Integration Strategies in the Visualization of Multifaceted Spatial Data

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der Technischen Wissenschaften

eingereicht von

Dipl.Ing. Johannes Sorger
Matrikelnummer 0225843

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Assoc.Prof. Dipl.Ing. Dr.techn. Ivan Viola
und Ao.Univ.Prof. Dipl.Ing. Dr.techn. Eduard Gröller
In Mitwirkung von: Dr.techn. Harald Piringer

Diese Dissertation haben begutachtet:

Torsten Möller

Barbora Kozlikova

Wien, 26. September 2017

Johannes Sorger

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Integration Strategies in the Visualization of Multifaceted Spatial Data

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Dipl.Ing. Johannes Sorger
Registration Number 0225843

to the Faculty of Informatics

at the TU Wien

Advisors: Assoc.Prof. Dipl.Ing. Dr.techn. Ivan Viola
and Ao.Univ.Prof. Dipl.Ing. Dr.techn. Eduard Gröller
In collaboration with: Dr.techn. Harald Piringer

The dissertation has been reviewed by:

Torsten Möller

Barbora Kozlikova

Vienna, 26th September, 2017

Johannes Sorger

Erklärung zur Verfassung der Arbeit

Dipl.Ing. Johannes Sorger
Wasagasse 31/22, 1090 Wien, Austria

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, 26. September 2017

Johannes Sorger

Acknowledgements

The work presented in this thesis has been realized in part at the VRVis research center [SOP+15, SOL+16], and in part in the VisGroup at the Institute of Computer Graphics and Algorithms at TU Wien [SMR+17, SMK+16]. At the end of this particular road that is marked by the submission of my PhD thesis, I feel a lot of gratitude towards a wide range of people who enabled my academic career and whom I had the pleasure to work with in both environments. Firstly, I thank Meister Eduard Gröller, who was the one constant factor, watching over my progress during my time both at the VRVis and in the VisGroup. He was always ready to diligently and patiently provide me with detailed feedback on the various intermediate states of the various paper drafts, as well as the drafts of this thesis. His support was essential in shaping the rough ideas and formulations of the individual works into the polished end-product that finally made it into this thesis. The research environment that the Meister established is unique in the world, and it was an honor to become a part of the famiglia. Thank you Meister! Next, I thank Harald Piringer, who supervised me during my time at the VRVis, for providing me with the opportunity to work in this great environment and on this interesting research topic. Harald taught me a lot about the scientific workflow through his very structured way of thinking and his attention to detail. Very special thanks go to Ivan Viola, who took me in, after my initial funding at the VRVis ended, and who provided me with the chance to finish my thesis by working on exciting topics in a very engaging environment. I learned a lot from Ivan during the many discussions that we had. I was always impressed with his ability to see unexpected angles and connections between the various topics. Ivan's support was essential in concluding this thesis, so thank you a lot Ivan! I also thank Katja Bühler, who supervised my master thesis and thus introduced me to the world of visualization research and the exciting environment that is academia in general and the VRVis in particular. Thank you Katja! I thank Georg Stonawski and Werner Purgathofer as the heads of the respective institutions for always being very supportive. Also a thank you to Agnes Knor, Anita Mayerhofer-Sebera, Max Höfferer, and all technicians, the respective souls and hearts of the VRVis and the VisGroup, who keep the operations running and who always readily provided me with all the support that I needed.

I thank all of my co-authors, project partners and collaborators. Without their support, none of this work would have been possible. Very special thanks go to Thomas Ortner and Peter Mindek, who were my brothers in arms during my time at the VRVis and the VisGroup respectively. Thank you for having been closely at my side during the roller coaster ride that is the sometimes nerve-wrecking life of a PhD student. I could not have made it without you guys.

A heartfelt thank you goes to my many colleagues who always supported me and with whom I had the pleasure of working at the VRVis and in the VisGroup. The friendly and productive atmosphere that you created made me look forward to go to work every morning. I was lucky to call you not only my colleagues but also my friends. Thank you Mitschi, Harri, Edith, Wimmi, Höltzi, Henni, Lui, Andi, Martin, Mike, and Artem and all the other Vollzeitequivalente at the VRVis. And thank you Manu, Rrrenata, Yun, Johanna, Gabriel, Alexey (the Govna), Highchao, smoking David, wild Viktor, Alexand0r, and Bruno at the VisGroup. Special thanks go to Clemens Arbesser, for carrying the torch of the Tealuminati, and always providing me with tea during our time at the VRVis, and to Thomas Auzinger for supporting me with adaptations of the dissertation template.

I also thank the reviewers of this thesis, Barbora Kozlikova and Torsten Möller, for taking the time to evaluate this work and for insightful discussions and feedback during its finalization.

Thanks go also to the funding agencies that provided financial support for the duration of my PhD studies. The works in this thesis were supported by the FWF-funded project P24597-N23 (VISAR) and the COMET K1 program of the Austrian Funding Agency (FFG) during my time at the VRVis, and by the Vienna Science and Technology Fund (WWTF) through project VRG11-010 and by the EC Marie Curie Career Integration Grant through project PCIG13-GA-2013-618680 during my time at the TU Wien.

Finally, I thank my family, my friends, and Müge for believing in me and supporting my efforts despite being mystified about what it actually was that was keeping me so busy and away from them all the time. I am lucky to be a part of you. Thank you.

Kurzfassung

Visualisierungsdesigner können auf eine Vielzahl an visuellen Kanälen zurückgreifen, die es ihnen ermöglichen, Datenattribute visuell darzustellen. Beispiele für solche Kanäle sind *Position, Größe, Orientierung, Farbe, Textur, Helligkeit und Bewegung.* Welche Datenattribute welchen visuellen Kanälen zugewiesen werden, kann dabei vom Designer *entschieden* werden. Theoretisch kann jedes Datenattribut durch jeden Kanal oder eine Kombination mehrerer Kanäle dargestellt werden. In der Praxis hängt die optimale Darstellungsform jedoch vom jeweiligen Datentyp und der Aufgabenstellung des Nutzers ab. Bei der Visualisierung von räumlichen Daten ist die Zuordnung von räumlichen Attributen zu visuellen Kanälen schon durch die Daten *vorbestimmt.* Der Designer hat in diesem Fall weniger Gestaltungsfreiheit als bei Daten ohne solch eine vorbestimmte Zuordnung. Räumliche Daten können zusätzlich zu diesen inhärenten räumlichen Attributen auch eine Vielzahl an weiteren Attributen aufweisen, die eine freie Zuordnung erlauben. Solch räumliche Daten werden auch als *facettenreiche räumliche Daten* bezeichnet. Der räumliche Bezug der Attribute in diesen Daten ist oft wichtig für die erfolgreiche Ausführung einer Aufgabenstellung. Die Kombination aus frei wählbaren und vorbestimmten Zuordnungen schränkt den Designer bei der Wahl der optimalen Darstellungsform stark ein, da die essentielle Information in sinnvoller Art und Weise mit dem inhärenten räumlichen Kontext *integriert* werden muss. Es bedarf daher spezieller Herangehensweisen, um Aufgabenstellungen, die auf der Analyse oder Präsentation von facettenreichen räumlichen Daten beruhen, lösen zu können.

Im Rahmen dieser Dissertation wird die *Integration von Darstellungsformen* im Kontext von facettenreichen räumlichen Daten erforscht. Der erste Teil dieser Arbeit beschreibt eine Integrationstaxonomie, die aus zwei Teilaspekten besteht: visuelle Integration und funktionale Integration. Visuelle Integration beschreibt, wie die diversen räumlichen und nicht räumlichen Darstellungsformen von facettenreichen räumlichen Daten visuell in Bezug zueinander gebracht werden können. Funktionale Integration wiederum beschreibt, wie Ereignisse und Interaktionen zwischen den jeweiligen visuell integrierten Darstellungsformen koordiniert werden können.

Im zweiten Teil dieser Arbeit werden Beiträge zur Visualisierungsforschung vorgestellt, die auf konkreten Anwendungen von Integration in Analyse- und Präsentationsszenarien beruhen. Das erste Set an Herausforderungen betrifft die Analyse von facettenreichen räumlichen Daten im Rahmen des in der Lichtplanung stattfindenden Entscheidungs-

findungsprozesses. Die Aufgabenstellung des Nutzers ist dabei, eine optimale Leuchtenkonfiguration für eine gegebene Innenraumszene zu finden. Der Nutzer muss sich dabei zwischen Duzenden bis Hunderten potentiellen Lösungen entscheiden. Die in dem Zusammenhang identifizierten Herausforderungen werden mit LiteVis gelöst, einem System, das die Visualisierung von Eingabeparametern mit der Visualisierung von allen relevanten Ausgabeparametern einer globalen Beleuchtungssimulation vereint. Die Integration dieser heterogenen Aspekte, gemeinsam mit einer neuartigen Visualisierung für Multi-Parameter Rankings, sind dabei die entscheidenden Faktoren, die einen effizienten Vergleich und die Analyse von Leuchtenkonfigurationen ermöglichen.

In Präsentationsszenarien kann sich der Nutzer nicht auf Interaktion verlassen, um Schlüsse aus einer Visualisierung ziehen zu können. Eine der Hauptherausforderungen in diesem Zusammenhang ist daher, visuell ansprechende, sinnhafte Darstellungsformen zu generieren, die für passive Benutzerrollen geeignet sind. In dieser Dissertation werden diesbezüglich zwei Herausforderungen im Bereich Visualisierung molekularer Daten adressiert.

In der ersten Aufgabenstellung, ist das Ziel, zwei unterschiedliche Darstellungsformen eines molekularen Modells in Bezug zueinander zu bringen. Der Bezug zwischen den Modellen wird dabei auf zeitlicher Ebene mittels Animation bzw. animierten Übergängen dargestellt. Eine Darstellungsform geht dabei kontinuierlich in die andere über. Eine neuartige Technik ermöglicht es, solche zeitlichen Übergänge in einer Art und Weise zu spezifizieren, so dass sie auf verschiede molekulare Datensätze angewandt werden können. Dadurch wird den Autoren von Animationen beträchtlicher Aufwand erspart, der mit dem händischen Erstellen solcher Animationen verbunden ist.

Die zweite Aufgabenstellung im Bereich Visualisierung von molekularen Daten betrifft die Präsentation von Übergängen zwischen zwei Entwicklungsstadien eines Mikroorganismus. Die Herausforderung dabei resultiert aus dem Fehlen von Informationen, die diesen Übergang auf molekularer Ebene beschreiben. Eine neuartige Technik hilft dabei, zu vermeiden, dass beim Darstellen eines Übergangs aufgrund der fehlenden Details falsche Informationen vermittelt werden. Die Technik basiert auf der kontinuierlichen visuellen Abstraktion der jeweiligen Entwicklungsstadien. Die Idee dahinter ist, beide Modelle auf eine Stufe zu abstrahieren, auf der der Bezug zwischen ihnen fehlerfrei vermittelt werden kann. In dem Zusammenhang wird erforscht, wie die verschiedenen Abstraktionsgrade und die Modelle der Entwicklungsstadien miteinander integriert werden können. Die Resultate der in dieser Dissertation präsentierten Arbeiten zeigen deutlich, wie vielseitig einsetzbar die Integration von Visualisierungen ist, um Herausforderungen in der Analyse und Präsentation räumlicher Daten zu bewältigen.

Abstract

Visualization designers have several visual channels at their disposal for encoding data into visual representations, e.g., position, size, shape, orientation, color, texture, brightness, as well as motion. The mapping of attributes to visual channels can be *chosen* by the designer. In theory, any data attribute can be represented by any of these visual channels or by a combination of multiple of these channels. In practice, the optimal mapping and the most suitable type of visualization strongly depend on the data as well as on the user's task. In the visualization of spatial data, the mapping of spatial attributes to visual channels is inherently *given* by the data. Multifaceted spatial data possesses a wide range of additional (non-spatial) attributes without a given mapping. The data's given spatial context is often important for successfully fulfilling a task. The design space in spatial data visualization can therefore be heavily constrained when trying to choose an optimal mapping for other attributes within the spatial context. To solve an exploration or presentation task in the domain of multifaceted spatial data, special strategies have to be employed in order to *integrate* the essential information from the various data facets in an appropriate representation form with the spatial context.

This thesis explores visualization integration strategies for multifaceted spatial data. The first part of this thesis describes the design space of integration in terms of two aspects: visual and functional integration. *Visual integration* describes how representations of the different data facets can be visually composed within a spatial context. *Functional integration*, describes how events that have been triggered, for instance, through user interaction, can be coordinated across the various visually integrated representations.

The second part of this thesis describes contributions to the field of visualization in the context of concrete integration applications for exploration and presentation scenarios. The first scenario addresses a set of challenges in the exploratory analysis of multifaceted spatial data in the scope of a decision making scenario in lighting design. The user's task is to find an optimal lighting solution among dozens or even hundreds of potential candidates. In the scope of a design study, the challenges in lighting design are addressed with LiteVis, a system that integrates representations of the simulation parameter space with representations of all relevant aspects of the simulation output. The integration of these heterogeneous aspects together with a novel ranking visualization are thereby the key to enabling an efficient exploration and comparison of lighting parametrizations.

In presentation scenarios, the generation of insights often cannot rely on user interaction and therefore needs a different approach. The challenge is to generate visually appealing, yet information-rich representations for mainly passive observation. In this context, this thesis addresses two different challenges in the domain of molecular visualization. The first challenge concerns the conveying of relations between two different representations of a molecular data set, such as a virus. The relation is established via animated transitions, i.e., a temporal form of integration between two representations. The proposed solution features a novel technique for creating such transitions that are re-usable for different data sets, and can be combined in a modular fashion.

Another challenge in presentation scenarios of multifaceted spatial data concerns the presentation of the transition between development states of molecular models, where the actual biochemical process of the transition is not exactly known or it is too complex to represent. A novel technique applies a continuous abstraction of both model representations to a level of detail at which the relationship between them can be accurately conveyed, in order to overcome a potential indication of false relationship information. Integration thereby brings the different abstraction levels and the different model states into relation with each other. The results of this thesis clearly demonstrate that integration is a versatile tool in overcoming key challenges in the visualization of multifaceted spatial data.

Contents

Kurzfassung **ix**

Abstract **xi**

Contents **xiii**

1 Introduction **1**
1.1 Definitions . . . 3
1.2 Thesis Overview . . . 4
1.3 Contributions . . . 7

I Integration Strategies **9**

2 Visual Integration **11**
2.1 Visual Channels in Spatial Data Visualization . . . 11
2.2 Adapting the Spatial Representation . . . 14
2.3 Composing Multiple Representations . . . 18
2.4 Visual Integration Challenges . . . 26

3 Functional Integration **29**
3.1 Introduction . . . 30
3.2 Related Work . . . 31
3.3 Model-Based Taxonomy . . . 32
3.4 Integration Techniques . . . 36
3.5 Discussion . . . 42
3.6 Conclusion and Outlook . . . 45

II Applications **47**

4 Integration in Parameter Space Exploration **49**
4.1 Introduction . . . 50
4.2 Related Work . . . 52

4.3 Lighting Design Background 54
4.4 Design Study of LiteVis 60
4.5 Implementation 68
4.6 Use Case Scenario 69
4.7 Evaluation 72
4.8 Discussion and Future Work 74
4.9 Conclusion 76

5 Storytelling Templates For Temporal Integration **79**
5.1 Introduction 80
5.2 Related Work 82
5.3 Metamorphers 84
5.4 Implementation 90
5.5 Results 92
5.6 Discussion 96
5.7 Conclusion 98

6 Masking Missing Information Via Visual Abstraction **101**
6.1 Introduction 102
6.2 Related Work 104
6.3 Forward & Inverse Abstraction 106
6.4 Implementation of Visual Abstraction Levels 111
6.5 Results 116
6.6 Discussion and Expert Feedback 118
6.7 Conclusion and Outlook 119

7 Conclusion **121**
7.1 Summary 121
7.2 Outlook 123

Bibliography **127**

Curriculum Vitae **141**

CHAPTER 1

Introduction

Visual communication for the purpose of the presentation and exploration of data, concepts, relationships, and processes can be described as one of the main goals of visualization [RTM$^+$03]. In order to give data attributes a visual representation capable of achieving this goal, visualization designers encode the values of data attributes in so called *visual channels.* Examples of these channels are *position, size, shape, orientation, color, texture, brightness*, as well as *motion* [WGK10]. As one channel should only encode the values of a single attribute, the amount of information that a visualization can encode – i.e., the *"visual budget"* that a designer can spend – is limited by the number of visual channels. The mapping of attributes to visual channels can be *chosen* by the visualization designer. For instance, should the value of a stock market share be represented by the position or the size of its visual representation? In theory, any data attribute can be represented by any of these visual channels or by a combination of multiple of these channels. In practice, the optimal mapping and the most suitable representation form strongly depend on the data as well as on the user's task. For instance, ordered data can be intuitively represented by channels that perceptually convey magnitude information, while categorical data is better represented by channels that convey identity information [Mun14]. In the visualization of spatial data, the data's spatial context is often important for successfully fulfilling an analysis task. *Spatial data* is a category of data where the *spatialization* is already *given* [TM04]. The spatialization refers to the spatial reference of attributes, such as the position, orientation, shape, and size. This means, each item in a data set has a given (inherent) value for its position – and potentially also its orientation, shape, and size – in relation to the physical three- (or two-) dimensional space, i.e., where it has been measured or simulated. Examples for such data are computed tomography (CT) or magnetic resonance imaging (MRI) scans that yield a volumetric image of a patient. The position of each voxel in the volume is given, and typically essential for a successful analysis of the medical image data. In these cases, the visual channels that are already encoding essential information from the data, i.e., through a given mapping, are not available to be used in the visualization design.

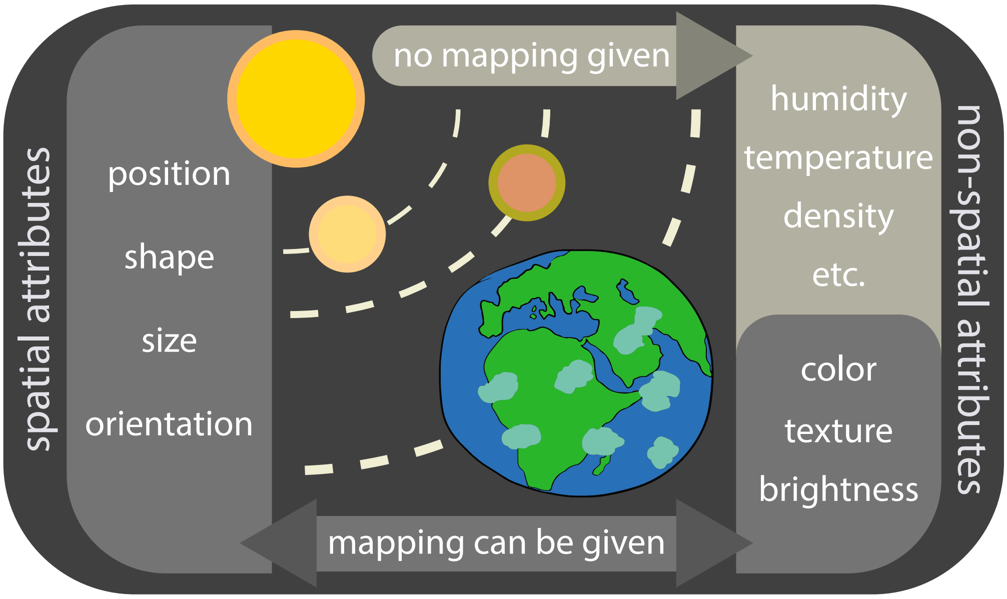


Figure 1.1: Inherent mapping in spatial data: spatial attributes have an inherent mapping to visual channels. Certain non-spatial attributes, i.e., color, texture, and brightness, can have an inherent mapping as well. For the remaining non-spatial attributes the mapping has to be chosen in order to visually convey the attribute values.

