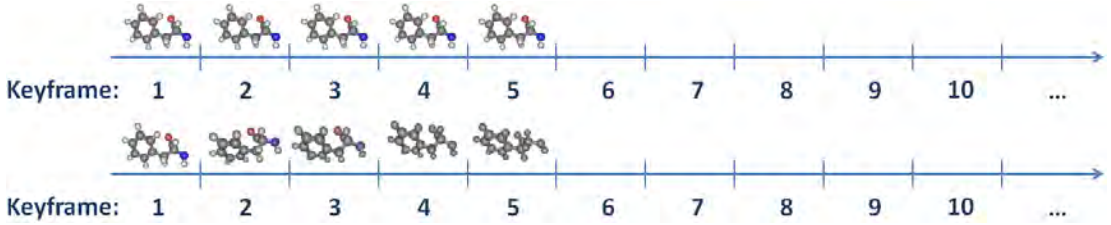


Figure 4.27: Visualization of a (top) typical case where no visibility smoothing is required. (bottom) Note that the residue is visible within the same keyframes, but the atom-colors slowly fade-out. Renderings produced using CAVER Analyst [Ana].

### 4.3.2 Color Interpolation

Once visibility smoothing is completed, it is ensured that individual molecular structures are visible continuously for a longer period of time. The minimum duration is dependent on user preferences and can be changed during the evaluation of the MD-simulation. In practice, a standard window size of five keyframes is used as default value. Nevertheless, if an element is not visible starting from the beginning of the simulation, namely keyframe 1, it will still pop up the first time it is rendered and suddenly disappear the last time it is rendered. Because of this sudden pop up, a context element would unnecessarily attract the user's attention.

To reduce this visual distraction, the first and most obvious possibility would be *alpha blending*. A blend function could be used, which changes the alpha values of individual structures in such a way that they slowly fade-in in the beginning and then fade-out at the end of their visibility. Unfortunately, alpha blending would interfere with the implemented ghosting procedure since multiple objects are then rendered semi-transparently and this would introduce additional problems concerning the rendering order. Instead, we added an *ease-in* and *ease-out* function for color interpolations. During ease-in, the colors of a molecule are slowly interpolated from the context color, by default it is gray (R=144, G=144, B=144), to the actual focus colors, defined by the CPK model [Kol65]. An example of this RGB ease-in color interpolation from a context color to individual CPK colors is shown in Figure 4.28.

During ease-out, the colors of a molecule are interpolated vice versa. Instead of switching the colors between two keyframes and within a fraction of a second, this process is again done continuously. An example of an ease-out interpolation is shown in Figure 4.29.

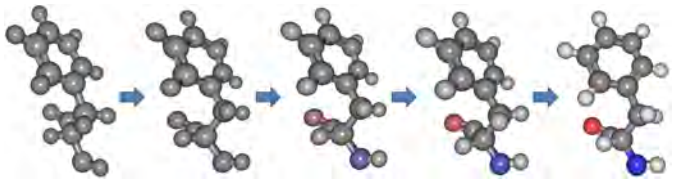

Figure 4.28: Smooth ease-in color interpolation from gray, as context color, to individual atom colors from the CPK model. Renderings produced using CAVER Analyst [Ana].

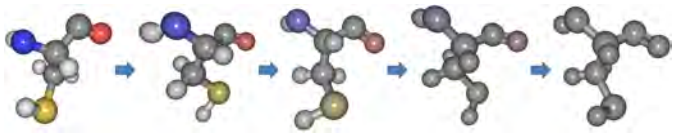

Figure 4.29: Smooth ease-out color interpolation from individual CPK atom colors, to the context color gray. Renderings produced using CAVER Analyst [Ana].

In the following, we will explain the concept of color interpolation that is independent of the playback speed and the number of keyframes, a residue is visible. The solution was a smooth color interpolation done in visualization time instead of keyframe time. As soon as a residue is rendered for the first time, we assign an individual stopwatch. This timer is then used for the interpolation of all atoms, the residue consists of. Using this keyframe independent time scale, any duration for ease-in and ease-out can be defined. After informal inspection, we set the ease-in and ease-out interval to 500 ms. The ease-in as well as the ease-out interval can vary and different durations can be used. Using this approach, the color interpolation is independent of the number of displayed keyframes.

Additionally, *Hermite interpolation* was used to provide basic functionality for color interpolation. A Hermite interpolation can be computed using a cubic polynomial, i.e., a polynomial of degree three. In addition to the starting and end point, in our case the colors $C_1$ and $C_2$, it requires a starting tangent $T_1$, an ending tangent $T_2$, and an interpolation value $x$ in the range of [0,1]. During ease-in, $C_1$ is the context color gray and $C_2$ represents the final CPK color:

$$C_{Hermite} = (2x^3-3x^2+1)\cdot C_1+(x^3-2x^2+x)\cdot T_1+(-2x^3+3x^2)\cdot C_2+(x^3-x^2)\cdot T_2 \quad (4.3)$$

Inspiration for our ease-in function comes from Penner [Pen] and Renaudeau [Ren]. In the final visualization a *quadratic ease-in* function, shown in Equation 4.4, was used.

$$f(x) = x^2 \quad (4.4)$$

A plot of the function from Equation 4.4 is shown in Figure 4.30.

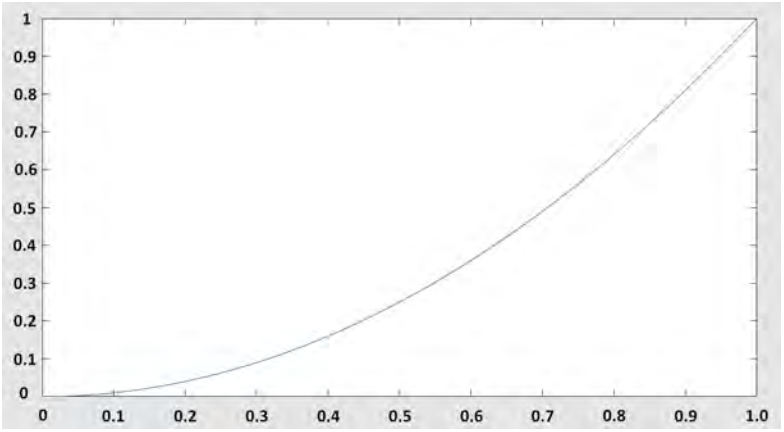


Figure 4.30: Quadratic ease-in function inspired by Penner [Pen] and Renaudeau [Ren]. Graph plotted using MATLAB 2016b [Mat16].

A general formula for linearly mapping a value $x_{AB}$ within the interval [A, B] to a new value $x_{ab}$ in the interval [a, b] is the following Equation 4.5:

$$x_{ab} = \frac{(x_{AB} - A) \cdot (b - a)}{(B - A)} + a \tag{4.5}$$

In order to be able to access the ease-in function depending on the elapsed time, we have to perform a linear mapping first. $F_{Time}$ represents the default duration of the ease-in in milliseconds, which is in our case 500 ms. This corresponds to a mapping from [0, $F_{Time}$], to an interval with a range from [0,1]. The following equation provides a formal definition:

$$[A, B] \mapsto [a, b] := [0, F_{Time}] \mapsto [0, 1]$$

This mapping can be done using Equation 4.5. Initially, $x_{AB}$ is in the range [0, $F_{Time}$] and represents the elapsed time since the ease-in process was started:

$$x_{ab} = \frac{(x_{AB} - 0) \cdot (0 - 1)}{F_{Time} - 0} + 0 = \frac{x_{AB}}{F_{Time}}$$

During the next step, the result of this mapping $x_{ab}$ is used to evaluate the quadratic ease-in function from Equation 4.4. The result of this evaluation $f(x_{ab})$ is already normalized and can therefore be directly used for the Hermite color interpolation from Equation 4.3.

In the following, we will provide an example of a residue, which is rendered nine times within 500 ms. Note that the *time* passed is just a representative of the passed time in milliseconds. In fact, the function will be executed much more often than only nine

times within 500 milliseconds, especially since our focus & context visualization runs in real-time with an average frame rate of about 100 FPS (Chapter 6 contains the exact hardware specifications and the tested datasets). The calculation results of this example are shown in Table 4.1 and the color transition is shown in Figure 4.31.

| Frame | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| time (in ms) | 0 | 75 | 140 | 205 | 300 | 365 | 410 | 485 | 545 |
| $x_{ab}$ | 0 | 0.15 | 0.28 | 0.41 | 0.6 | 0.73 | 0.82 | 0.97 | **1** |
| $f(x_{ab})$ | 0 | ∼0.01 | ∼0.08 | ∼0.17 | 0.36 | ∼0.52 | ∼0.66 | ∼0.93 | 1 |

Table 4.1: Ease-in calculation results for $x_{ab}$ and $f(x_{ab})$ using an example consisting of nine rendered frames. The *time* value corresponds to possible elapsed times in milliseconds that have passed since the ease-in process was started. The ease-in duration is additionally limited to 500 milliseconds and therefore $x_{ab}$ from frame 9 is clamped to 1 (colored red). Note that $f(x_{ab})$ was again rounded to two decimals.

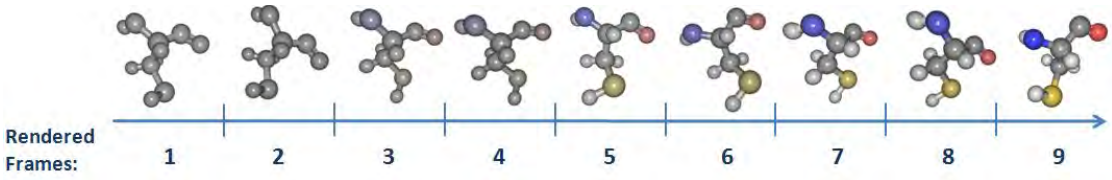


Figure 4.31: Example of a quadratic ease-in color interpolation, which is limited to 500 milliseconds instead of a certain number of visible keyframes. In this example the interpolation is performed nine times. Renderings produced using CAVER Analyst [Ana].

As soon as a residue is rendered for the second last keyframe, the ease-out process is started. Note that the ease-out color interpolation is independent from the number of keyframes and is only dependent on a predefined duration in milliseconds. Again, a stopwatch timer in combination with a predefined ease-out duration is used to fade-out individual residues. By default, the ease-out duration is 500 milliseconds. In the final visualization, a *quadratic ease-out* function, shown in Equation 4.6, was used:

$$f(x) = (2 - x)x \tag{4.6}$$

A plot of this function is shown in Figure 4.32. Note that the ease-out function still increases from 0 to 1 and therefore the starting color $C_1$ and $C_2$, and their tangents $T_1$ and $T_2$, need to be swapped. $C_1$ is then the focus CPK color and $C_2$ represents the context color gray.

Finally, we will provide an example with an ease-out duration of 500 milliseconds, within which a residue is rendered 11 times. The calculated results are shown in Table 4.2 and a visualization of this example is shown in Figure 4.33.

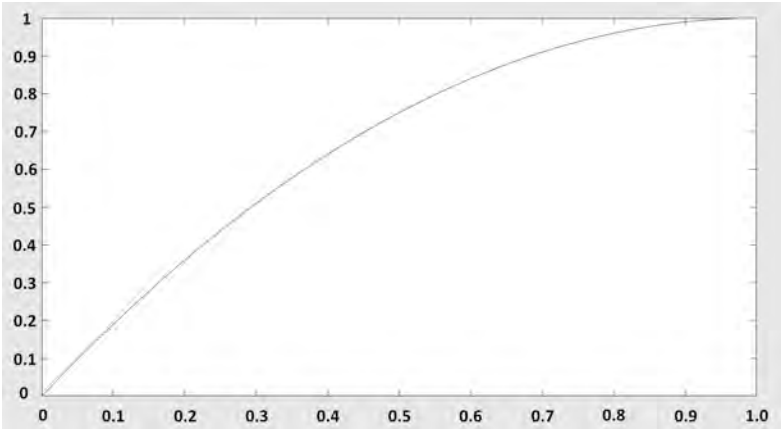


Figure 4.32: Quadratic ease-out function inspired by Penner [Pen] and Renaudeau [Ren]. Graph plotted using MATLAB 2016b [Mat16].

| Frame | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| time (in ms) | 0 | 50 | 105 | 145 | 195 | 250 | 285 | 340 | 395 | 455 | 505 |
| $x_{ab}$ | 0 | 0.1 | 0.21 | 0.29 | 0.39 | 0.5 | 0.57 | 0.68 | 0.79 | 0.91 | **1** |
| $f(x_{ab})$ | 0 | ~0.19 | ~0.38 | ~0.5 | ~0.63 | 0.75 | ~0.82 | ~0.9 | ~0.96 | ~0.98 | 1 |

Table 4.2: Ease-out calculation results for $x_{ab}$ and $f(x_{ab})$ using an example consisting of 11 keyframes. The *time* value corresponds to possible elapsed times in milliseconds that have passed since the ease-out process was started. The ease-out duration is additionally limited to 500 milliseconds, therefore $x_{ab}$ from frame 11 is clamped to 1 (colored red). Note that $f(x_{ab})$ was again rounded to two decimals.