Depending on the data type and the acquisition method, spatial data can possess given mappings for the other visual channels, i.e., color, texture, and brightness, as well. For the representation of the earth in Figure 1.1, for instance, position, shape, size, and orientation within the solar system are given. Similarly, we can extract values for the color, texture, and brightness at each point where we sample the surface of the earth. All of these attributes have an inherent (given) mapping to visual channels. Besides attributes with a given mapping, spatial data can also possess attributes where the mapping is not inherently given. For attributes, such as humidity, temperature, or density, a mapping has to be chosen in order to visually convey their values. Conveying the spatial context for these non-inherently mapped attributes is often important as well.

Depending on the number of attributes with a given mapping, the design space in spatial data visualization, i.e., the mapping of data attributes to visual channels, can be heavily constrained. If the mapping for most visual channels is already given, only few channels remain to encode additional relevant information – the visual budget is almost completely spent. In such cases, strategies for managing the visual budget in the spatial representation have to be employed to enable a depiction of non-spatial attributes in the spatial context. In other cases, a task requires the analysis of certain attributes of a spatial data set within the context of a completely chosen, i.e., non-spatial representation. In both cases, additional information or entire representations have to be integrated within the spatial data visualization.

Motivation & Goal

The *motivation* behind the work in this thesis is the exploration of the design space of spatial data visualization in terms of how chosen mappings of attributes to visual channels can be integrated with the constrained (given) mapping of spatial data representations.

The *goal* is to enable the design of visualizations that incorporate, or *integrate*, the representation of and interaction with all essential attributes and representation forms that are required to fulfill a task in spatial data visualization.

The content of this thesis is structured into two parts as illustrated in Figure 1.2: In Part I, we[1] investigate the design space of integration in terms of two aspects. In Chapter 2, *Visual Integration*, we discuss the strategies for managing the visual budget in a visualization of spatial data, i.e., by adapting or extending the spatial representation, in order to convey the entire essential information in a way that allows a user to fulfill a certain task. Such an adaption or extension corresponds to the *visual integration* of different kinds of mappings, i.e., visual representations, into a coherent visualization. In Chapter 3, *Functional Integration*, we investigate the ways in which the events in multiple data representations can be coordinated in order to form a single coherent visualization system. In Part II of this thesis, we present several novel integration techniques in the context of concrete application scenarios for exploration and presentation tasks. The following sections will clarify the terminology that is used in this thesis, before giving a more detailed overview of the individual chapters and contributions.

1.1 Definitions

A *visual mark* represents a single *data item*, e.g., a row in a tabular data set. Marks are classified according to the dimensions that they occupy: points (0D), lines (1D), areas (2D), and surfaces or volumes (3D). A mark can encode data attributes by variations of its position, shape, size, orientation, color, texture, brightness, and motion. These properties are referred to as *visual channels*. Visual channels can be separated into two groups: *spatial & retinal* [WGK10]. Position, shape, size, and orientation are spatial channels. Color, texture, and brightness are retinal channels. Motion affects all categories as it describes variations of each channel over time. The process of encoding an attribute by a visual channel refers to *mapping* the attribute to that channel.

The term *spatial data* describes data sets that have an association to physical (two- or three-dimensional) space. This means, the attribute vector that describes a data item contains spatial references to this physical space, i.e., where the respective data item was

---

[1] While this thesis is the result of the effort of a single author, the works, on which Chapters 3-6 are based, have been achieved within collaborative efforts. In all cases, the author of this thesis is also the main author of these works. In the remainder of this thesis, the plural first person form, i.e., "we", is used to reflect this collaborative effort (please refer to the individual chapter prefaces for the detailed author lists).

measured or simulated. These spatial references can take the form of attributes, such as position, orientation, size, and shape of the measured entity. These attributes are referred to as spatial attributes in the context of this thesis. Spatial attributes have an intrinsic mapping to the spatial visual channels.

Non-spatial data attributes do not possess any given mappings to spatial channels. Data that consists solely of non-spatial attributes is often also referred to as *abstract* data in the literature. However, in this thesis the term "non-spatial" is used exclusively, in order to avoid confusion with the terms "visual abstraction" and "abstracted representation".

A spatial data set typically contains non-spatial attributes as well. Data that contains spatial and non-spatial facets is referred to as *multifaceted spatial data* [KH13]. Some non-spatial attributes, can have an intrinsic mapping to the retinal visual channels of a representation. For all other non-spatial attributes, the mapping to visual channels has to be chosen by the visualization designer (or by the users, if the visualization system allows them to do so). In this thesis, the differentiation between attributes of spatial data with inherent (given) mapping and attributes without inherent (chosen) mapping to visual channels is important.

A *visual representation* or *representation* displays parts or all of the data within a common frame of reference, e.g., within the same view, where the visual marks typically share the same mapping of data attributes to visual channels. The *representation form* describes, which spatial frame of reference and which kind of mapping was chosen to display the data. A spatial representation form is defined as one that adheres to the given mapping of spatial attributes to spatial channels. A representation form is non-spatial if the visualization designer ignores any given spatial mapping. Examples for non-spatial representation forms are bar charts, scatterplots, and parallel coordinate systems. Spatial representation forms are, for instance, three dimensional volume representations, 3D geometric scenes, and 2D geographic maps. A *visualization system* can be composed of multiple representation forms. The *visual budget* describes how many visual channels are available for encoding data attributes within a single representation or across an entire visualization system.

1.2 Thesis Overview

The chapters in this thesis are structured into two parts as indicated in Figure 1.2. Part I of this thesis defines the design space of integration in terms of visual (Chapter 2) and functional integration (Chapter 3).

Chapter 2 describes the principal strategies for managing and increasing the visual budget within a spatial data visualization in order to incorporate all essential information and representation forms that enable a user to successfully fulfill a task. Visualization designers can: 1) adapt the spatial representation in order to embed additional attributes, or in order to make it more suitable for a specific task, and 2) extend the spatial representation with additional representations, either a) spatially, or b) temporally, or they can apply a

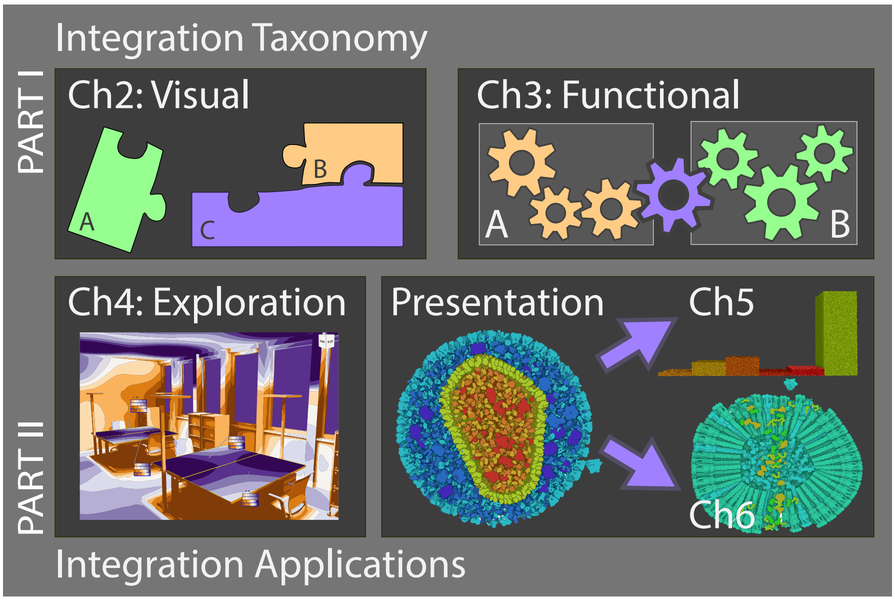


Figure 1.2: Overview of the thesis structure: Part I describes a taxonomy for visual and functional integration. Part II presents novel techniques in the context of integration applications for exploration and presentation scenarios.

combination of 1) and 2). The *adaption of the spatial representation* can be applied if the goal is to convey non-spatial attributes in a spatial context. The idea behind this strategy is to replace the given mapping of non-essential attributes by choosing to map more essential ones to the respective visual channels instead. The representation of the earth in Figure 1.1, for instance, could be adapted to encode the temperature in the surface color. The feasibility of this strategy reaches its limits, either when the number of attributes to encode is bigger than the number of available visual channels, or when the analysis of attributes cannot be efficiently handled within the inherent spatial context. Further, inherent obstacles of 3D spatial representation forms, such as occlusion and perspective distortion of objects, can impede users from accurately interpreting or even locating the information that they seek in a 3D spatial environment [ET08]. Regardless of the number of available visual channels, it is therefore often more efficient to introduce additional non-spatial views to a spatial representation. Each additional representation thereby comes with its own set of visual channels, and thus increases the visual budget of the visualization. Added representations can be distributed *spatially* or *temporally*. A common example for spatially integrated representations are *juxtaposed views* where views featuring separate representations are placed side-by-side. The temporal integration of visualizations refers to animated transitions, where one representation is gradually transformed into another one. Each form of visual integration comes with its own set of advantages and restrictions that determine when their respective application is feasible, as will be discussed in Chapter 2.

Chapter 3 describes the design space of functional integration. When dealing with multiple visually integrated spatial and non-spatial data representations, the question arises how to coordinate events between them so that they act in concert, i.e., as a unit of interconnected parts, within a coherent visualization system. Such events might be triggered interactively by the user or they might be triggered by the dynamic nature of the data. We refer to such coordinations as *functional integration* between representations. A common example for a functional integration is brushing & linking where the selection of data items in one representation triggers the visual indication (e.g., with a highlighting color) of all associated items in the other representations. We differentiate the types of functional integration based on whether an event has been triggered on the *Data*, the *Visual* (mapping), or the *Navigation* component of a representation – as well as on the component type that is affected in another (target) representation. We thereby regard the components and their integration in terms of which kinds of tasks they support, and we illustrate the various categories of functional integration with examples from literature.

Part II of this thesis showcases how the application of integration can support specific visualization tasks. Which types of visual and functional integration are thereby the most suitable ones, strongly depends on the data and the task. On a high level, visualization tasks can be classified into three categories: exploration, presentation (and fun) [BM13]. Chapters 4 through 6 describe novel integration techniques in the context of application scenarios for exploration and presentation tasks.

Chapter 4 addresses challenges in exploratory analysis of multifaceted spatial data in the scope of a decision making scenario in lighting design. There are multiple challenges associated with exploratory analysis in integrated visualization systems. One is to choose representation forms that complement each other [WBWK00]. Another is to functionally integrate the interactions with the individual representations in a way that facilitates the user's task. The user's task is to find an optimal lighting solution among dozens or even hundreds of potential candidates. The optimality of a solution is determined by a multitude of qualitative and quantitative factors. The assessment of qualitative factors, such as luminary aesthetics and illumination quality, strongly depends on their depiction in their original 3D spatial context, i.e., a 3D model of the scene where the illumination is simulated, measured, and displayed. The wide range of quantitative factors in each solution requires additional non-spatial representation forms that neglect the given spatial mapping in order to favor a more structured overview for the comparison of multiple attributes across many solutions. We demonstrate how to join the representations of different data facets that result from these task requirements in an integrated visualization system. We thereby propose a novel type of ranking visualization that considers the spatial and non-spatial data facets for finding an optimal lighting solution.

Chapter 5 addresses the presentation of hidden insights into molecular models of microorganisms, for instance, in regard to the structural hierarchy of a virus, or the quantities of proteins. Such insights are given by representation forms, such as exploded views or histograms, that might distort or completely abstract the original given mapping. The more such a representation of the virus differs from the original one, the harder it is for the user to grasp the relation between both representation forms, especially if the relation cannot be explored interactively. In presentation scenarios, the generation of insights cannot rely on user interaction and therefore needs a different approach. The challenge is to generate visually appealing, yet information-rich, representations for mainly passive observation. Visual integration via animated transitions is a way to present this relation in an engaging and intuitive manner. To allow visualization designers the application of such transitions to a wide range of data sets, we propose a technique for creating all aspects of such transformations in a re-usable way.

Chapter 6 addresses the challenge of presenting the transition between development states of molecular models where the actual biochemical process of the transition is not exactly known or it is too complex to model. These models can be displayed at very high detail, i.e., at atomic resolution. However, much less is known about how the individual elements of a model, i.e., proteins and cellular structures, relate to the elements of another model. There is a discrepancy between the level of detail at which the models and the relationship between them are described. To handle this issue, we present a technique for adapting the spatial representation of both models by continuously reducing the detail to a level at which the relationship between them can be accurately described. We thereby propose four levels of visual abstraction for molecular data that correspond to different levels of knowledge about the relationship between two models. Another challenge that we address in this context is the loss of information when reducing the visual detail of a model.

Chapter 7 concludes this thesis with a reflection on the visual and functional aspects of the integration design space in respect to the presented applications and in respect to the field of visualization in general.

1.3 Contributions

The high level contributions of this thesis are:

- The presentation of a taxonomy for visual and functional integration.
- The presentation of novel visualization techniques and applications for exploration and presentation scenarios in the context of integration.
- An application of and reflection on the integration taxonomy for the analysis and comparison of existing work and the presented original work.

The contributions of the individual chapters are:

In Chapter 2:

- The description of the design space of visual integration in the domain of multi-faceted spatial data, on the level of visual channels.
- A demonstration of decoding visual integration in existing visualization systems.

In Chapter 3:

- The description and exploration of the design space of functional integration based on a taxonomy of visualization component combinations.
- A demonstration of the range of potential functional integration applications based on the combination of visualization components, and illustrated by concrete examples from literature.
- An application of the taxonomy to classify and compare integrated visualization systems.

In Chapter 4:

- A demonstration of the importance of integration for exploration tasks in the scope of a design study for decision making in lighting design.
- A problem characterization of the application domain of lighting design. We thereby relate the domain problem to other areas of simulation-based decision making.
- A novel visualization that allows users to weigh spatial and non-spatial aspects of the simulation output when ranking multiple decision options.

In Chapter 5:

- A technique for creating temporal integrations, i.e., animated transitions, between two representation forms of molecular data.
- A description of the six different aspects of such a transition in the form of a pipeline for animated transitions.
- A set of rules for creating transitions that can be modularly combined and re-applied to different data sets.

In Chapter 6:

- A technique for creating temporal integrations, i.e., animated transitions, between molecular models where the actual biochemical process of the transition is not exactly known or too complex to represent.
- A definition of four levels of visual abstraction for molecular data that correspond to different levels of knowledge about the biochemical processes.
- A description of how to transition between these levels and between different data sets at each level.

In Chapter 7:

- A reflection on the design space of integration.
- A description of interesting areas for potential future research.

Part I

Integration Strategies

CHAPTER 2

Visual Integration

In this chapter, we will equip the reader with the necessary background to understand the visual integration design space. We thereby refer to terminology from existing work. The two principal strategies for visually integrating non-inherently mapped attributes with a spatial context are the adaption of the spatial representation (as discussed in Section 2.2), and the composition of multiple representations (as discussed in Section 2.3). The contribution of this chapter is thereby the discussion of the visual integration design space on the level of individual operations on visual channels (in Section 2.2) and the relations between individual visual channels (in Section 2.3). The challenges in visual integration in the context of multifaceted spatial data are presented based on prior work [OSP$^+$16] in Section 2.4. Before we elaborate these strategies, first we have to understand the available design space, i.e., the visual channels in spatial data representations, that we will discuss in the following section.

2.1 Visual Channels in Spatial Data Visualization

The first notion of a graphical vocabulary consisting of marks, positional (spatial), and retinal channels was defined by Bertin in his Semiology of Graphics [Ber83]. Bertin was the first to define a strict separation of content from container, i.e., the separation of data attributes from the graphical properties that encode these attributes. With this vocabulary, Bertin was addressing visual encodings of information on a *"flat sheet of paper"*, i.e., no 3D spatial data was yet considered. Other classification schemes followed that extended Bertin's work in respect to the capabilities of computationally generated graphic primitives on computer displays and similar viewing devices. Bartram [Bar97] added *motion* as a visual channel and thereby included *animation* into the design space of visual representations.

In this thesis we rely on the definition of visual channels that was proposed by Ward, Grinstein, and Keim [WGK10] as a summarization of the definitions found in literature, i.e., *position, shape (mark), size, orientation, color, texture, brightness,* and *motion.* Each channel is capable of encoding the values of a data attribute through variations in the respective channel's visual output, such as variations in brightness or variations of positions along an axis. As illustrated in Figure 2.1, we split the channels into a set of spatial and retinal channels. Motion thereby refers to a change of one of the other seven channels over time. An inherent mapping of motion corresponds to the mapping of time dependent data to visual channels, where the variations of the channel output over time signify the change of a data value over time. The mapping to the motion channel can also be chosen. In this case, motion encodes non time-dependent attributes. For example, the frequency at which the brightness of a visual mark changes, could encode the degree of importance of the corresponding data item in order to attract the attention of the user.

In Section 1.1, we touched upon the separation of visual channels into spatial and retinal channels. Channels can also be separated into identity channels and magnitude channels. Identity channels, such as shape, serve visual discrimination, while magnitude channels, such as size, are better suited for conveying quantities. However, in spatial data the visual channels do not exclusively serve visual discrimination or the suggestion of magnitudes. Instead, the channels can carry intrinsic semantics of the data that are not easily quantifiable. In tumor diagnosis, for instance, the position, shape, and size of a tumor in a medical image give the radiologist information about the condition of the patient. Additionally, the texture or intensity values (color, brightness) can suggest the degree of agressiveness.

Which types of channels are available to encode information depends in general on the kind of visual mark used. A bar chart, for instance, uses the x-axis of the position channel to differentiate bars that correspond to different categories. It would not make sense to encode an additional attribute within the y-axis of the bar position, as the relative length of the bars would be much harder to read. In spatial data, the given mappings and their intrinsic semantic create an additional constraint on the mapping. Changing the given mappings of spatial and, in certain cases, also retinal channels can change the intended meaning of a visual mark, for instance, when changing the size, shape, or location of the two-dimensional area mark that represents a country on a map.

Another characteristic of spatial data is that it can consist of hierarchically organized levels where the meaning of higher levels depend on the faithful depiction of objects or nodes from the lower levels. The voxels in a volume, for instance, do not convey much on their own. However, they can be explicitly or implicitly assigned to a hierarchical group that represents, for instance, an organ or a blood vessel. An explicit assignment of voxels to a group corresponds to a segmentation of the volume. A correlation of voxels can also be revealed by tuning the visual mapping of their retinal channels, e.g., with the help of a transfer function. Such a group carries a semantic that would be lost if the inherent positions of the individual voxels were to be changed.

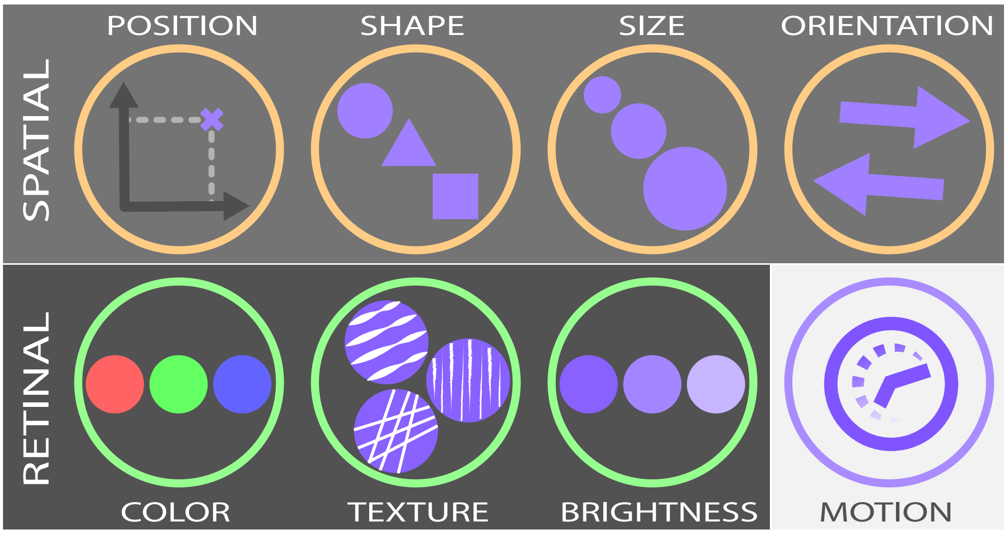


Figure 2.1: The eight visual channels: position, shape, size, orientation, color, texture, brightness, and motion.