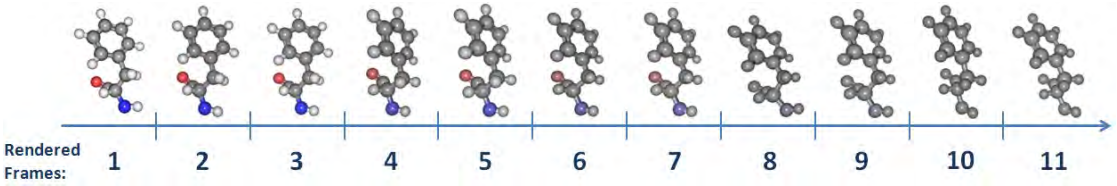


Figure 4.33: Example of a quadratic ease-out color interpolation, which is limited to 500 milliseconds instead of a certain number of visible keyframes. In this example the interpolation is performed 11 times. Renderings produced using CAVER Analyst [Ana].

### 4.3.3 Calculation of the Derivative

In the following, we will describe the benefits of calculating the derivative of an importance function and how it can be computed. The calculation of the derivative of one or a combination of multiple importance functions is necessary, if domain experts want to investigate drastic changes within such functions. An example could be that the speed of a ligand changes abruptly.

Although every importance function appears to be continuous, it is a discrete evaluation of importance values per keyframe. Therefore, the approximation of the derivative $f'(x)$ of such a function $f(x)$ can be calculated using the *symmetric difference quotient*, which is described in Equation 4.7. $x$ represents the domain of the function and $h$ the distance between two discrete function values:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} \tag{4.7}$$

A visualization of the computation of an approximated derivative, using the symmetric difference quotient, is shown in Figure 4.34.

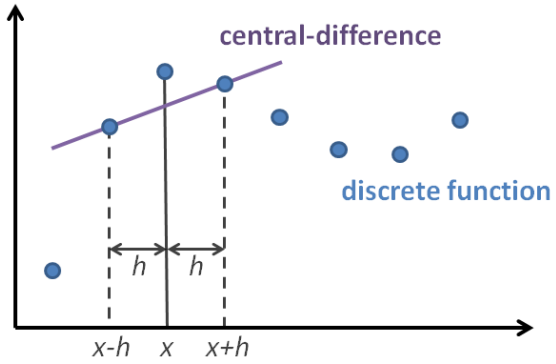


Figure 4.34: Visualization of the symmetric difference quotient. It shows the secant line that is used to approximate the slope of the tangent.

The smaller $h$ is, the better the tangent is approximated. To simplify the computation of the derivative, $f(x+h)$ represents the importance value of the next keyframe and $f(x-h)$ represents the importance value of the last keyframe. Importance functions are limited to the interval [0,1] and since we are interested in both, positive as well as negative slopes, the absolute value $|f'(x)|$ is used as importance value. Figure 4.35 shows an example of an importance function, consisting of 240 keyframes, and its derivative.

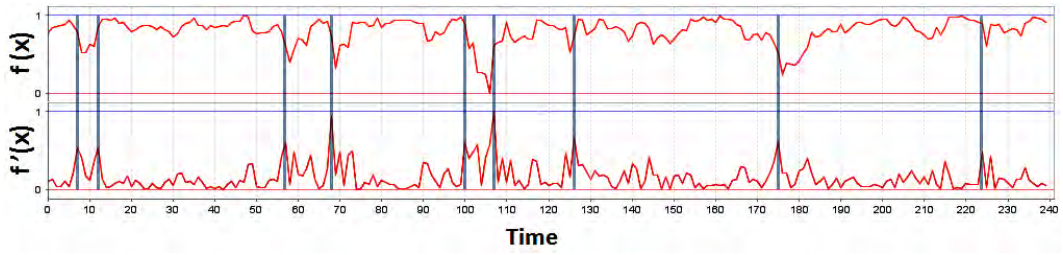


Figure 4.35: Visualization of (top) a normalized importance function and (bottom) its normalized derivative. Additionally nine peaks of the derivative are highlighted. Importance function and derivative provided by CAVER Analyst [Ana].

### 4.3.4 Motion Blur

Visibility smoothing of individual keyframes, including ease-in- and ease-out functions for color interpolations already try to prevent visual interrupts that attract the user's attention. Unfortunately, adaptive fast-forward playback of the simulation can still disrupt the motion perception and increase *change blindness*. Therefore, we additionally apply *motion blur* that on the one hand, reduces visual disruption and on the other hand, helps to better understand the movement of atoms and molecules, as objects leave visible traces. In addition, we will mention how motion blur can reduce negative side effects like *change blindness*.

Navarro et al. [NSG11] define *motion blur* as visible flow marks representing trajectories of objects. It is an essential reference, already known from our *human visual system*. According to the authors [NSG11], motion blur is commonly perceived as a very natural and pleasant visual effect since it helps our brain to better understand object movements. In contrast to recording devices such as analog or digital cameras and video cameras, where motion blur is created automatically, in rendered images or videos it must be simulated. The reason why this effect does not appear in rendered images is that such images correspond to recordings made with an infinitely short exposure time. According to Navarro et al. [NSG11], motion blur rendering can be divided into the following approaches: *analytic methods*, *geometric substitutions*, *Monte Carlo methods*, *post-processing methods*, *hybrid methods*, and *physically accurate methods*. The most appropriate approach for our visualization was the *post-processing method*. It represents a technique where the rendering of motion blur is independent of the level of detail within a scene. Instead of trajectories of individual objects, only the pre-rendered images, stored as textures, are used to blur the image, which is currently rendered. This makes it especially suitable for real-time applications.

In the following, we present a theoretical explanation of how motion blur was integrated into our CAVER Analyst [Ana] extension. It is performed after the ghosting procedure and therefore represents the last part of our rendering pipeline (the entire pipeline is explained in detail in Chapter 5).

An overview of the blurring algorithm is shown in Figure 4.36.

The blurring procedure can be divided into the following five components:

1. **Current frame $F_{Current}$ :** Motion blur represents the final step of our rendering pipeline and $F_{Current}$ is therefore the direct output of the ghosting procedure.
2. **Accumulated frames $F_{Accum}$ :** They represent a composition of all continuously rendered and subsequently blended frames from the past.
3. **Weight function:** The basis of the weight function is on the one hand, the temporal importance function, which defines the playback speed of the animated simulation, and on the other hand, a GUI slider. If the temporal importance value $W_{Frame}$ of a keyframe is low, the playback speed and additionally the motion blur intensity is increased. The GUI slider value $W_{Slider}$ is optional and not necessarily

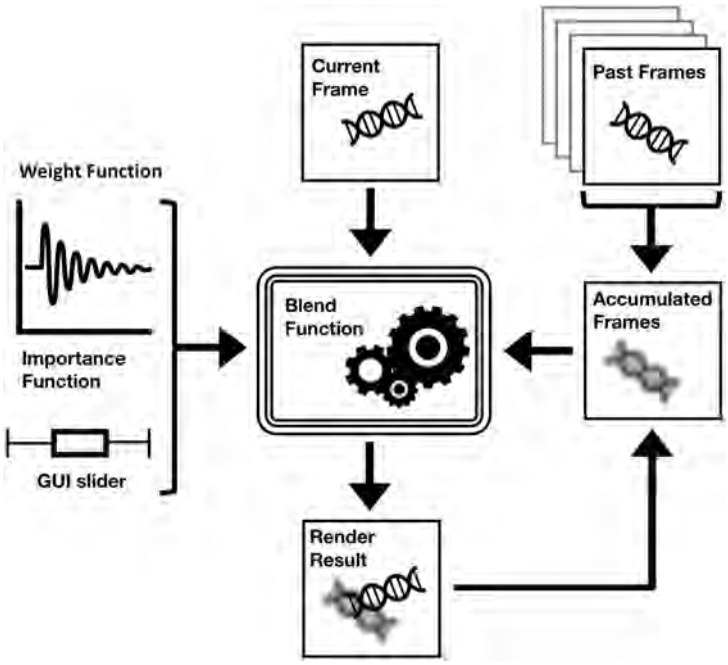


Figure 4.36: Overview of the motion blur procedure which uses a weight function to blend between the currently rendered frame and already accumulated frames. The final result is then stored as accumulated frame and used for the next iteration.

needed for the actual exploration. It allows the user to manually increase or decrease the overall strength of motion blur. Equation 4.8 provides an overview of the composition of the weight function:

$$W_{Sum} = W_{Frame} + W_{Slider} \tag{4.8}$$

$W_{Frame}$ is thereby in the range [0, 1]. The range of $W_{Slider}$ was chosen empirically. We defined a range (1,2] including 1.1 as default slider value. To prevent numerical errors evaluating Equation 4.9, $W_{Sum}$ is clamped to the interval (1,2]. For example, an error could occur if $W_{Sum}$ is exactly 1. Note that the upper bounds of both intervals, $W_{Slider}$ and $W_{Sum}$, could be increased arbitrarily.

4. **Blend function:** The blend function represents the main part of the procedure. Using $W_{Sum}$, it interpolates between the currently rendered frame $F_{Current}$ and the accumulated frames $F_{Accum}$. Equation 4.9 provides the used blend function:

$$F_{Blurred} = F_{Accum} \cdot \frac{1}{W_{Sum}} + F_{Current} \cdot \frac{W_{Sum} - 1}{W_{Sum}} \tag{4.9}$$

5. **Rendering result $F_{Blurred}$:** The final result represents the composition, which will be displayed on screen. Additionally, in preparation for the next rendering cycle $F_{Accum}$ is overwritten by $F_{Blurred}$.

Figure 4.37 shows a visualization of three different keyframes of an MD-simulation with unequal importance values.

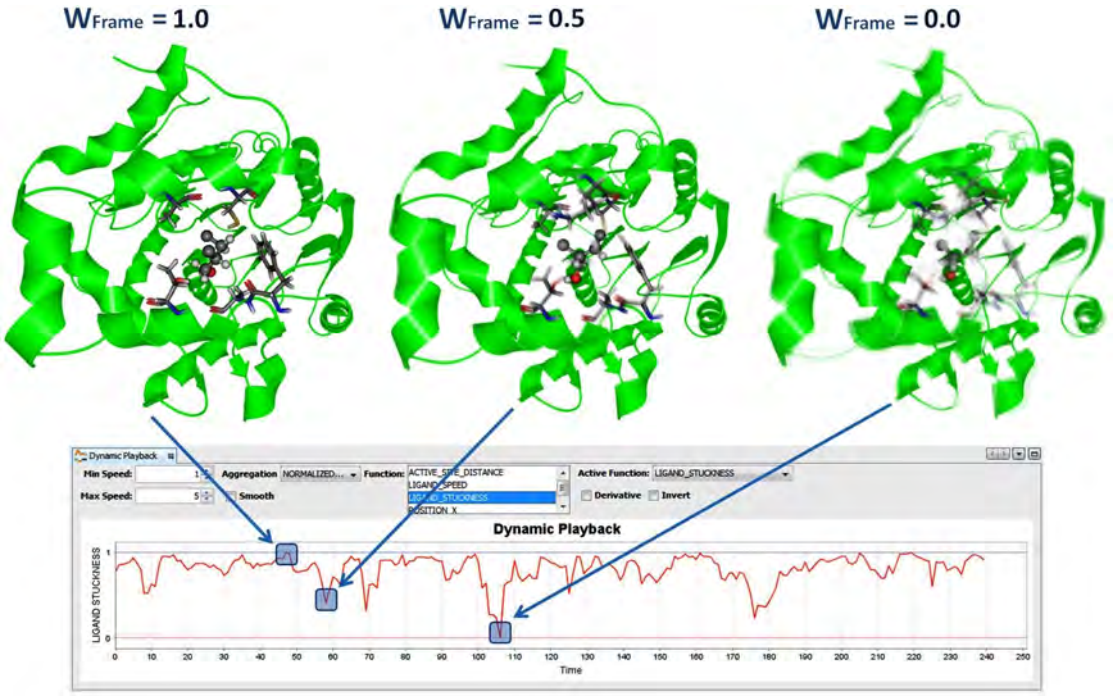


Figure 4.37: Visualization of three different example of motion blur intensities. (left) Keyframe 48 with an importance of 1 and a $W_{Sum}$ of 2. (middle) Keyframe 59 with an importance 0.5 and a $W_{Sum}$ of 1.6. (right) Keyframe 107 with an importance of 0 and $W_{Sum}$ of 1.1. Renderings produced using CAVER Analyst [Ana].