In other types of spatial data, such as 3D scenes containing geometric meshes, groups can be explicitly defined, e.g., in the sense of a scene graph, or they can be dynamically created, e.g., according to data semantics, or spatial proximity. Each hierarchy level can thereby possess its own set of visual channels. The spatial channels (position, shape, size, orientation) can be described based on global or local coordinates that correspond to a certain hierarchy level. A molecular model of a microorganism, for instance, consists of hierarchical compartments, such as the cell membrane or the nucleus, that correspond to the biological structures of the microorganisms. These compartments are formed by individual molecules that are positioned, oriented, and scaled in respect to the center point of the microorganism (the respective *microorganism's* local coordinate system). The molecules consist of individual atoms that, in turn, are positioned around the molecule's center (the respective *molecule's* local coordinate system). This differentiation of visual channels into different hierarchical sets is important when it comes to the visual abstraction of such spatial data, as we will discuss in Chapter 6.

Considering these characteristics of spatial data, the design space for encoding non-spatial attributes without an inherent mapping within a spatial representation form is limited. In the following section, we will explore the ways in which a spatial representation can be adapted in order to accommodate additional non-spatial information.

2.2 Visual Integration Strategy I: Adapting the Spatial Representation

The given mapping of visual channels in a spatial representation can be deliberately adapted, in order to increase the expressiveness and effectiveness of a visualization. The *expressiveness* determines whether a graphical representation conveys the desired information, while the *effectiveness* describes how well the visual encoding utilizes the capabilities of the display device and the human perceptual and cognitive system [Mac86]. In practice this means that a spatial representation can be adapted in order to make the essential information that is encoded better readable, e.g., with occlusion handling techniques, or to embed additional attributes that do not have an inherent mapping within the representation. This approach is a viable strategy when the essential spatial and non-spatial information can be expressively conveyed within a spatial context. For adapting a spatial representation, visualization designers have three types of operations at their disposal: they can *neglect*, *modify*, or *re-assign* the inherently given mapping. *Neglecting* the mapping to a visual channel means entirely removing the inherently mapped data attribute from the visual encoding. An example is the removal of inherent retinal attributes from a representation by displaying all visual marks with the same color. Color, brightness, and texture thus do not encode any information anymore. Such a removal of visual information can serve the reduction of non-essential details, in order to streamline a visual representation and to steer the users' attention towards more important visual aspects, i.e., to create a more effective visualization. The second type of operation, the *modification* of a visual channel, corresponds to a distortion, projection,

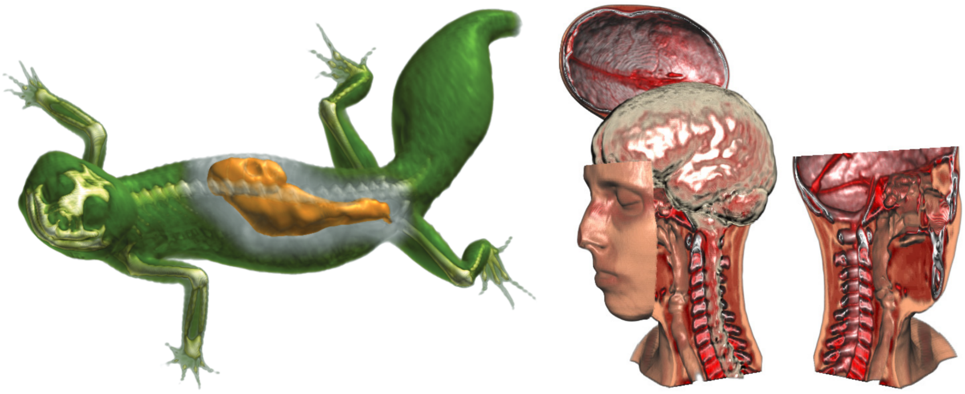
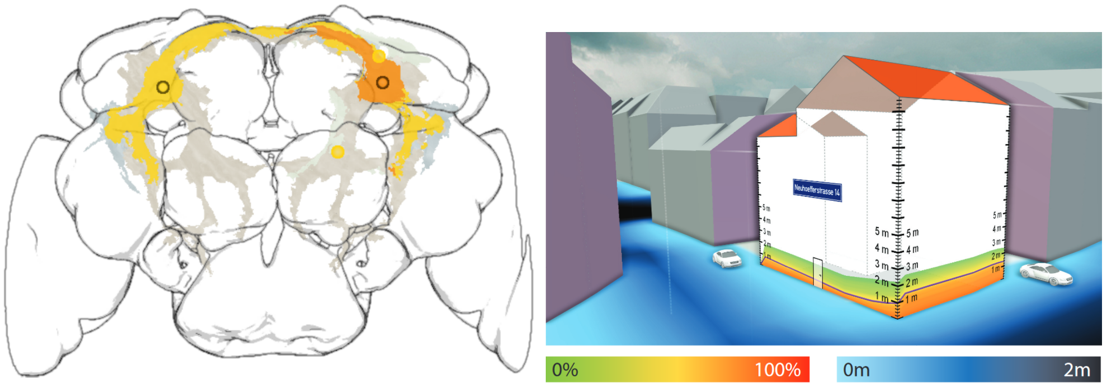

(a) (b)

Figure 2.2: Modifying visual channels in illustrative visualization: a) changing the transparency (in the color channel) of outer layers to reveal skeleton and organs in importance driven volume rendering [VKG04], (b) changing the position, shape, and orientation (spatial channels) of parts of a volume to create an exploded view showing the inner anatomy of a human head [BG06].



(a) (b)

Figure 2.3: Re-assigning visual channels: a) the retinal channels of the volumetric depiction of overlapping brain regions are re-assigned to encode the degree and the area of an overlap between individual regions [SMB+14], b) the retinal channels of the building's facade are re-assigned to encode the probability of water exposure during a flooding [CKS+15].

or other type of change to the original visual encoding without exchanging the data attribute that the visual channel is associated with. Such operations are often applied in illustrative visualization in order to reveal important details that would otherwise remain occluded by less important parts of the data. Depending on which parts of the data are important, retinal or spatial channels or a combination of both can be adapted. Figure 2.2a shows an example where the transparency (retinal) of one object is increased in locations that would occlude other more important objects. Figure 2.2b displays a situation where the retinal channels are preserved. The spatial channels, in this case, the position and orientation, are modified in order to remove occluding structures. The third type of operation for adapting a spatial representation, the *re-assignment* of visual channels, exchanges the given mapping of an attribute for a particular visual channel with another attribute where the mapping is not inherently given. In Figure 2.3a, for instance, the retinal channels of the volumetric depiction of segmented brain regions are re-assigned with derived information about areas of overlaps between regions. In Figure 2.3b, the retinal channels of a building are re-assigned with information about the probability of rising water levels during a flooding.

We refer to the first two types of operations, i.e., the neglecting and modification of visual channels, as *visual abstraction* of a representation. These operations take something away from a representation or distort it in a way that makes it easier to understand, i.e., more efficient. The re-assignment of attributes on the other side, differs from these two operations as it adds something to the representation, i.e., additional information without a given mapping, in order to make it more expressive. One, more, or all the visual channels of a spatial data representation can be adapted by these three types of operations.

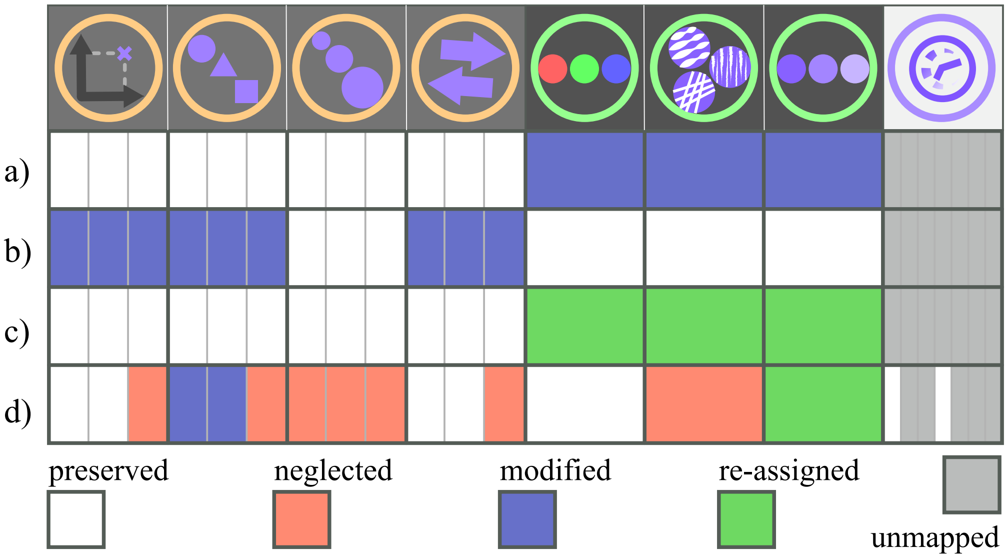


Figure 2.4: Schematic representation of the adaption operations applied within the previously discussed examples. The three subdivisions in the spatial channels correspond to the x, y, and z-axis respectively. a) Importance driven focus of attention (Fig. 2.2a): only the retinal channels are modified to reveal an occluded focus object. b) Exploded view (Fig. 2.2b): the spatial channels of the context are modified to reveal the occluded focus object. c) Figure 2.3: retinal channels are re-assigned to display additional information within a spatial context. d) Sport analysis (Fig. 2.5): visual abstraction of the spatial channels. Only the two-dimensional position and orientation as well as their change over time are preserved.

Figure 2.4 illustrates this design space that the visual channels define. Each column represents a visual channel. In the case of spatial channels, each column is split into three sub-columns representing the individual spatial dimensions. Motion is split into seven sub-columns as it can describe the change over time of each of the remaining channels. This schematic representation can be used to encode the types of operations that can be found in a certain representation in order to compare and classify different adaption approaches. Different colors indicate, whether a mapping was given or chosen, and which type of operation was applied. We encoded the adaption approaches for the examples that we discussed so far in this section in Figure 2.4.

In the distribution of modified and preserved channels in Figure 2.4a we can observe, how the approach in Figure 2.2a aims to convey the accurate spatial context of the displayed information by preserving the spatial channels. Figure 2.4b on the other hand, reflects how the approach in Figure 2.2b distorts the spatial mapping and preserves the given retinal information in order to reveal the data of interest. In both cases, the data is separated into a focus and a context subset. The adaptions of the mapping are thereby only applied on the context, to reveal items in focus. Still, neither approach neglects or re-assigns any of the given mappings. The spatial context therefore remains readable.

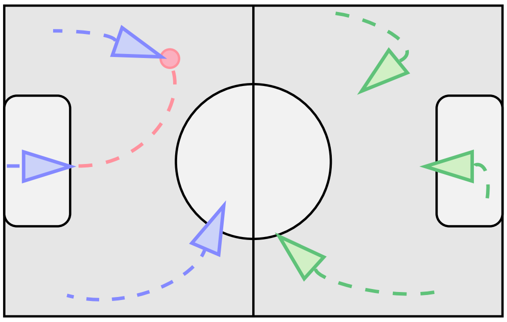


Figure 2.5: An adapted spatial representation of player moves in a soccer game for the purpose of post game analysis. The two-dimensional position, orientation, and motion of the players and the ball have been preserved.

The examples from Figure 2.3 can be encoded in this schematic as well. Figure 2.4c is valid for both examples: the spatial channels are preserved, and the retinal channels are re-assigned with information that has been derived from the spatial data, i.e., the amount of overlap between volumes and the probability of water levels.

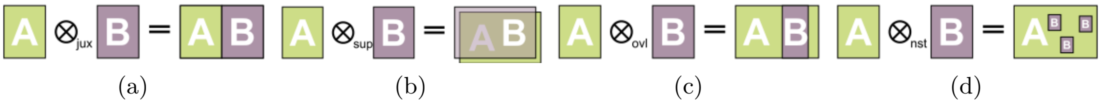
Another example of adapted spatial representations can be found in the analysis of sports games, such as the one in Figure 2.5. We can inspect the adaption of mappings in this representation in the schematic in Figure 2.4d. Here the positions and orientations of the players and the ball in relation to the playing field, as well as the respective team affiliations, are the essential attributes for efficiently analyzing team strategies. This is reflected in the white cells for the x- and y-axis of position and orientation that describe values in respect to the plane of the soccer field. The z-axis represents the depth in respect to the top-down perspective and has been neglected for these channels, since the field is presented in a two dimensional context. The shape is modified into a glyph that still can convey the rotations by its orientation. The color is modified to represent the player's team affiliation. The inherent mapping of motion is preserved for positions and orientations, as it encodes the movement of the player and ball over time. Further, the visualization designer can decide to re-assign the speed of a player to the brightness channel. The remaining inherent information, i.e., size, and texture, are not essential for the fulfillment of the analyst's task, and could even distract from the analysis. Their inherent mapping is therefore neglected.

When regarding the schematic depictions of the presented examples, different high-level patterns can be observed that are dictated by the respective visualization purpose. Figure 2.4a represents visualizations where the original shape and size of spatial entities are necessary to properly assess a data set. Figure 2.4b represents cases where the retinal properties of an object should be preserved for a proper data analysis. Figure 2.4c represents scenarios where the spatial channels serve as context for additional information that is re-assigned to the retinal channels. Figure 2.4d describes representations where the inherent temporal mapping of spatial channels has to be conveyed. These are just some examples of potential pattern classifications in the adaption of spatial representations.

In scenarios, where the expressiveness of a visualization strongly depends on conveying non-spatial relationships, the adaption of the spatial representation alone might not be enough to handle such tasks, as not all non-spatial relationships can be adequately depicted in a spatial context. Further, it might occur that the number of essential attributes that need to be displayed, exceeds the number of available visual channels in the spatial representation. In such cases, additional representations, each with their own set of visual channels, have to be allocated and integrated with the spatial representation. In the following section, we therefore discuss the different strategies of how multiple representations can be visually composed into a single coherent visualization.

2.3 Visual Integration Strategy II: Composing Multiple Representations

An expressive and effective visualization often requires more than one view on the data [Rob98]. Different representation forms and different visual mappings can reveal different types of relationships. While conveying the inherent spatial mapping is often essential for understanding spatial relationships, spatial data representations have limited capabilities for revealing non-spatial relationships. In order to reveal non-spatial relationships, a flexible mapping of spatial channels to non-spatial dimensions is required, e.g., for the axes of a scatterplot. The logical consequence of this situation is to combine multiple representation forms in order to join their strengths and balance out their weaknesses.

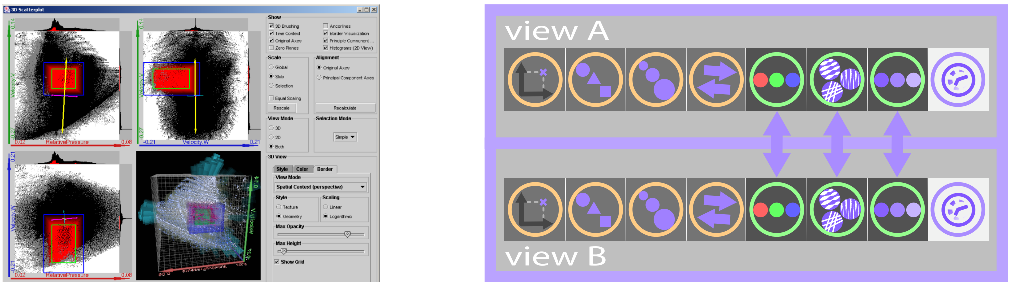


(a) (b) (c) (d)

Figure 2.6: Composite visualization paradigms according to Javed and Elmqvist [JE12]: juxtaposition, superimposition, overloading, nesting.

There are different ways, in which multiple representation forms can be joined. Javed and Elmqvist [JE12] define a taxonomy of composite visualization views (CVVs) based on five different composition strategies: juxtaposition, integration, superimposition, overloading, and nesting. In the following, we will give a brief description of each strategy. Further, we will describe what kinds of relationships each strategy is able to convey between composited representations, as well as the constraints that each type of composition puts on the mapping of the involved representations' visual channels.

Juxtaposition (Fig. 2.6a) corresponds to the coordinated multiple view (CMV) paradigm [Rob07], i.e., placing representations side-by-side in separate views. Juxtaposition poses no restrictions on the mapping of visual channels in each view. Therefore, they are well suited for freely exploring relations among multiple data dimensions. However, the way, in which these relations between visual marks should be conveyed across views, has to be explicitly defined by the visualization designer. To convey the relation,

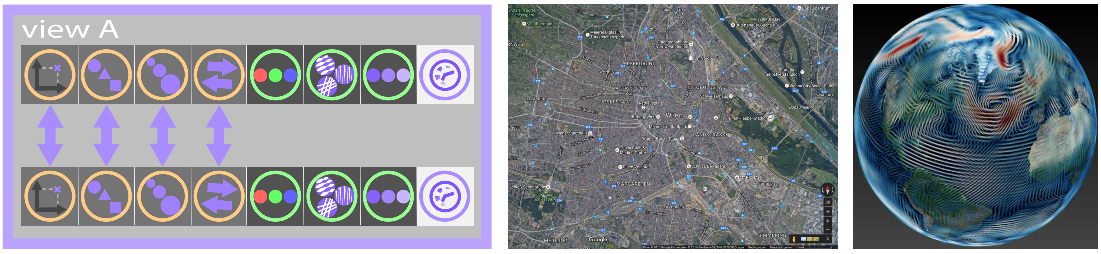


(a) (b)

Figure 2.7: Juxtaposition: a) representations are displayed in separate views that are placed side-by-side [PKH04], b) the retinal channels between representations can be coordinated in order to encode relations between selected items.

typically certain visual channels are coordinated between views, such as the color of visual marks that are interactively selected by the user (brushing&linking), as depicted in Figure 2.7a. As opposed to the other visual integration strategies, in juxtaposed representations, interaction is thus essential to establish these relations. We illustrate such a relation between juxtaposed representations in the schematic in Figure 2.7b. Since users have to switch their attention between multiple views while maintaining a mental image of the relations between data items, juxtaposition can be straining on the users' cognition [WBWK00].