Next, we will provide the exact calculation results of the above described keyframes:

**Keyframe 48** has a temporal importance value $W_{Frame}$ of 1. In combination with a default slider value $W_{Slider}$ of 1.1 and after the clamping to an interval of (1,2] this results in an overall weight $W_{Sum}$ of 2. Therefore, the two frames $F_{Accum}$ and $F_{Current}$ will be blended like the following:

$$F_{Blurred} = F_{Accum} \cdot 0.5 + F_{Current} \cdot 0.5$$

Assuming a real-time frame rate of at least 60 FPS, or in our case an average frame rate of 100 FPS, this results in a visually imperceptible blurring.

**Keyframe 59** has an importance value $W_{Frame}$ of 0.5. After the clamping and in combination with the default slider weight $W_{Slider}$, this results in an overall weight $W_{Sum}$ of 1.6. The blending will therefore be performed as follows:

$$F_{Blurred} = F_{Accum} \cdot 0.625 + F_{Current} \cdot 0.375$$

Using the GUI slider, the user can additionally increase or decrease the blurring strength. Note that an increased slider weight results in a lower motion blur intensity. Assuming that the slider value was manually changed to a new weight $W_{Slider}$ of 1.3, this results in an overall weight $W_{Sum}$ of 1.8 and the following blend function:

$$F_{Blurred} = F_{Accum} \cdot 0.\dot{5} + F_{Current} \cdot 0.\dot{4}$$

**Keyframe 107** has an importance value $W_{Frame}$ of 0. Using a default slider value $W_{Slider}$ of 1.1 results in an overall weight $W_{Sum}$ of 1.1 and the following blend function:

$$F_{Blurred} = F_{Accum} \cdot 0.9 + F_{Current} \cdot 0.1$$

To prevent motion blur from having a negative impact on the exploration of an MD-simulation, it is only activated if the position of the camera is not changed by the user. As soon as the user interacts with the visualization by changing the viewpoint, motion blur is deactivated. Otherwise, this would lead to distracting and disturbing motion streaks since flow marks are no longer the result of object movements but the consequences of camera changes. Especially orbital rotations around objects highly suffer from these negative effects.

As mentioned in the beginning of this section, motion blur could additionally help to reduce different negative effects like *change blindness*, which was introduced by Scott-Brown et al. [SBC07] based on security camera records. It is a negative effect that can result while observing motion. It describes problems that arise through short interruptions or cuts in videos, blinking, or any other visual interruptions. People are then no longer able to perceive even major visual changes.

In 1998, Simons and Levin [SL98] demonstrated this in a user-study. Participants were shown sequences of pictures. Between these pictures, they were shown a gray screen for 80 milliseconds. This short interruption was long enough that most of the participants were not able to see even large objects, like buildings, move, appear, or disappear.

Unfortunately, there are also negative side effect of using motion blur. Most obviously, it is the loss of fine details since parts of the scene are blurred. Motion blur can therefore be an undesirable effect if scenes contain a very high degree of spatial details that need to be perceived and processed by the user. We consider this to be negligible because the geometric level of detail has already been reduced using a spatial importance function. In addition, the playback speed of keyframes is only increased if their temporal importance value is less than 1 and therefore only less important keyframes are blurred.

CHAPTER 5

# Implementation

This chapter focuses on implementation details of the presented interactive real-time visualization of MD-simulations. We will provide an overview of programming and shader languages that were used to implement required features and functionality. Additionally, we will explain why a pre-computation step is required after the import of MD-simulation keyframes and why this pre-computation has to be recomputed if the spatial importance function changes during the exploration. In contrast, we will present the reason why the temporal importance function does not have to be recalculated even if a single function or a combination of several functions are changed.

The underlying framework, CAVER Analyst [Ana] 2.0 BETA, is based on the Java platform JDK 1.8, NetBeans IDE 8.2, and was developed using the NetBeans platform nb731. Furthermore, OpenGL was used as graphics API and the corresponding vertex, geometry, and fragment shaders were written in GLSL.

An overview of the entire system, representing our visualization pipeline, is shown in Figure 5.1. Our spatio-temporal focus & context visualization consists of the following six necessary steps: First, the import of already computed MD-simulation keyframes. Second, the evaluation of the spatial importance function, which creates subsets of focus and context elements per keyframe. Third, visibility smoothing which guarantees that individual objects are visible for a predefined number of keyframes. Fourth, visibility management through ghosting, which prevents occlusions of focus elements. Fifth, the evaluation of the temporal importance function, which defines how often a keyframe is rendered. Sixth, motion blur to reduce visual clutter and provide visual feedback of the currently used playback speed.

Similar to Chapter 4, we will divide the explanation of implementation details into Section 5.1 *Spatial Importance* and Section 5.2 *Temporal Importance*.

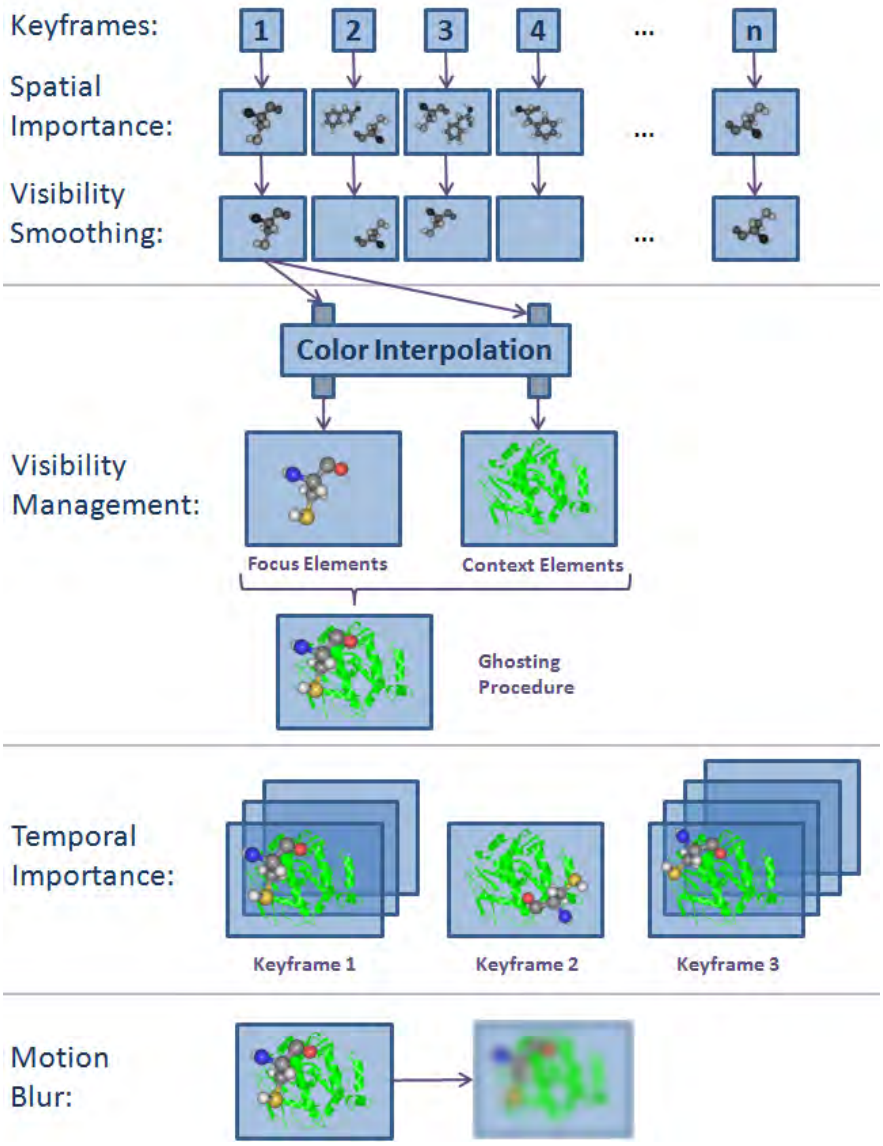


Figure 5.1: Overview of the entire visualization pipeline consisting of six important steps (from top to bottom): (Keyframes) Initially, a pre-computed dataset of MD-simulation keyframes is imported. (Spatial Importance) Next, the user defines a spatial importance function that is used to create subsets of visible focus & context elements per keyframe. (Visibility Smoothing) Based on these subsets, visibility smoothing is performed. (Visibility Management) During the next step, ghosting is performed based on the already colored molecular structures. (Temporal Importance) The next step is dependent on the temporal importance function. The more important a keyframe is, the more often the keyframe is rendered. (Motion Blur) Finally, already rendered images are blurred depending on the currently used playback speed.

## 5.1 Spatial Importance

After importing the MD-simulation keyframes, the user can select a spatial importance function using an already described Java Swing GUI element provided by CAVER Analyst [Ana]. Depending on the chosen spatial importance function, we subsequently assign default rendering strategies to elements. All representation types can be changed arbitrarily by the user during runtime. To enable this, we will present all required implementation details in the next subsections.

### 5.1.1 Rendering Strategies for Focus & Context

The functionality, to be able to arbitrarily change rendering strategies during runtime, was implemented using two *java.util.HashMap<Integer, Set<Residue>>* collections. One is used for focus and one for context elements. The *Residue* class represents a container where single amino acid residues are stored in. Among other properties, it contains the list of atoms it consists of, the chain it belongs to, the exact residue type, and the sequence number. The *java.util.HashMap.entrySet()* contains a *java.util.HashSet<Residues>* collection of all assigned residues. The integer values stored in the *java.util.HashMap.keySet()* represent individual keyframes of an MD-simulation. The size of the *keySet* represents the number of imported keyframes. An integer value, representing the currently displayed MD-simulation keyframe, can then be used to access all elements associated with this keyframe. A visualization of our HashMap collection, used to store molecular elements per keyframe, is shown in Figure 5.2.

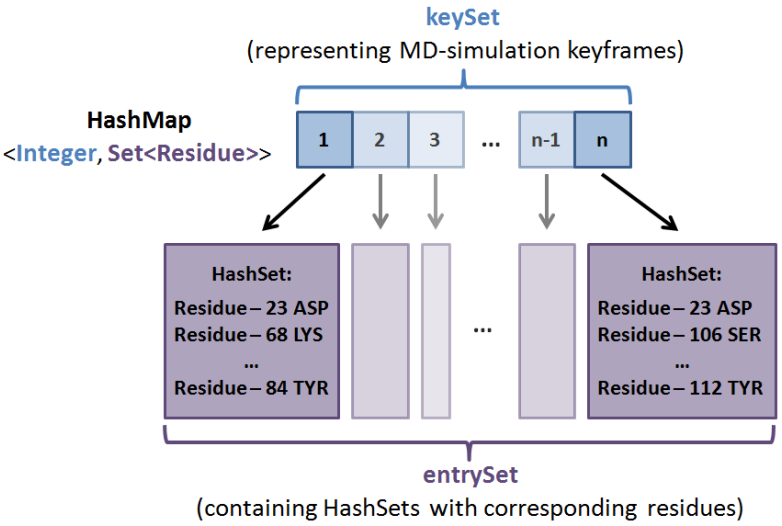


Figure 5.2: Visualization of a HashMap collection, which is used to store and access either focus or context elements. The *keySet* contains the number of the corresponding MD-simulation keyframe and the *entrySet* contains HashSets of residues that are assigned to these keyframes.

This HashMap data structure is advantageous because the selected rendering strategy is independent of the collection, since both HashMap collections only store high-level information about which elements need to be rendered per keyframe. As a result, the rendering strategy can be changed at runtime and no further pre-computations are necessary. In contrast to this, the preprocessing must be recomputed if the spatial importance function changes. For example, if the user wants to focus on different molecular elements. This is necessary since the two HashMap collections, for focus and context elements, otherwise contain wrong *entrySets* of residues.

### 5.1.2 Visibility Management

While computing the *spatial importance* of visually complex and dense molecular scenes, the most important step is the *ghosting* procedure. In the following, we will present all required implementation details.

To perform ghosting, two *frame buffer objects (FBO)* that provide both, a color and a depth attachment, are necessary. The first FBO is used to render all focus elements and the second FBO is used to render both, focus and context elements. The four resulting textures, two depth and two color textures, are then passed to a fragment shader. Depending on the depth values of the two depth textures, color pixels are composed using an OpenGL fragment shader. In the shader, the following three possibilities, which are additionally shown in Figure 5.3, are checked by if-else statements:

1. The depth value of the focus pixel is exactly 1, i.e., the distance to the far plane. This means that there is no focus element and therefore, the output fragment of the shader only consists of the pixel from the context color texture.
2. The depth value of the focus pixel is smaller than the depth value of the context pixel. In this case, the focus pixel occludes the context pixel. The output of the fragment shader is therefore only dependent on the focus color texture.
3. The depth value of the context pixel is smaller than the depth value of the focus pixel. This time, a context pixel occludes the focus pixel which corresponds to a situation where ghosting is required. Therefore, the context color pixel including the 8-connected neighbors are converted to gray values and convoluted with a vertical- and horizontal Sobel kernel. Next, the results are combined using a linear interpolation between the two pixels from the focus and context color texture. This result represents the return value of the fragment shader.

## 5.2 Temporal Importance

In the following section we will present implementation details of our CAVER Analyst [Ana] extension regarding the temporal importance function. We will explain how visibility smoothing, keyframe independent color interpolation, and motion blur can be implemented using Java and OpenGL.

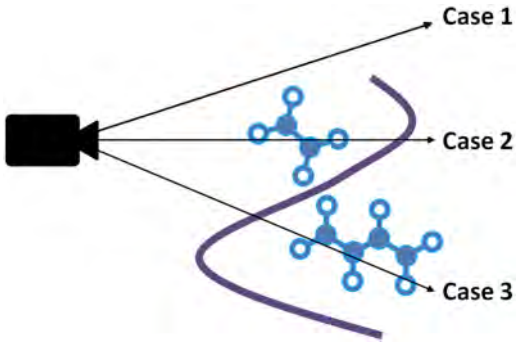


Figure 5.3: Comparison of three different cases, the fragment shader has to check using if-else statements: (Case 1) The depth value of the focus depth texture is 1 which does not require any blending. (Case 2) The focus depth value is smaller than the context depth value and therefore no ghosting is required. (Case 3) The context depth value is smaller than the focus depth value and therefore ghosting is required.

### 5.2.1 Visibility Smoothing

In the following, we will present the three essential implementation steps that were necessary to perform visibility smoothing. In addition, a visualization of the required data structures, including some example values, are shown in Figure 5.4.

1. Initially the number of keyframes, a residue is visible, is calculated and stored in a *java.util.HashMap<Integer, Map<Residue,Integer>>* collection. The *keySet* corresponds to the MD-simulation keyframe. The *entrySet* contains HashMaps of all residues that are visible within that particular keyframe including an integer value storing the number of keyframes this residue is still visible. This part of the algorithm is shown in Figure 5.4 (left).
2. During the next step, the algorithm iterates over the collection and if the number of visible keyframes is above a chosen threshold, by default five keyframes, the residues are copied to a *java.util.HashMap<Integer, Set<Residue>>* collection containing the smoothed residues. In contrast, if this integer threshold is below the default visibility duration, residues are discarded. The second part of the algorithm, including the collection, which stores the smoothed residues, is shown in Figure 5.4 (right).
3. Finally, the algorithm searches for sections where residues are not visible for less than the default threshold of five keyframes. Residue 141 PHE represents an example, which is shown in Figure 5.4. This residue is visible from keyframe 1–26 and from keyframe 28–100. The third step of the algorithm, for example, adds residue 141 PHE to the entrySet of keyframe 27.

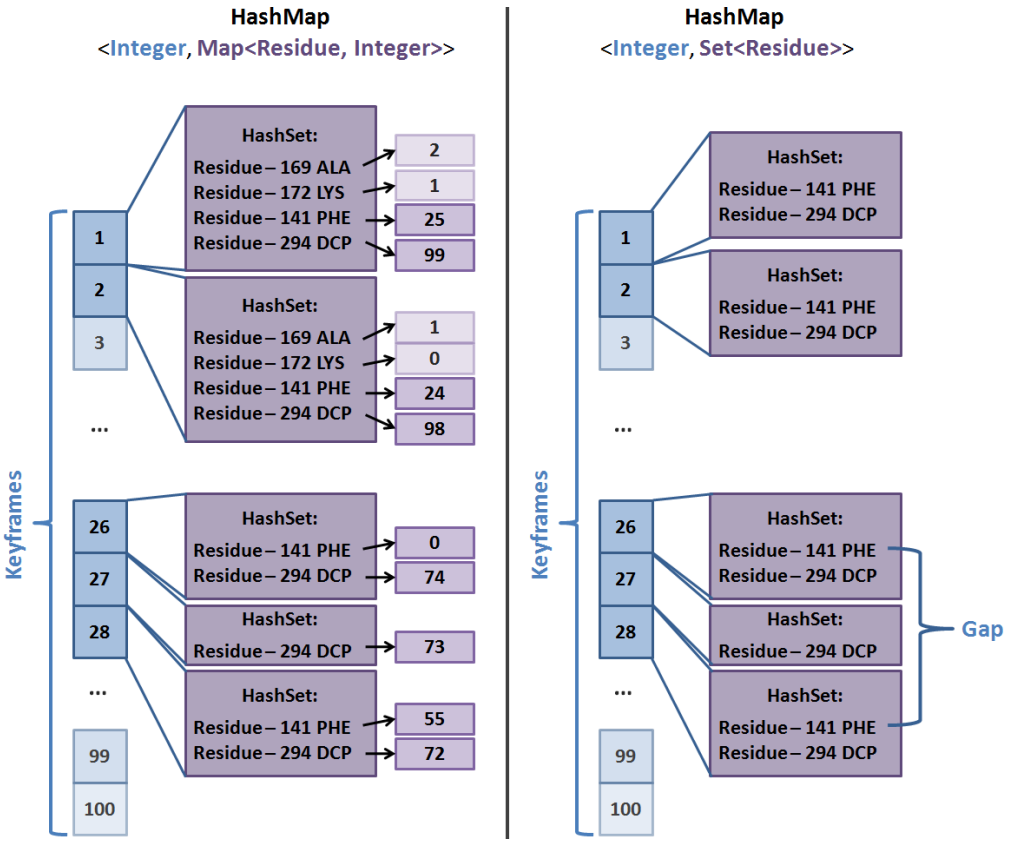


Figure 5.4: Visualization of the collections used for visibility smoothing. (left) A HashMap that is created during the first step of the smoothing procedure and stores collections of residues including the total number of keyframes these residues are visible. (right) A HashMap is created during the second step of the smoothing procedure. It contains smoothed residues, which are all visible for at least the CAVER Analyst [Ana] default length of five keyframe. This HashMap is subsequently used to detect gaps where residues are not rendered for less than five keyframes. An example of such a gap is Keyframe 27.

### 5.2.2 Color Interpolation

The following approach was used for the ease-in and ease-out color interpolation. A *java.util.HashMap<Residue, StopWatch>* collection is used to assign stopwatches to individual residues. The *keySet* consists of single residues and the *entrySet* contains the corresponding instances of the *org.apache.commons.lang3.time.StopWatch* class. During the rendering process, this collection can then be used to find out how much time has already passed since the ease-in or ease-out procedure started. Since color interpolation is only dependent on the elapsed time, this value can be used for the Hermite interpolation. As soon as the ease-in or ease-our procedure is finished, the timers are stopped and the entry is removed from the collection. A visualization of the collection, used to assign individual stopwatches to residues, is shown in Figure 5.5.

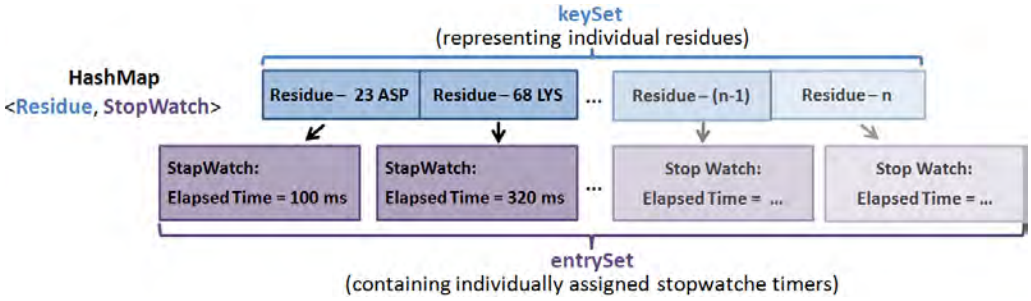


Figure 5.5: Visualization of a HashMap collection, which is used to assign individual StopWatch timers from the *entrySet* to residues from the *keySet*.

### 5.2.3 Motion Blur

The following section describes how *motion blur* was implemented. Inspiration for the technical implementation of motion blur comes from *accumulation buffers*, which are widely used in OpenGL. This approach was adapted according to the underlying CAVER Analyst [Ana] framework and provides a tailor-made implementation to our requirements. In our case, three different FBOs are used. To simplify the explanation, they are called $FBO_{MotionBlur}$, $FBO_{Accum}$, and $FBO_{Current}$ and the following implementation steps are required:

1. The first step is binding and clearing $FBO_{MotionBlur}$. It is the render target and stores the result of the motion blur procedure. Next, the color attachment of $FBO_{Current}$ is bound since it contains the rendering result of the current render cycle. If $FBO_{Accum}$ already contains accumulated images, its color attachment is bound. Otherwise, the color attachment from $FBO_{Current}$ is also used once as accumulation FBO placeholder until $FBO_{Accum}$ contains a valid color attachment. Next, a uniform float value, representing the blend weight, is passed to the fragment shader that subsequently performs the blending.
2. During the next step, $FBO_{Accum}$ is bound and cleared. Subsequently, the color attachment of $FBO_{MotionBlur}$ is bound as it already contains the final blended image. Then, a copy program is executed to copy and overwrite the color attachment of $FBO_{Accum}$. This serves as preparation for the next render cycle.
3. The color attachment of $FBO_{MotionBlur}$ is then displayed on the screen which completes the motion blur procedure.

CHAPTER 6

# Use Cases

The following chapter presents two possible real-world use cases, which were created in collaboration with domain experts from the Loschmidt Laboratories in Brno, Czech Republic. These examples illustrate how diverse and varying MD-simulations can be and therefore how adaptive our visualization approach must be. Section 6.1 presents the first use case, which explores the influence of individual residues on the ligand and Section 6.2 presents the second example, which examines individual water molecules and how they interact with a protein. Both use cases have already been presented twice, to multiple researchers, to ensure that they improve their workflow, allow them to develop new hypotheses more efficiently and faster, and gain novel insights into MD-simulation datasets.

A fundamental requirement is that the exploration of the simulation can be done in real-time. According to Akenine-Möller et al. [AMHH08], 15 FPS are at least required for real-time applications, but most video games aim for 60 FPS or more. The authors [AMHH08] additionally state that starting from 72 FPS differences to higher FPS are no longer detectable. If the user wants to interact with the visualization, this should be possible without FPS drops or additional waiting times. For example, if rendering strategies of individual elements are changed, the strength of motion blur is adapted, the color of objects is changed, or the importance function is modified. In addition, it should be possible that users can freely move the camera and thereby always have the best view of focus elements without having to wait for additional computations.

The basic procedure for both use cases is the following: The user loads a dataset of an MD-simulation consisting of *Protein Data Bank* (PDB) files. It represent a standard file format containing atoms and atomic coordinates from macromolecules. Next, focus and context elements are chosen using a spatial importance function. Afterwards, the preprocessing is started which analyzes the whole simulation frame by frame. Once the preprocessing is completed, the user can select a single or a combination of multiple

temporal importance functions from a list. As soon as a function is selected, the MD-simulation can be explored using an adaptive playback speed and all interactions can be performed in real-time.

The following operating system and hardware were used for the analysis of the use cases:

| | |
|---|---|
| **Operating system:** | 64-bit Windows 7 Home Premium |
| **CPU:** | Intel(R) Core(TM) i7-3930K CPU @ 3.20GHz |
| **GPU:** | NVIDIA GeForce GTX 1070 with 8 GB memory |
| **RAM:** | 16 GB |

## 6.1 Influence of Residues on the Ligand

The following use case should help to investigate how residues interact with a ligand. This practical example of an MD-simulation is based on a protein from *The Protein Data Bank* [BWF+], which is called dhaa and consists of 2 162 simulation keyframes (∼370 KB per keyframe and a total size of ∼785 MB), 316 residues, 293 amino acid residues, 23 ligands, and 4 684 atoms. The ligand in focus is named 294 DCP and consists of 5 carbon, 6 hydrogen, and 1 oxygen atom(s).

The first use case can be defined by the following properties:

**Research question:** How do residues that are close to the ligand influence its movement, interaction, and binding with other molecules?

**Spatial focus:** Ligand (294 DCP).

**Temporal focus:** For this use case, a (smoothed) combination of the stuckness and speed of the ligand is used as temporal focus.