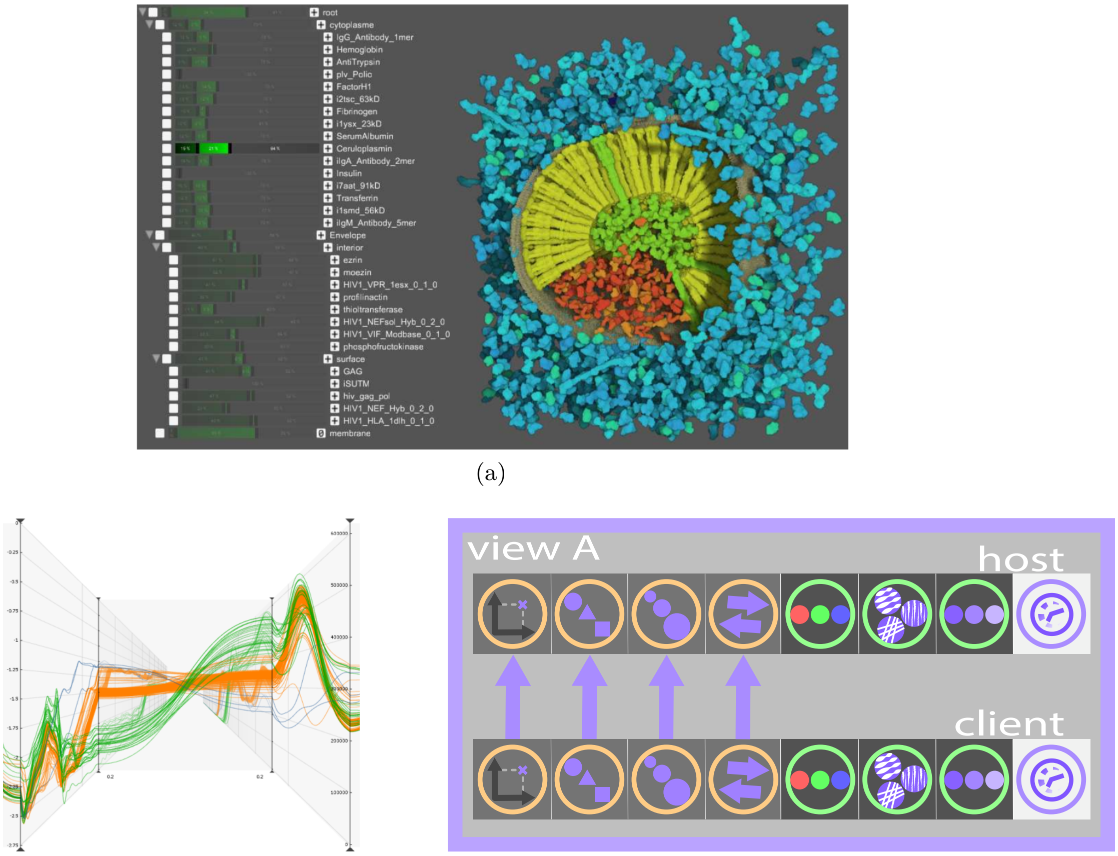
Superimposition refers to the overlaying of different representations within the same view (Fig. 2.6b). All superimpositioned layers thereby share the same spatial frame of reference, i.e., the spatial channels convey the same meaning for each layer. This restriction is illustrated by the schematic in Figure 2.8a. Relations between the layered representations are thus implicitly determined by the relative differences between the values of the spatial



(a) (b) (c)

Figure 2.8: Superimposition: a) the mapping of spatial channels across representations is restricted, as positions, shapes, sizes, and orientations have to convey the same meaning across all layers, b) superimposed cartographic layers [Inc], c) glyphs encoding direction and pressure data of wind predictions are superimposed on a 3D spatial rendering of the earth [Dol07].

channels of each layer. As such, this form of visualization composition is especially well suited for conveying spatial relationships between the composited representations. A common example of superimposition is the overlaying of different cartographic layers, such as a relief map and a street map, in the visualization of geographic information systems (GIS) (Fig. 2.8b). Another common application is flow visualization, where glyphs representing, for instance, the flow direction and speed, are rendered on top of a spatial representation for reference (Fig. 2.8c). Since the visualizations are stacked, each one can make use of the entire available display space. On the downside, superimposition does not scale well due to increased occlusion that can occur when overlaying many visually dense representations.



(a)

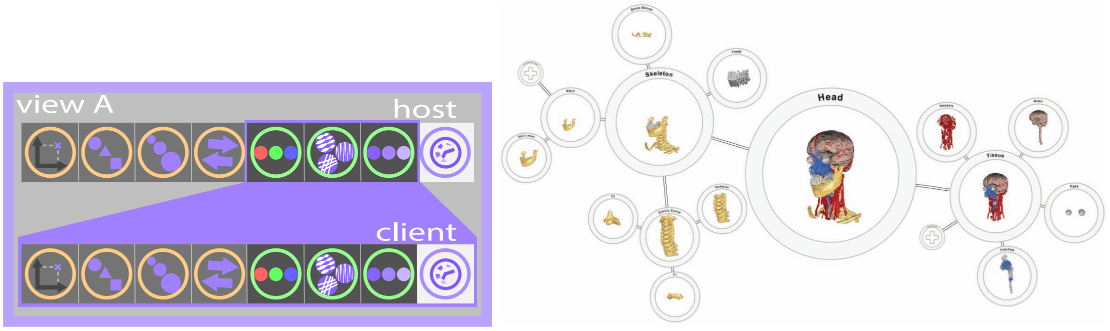
(b)

(c)

Figure 2.9: Overloading: a) a spatial representation of molecules overloaded with histograms of protein type visibilities [LMMS+16] – the spatial relation does not convey any meaning, b) two time series plots are overloaded with a parallel coordinate plot that indicates relations between both time series [GRPF16] – the spatial relation between host and client conveys a meaning, c) if the spatial relation between host and client should convey a meaning, the mapping of spatial channels is constrained by the host.

Overloading (Fig. 2.6c) composes two representations within the same view as well. However, in this case, the overloaded representation assumes the role of the host, and the remaining representations are the designated clients. The client representations are thereby embedded into the spatial frame of reference of the host. As opposed to superimposition, the values of spatial channels can have different meanings for the host and the client representations. Nevertheless, just like in superimposition, the relative positions between the visual marks of host and client can convey a meaning, as well. However, in this case, the conveyed relations are rather of non-spatial than of spatial nature. Overloading is therefore well suited to indicate non-spatial relationships within the host representation through the encoding of the client representation. An example of overloading is displayed in Figure 2.9b: time series plots are overloaded with a parallel coordinate plot [GRPF16]. The parallel coordinate plot is bound to the spatial layout of the time series in order to convey non-spatial relations between them. In order to create a meaning between the spatial channels of host and client representations, the visualization designer has to deliberately choose the spatial frame of reference of the client representation in a way that conveys the intended meaning. The mapping of spatial channels can thereby be constrained by the host representation, as illustrated in Figure 2.9c. Figure 2.9a displays a spatial representation of a molecular data set that is overloaded with histograms that convey the visibility of proteins. However, in this case, the spatial relation of client and host does not convey any meaning.

Nesting (Fig. 2.6d) can be described as a special form of adapting a host representation, where the retinal channels of the host are re-assigned to depict an entire client representation (as opposed to re-assigning a channel with a different attribute). This relation is depicted in the schematic in Figure 2.10a. A client representation thereby typically encodes additional information about the data item that corresponds to the mark in which it is hosted. This pattern is well suited for conveying relationships between the



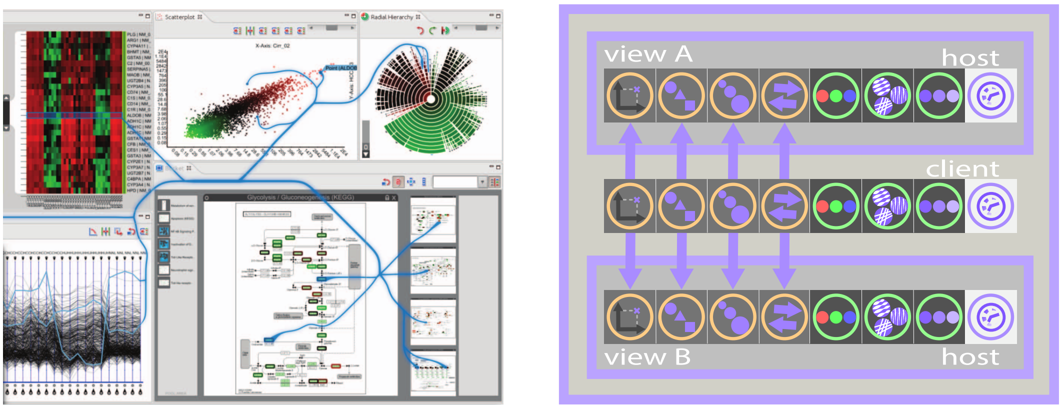
(a)

(b)

Figure 2.10: Nesting: a) the retinal channels of one representation serve as the host for another (client) representation, b) volumetric renderings of anatomic structures are hosted within the visual marks (the nodes) of a graph that encodes the structures' hierarchical relationship [BVG10].

spatial channels, e.g., the position, of data items in the host representation and the information that is mapped to the visual channels of the respective client representations. Figure 2.10, for instance, displays hierarchic clusters of an anatomical 3D model that are nested within the nodes of a graph hierarchy. The graph hierarchy thereby corresponds to the 3D model's hierarchy. The downside of this technique is the limited display space that is available to each client representation, as it has to be contained within the area of the visual mark in the host representation. The limited display area of the client is therefore not well suited for complex representations, as they might be hard to read.

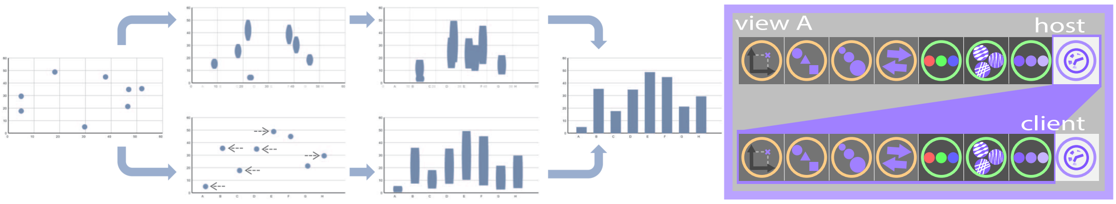
Integration is the final visualization composition pattern proposed by Javed and Elmqvist [JE12]. Similarly to juxtaposition, this pattern places representations side-by-side in separate views. However, integration differs in the way in which the relation between visual marks is established across views. The relation is established via explicit visual links that connect the visual marks of corresponding data items, as depicted in Figure 2.11a. These visual links are independent of the visual channels of the respective views. Due to the visual links, the relations are encoded more explicitly than in juxtaposition. However, integration scales badly as the links have to avoid occlusion of other data items.
We found that Javed and Elmqvist's definition of integration is actually redundant, as this strategy can be described as the combination of two other visual composition strategies: juxtaposition and overloading. In this sense, juxtaposed representations are simply overloaded with an additional representation, i.e., a graph, that connects corresponding marks across juxtaposed representations. The graph is thereby the client representation and has to adjust its spatial mapping to the host representations (the juxtaposed views) in order to convey the intended meaning. We illustrate this concept in the schematic in Figure 2.11b.



(a)

(b)

Figure 2.11: Integration according to Javed and Elmqvist [JE12]: a) explicit encoding of relations across juxtaposed views with visual links [SWS+11], b) our schematic describes the visual link as a proper representation layer that is overloading juxtaposed views.



(a) (b)

Figure 2.12: Animated transitions: a) one representation is transformed into another one over time [HR07], b) the transition is encoded in the motion channel, e.g., by interpolating the visual channel values of both representations over time.

It should further be noted that the term *integration* that Elmqvist and Javed use to describe the above discussed CVV pattern, is used with a different connotation in other visualization literature. Balabanian [Bal10], for instance, differentiates between integrated views and linked views. According to his definition, linked views are visualizations that are placed side-by-side (corresponding to juxtaposition and integration). Integrated views, on the other side, compose multiple visualizations within the same view (corresponding to superimposition, overloading, and nesting). In other works [OSS$^+$16, SRH$^+$09], as well as in the scope of this thesis, integration simply means the visual and functional coordination of multiple representations, regardless of whether they are composed within the same view or across multiple views.

The five patterns that were proposed by Javed and Elmqvist solely describe the *spatial* composition of representations. However, also a *temporal* composition of representations is possible, i.e., in the form of animated transitions. An animated transition integrates two representations over time within the same view (see Fig. 2.12a). Each representation can thereby use the entire available display space. This form of composition is capable of conveying spatial and non-spatial relationships between two representations in an intuitive and engaging manner [TMB02]. The relation between representations is established by transforming the mapping of visual channels over time, e.g., via interpolation. The animation conveys, how a visual mark in representation A transforms to a corresponding mark in representation B. Animation thereby does not pose any constraints on the mapping of spatial and retinal channels. However, since the transition is described via the change of each channel over time, it occupies the motion channel, as indicated in Figure 2.12b. Due to the transition, the two representations are never shown at the same time. Animation is therefore rather suitable for presentation tasks and less for exploration/analysis tasks. Further, special care has to be taken when authoring an animated transition. Depending on the number of visual marks on display, clutter, occlusion, and visual stimuli overload can be a challenge. For instance, when too many elements on the screen are moving at the same time, are occluded, or are moving too fast, the relation can get lost in transition. We will discuss the different steps that have to be considered when authoring a transition in Chapter 5.

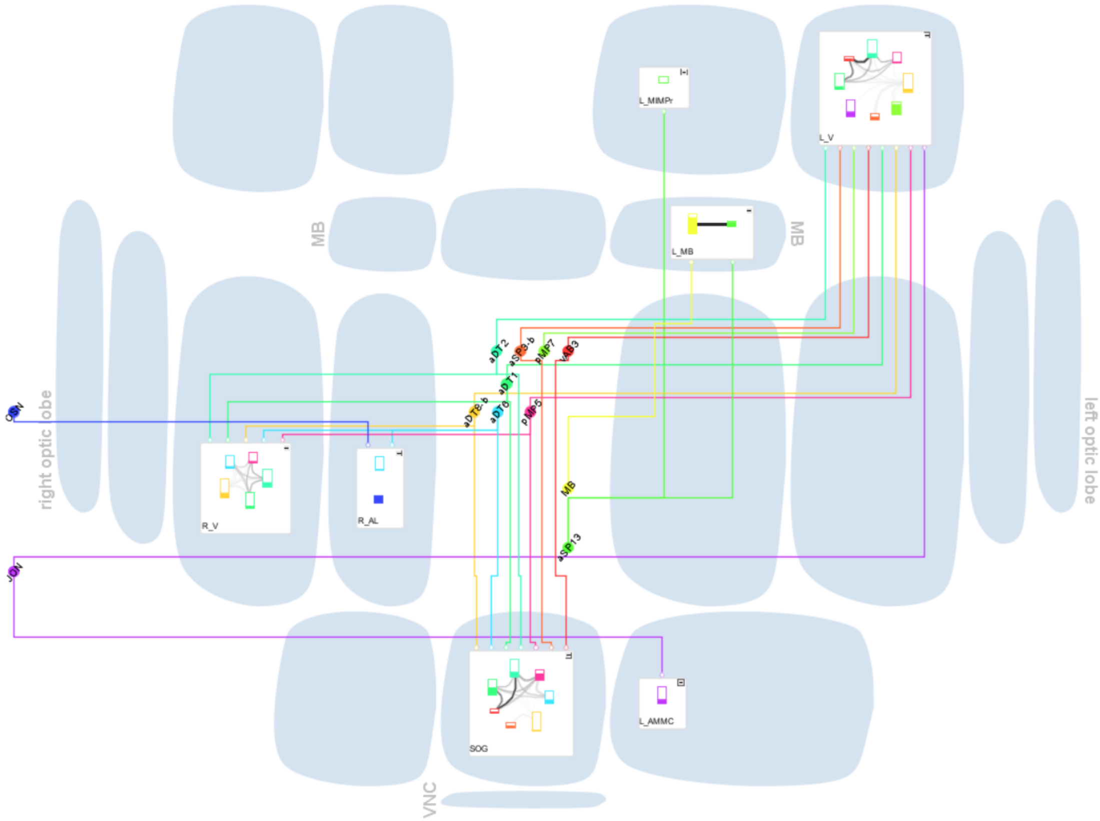


Figure 2.13: A combination of multiple visual integration strategies in the visualization of potential neuronal connectivity within the fruit fly's brain [SBS+13]. The visualization consists of three visually integrated representation layers: the background (layer 1), the network graph (layer 2), and the nested graphs (layer 3). The schematic depiction of the applied visual integration strategies can be found in Figure 2.14.

This concludes the description of the six proposed strategies for the visual integration of representations. The appropriate strategy for a certain task is typically chosen based on the kind of relationship that should be highlighted between two representations and based on the type of constraints that a strategy poses on the mapping of the respective representation's visual channels. To recap: juxtaposition (and integration) can convey relationships between arbitrary dimensions but in separate views; superimposition is suitable for conveying one-to-one spatial relationships; an overloaded client representation can convey non-spatial relationships within the host representation; nesting conveys the relation between a host mark and a client representation; animated transitions can convey relations between different mappings without the need for interactive exploration.

If one of these strategies is not sufficient to convey a specific insight, also combined approaches are possible (as we saw in the example in Figure 2.11). In Figure 2.13, we give another example of combined visual integration strategies, where potential neuronal

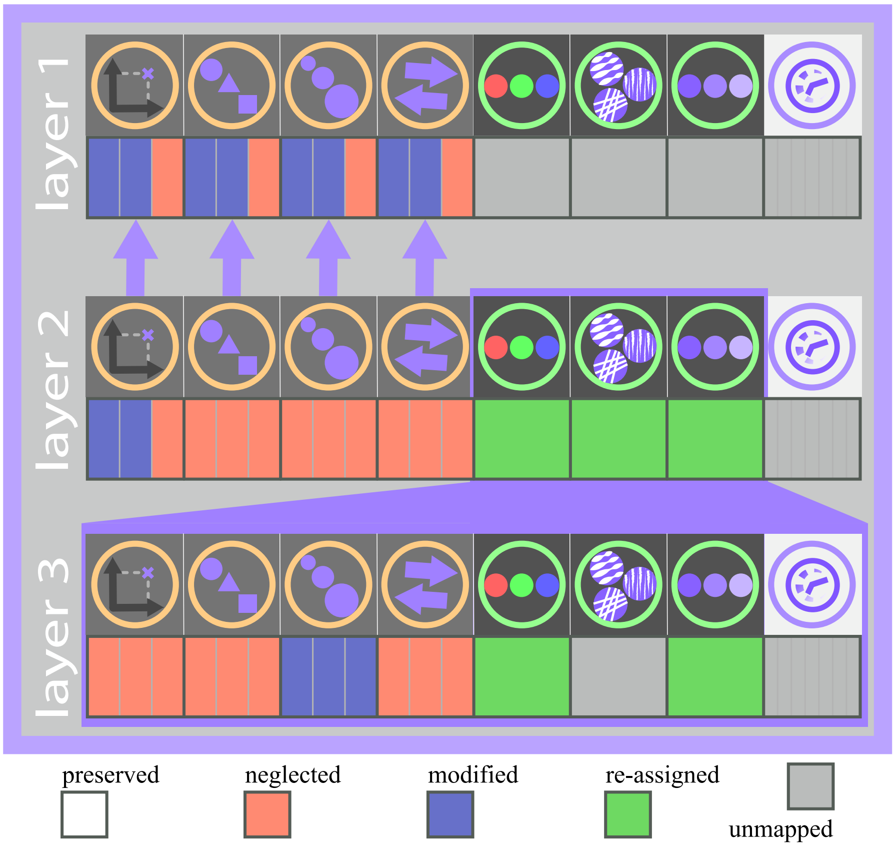


Figure 2.14: The schematic depiction of the visual integration strategies that are applied in the visualization of potential neuronal connectivity in Figure 2.13. Layer 1 (background): all spatial channels have been modified to achieve a visually abstracted representation of the fly's brain template. Retinal channels have been neglected. Layer 1 is overloaded with layer 2, i.e., a network of smaller brain regions and the associated neurons. The graph is a non-spatial representation form, i.e., all spatial channels have been neglected. Only the modified positional encoding remains to depict the correspondence of the graph nodes to the larger brain regions in layer 1. The retinal channels in layer 2 are re-assigned in order to allow nesting of the representations in layer 3. Layer 3 features graphs that encode the overlap between segmented neuronal volumes in each smaller brain region. Here, the only preserved spatial attribute is the relative size of the volumes, modified and mapped to the height of a node. The node's fill color is re-assigned to represent the overlap of the volume with a brain region. The brightness of an edge between two nodes encodes the overlap volume between them.

connections in the brain of a fruit fly are displayed. The visualization is composed of three different representation layers that are all originally based on spatial data, i.e., the segmented volumes of neurons within the brain. The background (layer 1) is an *adapted spatial representation* of the brain template of the fruit fly that is *overloaded* with a graph (layer 2) representing connections of neurons (round colored nodes) to smaller brain regions (large square nodes). The graph's large nodes are, in turn, each *nested* with an additional graph (layer 3) of potential neuronal connections between the neurons in the respective brain region. The schematic in Figure 2.14 depicts the adaption of the visual channels as well as the relation between integrated layers.