**Spatial focus area:** The area is defined by residues surrounding the ligand whose van der Waals radii overlap is greater or equal to -0.4 Å, as recommended by *UCSF Resource for Biocomputing, Visualization, and Informatics* [Chi].

**Context:** The whole protein.

**Rendering strategy focus:** The ligand is rendered using either a simplified sticks or a more detailed balls & sticks representation, in combination with the CPK model for atom colors.

**Rendering strategy focus area:** Surrounding residues are also rendered using either a simplified sticks or a detailed balls & sticks representation. Similar to the ligand, the CPK model is used for coloring atoms.

**Rendering strategy context:** The protein in context is rendered using a simplified ribbon representation with a uniform coloring chosen by the user.

As mentioned at the beginning, the next step after selecting focus and context elements is a one-time preprocessing step. The times, which are required for this particular use case, depending on the number of imported MD-simulation keyframes, are shown in

Figure 6.1. For example, if all 2 162 keyframes are imported and preprocessed, it takes about 69 seconds. Additionally, it shows that the required times rise exponentially with increasing numbers of keyframes, since keyframes are not simply processed individually, but also in relation to previous and future keyframes.

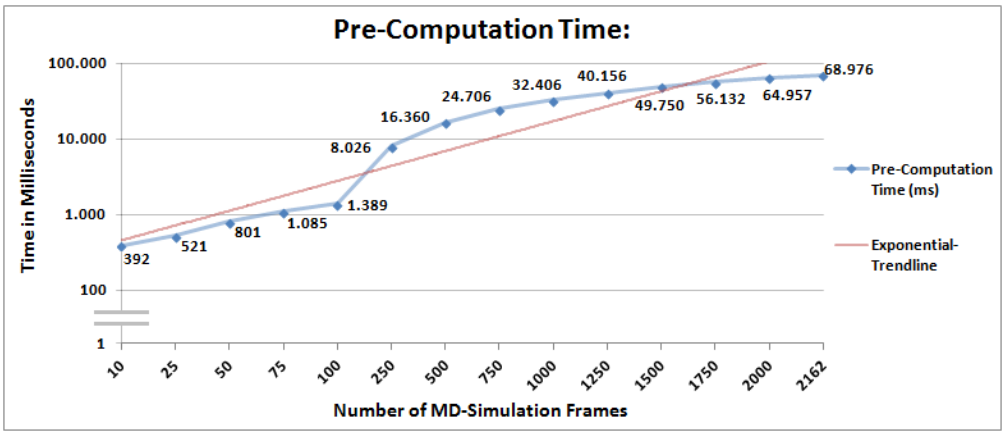


Figure 6.1: Visualization of the required duration of the preprocessing on a logarithmic time scale. The duration was measured using different numbers of keyframes and additionally, a trendline shows the exponentially increasing time requirements.

Once the preprocessing is completed, the user can interact with the visualization in real-time. Figure 6.2 shows a visualization of the FPS achieved on average. As the number of keyframes increases, the average number of FPS settles at around 100 FPS. However, it is interesting that especially a very short selection of the MD-simulation consisting of 10 keyframes provides a lower number of average FPS, namely 83 FPS. This is due to the fact that especially in this case, the color interpolation never stops, since an ease-in is always immediately followed by an ease-out. Constantly interpolating colors of atoms requires more computing time and therefore lowers the number of frames-per-second.

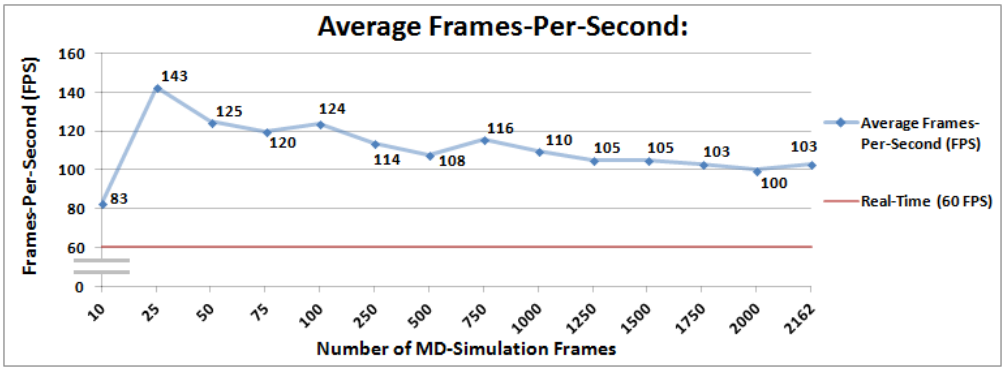


Figure 6.2: Overview of average numbers of FPS while interacting with the visualization of the MD-simulation. Additionally, a line marker showing the minimum frame rate of 60 FPS, as lower threshold for real-time applications, is displayed.

Finally, two different visualization results of the MD-simulations are shown. The first example is shown in Figure 6.3 and it visualizes a detailed rendering of the ligand using a balls & sticks representation and a more simplified sticks rendering strategy for surrounding residues. All residues and the ligand are colored using the CPK color model. The protein is rendered using a simplified cartoon representation and a uniform coloring. Furthermore, the importance functions are selected according to the defined use case properties: a smoothed combination of the speed and stuckness of the ligand.

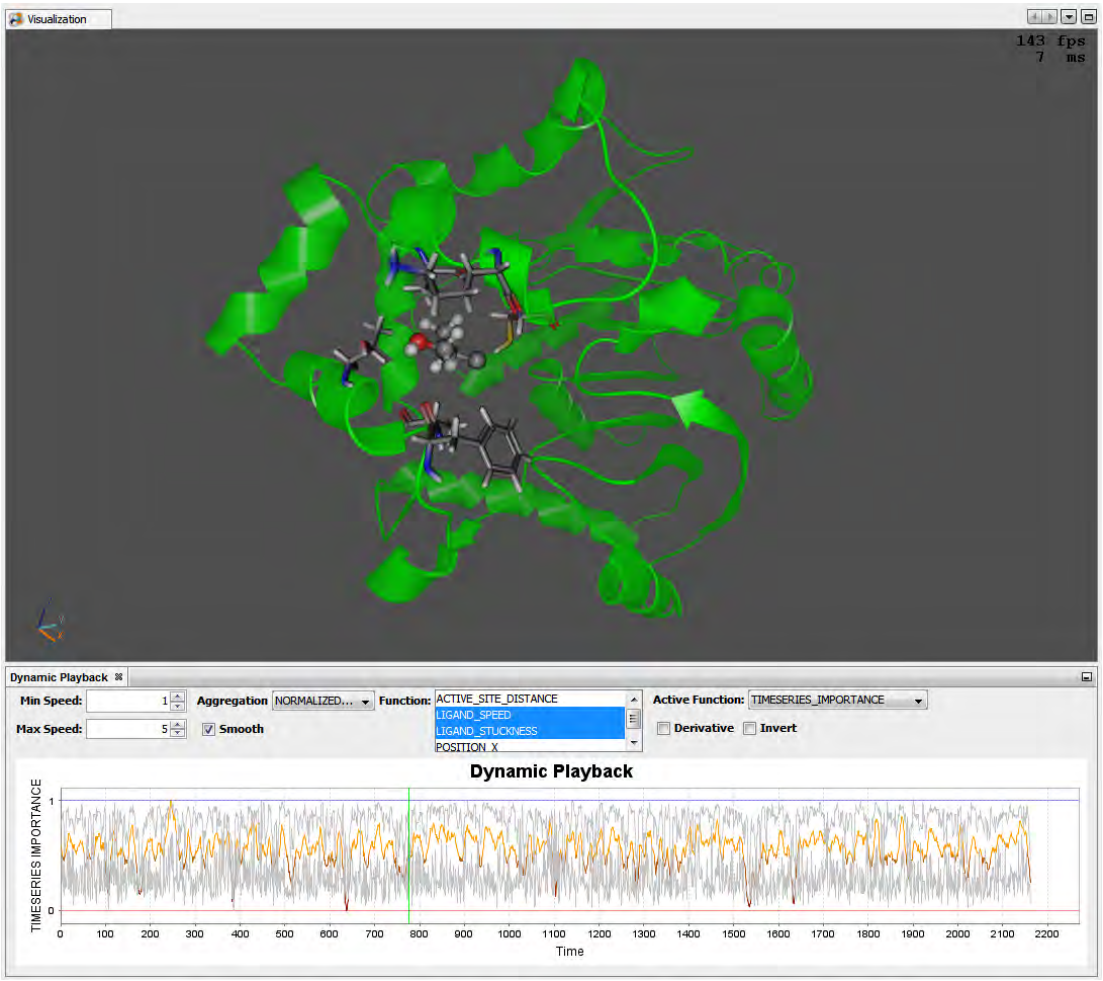


Figure 6.3: Visualization of the first use case using a detailed balls & sticks rendering strategy for the ligand, a simplified sticks rendering strategy for surrounding residues, and a simplified cartoon representation for the context protein. The ligand and residues are colored using the CPK model and the context protein is colored using a uniform coloring. The used importance function is a combination of speed and stuckness of the ligand and the combined function is additionally smoothed. Graphical user interface and rendering result of CAVER Analyst [Ana].

The second example is shown in Figure 6.4. Instead of the ligand, it emphasizes surrounding residues, which are rendered using a space-filling representation in combination with the CPK color model. This time, the ligand and the surrounding context protein are simplified. The ligand is rendered using a sticks representation and the protein is rendered using a cartoon representation. Both are colored with different uniform colorings. Similarly to the first example, the speed and stuckness of the ligand define the importance function, but this time it is not smoothed and therefore the exact calculation of the temporal importance function is used as playback speed.

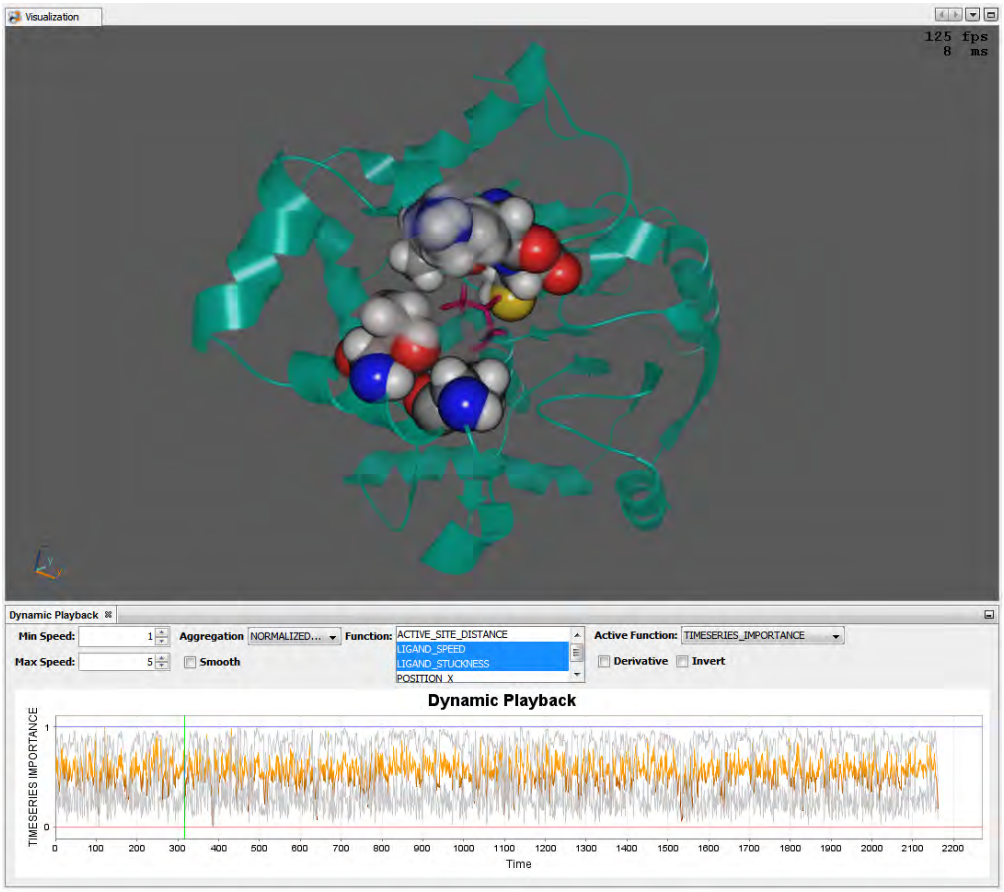


Figure 6.4: Another visualization of the first use case using a simplified sticks representation and a uniform coloring for the ligand. Surrounding residues are rendered using a detailed space-filling representation in combination with the CPK color model. The protein is again simplified using a cartoon representation but a different uniform color was selected. The importance function is, similar to the last example, a combination of ligand speed and stuckness, but now the resulting function is no longer smoothed. Graphical user interface and rendering result of CAVER Analyst [Ana].