2.4 Visual Integration Challenges

Before concluding this chapter, we will highlight some of the challenges in visualizations that are composed of spatial and non-spatial representation forms. We thereby refer to prior work, in which we address challenges and recurring design questions in such integrated systems in the scope of a methodology [OSP$^+$16]. The methodology examines the three visual perception tasks that Elmqvist and Tsigas identified in the context of a taxonomy of 3D occlusion management techniques [ET08]. These tasks are target discovery, target access, and spatial relation. In order to illustrate these tasks and the associated challenges, we will regard them in the context of a tunnel maintenance scenario (Fig. 2.15) where the cracks in a tunnel wall are the data items of interest.

Target discovery is the task of finding a specific item within a scene. A specific crack could be located on any of the displayed tunnel walls. *Target access* is concerned with retrieving the graphically encoded information that is associated with a target. Accessing spatial attributes, such as the length, orientation, and diameter of a crack can be essential for a proper assessment of a tunnel's stability. *Spatial relation* refers to the retrieval of information that is encoded in the spatial relationship between multiple targets and their spatial context. The constellation of multiple cracks to each other, as well as their location on a tunnel wall (spatial context), can give critical information about a tunnel's stability.

In the context of 3D spatial visualizations, there are four inter-dependent factors that can impede these three visual perception tasks: the view-frustum, the viewing angle, the viewing distance, and occlusion. The *view frustum* is only capable of displaying a small part of a potentially complex 3D spatial scene (see Fig. 2.15). Without any support, the task of *target discovery*, e.g., locating a specific crack in the tunnel, can thereby succumb to a game of "hide and seek", as objects might lie fully or partially outside the view frustum. Similarly challenging would be the localization of multiple cracks within the view frustum when trying to analyze their *spatial relation* to each other. *Viewing angle* and *viewing distance* can determine whether *target access* of the spatial properties of an object, such as a crack in a tunnel wall, is successful or not. If the viewing angle and the distance are not appropriate, the detailed spatial features of an object might be too distorted or too small for a proper target access. All three visual perception tasks are

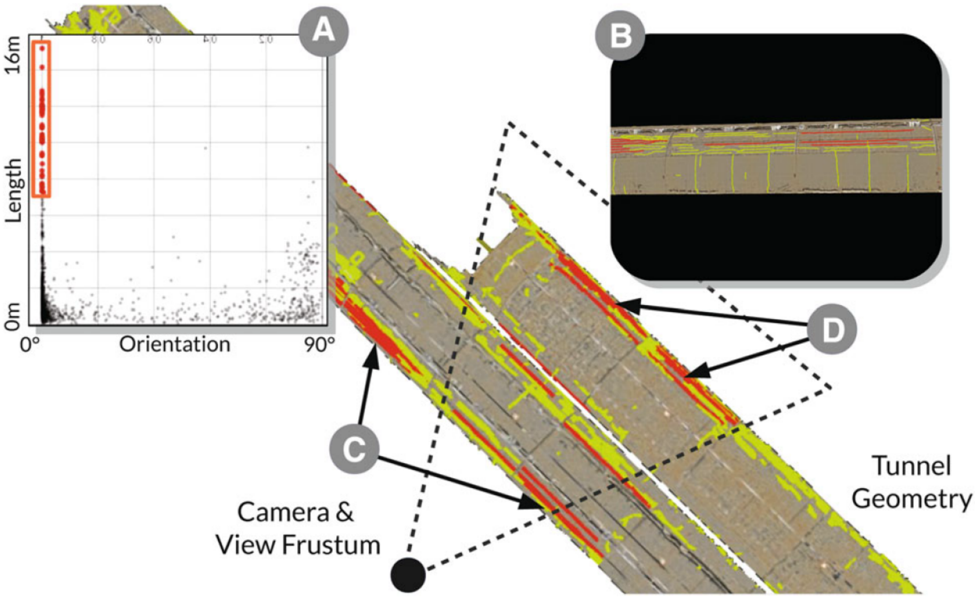


Figure 2.15: Tunnel cracks as data items in a non-spatial (A) and spatial (B) context. While the visual perception tasks of target discovery, target access, and spatial relation can be easy to solve in a non-spatial representation (A), data items in 3D spatial scenes might be only readable from a certain viewing angle, they can lie outside the view frustum (C) or they can be occluded (D) [OSP+16].

further impeded through full or partial *occlusion*, caused either by other objects or by the context geometry, e.g., the tunnel walls.

If these obstacles are not properly addressed, also certainty can be impacted in a negative way, as users have no way of knowing if they discovered all relevant (selected) targets in the 3D scene. A proper functional integration of spatial and non-spatial views that explicitly address the aforementioned obstacles, is the key to support the perception tasks in 3D spatial visualization environments. The non-spatial representation thereby provides the user with an overview, and the spatial representation supplies the details on demand. Users can locate an item according to non-spatial or derived spatial attributes within the overview and then, through special functional integration mechanisms between spatial and non-spatial views, automatically locate and inspect the item in the spatial context, thus saving them from manually searching for the item in the spatial view.

In our methodology [OSP+16], we propose several strategies that target the aforementioned obstacles. *Guided navigation* automatically adjusts the camera in the spatial view to either an overview position when multiple entities are selected or to a detail position when only one entity is selected. The overview position supports the recognition of spatial relations by keeping entities that have been selected in another (non-spatial) view within the spatial view-frustum. The detail position supports target discovery and access by displaying a selected tunnel crack in a frontal view at close proximity. *Adaptions of the spatial representation* are applied for occlusion handling of context geometry, i.e., the tunnel walls, to guarantee that all brushed entities within the spatial view frustum are visible. Another adaption of the spatial representation is triggered when the camera distance to selected items is too large to allow a proper visual discrimination. In this case, the original shape is modified to a glyph representation that allows the user a visual discrimination even at large distances in order to convey the spatial relation of selected items.

Here we discussed specific solutions to a specific application scenario. However, the identified challenges in visualization systems that integrate heterogeneous, i.e., spatial and non-spatial, representation forms also hold true for other application scenarios. In Part II of this thesis, we present novel techniques in other concrete integration application scenarios. Before that we will address another important aspect of integration: the functional integration of visually integrated representations that is required to enable linked interactions between the various data representations in a visualization system.

CHAPTER 3

Functional Integration

This chapter is based on the following publication:

Johannes Sorger, Thomas Ortner, Harald Piringer, Gerd Hesina, and Eduard Gröller. *A Taxonomy of Integration Techniques for Spatial and Non-Spatial Visualizations.* In 20th International Symposium on Vision, Modeling and Visualization (VMV 2015). October 2015 [SOP+15].

VISUAL integration enables a visualization system to convey the entire range of attributes that is required to fulfill a task, within the most suitable representation form. A functional integration between the visually integrated representations is required in order to actually join the strengths of the involved representations, i.e., by integrating their functionalities. Functional integration describes, how one or more views on (or representations of) the data react to the changes in another view or representation. Without such functional linking, each representation, no matter if visually integrated or not, remains an isolated entity. In order to retrieve the desired information from such isolated representations, the user would have to interact with each one individually. Functionally integrated representations are enabled to show relevant information in relation to the information that the user currently interacts with in another representation, e.g., by filtering or highlighting corresponding data entities.

In this chapter, we explore the ways in which a functional link between two representations can be established. In the scope of a taxonomy, we thereby describe how a functional link can be triggered within one representation, and how an integrated representation can react. A link can determine, *what* kind of data is shown, *how* it is shown, and, especially in spatial representation forms, from *which perspective.* According to these three questions, we structure a visualization into three components: the *Data*, the *Visual*, and the *Navigation* component. The Data component governs which parts of the data are shown in which granularity. The Visual component is responsible for the way in which attributes are mapped to visual channels. The Navigation component handles modifications of the view frustum. Each component supports certain interactions and events that correspond to user intents. A functional integration type is defined by the specific combination of the component on which an interaction is issued and the component where as a result an event is triggered. In the remainder of this chapter, we will refer to functional integration simply as integration for the sake of brevity.

3.1 Introduction

The visual representation of data is traditionally classified into methods that assume an inherent mapping from data values to spatial coordinates and into methods for data lacking explicit spatial references, where the spatialization is chosen [TM04]. In practice, however, users often need to analyze data that contains multiple facets, like spatio-temporal and multivariate data characteristics [KH13]. In flow visualization, for instance, non-spatial representation forms are used as an overview to select and highlight interesting attribute values in a spatial volumetric flow representation [Dol07]. In traffic simulation and road planning, traffic data is assessed directly within the spatial context of a 2D or 3D map, while non-spatial views serve the statistic analysis of trends and non-spatial data correlations [WYL+14].

The benefits of visualizations that integrate spatial and non-spatial data facets have been repeatedly emphasized in visualization literature [SRH+09, TM04]. These benefits were also the topic of panel discussions at the IEEE Visualization conferences in 2003 and 2006 [HWM+06, RTM+03]. Fuchs and Hauser make a strong case for the application of multi-method visualization: *"a tight integration of multiple techniques gives a key advantage towards understanding the investigated data"* [FH09]. The authors identify three main advantages of multi-method visualization: improved effectiveness (each part of the data is visualized by the most appropriate method), minimizing visual clutter, and separation between the questions of how and what to visualize. Kehrer and Hauser state that there is a lack of general concepts for handling the heterogeneity of multifaceted data [KH13].

While basic coordination techniques like brushing & linking are quite common, the ways in which spatial and non-spatial representations can benefit through integration are manifold. With the absence of a proper formalism though, it is difficult to describe and discuss this design space. We therefore propose such a formalism by describing the ways in which heterogeneous representation forms can be integrated. We thereby describe spatial and non-spatial representation forms based on a separation into a Data, a Visual, and a Navigation component. Our model can classify systems in terms of how the components of two representations are integrated. While this model holds also true for the integration of homogeneous representation forms, in the context of this taxonomy, we will focus on the integration of heterogeneous, i.e., spatial and non-spatial representations.

The contributions of this chapter are:

- The specification of a taxonomy that defines the design space for integrating spatial and non-spatial representations.
- An in-depth analysis of this design space based on state-of-the-art approaches from the literature.
- A demonstration of the classification and comparison capabilities of our for real-world integration scenarios.

3.2 Related Work

Publications that explicitly address the integration of spatial and non-spatial representations are still rare. In their state-of-the-art report on visualization of multivariate scientific data, Fuchs and Hauser classify techniques by data type, i.e., scalar, vector field/flow, and tensor field visualization, and by the stages of the visualization pipeline where these techniques are applied [FH09]. Kehrer and Hauser give a survey of multi-faceted scientific data visualization [KH13]. The authors describe five different facets of scientific data by which the discussed techniques are categorized: spatiotemporal, multivariate, multimodal, multirun, and multimodel. For each facet, techniques are divided into approaches for representation, computational analysis, and interaction. Both surveys stress the importance of multi-method visualizations.

In their high level visualization taxonomy [TM04], Tory and Möller classify visualization algorithms based on whether they handle data discretely or continuously, and whether the spatialization is chosen, constrained, or given. With their taxonomy, they aim to inspire research ideas in hybrid visualization areas.

Boukhelifa et al. [BRR03] propose a model for describing coordination in exploratory multiple-view visualizations. The model uses the visualization pipeline to convey, which pipeline stages in connected views are linked through a coordination object. The authors specify the basics of coordination in a system: coordination entities, type, chronology, scope, granularity, initialization, updating, and realization. This coordination of individual pipeline stages across views is an important source of inspiration to us, as we explore the combination of visualization components across integrated representations. In her book, Munzner also weighs in on the coordination of juxtaposed views [Mun14]: *"The main design choices for juxtaposed views cover how to coordinate them: which visual encoding channels are shared between them, how much of the data is shared between them, and whether the navigation is synchronized."* We describe these three aspects by the three visualization components – as such they represent a high-level view of the visualization pipeline that the coordination model of Boukhelifa et al. [BRR03] is based on. In contrast to Munzner and Boukhelifa et al., we do not restrict our model to multiple-view systems, as all types of composite visualizations can be described.

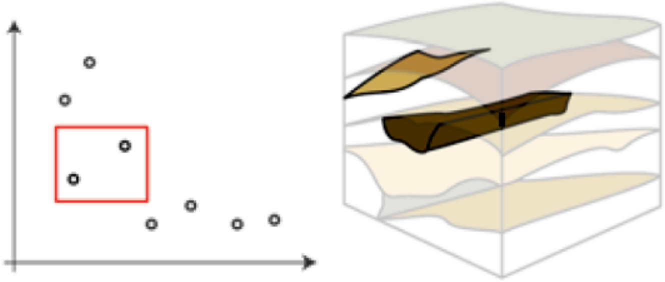


Figure 3.1: Selection in a scatterplot (non-spatial source representation) causes highlighting of the corresponding objects in the 3D view (spatial target representation).

The design space of composite visualizations is described by Javed and Elmqvist [JE12], as discussed in the previous chapter. The authors suggest a theoretical model that unifies the coordinated multiple-view paradigm with other strategies for combining visual representations, i.e., juxtaposition, superimposition, overloading, nesting, and integration. Functional integration, as we describe it, can take place between all types of composite visualizations. The works by Javed and Elmqvist [JE12] and Boukhelifa et al. [BRR03] treat the issue of handling multiple visualizations in a single framework from a visual and a functional perspective respectively. In this chapter, we go one step further, and explore the resulting design space in respect to heterogeneous representation forms.

Prior work on analyzing interactions within a heterogeneous visualization system has been proposed by Balabanian et al. [BVG10]. The authors categorize heterogeneous visualization techniques based on a 3×3 matrix that describes whether interactions are issued within spatial, non-spatial or integrated representations. We explore this design space in more detail, as we partition each representation into the three visualization components in order to regard their potential combinations. Furthermore, we do not consider an *integrated space* as a third category besides the *spatial* and the *non-spatial* one, as we partition an integrated visualization into the individual spatial and non-spatial representations from which it is composed.

The design space of integrated representations is acknowledged in the visualization literature – in respect to visual [JE12] and an functional integration [BVG10, BRR03]. However, save for the work of Balabanian et al. [BVG10] the intricacies of joining spatial and non-spatial representations are not discussed.

3.3 Model-Based Taxonomy

3.3.1 Overview

We define integration as the coordination of visualization components where each component stems from a different representation. In our taxonomy, we focus on the integration of heterogeneous, i.e., spatial and non-spatial, representation forms. The representation that a user interacts with is referred to as the *source representation.* The representation that is affected through the integration is referred to as the *target representation.* A common example of integration between two representations can be seen in Figure 3.1, where the selection of data points in a scatterplot (non-spatial source representation) causes the highlighting of the corresponding objects in a three-dimensional view (spatial target representation). As the key idea of the taxonomy, different types of integration can be discriminated based on which visualization components are linked in the source and target representation. In our model, components can be combined like building blocks, in order to form an integration. Simple integration types consist of only two components. However, also more complex combinations are possible.

A functional link between two integrated representations is always triggered through user interaction. In the aforementioned example, the selection of objects in a scatterplot corresponds to an interaction on the Data component of the non-spatial representation. The integration affects the Visual component in the spatial target representation by highlighting the selected objects in the 3D view. Each component supports certain types of input (interactions) and output (feedback) that are able to fulfill certain types of user intents. The Data component handles, for instance, the selection of data items, while the Visual component is responsible for visually indicating selected items.

As interaction is a well studied topic in the visualization community, we rely on established definitions, i.e., the user intents by Yi et al. [YaKSJ07], for describing the types of interactions supported by the individual components (Fig. 3.2). Brehmer and Munzner [BM13] give a good overview of other works on interaction terminology, as well as a comparison of definitions that are equivalent to the ones of Yi et al.

3.3.2 Definition of Representation Forms

For **spatial representation forms**, data attributes are mapped to their inherent positions in three-dimensional space, e.g., volume- and flow-visualization, real-time rendering, or GIS. Mapping to two-dimensional space is also considered as spatial, if the third dimension is negligible, e.g., slicing in volume visualization, 2D maps in GIS, certain flow visualization scenarios. The mapping from data to 2D or 3D space is therefore inherent [TM04].

We define **non-spatial representation forms** as encompassing all types of visualizations where the spatialization of the data's representation is chosen [RTM$^+$03]. Explicit spatial references of non-spatial attributes are either missing or visually abstracted (i.e., neglected or re-assigned as discussed in Section 2.2). Temperature in a climate simulation, for example, can be visualized in a non-spatial context as a histogram or in its inherent spatial context at the position in the volume where it was measured. Depending on the data type, non-spatial representations may include multivariate representation forms (e.g., parallel coordinates, glyphs), hierarchical representation forms (e.g., TreeMaps [Shn92]), graph representations, and other representation forms, such as text visualizations.

3.3.3 Notation

We abbreviate an integration between two visualization components by their initial letters, joined by an arrow. The example from Section 3.3.1, where a selection on the Data component triggered a highlighting operation in the Visual component, is therefore denoted as $D \rightarrow V$. The integration direction, e.g., spatial to non-spatial or non-spatial to spatial, is indicated in the subscript – *"s"* corresponds to a spatial representation, and *"a"* corresponds to a non-spatial representation: $D_s \rightarrow V_a$, $D_a \rightarrow V_s$, and $D_{a/s} \rightarrow V_{s/a}$ for bidirectional integration.

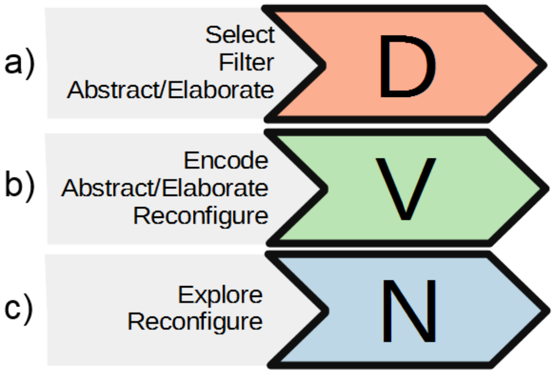


Figure 3.2: The Data, Visual, and Navigation components represent high level groups of visualization pipeline stages. Each component supports different types of user intents.

3.3.4 Visualization Components

We base our model components on the visualization pipeline as an established concept for describing the individual stages of a visualization. The pipeline typically consists of the original data, processed data, mapping, rendering, and image stages [FH09, HM90]. For a more streamlined model, we summarize the data related stages (original data and processed data) into the Data component, and the stages related to the visual output of a visualization (mapping, rendering, image) into the Visual component. Some models also acknowledge an additional view transform (or navigation) stage [BRR03, CR98] in order to encompass not only interaction with the visualization pipeline stages but also with the actual view on a 2D or 3D representation. We thus also include a Navigation component in our model. References to these three components can also be found in literature. Card, Mackinlay, and Schneiderman [CMS99] present a visualization reference model that also considers interaction, describing three primary transformations for mapping spatial data to visual representations: data transformations, visual mappings, view transformations.

Interactions are carried out *directly or indirectly* [Rob07] on a component in order to fulfill supported user intents. Indirect interaction is handled via menus or widgets, e.g., sliders and buttons. Direct interaction takes place directly on the elements of a representation, e.g., when brushing points in a scatterplot.

Data Component

The Data component (Fig. 3.2a) handles the question of *what* to visualize [FH09]. It comprises all parts of a visualization that are directly related to the data which the visualization is based on, i.e., original and processed data.

Interaction with the Data component supports the user intents of *select, filter*, and *abstract/elaborate* [YaKSJ07]. **Select** marks interesting data for further examination, while **filter** removes data according to user specified conditions. **Abstract/elaborate** corresponds to the aggregation of data, as well as the derivation of new data.

As an integration *source* (D→X), the Data component can supply other components with information on which data they should process. As an integration *target* (X→D), the Data component can receive information on how to process existing data as well as new data that was generated by other components.