## 6.2 Influence of Waters on the Protein

The second use case was inspired by a recent publication by Vad et al. [VBJ+17], which focuses on water flows through proteins. Therefore, the protein is subdivided into three different parts: The first part is called *active site* or *focus region*, which represents the region inside a protein where a chemical reaction, like a binding, takes place. The second part is called *inner region*, which represents the entire inside of a protein. The last part is called *outer region*, which represents the region outside a protein. An example is shown in Figure 6.5.

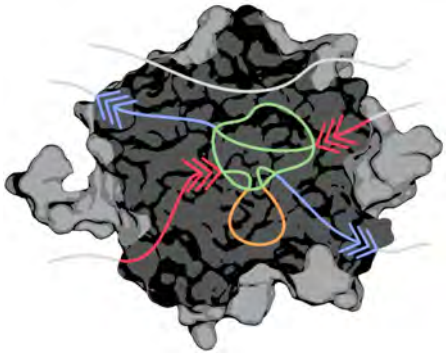

Figure 6.5: Visualization of different regions of interest within a protein. The color of a path indicates the type of a water molecule: (*Gray*) *Outside* waters or waters that never reach the active site. (*Red*) *To active* waters entering the protein. (*Green*) *Active* waters within the active site. (*Orange*) *Inside* waters within the protein but outside the active site, and (*Blue*) *From active* waters leaving the protein. Image from Vad et al. [VBJ+17].

For this use case, we have reused the *watergate color model*, introduced by Vad et al. [VBJ+17]. Additionally, we render active, inside, to active, and from active waters using a detailed balls & sticks rendering strategy as they represent the focus of the analysis, and outside waters using a simplified sticks rendering strategy since they only provide additional context information. Figure 6.6 shows an overview of the used rendering strategies and colors.

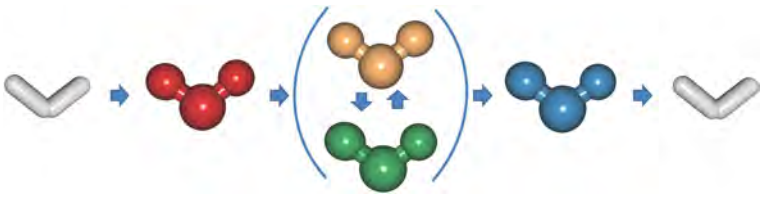

Figure 6.6: Visualization of different states of water molecules. Gray outside waters are simplified using a sticks representation and waters that are already inside the protein are rendered using a detailed balls & sticks representation. The coloring is similar to the color model introduced by Vad et al. [VBJ+17]. Renderings produced using CAVER Analyst [Ana].

This use case examines how individual water molecules interact with a protein. It is based on Protein 1cqz, from *The Protein Data Bank* [BWF+] and consists of 8 799 residues, 310 amino acid residues, 8 488 water residues, 1 ligand, and 30 457 atoms. The total length of the observed MD-simulation is 1 000 keyframes (∼2.47 MB per keyframe and a total size of ∼2.41 GB).

The second use case can be defined by the following properties:

**Research question:** Which, when, where, and how many waters reach the active site within a protein? How long do they stay there? What is their exact movement? When and where do they leave the protein again?

**Spatial focus:** According to the definition of Vad et al. [VBJ+17], the spatial focus is on the following waters: *active*, *inside*, *to active*, and *from active* waters including the spatial position of waters if they change from one state to another.

**Temporal focus:** In this use case, multiple temporal importance functions or combinations are used: First, an importance function that is dependent on the visible number of *from active* and *to active* waters. Second, an importance function that indicates the amount of water that changes between two states. Third, the distance of waters to the active site and the RMSD of the protein.

**Spatial focus area:** According to the definition of Vad et al. [VBJ+17], the spatial focus area is defined by *outside* waters, or none.

**Context:** The whole protein.

**Rendering strategy focus:** Waters that are inside the protein are rendered using a detailed balls & sticks representation. If the focus is on waters that change states, a space-filling representation is used instead. Water molecules in focus are colored using the watergate color model introduced by Vad et al. [VBJ+17].

**Rendering strategy focus area:** Waters that are outside of the protein are rendered using a simplified sticks representation. In this case either the CPK model, the watergate color model, or a uniform coloring depending on user preferences is used. Optionally, the user can hide context waters so that they are not rendered.

**Rendering strategy context:** The protein is rendered using a simplified ribbon representation with a uniform coloring chosen by the user.

Again, the next step after defining focus and context is the one-time preprocessing. The duration required for this step is shown in Figure 6.7. Similar to the pre-computation time required for the first use case, the duration of the second use case also rises exponentially the more keyframes are considered.

Figure 6.8 shows the FPS achieved on average including a line marker for the minimum frame rate of 60 FPS. Thereby, it is clearly visible that our visualization can be accessed in real-time and even a large amount of keyframes can be explored easily.

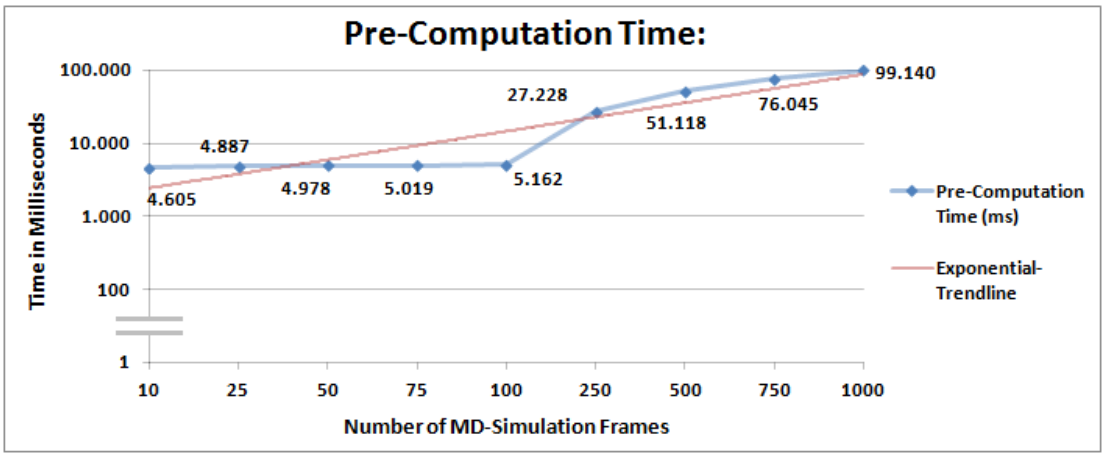


Figure 6.7: Visualization of the amount of time spent on the pre-computation of the MD-simulation using different numbers of keyframes. A logarithmic time scale is used for the measured time in milliseconds and an exponential trendline shows the increasing time requirements.

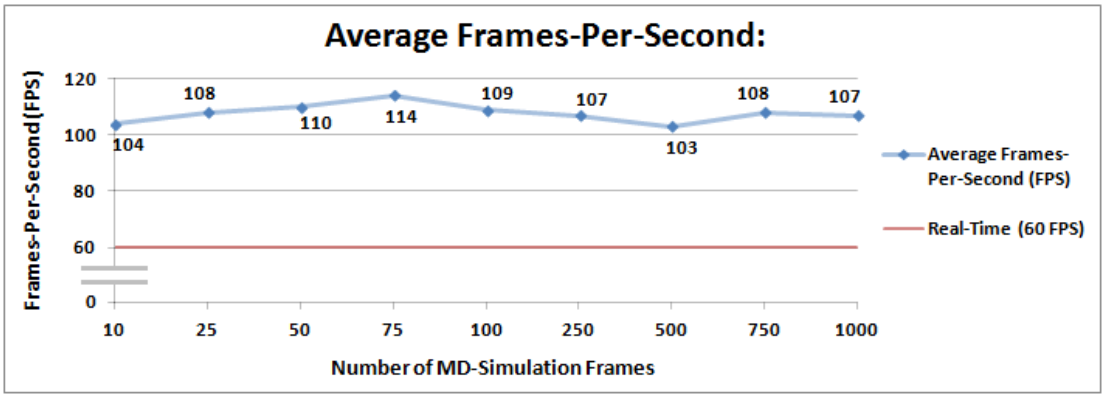


Figure 6.8: Visualization of the average number of FPS including a line marker showing the minimum frame rate of 60 FPS, which we aim for as lower threshold for real-time applications. The number of average FPS settles constantly at around 100 FPS.

Finally, to complete this chapter, we will present two examples of possible results of the second use case. The first example is shown in Figure 6.9. *Active*, *inside*, *to active*, and *from active* waters are rendered using a detailed balls & sticks representation and *outside* waters are rendered using a simplified sticks representation. All waters are colored using the watergate color model and the protein is rendered using a simplified green cartoon representation. The importance function is defined according to the above mentioned properties of this use case: a combination of the number of *to active* and *from active* waters. Additionally, the resulting temporal importance function is smoothed.

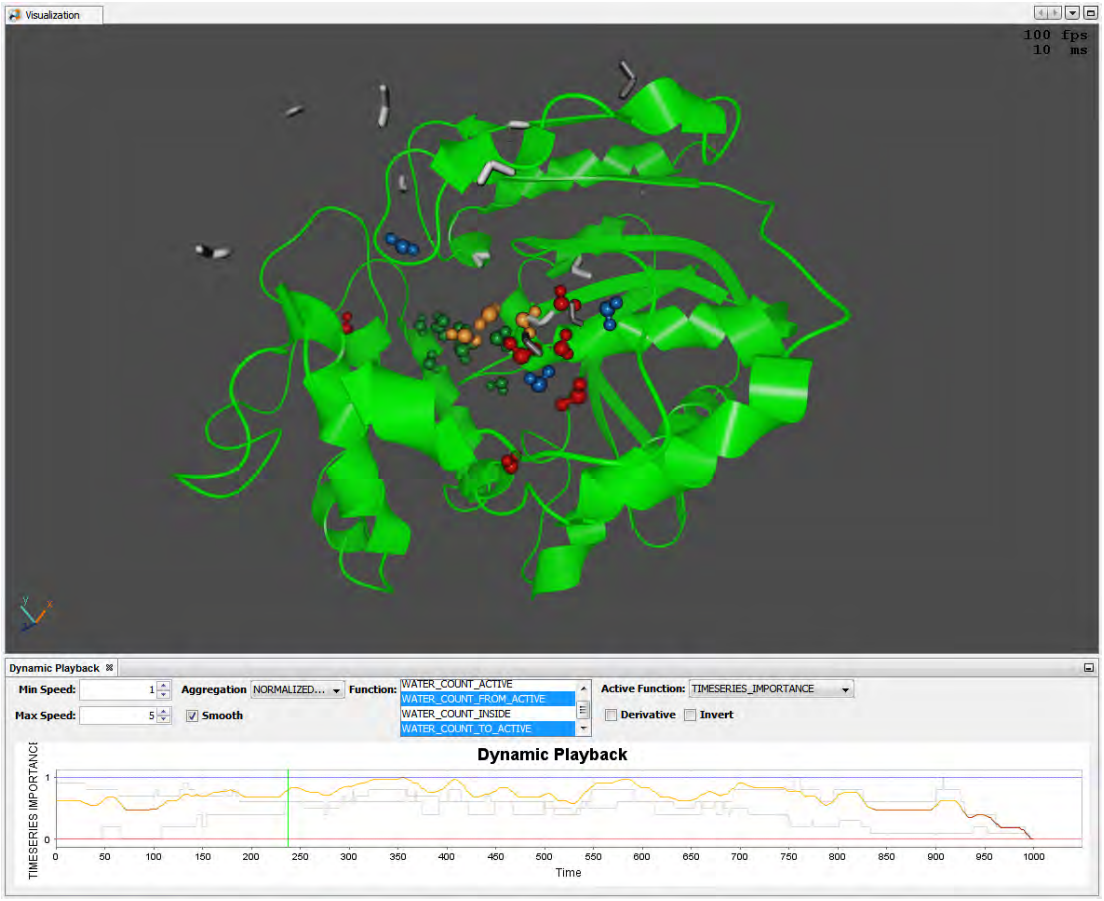


Figure 6.9: Visualization of the second use case. Waters already inside the protein are rendered using a detailed balls & sticks representation and outside waters are rendered using a simplified sticks rendering strategy. The colors of all waters are chosen according to the watergate color model. The context protein is rendered using a simplified cartoon representation with a uniform coloring. Additionally, the used importance function is a smoothed combination of the number of *to active* and *from active* waters. Graphical user interface and rendering result of CAVER Analyst [Ana].