Visual Component

The Visual component (Fig. 3.2b) is concerned with the question of *how* to visualize the supplied data. It comprises all parts of a visualization responsible for generating its final output image, i.e., the mapping, rendering, and image stage [FH09]. The Visual component defines, which data attributes are mapped to which visual channels in order to determine the visual appearance of a data item.

Interaction with the Visual component supports the user intents of *encode, abstract/elaborate*, and *reconfigure* [YaKSJ07]. **Encode** corresponds to assigning visual channels to data attributes, e.g., changing the color of a visual mark in a flow visualization in dependence of their velocity. **Abstract/elaborate** enables users to add or remove detail from a representation, e.g., by encoding more or fewer attributes in a glyph, or by visually abstracting a spatial representation. Abstract/elaborate also corresponds to image processing methods that derive new data from visual attributes, such as visibility. **Reconfigure** changes the frame of reference in a representation in order to gain another perspective on the displayed data. The reconfigure intent in the context of the visual component thereby corresponds to modification or re-assigning operations on the visual channels, that we discussed in Section 2.2. Examples for reconfigure are changing the order of axes in a parallel coordinates plot to reveal hidden patterns, or exploding a view in a spatial representation to avoid occlusion.

As an integration source (V→X), the Visual component provides the target representation with information about the mapping or changes thereof, i.e., which data attributes are being mapped to which visual channels. Further, the Visual component can provide derived information from image processing as well. As an integration target (X→V), the Visual component can give visual feedback to interactions in the source representation, e.g., by adapting a representation in respect to the received input information.

Navigation Component

The Navigation component (Fig. 3.2c) is responsible for changing the viewing position and/or direction on the visualized data. It is simultaneously concerned with the questions of the Data and the Visual components, i.e., what to see (view port) and how to look at it (viewing distance, and angle) but without directly affecting the Data or Visual components. Especially in spatial representations, Navigation is an essential component, as due to the size of a scene or due to the (self-)occlusion of objects not all relevant data can be displayed simultaneously.

Navigation of a visualization supports the user intents of *explore* and *reconfigure*. **Explore** corresponds to interactions like panning on a large graph representation, or flying through a 3D scene. **Reconfigure** corresponds to view rotations in spatial representations, e.g., rearranging the view on a volume through rotation.

Navigation as an integration source (N→X) can supply the target components with updates about the viewing position or direction. This information can then be used, for instance, to steer data aggregation or the visual level of detail. Navigation as an integration target (X→N) can update the viewing distance or direction in relation to information supplied by the source representation, for instance, by transforming the camera so that it captures specified data items.

3.4 Integration Techniques

The pairwise integration of the three visualization components yields nine types of integration techniques that we will present in the following. We cluster the integration of visualization components into three categories: Data Operations, Data Indication, and Visual Consistency.

The technique descriptions are based on state-of-the-art examples that we encountered during our literature research. The specific examples serve to illustrate the specific techniques. However, they do by no means represent a complete listing of all potential permutations of the presented techniques. Instead, the descriptions should provide the reader with the knowledge on how to devise the required type of integration for a given task. In this work, we focus on the discussion of examples for integration techniques between heterogeneous, i.e., spatial and non-spatial, representation forms. However, each technique that we present in this section remains also valid for the integration of homogeneous representation forms.

3.4.1 Data Operations

Data Operations describe all integration types that affect the Data component in the target representation. Data operations can be categorized into data manipulation, i.e., selection and filtering, data derivation, and data aggregation.

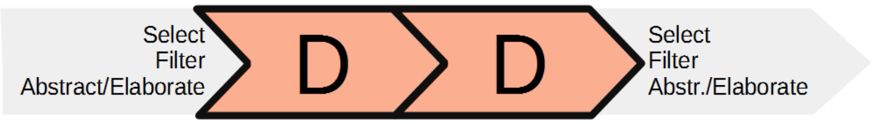


Figure 3.3: A functional link between two Data components can reflect data operations across integrated representations, i.e., by triggering selection, filter, aggregation (abstract), and derivation (elaborate) operations in respect to interactions on the data.

D→D enables techniques where operations on the Data component in the source representation can be reflected on the Data component in the target representation. Figure 3.3 displays the user intents that can be linked by this type of integration.

Data manipulation: User interactions on the data in the source representation determine which data items are loaded or filtered in the target representation. The visualization of traffic trajectory data by Wang et al. [WYL$^+$14] lets users load additional traffic streams into the spatial view by brushing the non-spatial representations in a histogram ($D_a \rightarrow D_s$). The tool for geographical data analysis by Turkay et al. [TSH$^+$14] lets the user issue spatial queries by drawing a path on a map. non-spatial data that correspond to the path positions is then loaded into a graph matrix for further analysis ($D_s \rightarrow D_a$).

Data aggregation: User interactions on the data in the source representation determine the level of granularity at which data is processed in the target representation. In the visualization of fiber tracts by Jianu et al. [JDL09], the manipulation of clusters in a dendrogram is reflected in the level of aggregation of 3D fiber tracts ($D_a \rightarrow D_s$).

Data derivation: Data that the user interacts with in the source representation is used for the derivation of new data in the target representation. In their visualization of mobility in public transportation systems, Zeng et al. [ZFA$^+$14] generate an isoflow tree representation of traffic data (non-spatial) through selection of a spatial starting point on a traffic map ($D_s \rightarrow D_a$).

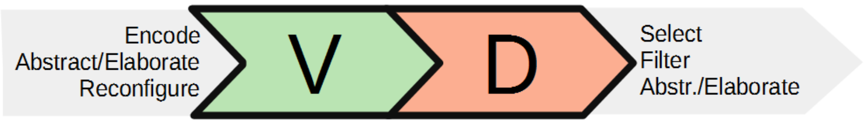


Figure 3.4: The Visual component can supply information about attribute mappings to visual channels (encode, reconfigure), as well as aggregated and derived visual information (abstract/elaborate) to the Data component.

V→D enables the Data component of the target representation to use the Visual component as an information source (Fig. 3.4).

Data aggregation: Bruckner and Möller [BM10] developed a tool for *visual parameter steering* that supports artists in designing complex visual effects based on particle simulations, such as fire or smoke. After running numerous particle simulations, the results are clustered based on their visual appearance ($V_s \rightarrow D_a$).

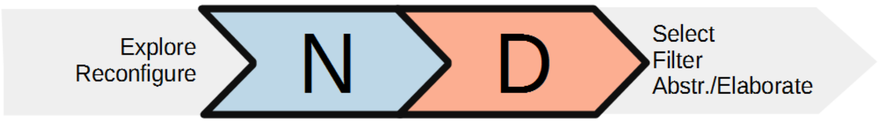


Figure 3.5: A functional link between the Navigation component and the Data component supports the steering of data in the target representation based on the camera position and orientation in the source representation. Navigation thereby determines which data items are situated within the view frustum (explore), and from which distance/angle they are inspected (reconfigure). This information can be used by the target representation to select or filter corresponding items, or to abstract/elaborate the data, e.g., based on the distance.

N→D: enables techniques where navigation in the source representation can steer the Data component in the target representation. Data in relation to positional or directional updates, can be selected, filtered, abstracted, or elaborated in the target representation (Fig. 3.5).

Data aggregation: Chang et al. [CWK⁺07] use navigation in a 3D view to control the clustering of demographic data, which in return is visualized in a matrix view and a parallel coordinates plot ($N_s \rightarrow D_a$). This type of integration occurs often in combination with N→V (see Sections 3.4.2 and 3.4.4).

3.4.2 Data Indication

Data indication encompasses integration types that highlight (indicate) data objects in the target representation that are related to interactions in the source representation. This indication of related information facilitates orientation between visually integrated representations.

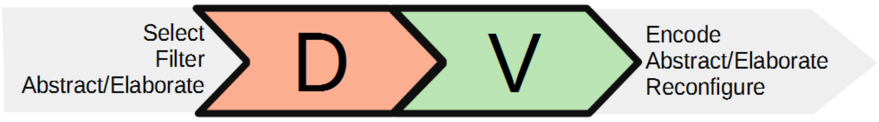


Figure 3.6: A functional link between the Data component and the Visual component allows the target representation to visually react to interactions on the data in the source representation. The target representation can highlight selected data (encode), change the visual level of detail (abstract/elaborate), or change the mapping of attributes to visual channels (reconfigure).

D→V encompasses techniques where operations on the data in one view are visually indicated in another view (Fig. 3.6). Brushing & linking is a common example for this type of integration. In WEAVE [GRW⁺00], for instance, non-spatial representations are used to highlight features in 3D volumes. The volume can also be brushed for highlighting the corresponding data point in the non-spatial views ($D_{a/s} \rightarrow V_{s/a}$).

Since in complex 3D visualizations objects can be occluded or lie outside the view frustum, it can be challenging for users to locate the 3D representations of data items that have been selected in an non-spatial view. Berge et al. [SzBBKN14], for instance, visually abstract the 3D spatial representation in order to make volume segments visible after they have been selected in the non-spatial representation ($D_a \to V_s$).

Another common application of $D \to V$ is mapping data values to visual channels. For many applications, $D_a \to V_s$ is essential to explore the spatial distribution of non-spatial values. In scientific visualization, for instance, it is common to use a transfer function for mapping non-spatial attributes, such as the temperature in the volume data of an engine block, to color and transparency ($D_a \to V_s$) [MFNF01]. Jianu et al. [JDL09] encode the spatial similarity of brain fiber tracts in a color that is shared among spatial and non-spatial fiber tract representations ($D_s \to V_a$, as well as $D_s \to V_s$)).

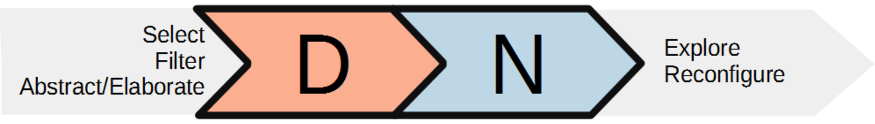


Figure 3.7: A functional link between the Data component and the Navigation component enables guided navigation in the target representation in respect to, e.g., selected, parts of the data in the source representation.

D→N encompasses techniques where the data component supplies the navigation component with information, e.g., about which objects are selected (Fig. 3.7). The navigation component in the target representation then transforms the selected objects into the view, in the sense of *guided navigation*. Guided navigation can help to alleviate issues of localization and occlusion in both representations. A technique by Viola et al. [VFSG06] selects an optimal viewpoint from pre-computed camera positions for a specified volume segment ($D_a \to N_s$).

In the context of occlusion, and the question of whether a chosen perspective on a 3D data representation is meaningful, finding a proper metric for determining viewpoint optimality is a challenge in 3D spatial data visualization. For non-spatial representations, the situation is simpler. In BrainGazer [BSG$^+$09], for instance, a list view automatically scrolls to the entry of a segment that has been selected in the 3D view ($D_s \to N_a$).

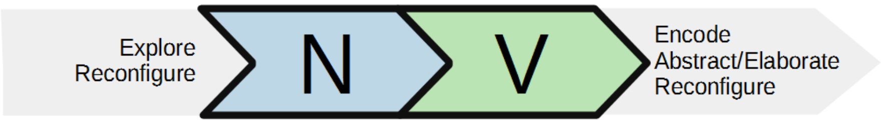


Figure 3.8: $N \rightarrow V$ supports the adaption of visual encodings in the target representation in respect to the camera position, and orientation in the source representation. Navigation thereby determines which data items are situated within the view frustum (explore), and from which distance/angle they are inspected (reconfigure). This information can be used by the target representation to highlight items in the view (encode), or to change the visual level of detail (abstract/elaborate), e.g., in respect to the distance.

N→V can change the displayed visual information based on user navigation (similar to $N \rightarrow D$) (Fig. 3.8). In their comparative blood flow visualization, van Pelt et al. [vPGL$^+$14] annotate the spatial vessel representation with non-spatial glyphs about local blood flow information. Zooming in the spatial view changes the type of representation in the annotations according to the available screen space or to the distance between the camera and the vessel ($N_s \rightarrow V_a$).

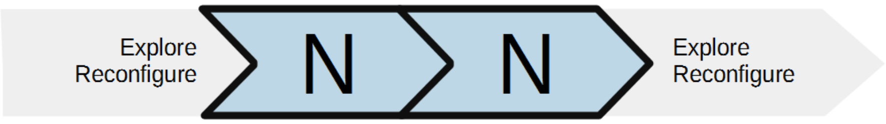


Figure 3.9: A functional link between two Navigation components enables navigational slaving between integrated representations.

N→N corresponds to synchronized navigation, i.e., navigational slaving [WBWK00], across representations. It allows users to simultaneously explore data in both representations (Fig. 3.9).

To enable an $N \rightarrow N$ integration, the respective representations either need to have at least one common dimension in their spatial frame of reference, or a function that maps transitions along a dimension in one representation to transitions along a dimension in the other representation. In *Biopsy Planner* [HMP$^+$12], for instance, users can specify a needle pathway into the brain. A line graph shows the distance to the closest blood vessel along this needle pathway. Consequently, the slice views that displays the pathway and the line graph share a dimension. Users can navigate along the x-axis of the line graph, which adapts the slicing position of the slice views ($N_a \rightarrow N_s$).

V→N encompasses techniques where information from the Visual component is used for steering the Navigation component (Fig. 3.10). For this type of integration, we found no example in the literature that would integrate a spatial with a non-spatial component. Viola et al. [VFSG06] apply this technique for spatial-to-spatial integration in volume visualization by transforming the camera to the optimal viewing position of an occluded object ($V_s \rightarrow N_s$).

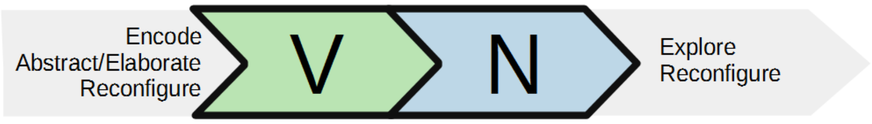


Figure 3.10: A functional link between the Visual and the Navigation component enables the adaption of the camera position and orientation in respect to visual information, such as visibility.

3.4.3 Visual Consistency

Visual Consistency encompasses integration types that support the *Rule of Consistency*, i.e., *"Make the interfaces for multiple views consistent, and make the states of multiple views consistent."* [WBWK00]. Wang Baldonado et al. state that visual consistency facilitates the use of coordinated multiple views by making comparisons easier.

V→V enables techniques that visually link items from the source representation to their counterparts in the target representation, e.g., by using the same visual mapping for the same data attributes in both representations (Fig. 3.11). An example is a focus-and-context visualization [SzBBKN14] where volume segments and their non-spatial counterparts share the same color across representations ($V_{a/s} \rightarrow V_{s/a}$).

V→V also describes certain types of nesting, i.e., when rendered (screen space) information from the source representation is mapped to the visual marks in the target representation. In a visualization of sparse traffic trajectory data [WYL$^+$14], overlays with traffic flow data are displayed in the spatial context of a 2D street map ($V_a \rightarrow V_s$). NeuroLines [AABS$^+$14] displays 3D volume renderings as annotations for a non-spatial neural pathway representation ($V_s \rightarrow V_a$).

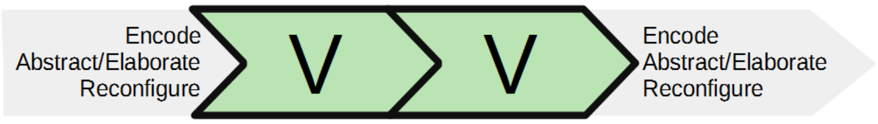


Figure 3.11: A functional link between two Visual components enables the synchronization of visual information between integrated representations, as well as the incorporation of visual information from the source representation.

3.4.4 Integration of Multiple Components

In some scenarios it makes sense that an interaction with a component has multiple parallel or sequential effects, i.e., the integration of multiple components is required. An example for *parallel integration* is integrated semantic zooming. Here, the distance to a data object determines the aggregation ($N_s \rightarrow D_a$) and the visual mapping ($N_s \rightarrow V_a$) of the data representations in a target representation (see for instance, Chang et al. [CWK$^+$07]). *Sequential integration* describes a chain of simple integrations where each target component becomes the source for the next component in the chain. On a technical level, brushing&linking could for instance be described as a sequence of $D_a \rightarrow V_a \rightarrow V_s$ or $D_a \rightarrow D_s \rightarrow V_s$, depending on the respective implementation. A data item is selected, the item is visually highlighted in the same representation ($D_a \rightarrow V_a$), and the highlighting is linked with another representation ($V_a \rightarrow V_s$).

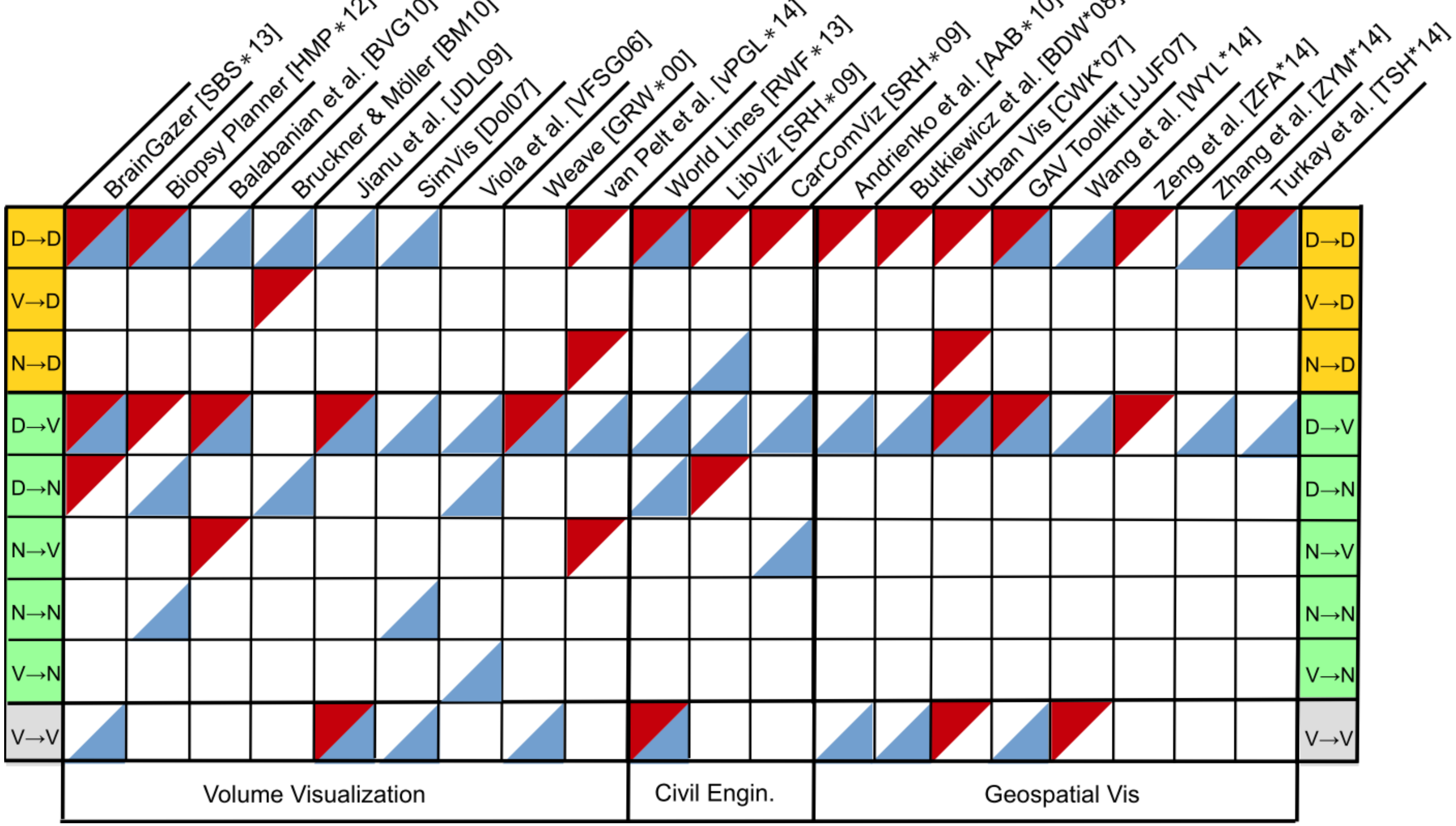
3.5 Discussion

3.5.1 Model Application

In Table 3.1 we compare the integration techniques that we derived from the cited literature. The different publications, listed as columns, are grouped by field - Scientific Visualization, Civil Engineering, and Geospatial Visualization. The rows represent the nine integration types, which are grouped according to the categorization discussed in Section 3.4: Data Manipulation (yellow), Data Indication (green), Visual Consistency (gray). To further illustrate the applicability of our model, we analyze two systems as representative examples for the two dominant integration directions.

SimVis [Dol07] is a framework for the interactive visual analysis of large, multi-dimensional flow data that result from Computational Fluid Dynamics (CFD) simulations. Multiple non-spatial views, such as scatterplots, histograms, or parallel coordinates, enable the exploration of the simulated flow attributes. The non-spatial views are linked to a spatial view that displays a three-dimensional representation of the flow data. In SimVis, the non-spatial views are used to explore the spatial view, i.e., to highlight patterns in the volume data that could be of interest. The dominance of interaction originating from the non-spatial representation is also reflected in the relation of blue to red cells in the respective column of Table 3.1.

UrbanVis [CWK$^+$07] is a visualization tool for the exploration of multi-dimensional data in an urban context. The tool provides a 3D view for the spatial exploration of an urban environment, and a non-spatial view for exploring the multi-dimensional information that is associated with spatial clusters in the 3D scene. The scenario here is the opposite case to the previous example, i.e., the spatial view is used to explore the non-spatial data. This manifests itself in a 4:1 ratio of red to blue cells in the respective column in Table 3.1. Interaction in the spatial view determines, what data ($\rightarrow D$) is chosen and how ($\rightarrow V$) it is displayed in the non-spatial view. The non-spatial view itself can then be explored independently, with no implications on the spatial representation.



	BrainGazer [SBS*13]	Biopsy Planner [HMP*12]	Balabanian et al. [BVG10]	Bruckner & Möller [BM10]	Jianu et al. [JDL09]	SimVis [Dol07]	Viola et al. [VFSG06]	Weave [GRW*00]	van Pelt et al. [vPGL*14]	World Lines [RWF*13]	LibViz [SRH*09]	CarComViz [SRH*09]	Andrienko et al. [AAB*10]	Butkiewicz et al. [BDW*08]	Urban Vis [CWK*07]	GAV Toolkit [JJJF07]	Wang et al. [WYL*14]	Zeng et al. [ZFA*14]	Zhang et al. [ZYM*14]	Turkay et al. [TSH*14]	
D→D	🟥🟦	🟥🟦	🟦	🟦	🟦	🟦			🟥	🟥🟦	🟥	🟥	🟥	🟥	🟥	🟥🟦	🟦	🟥	🟦	🟥🟦	D→D
V→D				🟥																	V→D
N→D									🟥		🟦				🟥						N→D
D→V	🟥🟦	🟥	🟥🟦		🟥🟦	🟦	🟦	🟥🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟥🟦	🟥🟦	🟦	🟥	🟦	🟦	D→V
D→N	🟥	🟦		🟦			🟦			🟦	🟥										D→N
N→V			🟥						🟥			🟦									N→V
N→N		🟦				🟦															N→N
V→N							🟦														V→N
V→V	🟦				🟥🟦	🟦		🟦		🟥🟦			🟦	🟦	🟥	🟦	🟥				V→V
	Volume Visualization									Civil Engin.			Geospatial Vis								

Table 3.1: A comparison of integrated interactions derived from the cited literature in this chapter. The different publications, listed as columns, are grouped by field. The rows represent nine types of integration, which are grouped according to the categorization shown in Section 3.4: Data Manipulation (yellow), Data Indication (green), Visual Consistency (gray). Blue arrows designate non-spatial-to-spatial, and red arrows spatial-to-non-spatial integration.

Integration Patterns

The dominant integration direction in the discussed applications depends on the task that a user should be able to achieve. The task also determines the role that the source and target representations will assume.

In SimVis, the task is to analyze simulation results, e.g., in order to find anomalies in the measurements. The non-spatial source representation is used for interactive data exploration, while the spatial target representation gives visual feedback. In Table 3.1, this is reflected in the dominance of non-spatial-to-spatial integration types (blue triangles) for volume visualization applications. Brushing&linking in volume visualization falls into this pattern. Here, non-spatial representations, such as histograms of intensity distributions, are used to select certain attribute ranges in order to change the mapping of the corresponding 3D spatial glyphs.

In UrbanVis, the user's task is to analyze census data in order to draw conclusions about relationships between living conditions and locations. The spatial source representation is used for the dynamic extraction and derivation of data, whereas the target representation is responsible for providing contextual detail that supports the analysis of the derived non-spatial information. This pattern is employed in applications that emphasize spatial exploration. Here the non-spatial view holds additional information about user defined regions in the spatial scene. This results in a strong integration from the spatial to the non-spatial representation, as it can be found in applications from civil engineering and geospatial visualization. In Table 3.1, this is reflected in the first row, by the strong spatial-to-non-spatial integration (red triangles) for direct data-to-data integrations ($D \rightarrow D$) in the categories of civil engineering and geospatial visualizations.

Both of these patterns enable a *feedback loop* [War00], in one case from the non-spatial to the spatial representation and in the other case from the spatial to the non-spatial representation. One representation gives thereby feedback to the interactions in the other representation and enables an iterative refinement of operations on the data. If integration techniques are applied in both directions, we speak of *balanced integration* across representations. Integration techniques thereby complement each other and enable users to explore the spatial and the non-spatial representation in a back and forth fashion.

Visualization Components

The suitability of a visualization component as an integration source or an integration target depends on the component's input and output capabilities, i.e., the user intents that a component supports in terms of interaction, and the information that a component can receive, process, and output in order to support a user intent.

Since the user typically interacts directly with the data ($D \rightarrow X$) or navigates a view on the data ($N \rightarrow X$), D and N are a very common integration source. Interactions on the data prevail in Table 3.1 ($D \rightarrow D$ and $D \rightarrow V$). From a technical perspective, it does not matter if the input information that a component receives comes directly from a user's interaction or from the output of another component. A component that is a suitable integration source therefore also represents a suitable integration target.

The Visual component is well suited as an integration target since it can transform the incoming information into visual feedback. Table 3.1 clearly shows the strong occurrence of integration techniques with the Visual component as an integration target, especially for interactions on the Data component in the source representation ($D \rightarrow V$).

Direct interactions with the Visual component as integration source, however, are less frequent, except for $V \rightarrow V$. Here, typically a change in the visual mapping is synchronized between representations, for instance, when trying to reveal patterns through manipulation of the mapping via transfer functions. The output of the Visual component, however, is not easily transformed into input for other components. A suitable integration target therefore is not automatically a suitable integration source. The goal of a system designer should be to pick the type of integration that supports the user intents required for fulfilling a given task.

3.5.2 Model Validation

Beaudouin-Lafon describes three metrics by which interaction models can be evaluated [BL04]: descriptive power, evaluative power, and generative power.

We argue that **descriptive power** is given, since our taxonomy's model is based on the general visualization pipeline. This means the taxonomy can describe the integration of visualization components regardless of the involved representation forms. Moreover, the taxonomy is not restricted to any particular application domain but may be applied to all types of data, for example, from computational fluid dynamics, or urban planning.

With the model application in Section 3.5.1, we aim to demonstrate that **evaluative power** is given. Different systems can be compared by analyzing their integration patterns. Alternative implementations can be suggested after identifying the user intents that are involved in fulfilling a certain task as well as the compatible integration types.

We argue that with the disclosure of the integration design space, **generative power** is given, as well. By describing the properties of individual visualization components in each representation form as well as how they can be combined with each other, the design space is revealed to the visualization designer.

3.6 Conclusion and Outlook

The integration of representations enables efficient exploration of multifaceted spatial data. In this chapter, we presented a model that describes the design space resulting from the combination of visualization components from spatial and non-spatial representations. Our taxonomy can be applied to draw conclusions and identify patterns and correlations across visually integrated representations.

While we focused on the integration of heterogeneous representation forms, our model could well be applied to visually integrated representations in general. In terms of future research perspectives, it would be interesting to apply our taxonomy to a broader suite of literature in the scope of a state-of-the-art report. By providing a stable base for the development of novel ideas, we hope to contribute to the understanding of this branch of research that, while finding more and more applications in today's scientific community, has never been discussed and structured on a detailed enough level.

By concluding this chapter, we reached the end of Part I of this thesis. So far, we equipped the reader with the knowledge about the various ways in which multiple representations can be visually composed in order to convey the information required to fulfill a task in the most suitable visual context. Subsequently, we described how interactions or events can be functionally linked across these visually composed representations in order to build a coherent visualization system. In Part II of this thesis, we now move on to apply this knowledge to concrete application scenarios for exploration and presentation tasks.

Part II

Applications

CHAPTER 4

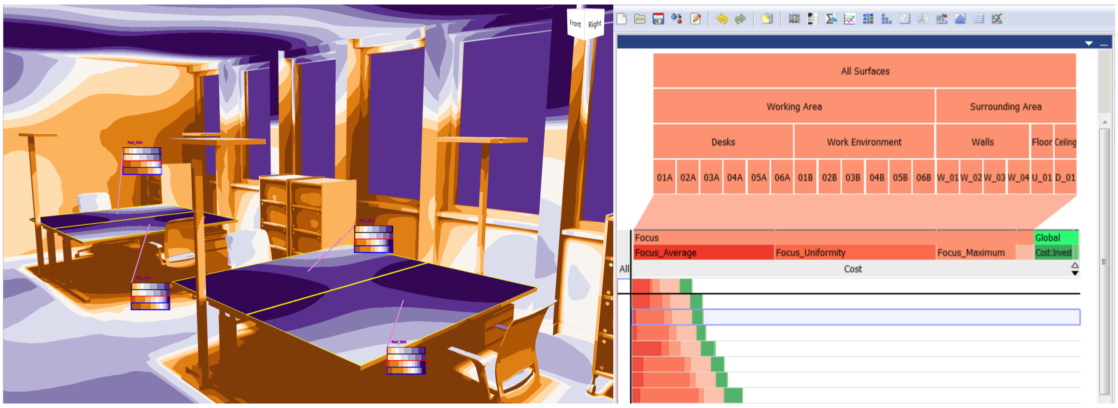
Integrating Spatial and Non-Spatial Data Facets in Parameter Space Exploration

For Exploratory Visualization in Lighting Design

This chapter is based on the following publication:

Johannes Sorger, Thomas Ortner, Christian Luksch, Michael Schwärzler, Eduard Gröller, and Harald Piringer. *LiteVis: Integrated Visualization for Simulation-Based Decision Support in Lighting Design.* IEEE Transactions on Visualization and Computer Graphics, 22(1):290-299, January 2016 [SOL$^+$16].

THE first integration application scenario takes us to the domain of lighting design. State-of-the-art lighting design is based on physically accurate lighting simulations of scenes such as museums or offices. Simulation outcomes support lighting designers in the creation of lighting configurations, which must meet contradicting customer objectives regarding quality and price, while conforming to industry standards. An efficient decision making process requires rapid feedback cycles between simulation specification and analysis, and a detailed comparison of multiple configurations. The spatial context of the simulation, as well as non-spatial representations that enable a quantitative comparison of simulation results are essential for a proper decision making process. Multiple visual integration strategies and a tight functional integration are required to bring the representations of the multifaceted simulation data together in a system that efficiently enables this task.



(a) (b)

Figure 4.1: Comparing lighting parametrizations according to different metrics in the LiteVis workspace. The Simulation View (a) displays a false color rendering encoding illumination quality. For each desk, floating annotations show a binned histogram of illumination values from selected simulation runs. The Simulation Ranking View (b) shows a ranking of simulated lighting parametrizations according to user defined importance values for spatial measurement surfaces and abstracted result indicators.

In the following Section 4.1, we regard the challenges that emerge in the application setting of decision making in lighting design and give an overview of the contributions of this chapter. We discuss related approaches that address specific challenges for parameter space exploration and multi-objective optimization tasks in Section 4.2. In Section 4.3, we analyze the data and the tasks that are involved in finding an optimal solution, before presenting our specific approach to solving the identified challenges in Section 4.4. We demonstrate the applicability of our approach in Section 4.6 based on a use case scenario. Finally, we evaluate our solution based on user feedback from lighting design experts as well as reflections on the design process in Section 4.7.

4.1 Introduction

Architectural lighting design has a strong impact on the atmosphere and aesthetics of buildings and open spaces. The illumination also affects how working environments support productivity and creativity. However, finding an acceptable trade-off between often contradictory customer requests regarding the quality and price of a lighting setup while conforming to industry standards is a challenging task. Physically accurate lighting simulations have long been used to support lighting designers in planning and communicating potential solutions to customers. Given a 3D geometric model of the scene as well as the position, type, and additional properties of the involved luminaries, a simulation computes the illuminance for each part of the scene.

Commercial software for lighting design [Relb, Rela, Lig] typically takes up to several hours to compute the result of a single setup, and focuses on inspecting a single solution rather than comparing many solutions. As a consequence, the workflow in lighting design has traditionally been restricted to computing a small number of alternative setups in a trial-and-error fashion. As an additional challenge, the tools that are required for analyzing aesthetic and financial aspects of a given lighting solution [Zum] are neither visually nor functionally integrated with each other. How fast a convincing solution could be found, has therefore been highly dependent on the experience of the lighting designer.

Recent advances in lighting simulation reduced the effort for computing a physically accurate illumination to a few seconds [LTH+13]. Technologically, this enables significant improvements in the workflow of lighting designers. First, it is now possible to examine the design space much more systematically and comprehensively by computing hundreds of potential solutions as a starting point of the design process. Second, lighting designers may now define and compute additional setups on-the-fly, e.g., during a workshop with a customer.

However, existing software tools in lighting design do not support these new possibilities well. For this reason, we conducted a design study in collaboration with lighting design experts. The result is LiteVis, a system for efficient decision support in lighting design. In contrast to existing tools, a focus of LiteVis is the comprehensive comparison of multiple solutions. LiteVis tightly integrates global illumination-based lighting simulation, a spatial representation of the 3D scene, as well as non-spatial representations of setup parameters and result indicators, as depicted in Figure 4.1. Specifically, LiteVis relies on hierarchically structured measurement surfaces in the 3D scene for defining quantitative quality indicators (Fig. 4.1b). Weighting these spatial indicators along with non-spatial indicators, such as monetary costs, supports a holistic view of all relevant decision factors while making the decision maker's preference explicit and reproducible.

The *contributions* of this chapter can be summarized as follows:

- A design study of decision making in lighting design resulting in the system LiteVis.
- A problem characterization of the application domain of lighting design, including an abstraction that relates the application problem to other areas of simulation-based decision making.
- A novel visualization for ranking multiple decision options based on weighting hierarchically structured spatial and non-spatial objectives.
- A report on feedback from domain experts and a reflection on the design process.

4.2 Related Work

4.2.1 Lighting Design

In the scientific community, several approaches to semi-automatic or user-guided, interactive lighting design have been proposed, e.g., sketch-based methods, where the user "paints" the scene parts to be lit [SDS+93, OMSI07, SC07, PBMF07, LHH+13], as well as procedural methods [SW14]. Additionally, the direct specification of lighting-induced features, such as shadows or highlights [PF92, PTG02], have been suggested to trigger the generation of an ideal lighting solution. While all these approaches tackle the problem in plausible ways, they have not found their way into today's industry standard lighting design tools. Potential reasons are that some constraints are hard to express mathematically (e.g., regarding aesthetics), or the wish for artistic freedom. Glaser et al. [GTCD03] have approached the lighting design problem by developing various 2D visualization prototypes, inspiring our work in terms of analytical aspects, but without integrating spatial 3D and non-spatial 2D views within a common tool.

Commercial lighting design applications, on the other hand, leave the task of placing and adjusting the light sources in a scene to the skills of a lighting designer. Tools, such as Relux [Relb], Dialux [Rela], Agi32 [Lig], rely on Radiosity and/or Raytracing-based simulation kernels. They visualize the simulated results on measurement areas placed in the 3D scene using false color visualizations. However, none of them supports the user in the comparison of different solutions. Separate tools, such as ecoCALC [Zum], tackle related problem domains like financial aspects. Even though their respective outputs are needed to form an overall decision, the tools used in the workflow of finding an ideal lighting solution are often isolated from each other, which makes a holistic exploration of the problem space cumbersome.

We base our approach on a lighting modeling system that relies on a simulation kernel recently proposed by Luksch et al. [LTH+13]. By providing a faster lighting simulation as compared to Radiosity-based approaches, it enables shorter cycles of scene parametrization and evaluation.

4.2.2 Integrated Visual Parameter Exploration

The field of visual parameter space exploration has shown significant progress in the last years. Beham et al. [BHGK14] developed a composite visualization that integrates the non-spatial parameter space of geometry generators with the spatial output space of the resulting shapes. Illustrative parallel coordinates allow the user to study the sensitivity of parameters in a global-to-local drill down fashion. Similarly, in LiteVis, the relation between the input space and the output space can be explored. However, in our case, both have spatial and non-spatial properties. Coffey et al. [CLEK13] propose a tool for simulation-based design that provides integration of forward design (input manipulation) and inverse design. Inverse design lets the user query a database of pre-computed samples by specifying the desired simulation output directly in the 3D scene. This specification of goals in the 3D scene shares the same principle as the specification of spatial objectives in LiteVis (see Sec. 4.4.2). Bruckner and Möller [BM10] propose a tool for the exploration of clustered time series data of physical fluid simulations for visual effects design. The simulation result is thereby of primary concern to the user. In contrast to the solution by Bruckner and Möller, we do not have to deal with dynamic scenes but our scenario demands an input- as well as an output-driven approach.

4.2.3 Multi-Objective Decision Making

In lighting design, multiple objectives must be optimized simultaneously. This is a common issue in many application domains. For this reason, multi-objective optimization (MOO) has long been an active field of research (see, e.g., Köksalan et al. [KWZ11] for a survey). As a single best solution does not exist for MOO-problems, one approach is to offer the decision maker multiple solutions that have been (semi-)automatically generated. Miettinen and Mäkelä, for instance, introduced an interactive method called NIMBUS [MM06] that asks the user to repeatedly examine the values of objective functions calculated for a current solution.

There are several approaches for visualizing the Pareto Frontier. The Pareto Frontier is a set of multi-objective optimal solutions. For none of the solutions within the set, the result of a single objective can be improved without worsening the result of another one. Korhonen and Wallenius [KW08] stress that visualizing the Pareto Frontier for more than three objectives is difficult. The authors classify several visualization techniques for multi-criterion decision making based on the cardinality of the result set. Lotov et al. [LBK04] employ scatter-plot matrices to show bi-objective slices of the Pareto Frontier. In order to support decision making, Andrienko and Andrienko [AA01] propose several extensions to parallel coordinates, the prevalent technique for visualizing Pareto Frontiers [BC03]. More recently, Chen et al. [CAS+13] employ self-organizing maps for projecting all Pareto-optimal solutions to a 2D radial visualization. However, most of these approaches do not scale to a very high number of objectives, e.g., up to hundreds in lighting design if spatial aspects are taken into account (see Sec. 4.3.1). Moreover, inspecting the entire set of Pareto-optimal solutions can still be inefficient.