The second example is shown in Figure 6.10. Instead of rendering all waters that are currently inside the protein, it highlights when waters change between two states using a space-filling representation. The color is again dependent on the watergate color model. Furthermore, outside waters are rendered using a simplified sticks representation but instead of a uniform color, the CPK color model is used. The surrounding protein is simplified using a cartoon representation and a uniform color is applied. In contrast to the previous example, the importance function is now only dependent on the number of changing water molecules and additionally, the importance function is no longer smoothed.

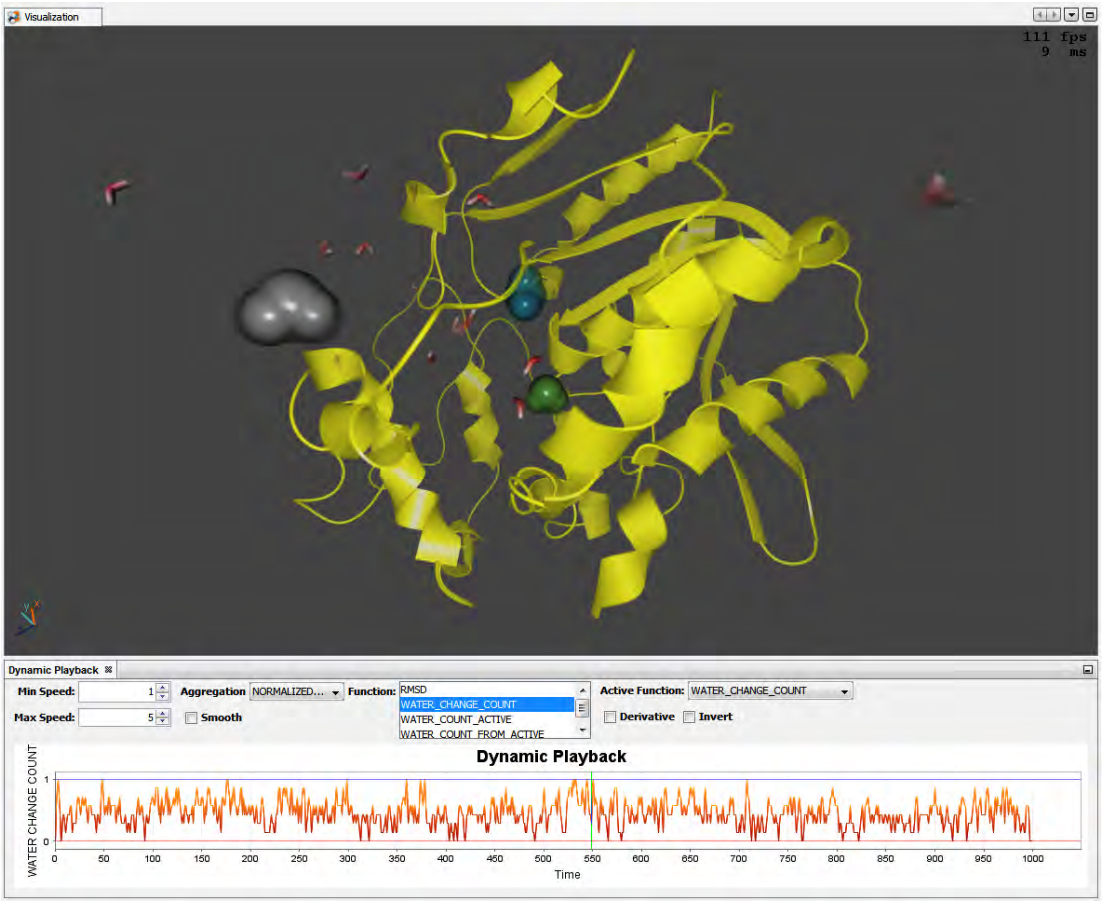


Figure 6.10: Different visualization of the second use case. Outside waters are rendered using a simplified stick representation, but instead of a uniform color, they are colored using the CPK model. Waters that change from one state to another are rendered using a more detailed space-filling strategy and their color is chosen according to the watergate color model. The protein is again rendered using a simplified cartoon representation, but a different uniform color was chosen by the user. This time, the importance function depends on the number of waters that change state per keyframe and the resulting function is no longer smoothed. Graphical user interface and rendering result of CAVER Analyst [Ana].

In summary, the two use cases have shown that MD-simulations of dense molecular scenes can now be explored in 3D using our novel spatial and temporal focus & context visualization. Additionally, we have shown that our visualization provides an average frame rate of 100 FPS and is therefore well suited as real-time application.

Without our novel focus & context visualization, it would not be possible to explore these large, detailed, and complex MD-simulation datasets efficiently in 3D and in real-time. If biochemists try to analyze the interactions of a single ligand surrounded by hundreds of residues, or a protein surrounded by thousands of water molecules, scenes

are too dense and the analysis of long MD-simulations is too time-consuming to draw meaningful conclusions from it. Therefore, relevant spatial and temporal events can easily be overlooked.

Additionally, we already received particularly positive feedback on the ghosting procedure, which renders context elements semi-transparently and simultaneously highlights edges of context structures. Biochemists have noted that this functionality is not only highly useful while exploring MD-simulations but should be an integral part of the CAVER Analyst [Ana] framework for various analysis.

CHAPTER 7

# Conclusions and Future Work

The following final chapter summarizes our contribution to the novel visualization approach for the interactive exploration of MD-simulations in real-time. Additionally, possible future problems and improvements through further developments are discussed.

## 7.1 Conclusion

This work was developed in close collaboration with international universities, such as Masaryk University and the University of Bergen, together with biochemists and domain experts from the Loschmidt Laboratories in Brno, Czech Republic. From the very beginning, part of this cooperation were two-weekly on-site meetings in Vienna, which focused on theoretical concepts and approaches. Subsequently, the meetings became more practical and focused on exact implementation details. In addition to these on-site meetings, online correspondences, such as conference calls or e-mails, were conducted. In the last months of the development, these online correspondences were performed on a weekly basis. Altogether there were 20 on-site meetings in Vienna and additionally 10 online conferences with experts from Bergen and Brno. Furthermore, we organized two live presentations in Brno, where the current state of research was presented to eight biochemists from the Loschmidt Laboratories. During these presentations, biochemists had the possibility to give feedback on the current development status of our visualization. Furthermore, there was a frequent online exchange of computed MD-simulation datasets and e-mail correspondences for the development of practical use cases.

The aim of this work was not to invent novel importance functions or to verify how relevant individual molecular elements are during explorations of dense, complex, and animated MD-simulation. We did not want to evaluate which already implemented temporal and spatial importance functions are best suited for the exploration of MD-simulations. Our aim was to use already existing importance functions and apply them to a novel spatio-temporal focus & context visualization that lowers the density of highly

complex molecular scenes, prevents occlusions of focus elements, supports different levels of detail, allows the user to analyze the playback in a more compact manner, reduces visual clutter using motion blur, and therefore helps the user analyzing an animated MD-simulation, without missing important spatial and temporal events. Therefore, the spatial importance combines different well established levels of geometrical detail, such as dots, sticks, balls & sticks, space-filling, surface, and cartoon representations. Additionally, molecules can be colored using various coloring strategies like a default CPK color model, a watergate color model, a custom color selection, and a categorical color coding. In addition, it was necessary to developed a visibility management system, which guarantees that context elements never occlude focus elements by applying an additional ghosting procedure. We furthermore introduced visibility smoothing to prevent unnecessary visual distractions of the user and presented a color interpolation, using ease-in and ease-out functions, which is independent of the playback speed and the number of displayed keyframe. Finally, we implemented motion blur that on the one hand, reduces visual clutter from animated MD-simulations and on the other hand, encodes the playback speed of individual molecular elements in a natural way since motion blur is an essential reference, already known from our human visual system.

As shown by two different use cases, this real-time focus &context visualization approach is applicable on completely different research tasks. Before that, MD-simulations were mainly examined using statistical tests in combination with static graphs. After an initial preprocessing step, the dataset can be interactively explored in real-time and even important properties such as render strategies, color models, etc. can be changed during runtime. Furthermore, we showed that in both cases an average of 100 FPS is achieved, which is way above the necessary 60 FPS threshold for real-time applications.

## 7.2 Future Work

In the following section, we present possible improvements and extensions for future implementations. Through our close collaboration with biochemists, we have continuously received suggestions and new ideas for possible improvements. This ongoing feedback highlights their interest in our real-time visualization, which gives them the possibility to analyze MD-simulation datasets in real-time and 3D, which was not possible for them before. In addition, it shows that biochemists are very interested in including this CAVER Analyst [Ana] extension in their daily research and use it to publish novel insights.

In the following, we will discuss the importance of additional use cases and will explain how supercomputers can be used to further improve our visualization. Our main focus is on creating additional innovative use cases. This is especially interesting as domain experts are then entering unknown terrain since it has not been possible to explore these research areas interactively, using a spatio-temporal importance function. Therefore, it would be particularly interesting to perform and design additional use cases. A possible new case would be a combination of the two use cases we have already presented since the influence of residues on the ligand is already analyzed but the influence of water

molecules on the ligand is currently ignored. Therefore, it would combine the ligand including residues and waters in combination with the protein. With that, it would be possible to generate new hypotheses about how waters affect the interactions, movements, and bindings of the ligand.

Another possibility is reducing the time that is currently required for preprocessing MD-simulation keyframes. This could be done by supporting additional scripts that allow researchers to outsource preprocessing computations to supercomputers. Since they have access to these computers, all relevant individual or combinations of importance functions and most frequently used distance thresholds, between focus and context elements, could be calculated in advance and then accessed on demand. If these calculations can be stored locally or with insignificant additional delays due to network transfers, this would be an intense saving of time as the duration of pre-computations increases exponentially and there would be no more waiting times to interact with the visualization. This is especially relevant, as we assume that the number of keyframes will increase in the future from thousands to millions, or billions of simulation frames. Currently, the pre-computation leads to waiting times between a fraction of a second to a few minutes. This would increase exponentially as the number of additional keyframes rises. Therefore, it would be advantageous to have a possibility to export and import already computed calculations. If this computation was executed once, it can be exported and used on any other computer.

In the following, we will focus on feedback regarding the graphical user interface, interactions with the visualization, and additional functionality required by researchers. An example of a potential improvement includes the possibility to add various additional importance functions to the framework. Our CAVER Analyst [Ana] extension could provide a multitude of functions that can be selected or combined by the user. Possibilities are functions for distances, angles, edge bonds, and linear interaction energies. Another possibility would be to allow the user to import arbitrary external functions. This would enable other programs to generate various 2D functions that can then be imported and used as temporal importance functions. Domain experts additionally want to change these functions during runtime, for example, by using the mouse to increase or decrease the importance of certain sections or modify the importance of individual keyframes.

Another suggestion for improvement was to allow the user to change the distance threshold from context objects to the element in focus. For example, if the ligand is in focus, the distance to surrounding context residues should be adjustable depending on user preferences or GUI settings. Currently, this distance is hard-coded but in future versions it should be changeable at runtime without any additional pre-computing steps. In addition, it is currently not possible to focus on multiple ligands although scientists would like to be able to analyze several ligands simultaneously. The dataset from the first use case on the investigation of residues and their influence on the ligand, for example, already contains 23 ligands which might be interesting as well. It will be particularly important to investigate how many different focus objects can be analyzed by the user, at the same time, without missing important events.

During the last presentation, biochemists noted that motion blur is on the one hand, a very natural way to encode speed and acceleration, but on the other hand, it would be advantageous to know the exact speed-up in numbers. Future versions of this CAVER Analyst [Ana] extension will therefore provide an additional GUI element showing the exact acceleration in numbers. Thereby, researchers will know exactly how much the movement of molecules has been accelerated, for example, that the ligand moves twice as faster as originally simulated within the MD-simulation.

In addition, it is currently required to adjust the strength of motion blur using a GUI slider. This should make it possible to support computers with less powerful graphics hardware. As mentioned above, a low number of FPS could cause distracting visual artifacts. If the computer is unable to render at least 60 FPS, the strength of motion blur could be adjusted automatically depending on the average number of FPS. An automatic approach could replace the currently used manual solution.

Another problem that could arise in the future during explorations of even longer MD-simulations, consisting of millions and billions of keyframes, is an over-plotted 2D representation of the importance function. If the function consists of much more keyframes than there are horizontal pixels in the GUI, this function can no longer be displayed. A solution for this could be an additional focus & context approach like non-linear video editing tools already have it now.

# Bibliography


[AMHH08] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. *Real-Time Rendering*. A. K. Peters, Ltd., Natick, MA, USA, 3rd edition, 2008.

[Ana] Development Team Caver Analyst. CAVER Analyst - softwaretool for protein analysis and visualization. `http://www.caver.cz/`. [Accessed 01-August-2018].

[AW57] Bernie J. Alder and Thomas E. Wainwright. Phase transition for a hard sphere system. *The Journal of Chemical Physics*, 27:1208–1209, 1957.

[AW59] Bernie J. Alder and Thomas E. Wainwright. Studies in molecular dynamics. i. general method. *The Journal of Chemical Physics*, 31:459–466, 1959.

[BAAR12] Jiamin Bai, Aseem Agarwala, Maneesh Agrawala, and Ravi Ramamoorthi. Selectively de-animating video. *ACM Transactions on Graphics - Proceedings of ACM SIGGRAPH*, 31:66:1–66:10, 2012.

[BALP09] Alexandra Baer, Friederike Adler, Daniel Lenz, and Bernhard Preim. Perception-based Evaluation of Emphasis Techniques Used in 3D Medical Visualization. *Vision, Modeling, and Visualization Workshop (VMV)*, pages 295–304, 2009.

[BGKG06] Stefan Bruckner, Sören Grimm, Armin Kanitsar, and Eduard Gröller. Illustrative context-preserving exploration of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12:1559–1569, 2006.

[Bli77] James F. Blinn. Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph.*, 11:192–198, 1977.

[BVMG08] Jean-Paul Balabanian, Ivan Viola, Torsten Möller, and Eduard Gröller. Temporal styles for time-varying volume data. *Proceedings of 3D Data Processing, Visualization, and Transmission*, pages 81–89, 2008.

[BWF+] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The Protein Data Bank. `http://www.rcsb.org`. [Accessed 16-August-2018].

[CBFR08] Nuno Cerqueira, Natercia Bras, Pedro A. Fernandes, and Maria J. Ramos. MADAMM: A multistaged docking with an automated molecular modeling protocol. *Proteins Structure Function and Bioinformatics*, 74:192–206, 2008.

[CDF+06] Forrester Cole, Doug DeCarlo, Adam Finkelstein, Kenrick Kin, Keith Morley, and Anthony Santella. Directing gaze in 3D models with stylized focus. *Eurographics Symposium on Rendering*, pages 377–387, 2006.

[CDSRC08] Alexandre Carvalho, A. Augusto De Sousa, Cristina Ribeiro, and Emília Costa. A temporal focus + context visualization model for handling valid-time spatial information. *Information Visualization*, 7:265–274, 2008.

[Chi] UCSF Chimera. Resource for Biocomputing Visualization, and Informatics. An Extensible Molecular Modeling System - Find Clashes/Contacts. `https://www.cgl.ucsf.edu/chimera/docs/ContributedSoftware/findclash/findclash.html`. [Accessed 17-September-2018].

[CLCC09] Kai-Yin Cheng, Sheng-Jie Luo, Bing-Yu Chen, and Hao-Hua Chu. SmartPlayer: User-centric video fast-forwarding. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 789–798, 2009.

[CWM99] Frank Albert Cotton, Geoffrey Wilkinson, and Carlos A. Murillo. Advanced inorganic chemistry. *Wiley-Interscience*, Sixth edition, 1999.

[DO07] Ajay Divakaran and Isao Otsuka. A video-browsing-enhanced personal video recorder. *International Conference of Image Analysis and Processing - Workshops*, pages 137–142, 2007.

[DRB+08] Pierre Dragicevic, Gonzalo Ramos, Jacobo Bibliowitcz, Derek Nowrouzezahrai, Ravin Balakrishnan, and Karan Singh. Video browsing by direct manipulation. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 237–246, 2008.

[FS01] Daan Frenkel and Berend Smit. Understanding molecular simulation. *Academic Press, Inc.*, 2nd edition, 2001.

[Hau04] Helwig Hauser. Generalizing focus+context visualization. *Technische Universität Wien, Österreich, Fakultät für Informatik*, Habilitationsschrift:1–153, 2004.

[HHWH11] Benjamin Höferlin, Markus Höferlin, Daniel Weiskopf, and Gunther Heidemann. Information-based adaptive fast-forward for visual surveillance. *Multimedia Tools Appl.*, 55:127–150, 2011.

[HJ05] Wolfgang Hurst and Philipp Jarvers. Interactive, dynamic video browsing with the zoomslider interface. *2005 IEEE International Conference on Multimedia and Expo(ICME)*, pages 558–561, 2005.

[HKH+12] Markus Höferlin, Kuno Kurzhals, Benjamin Höferlin, Gunther Heidemann, and Daniel Weiskopf. Evaluation of fast-forward video visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18:2095–2103, 2012.

[Hof65] August Hofmann. On the combining power of atoms. *Proceedings of the Royal Institution*, 4:401–430, 1865.

[HOvG02] Tomas Hansson, Chris Oostenbrink, and Wilfred F. van Gunsteren. Molecular dynamics simulations. *Current Opinion in Structural Biology*, 12:190–196, 2002.

[KC14] Brittany Kondo and Christopher Collins. DimpVis: Exploring time-varying information visualizations by direct manipulation. *IEEE Transactions on Visualization and Computer Graphics*, 20:2003–2012, 2014.

[Kek65] August Kekulé. Studies on aromatic compounds. *Liebigs Annalen der Chemie*, 137:129–196, 1865.

[KKF+17] Barbora Kozlíková, Michael Krone, Martin Falk, Norbert Lindow, Marc Baaden, Daniel Baum, Ivan Viola, Julius Parulek, and Hans-Christian Hege. Visualization of biomolecular structures: State of the art revisited. *Computer Graphics Forum*, 36:178–204, 2017.

[KMH01] Robert Kosara, Silvia Miksch, and Helwig Hauser. Semantic depth of field. *Proceedings of the IEEE Symposium on Information Visualization*, pages 97–104, 2001.

[Kol65] Walter L. Koltun. Precision space-filling atomic models. *Biopolymers*, 3:665–679, 1965.

[Lea09] Andrew R. Leach. Molecular modelling : principles and applications. *Pearson Prentice Hall*, Second edition, 2009.

[LMH+07] Xueliang Liu, Tao Mei, Xian-Sheng Hua, Bo Yang, and He-Qin Zhou. Video collage. *Proceedings of the 15th ACM International Conference on Multimedia*, pages 461–462, 2007.

[LR71] Bon-Su Lee and Frederic M. Richards. The interpretation of protein structures: Estimation of static accessibility. *Journal of Molecular Biology*, 55:379–IN4, 1971.

[LTF+05] Ce Liu, Antonio Torralba, William T. Freeman, Frédo Durand, and Edward H. Adelson. Motion magnification. *ACM Transactions on Graphics - Proceedings of ACM SIGGRAPH*, 24:519–526, 2005.

[Mat16] The Mathworks, Inc., Natick, Massachusetts, United States. *MATLAB version 9.1.0.441655 (R2016b)*, 2016.

[Mil94] Gavin Miller. Efficient algorithms for local and global accessibility shading. *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, pages 319–326, 1994.

[MM04] Elke Moritz and Joerg Meyer. Interactive 3d protein structure visualization using virtual reality. *Proceedings. Fourth IEEE Symposium on Bioinformatics and Bioengineering*, pages 503–507, 2004.

[MTC06] Claudio Montani, Marco Tarini, and Paolo Cignoni. Ambient occlusion and edge cueing for enhancing real time molecular visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12:1237–1244, 2006.

[NSG11] Fernando Navarro, Francisco J. Serón, and Diego Gutierrez. Motion blur rendering: State of the art. *Computer Graphics Forum*, 30:3–26, 2011.

[OGF+10] Sean O'Donoghue, David S. Goodsell, Achilleas S. Frangakis, Fabrice Jossinet, Roman Laskowski, Michael Nilges, Helen R. Saibil, Andrea Schafferhans, Rebecca C. Wade, Eric Westhof, and Arthur J. Olson. Visualization of macromolecular structures. *Nature methods*, 7:42–55, 2010.

[Ols18] Arthur J. Olson. (in press). Perspectives on Structural Molecular Biology Visualization: From Past to Present. *Journal of Molecular Biology*, doi: `https://doi.org/10.1016/j.jmb.2018.07.009`, 2018. [Accessed 06-October-2018].

[Pen] Robert Penner. Easing Equations. `http://gizma.com/easing/`. [Accessed 01-August-2018].

[Per05a] James A. Perkins. A History of Molecular Representation Part 2: The 1960s - Present. *The Journal of Biocommunication*, 31(2): `http://jbiocommunication.org/issues/31--2/feature2.html`, 2005. [Accessed 06-October-2018].

[Per05b] James A. Perkins. A History of Molecular Representation Part One: 1800 to the 1960s. *The Journal of Biocommunication*, 31(1): `http://jbiocommunication.org/issues/31--1/features3.html`, 2005. [Accessed 06-October-2018].

[Pho75] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18:311–317, 1975.

[PIB+11] Robert Patro, Cheuk Y. Ip, Sujal Bista, Dave Thirumalai, Samuel S. Cho, and Amitabh Varshney. MDMap: A system for data-driven layout and exploration of molecular dynamics simulations. *IEEE Symposium on Biological Data Visualization*, pages 111–118, 2011.

[PIV10] Robert Patro, Cheuk Y. Ip, and Amitabh Varshney. Saliency guided summarization of molecular dynamics simulations. *Scientific Visualization: Advanced Concepts*, 1:321–335, 2010.

[Rah64] Aneesur Rahman. Correlations in the motion of atoms in liquid argon. *Phys. Rev.*, 136:A405–A411, 1964.

[RAPP06] Alex Rav-Acha, Yael Pritch, and Shmuel Peleg. Making a long video short: Dynamic video synopsis. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:435–441, 2006.

[Ren] Gaëtan Renaudeau. Easing Functions. `https://gist.github.com/gre/1650294`. [Accessed 01-August-2018].

[Ric81] Jane S. Richardson. The anatomy and taxonomy of protein structure. *Academic Press*, 34:167–339, 1981.

[RLN07] Ruth Rosenholtz, Yuanzhen Li, and Lisa Nakano. Measuring visual clutter. *Journal of Vision*, 7:17, 2007.

[SBC07] Kenneth C. Scott-Brown and Patrick D. J. Cronin. An instinct for detection: psychological perspectives on CCTV surveillance. *The Police Journal*, 80:287–305, 2007.

[SL98] Daniel J. Simons and Daniel T. Levin. Failure to detect changes to people during a real-world interaction. *Psychonomic Bulletin & Review*, 5:644–649, 1998.

[VBJ+17] Viktor Vad, Jan Byška, Adam Jurčík, Ivan Viola, Eduard Gröller, Helwig Hauser, Sergio M. Margues, Jiri Damborský, and Barbora Kozlíková. Watergate: Visual exploration of water trajectories in protein dynamics. *Eurographics Workshop on Visual Computing for Biology and Medicine*, pages 33–42, 2017.

[vdZLBI11] Matthew van der Zwan, Wouter Lueks, Henk Bekker, and Tobias Isenberg. Illustrative molecular visualization with continuous abstraction. *Computer Graphics Forum*, 30:683–690, 2011.

[VFSG06] Ivan Viola, Miquel Feixas, Mateu Sbert, and Eduard Gröller. Importance-driven focus of attention. *IEEE Transactions on Visualization and Computer Graphics*, 12:933–940, 2006.

[VKG04] Ivan Viola, Armin Kanitsar, and Eduard Gröller. Importance-driven volume rendering. *Proceedings of the Conference on Visualization*, pages 139–146, 2004.

[Waa73] Johannes Diderik Waals. Over de continuiteit van den gas- en vloeistoftoestand. *PhD thesis, Univ. Leiden*, 1873.

[WAH+09] Marc Wolter, Ingo Assenmacher, Bernd Hentschel, Marc Schirski, and Torsten Kuhlen. A Time Model for Time-Varying Visualization. *Computer Graphics Forum*, 28:1561–1571, 2009.

[War04] Colin Ware. Information visualization: Perception for design. *Morgan Kaufmann Publishers Inc.*, 2nd edition, 2004.

[WHA07] Wesley Willett, Jeffrey Heer, and Maneesh Agrawala. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13:1129–1136, 2007.

[WHTPK09] Marc Wolter, Bernd Hentschel, Irene Tedjo-Palczynski, and Torsten Kuhlen. A direct manipulation interface for time navigation in scientific visualizations. *2009 IEEE Symposium on 3D User Interfaces*, pages 11–18, 2009.

[WMY+03] Barbara M. Wildemuth, Gary Marchionini, Meng Yang, Gary Geisler, Todd Wilkens, Anthony Hughes, and Richard Gruss. How fast is too fast?: Evaluating fast forward surrogates for digital video. *Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, 10:221–230, 2003.

[ZIK98] Sergej Zhukov, Andrej Iones, and Grigorij Kronin. An ambient light illumination model. *Rendering Techniques '98*, pages 45–55, 1998.