A common approach is therefore to weight the objectives in order to attain a score, which can then be used for ranking possible solutions for decision making. LineUp [GLG+13] is an interactive approach for weighting multiple objectives and visualizing the resulting ranking. While LineUp inspired our approach for decision support, we had to adapt our ranking approach to the special requirements that arise from the hierarchical structure of spatial objectives in LiteVis, as we will describe in Section 4.4.2. As an additional problem for most approaches in lighting design, each solution refers to the illumination of an entire scene and thus has complex characteristics, such as aesthetics, which are hard to quantify.

4.3 Lighting Design Background

This section provides a brief introduction to the field of lighting design as far as necessary for understanding the design decisions that went into LiteVis. This information is based on experiences gained from a tight collaboration with experts in lighting design for nearly five years. After describing the data and the tasks, we motivate the key design goals of LiteVis. In a problem abstraction, we put the domain specific challenges into a broader context.

4.3.1 The Data: Simulation in Lighting Design

Modern lighting design is based on lighting simulations which compute an output illumination for a particular scene and luminary setup. As for many simulation types, the *input* of a lighting simulation can be classified as control parameters, environmental parameters, and model parameters. We refer to a particular assignment of values to all parameters that are necessary for starting a simulation as *parametrization.* We refer to a set of a parametrization and the resulting simulation output as *solution* or *simulation run.*

The *control parameters* in lighting design, are *luminaries* and their position and orientation in the scene. They are placed by designers to achieve certain goals regarding the appearance and budgetary constraints of a solution. Luminaries are placed using controls similar to those in geometric modeling tools. Each luminary has several parameters. The most important ones in the context of this work are:

- *Type*: Type corresponds to a particular product of a manufacturer, which is typically classified by the application as, e.g., floor lamps, ceiling lamps, or wall mounted lamps.
- *Wattage*: This parameter describes the illumination power and energy consumption of a luminary.
- *Dim profile*: A luminary can be dimmed, resulting in lower power consumption and increased longevity, but also lower performance.

Environmental parameters describe aspects of the simulation which are not directly controlled by the user or vary over time:

- *Scene*: Contains the 3D geometry of the scene such as desks, windows, walls, etc. Geometry can absorb and reflect light based on material properties, and also cast shadows, i.e., as an occluder.
- *Environmental conditions*: Such conditions determine external influences, like sunshine that illuminates a scene based on the time of day/year and the weather condition. These factors can optionally be included in the simulation.

Model parameters are the implicit parameters of the lighting simulation algorithm responsible for transforming the simulation input into the simulation output. Model parameters typically define a trade-off between accuracy and speed. An example is the number of times the light bounces in the calculation of the indirect illumination. These parameters are not of direct concern to the lighting designer and therefore play a minor role in the scope of this work.

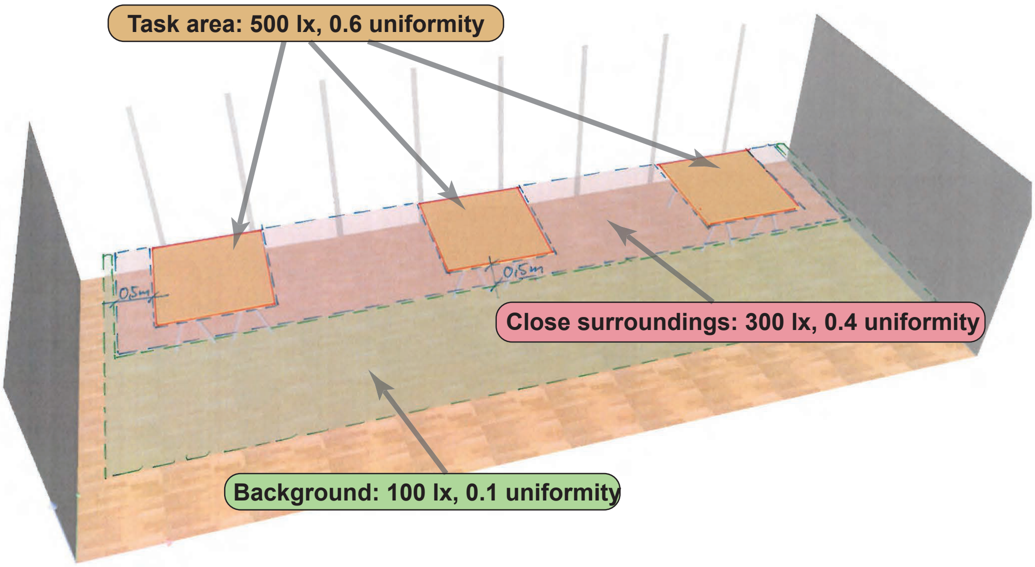

Task area: 500 lx, 0.6 uniformity
0,5m
0,5m
Close surroundings: 300 lx, 0.4 uniformity
Background: 100 lx, 0.1 uniformity

Figure 4.2: A conceptual sketch created by a lighting design expert, defining the location of the measurement surfaces and their target values in an office scene according to industry standards [DIN11].

The output of the lighting simulation comprises an illuminance value (measured in lux) for each texel in the scene. Using this direct output for 3D rendering of the scene is suitable for a qualitative inspection of the results. However, for most tasks, the direct simulation output is too detailed. The standard approach in lighting design for condensing the data and assessing the compliance to industry standards is based on pre-defined *measurement surfaces* on top of the geometry. Each measurement surface corresponds to a semantic part of the scene such as a particular desk, a certain part of the wall, a door, etc. A scene typically comprises multiple types of surfaces, which are structured hierarchically. For example, a top-down classification of a particular measurement surface could be "office – working area – all desks – desk 3".

Measurement surfaces aggregate the illuminance of their geometry to define quantitative *local indicators*. The most important local indicators in lighting design are the *minimal*, the *maximal*, and the *average illumination*, as well as *uniformity*. Uniformity is defined as the ratio of the minimal illumination to the average illumination and describes the evenness of the distribution of illuminance values on a surface.

It is common to define target values for these local indicators per measurement surface and to assess the suitability of a solution in terms of the difference to the respective target values. Different classes of measurement surfaces are subject to varying degrees of constraints regarding the illumination quality, which is reflected in different target values for each class (see Fig. 4.2). In addition to a customer-driven specification of target values, several classes of measurement areas, such as desks, must also meet industry norms and standards [DIN11].

This description of lighting simulations in the context of lighting design is generally applicable to commercial systems as well as to the underlying simulation in LiteVis [LTH+13]. In all cases, the simulation kernels of various software tools involve global illumination techniques. In commercial systems, the effort for computing the illumination of a single parametrization takes up to several hours, depending on the complexity of the scene and the configuration of the luminaries. Using the GPU and clustering virtual point lights into a set of virtual polygon lights, the underlying simulation in LiteVis, however, reduces the effort for obtaining comparable results to a few seconds. We refer to Luksch et al. [LTH+13] for additional technical details of the lighting simulation.

Besides the raw simulation output and the local result indicators that are derived from this output, there are *global indicators* that are not tied to specific spatial (local) parts of the simulation result. Global indicators of concern to the decision making process are the *investment cost* and the *run-time cost*, i.e., energy consumption of each luminary type. The collective luminary configuration in a parametrization determines these global indicators. The accumulated budgetary values typically represent a significant tradeoff to an excellent illumination quality.

In summary, the key entities of LiteVis are the simulated parametrizations (or simulation runs). Regarding the data model, each parametrization comprises a multivariate facet and a geometric facet. The multivariate facet includes (1) the parameters per luminary such as wattage, (2) environmental variables, such as the simulated time of day, (3) the local indicators per measurement surface and their deviation from the target values, and (4) the global indicators describing monetary cost. The geometric facet of the simulation output describes the spatial properties of the illumination in terms of an illuminance value per texel.

4.3.2 Task Analysis

Our collaboration with experts in lighting design enabled us to identify the following recurring tasks (Fig. 4.3). The subsequent list of tasks is based on insights gained from semi-structured interviews and contextual inquiries [HJ93].

- **T0 - Scene setup**: After the scene is modeled by a 3D artist, measurement surfaces are defined on top of the geometry, e.g., for desks and walls. The designer defines target values and constraints for local result indicators, such as uniformity per measurement surface, in accordance with customer requirements and industry norms.

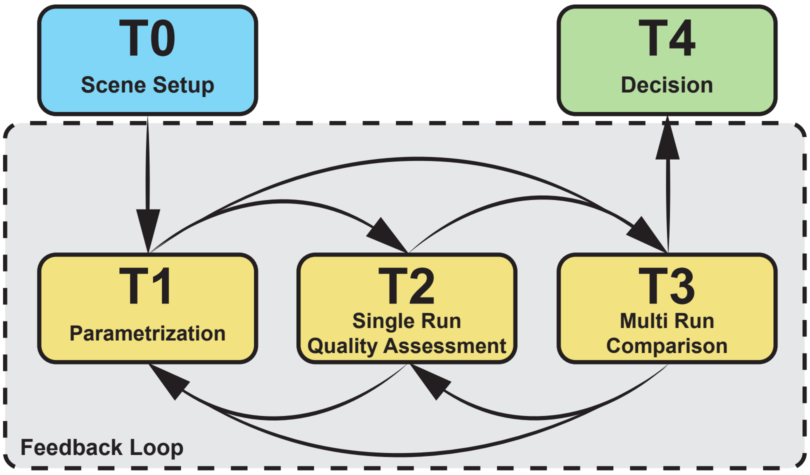


Figure 4.3: The five steps of the lighting designer's workflow. After initialization of the scene (T0), the lighting designer works in an iterative fashion (T1-T3) until he or she finds a satisfactory solution (T4).

- **T1 - Scene parametrization**: The lighting designer defines one or more parametrizations for a subsequent simulation. This particularly includes the placement and parametrization of a specific set of luminaries in the scene according to customer requirements, budget constraints, industry standards, and the designer's experience in order to achieve specific aesthetic effects. The designer may also specify certain environmental conditions, such as the position of the sun. The initial definition of parametrizations is mainly based on the experience and skill of the designer, while later iterations are based on insights from previous solutions. In all cases, the generation of parametrizations may be partly automated based on parameter variations, e.g., for varying values of luminary wattage.
- **T2 - Assessment of single solutions**: Once a simulation has been completed, its qualitative and quantitative output is typically evaluated individually in order to quickly reject inappropriate solutions. For the qualitative evaluation, the aesthetic appearance of the illuminated scene is assessed within its original spatial context in false color renderings that encode, for instance, the strength of shadow gradients. The quantitative evaluation is based on the assessment of result indicators of measurement surfaces, for instance, in relation to the upholding of industry norms or customer defined target values, typically within the context of non-spatial representation forms. The assessment of both spatial and non-spatial representations of the simulation data is therefore essential for fulfilling T2.

- **T3 - Comparison of multiple solutions**: Multiple candidate solutions are compared in detail in order to understand in which regard one is superior. One goal is to further reduce the set of possible candidates by excluding solutions that do not satisfy certain quantitative criteria. Another goal is to identify key trade-offs, e.g., between a certain aesthetic aspect and monetary cost. Traditionally, this task is performed using side-by-side comparisons of false color renderings and the corresponding data sheets – an approach that scales badly for the comparison of more than half a dozen of solutions.
- **T4 - Decision**: In order to arrive at a final decision, it is common to weigh the various result indicators. Qualitative aspects and a detailed comparison of the illumination of multiple solutions still play an important role at this stage. In many cases, this task is performed by the lighting designer together with the customer.

The traditional strategy for decision making in lighting design is based on trial-and-error. As computing a single solution took up to several hours, only very few alternatives were typically considered in a rather sequential manner. In some cases, an initial guess that appeared to be "good enough" was used without further comparison. In other cases, it could take up to several work days to identify an acceptable solution, especially for less experienced designers or very complex scenes.

In LiteVis, the description of these tasks remains valid. However, the lighting simulation underlying LiteVis enables two important improvements to this workflow. First, it enables to initially specify and compute a set of numerous samples in one step within T1. For example, hundreds of variations regarding luminary placements, wattages, and environmental parameters such as external lighting conditions can be simulated automatically over night. This provides a more global understanding of the space of possible solutions at a very early stage of the design process. The understanding supports the fast identification of acceptable designs while avoiding to miss interesting solutions.

Second, the simulation is fast enough to enable the exploration of additional solutions in real-time. The improved feedback loop makes it possible to reach a decision during a single meeting with a customer. Previously, multiple consecutive meetings could have been necessary.

4.3.3 Design Goals of LiteVis

The design goals of LiteVis are motivated by the gap between the state-of-the-art software in lighting design and the new possibilities for improving the workflow that are enabled by the fast simulation.

- **G1 - Integration of the simulation output space**: Currently, separate tools are necessary for the qualitative inspection of the simulated illumination, the quantitative analysis of result indicators, and the examination of budgetary constraints. This separation makes comparing multiple solutions cumbersome. A goal of LiteVis is thus to *integrate* these separate components of the simulation output space within a single framework in order to provide a convenient access to them.

- **G2 - Integration of scene parametrization and evaluation**: The lighting simulation (T0, T1) and the evaluation of the solution (T2, T3, T4) are currently isolated from each other as well, since they are performed with separate tools. The lack of integration between these two components impedes an efficient feedback loop between the creation and the evaluation of a parametrization. Consequently, a goal of LiteVis is to allow the user to easily trigger new simulation runs based on the inspection of prior results, and to display new results immediately when they become available.
- **G3 - Effective comparison of multiple solutions**: The comparison of solutions is essential for defining additional iterations and reaching a final decision. However, current simulation tools in lighting design do not offer the designer appropriate means to understand, compare, and make decisions based on alternatives. The status quo for comparing multiple solutions is to assess the corresponding data sheets side-by-side, which is ineffective for decision making. A goal of LiteVis is to enable a direct comparison of solutions in regard to their qualitative and qualitative aspects, as well as a sensitivity analysis to assess the impact of parameter variations. To support this goal, strategies for the comparison of solutions in a spatial, as well as in a non-spatial context had to be developed.
- **G4 - Explicit and reproducible decision support**: Measurement surface types vary in their target values for illumination indicators and typically carry different semantic importances for upholding these values. For example, the illumination quality on a table surface is more important than on a wall. Current tools consider neither distances from local target values, nor semantic importance factors. Moreover, different stakeholders may put more emphasis either on qualitative or on financial aspects. A key goal of LiteVis is therefore to support reproducible decision making based on an explicit preference specification for illumination-related as well as cost-related indicators. A related goal and a prerequisite is to enable the specification of target values for indicators.

4.3.4 Problem Abstraction

In this sub-section we characterize the domain of simulation-based lighting design based on the conceptual framework for visual parameter space analysis by Sedlmair et al. [SHB$^+$14], in order to offer a more generalized view of the problem space. In the terms of the framework, the lighting simulation is a deterministic computational *input-output model*. Control parameters, environmental parameters, and model parameters are thereby the input (see Sec. 4.3.1) that generates the illumination per texel as *direct output*. The direct simulation output can be considered as a *complex object*. Apart from a qualitative assessment, this complex object requires a derivation step for subsequent decision making. In our context, the derivation is an aggregation of local indicators based on measurement surfaces.

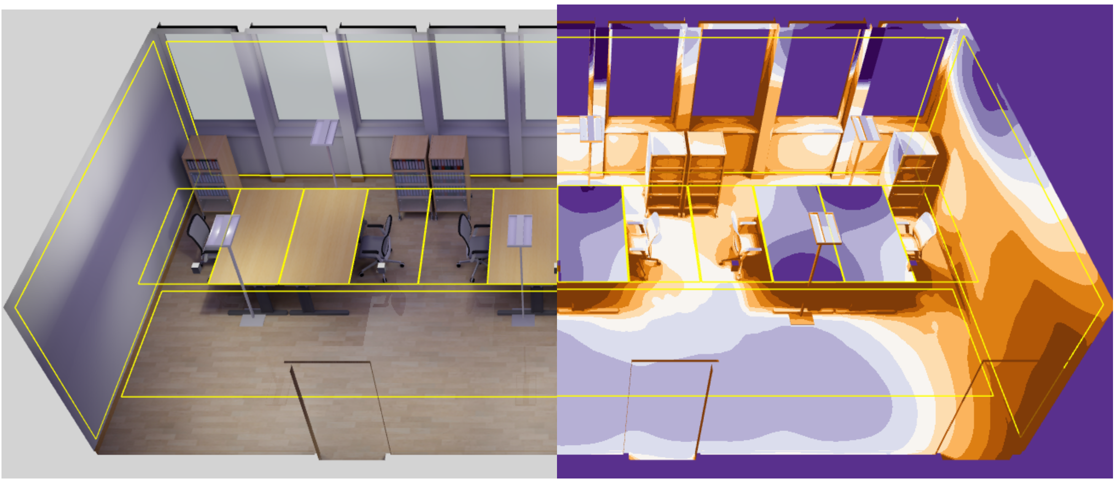

(a) (b)

Figure 4.4: a) The Simulation View displaying luminaries and measurement surfaces (yellow) in an office scene. b) The false color render mode conveys negative and positive distances to a specified illumination target value.

The strategy for navigating the parameter space has traditionally been an *informed trial-and-error* approach. LiteVis supports a shift of the strategy towards a *global-to-local* exploration. The primary task of the lighting designer is the *optimization* of multiple competing objectives corresponding to the indicators for global and local indicators. *Sensitivity analysis* can be of potential interest to lighting designers as well. Due to the hierarchical structure of the measurement surfaces, the corresponding objectives can also be considered hierarchical, which has motivated key design decisions of LiteVis.

4.4 Design Study of LiteVis

4.4.1 System Overview

The design of LiteVis comprises two tightly integrated parts, i.e., the spatial *Simulation View*, and the non-spatial *Analysis Views*. The features that we introduce in this section are demonstrated in motion in the supplemental material video [SOL$^+$].

The Simulation View (Fig. 4.4a) provides a spatial display of the scene. While the 3D geometry of the scene is modeled externally in a 3D modeling tool, the Simulation View supports an interactive definition of measurement surfaces for scene setup (T0). It also supports the parametrization of new simulation runs (T1). To create a parametrization, the user can create luminaries and interactively place their 3D representations in the scene via dragging. The user can inspect and modify luminary parameters, such as the value for wattage. Optionally, the user can specify a value range to define and simulate multiple parametrizations.

After computing simulation results, the Simulation View supports a qualitative assessment of the direct output of a single simulation run (T2) as well as a comparison of multiple runs (T3, see Sec. 4.4.3 and Sec. 4.4.4). As an alternative to rendering the illumination in a realistic way, a visual abstraction of luminance intensities conveys negative and positive distances to the respective target values in the *false color render mode* (Fig. 4.4b). While the comparison of parametrizations in the Simulation View is novel in the lighting designer's workflow, the scene parametrization and false color rendering represent familiar aspects of their routine.

The Analysis Views are a suite of non-spatial representations that enable the comparison, selection, filtering and aggregation of local and global result indicators, as well as the underlying input parameters of all simulation runs. Most importantly, the Simulation Ranking View enables a direct comparison of multiple solutions regarding their overall superiority as defined by a weighting of local and global indicators (see Sec. 4.4.2). Additionally, various well established representation forms, such as parallel coordinates, bar charts, and spreadsheets, may be used to explore the multivariate attributes of the solutions (see for instance Fig. 4.8d).

Integration strategies: in order to tightly couple the Simulation View with the various Analysis Views, we implemented the following integration strategies. Selecting one or more simulated solutions in the Analysis Views loads the corresponding parametrizations and illuminations into the Simulation View for a detailed inspection and spatial comparison. Depending on whether one, two, or more solutions are selected in an Analysis View, different visual integration strategies are triggered in the Simulation View in order to support the inspection of a single solution, the pairwise comparison of two solutions (see Sec. 4.4.4), or the comparison of many solutions (see Sec. 4.4.3). Additionally, the user can create variatons of already simulated parametrizations on the fly, and inspect the differences to the original solution. Creating and simulating a new parametrization in the Simulation View automatically extends the set of solutions in the Analysis Views. Selecting simulation run and measurement surface representations in either view, highlights their respective counterparts in any other view where they are represented. The visual connection of related entities thereby supports the user in maintaining a mental connection between individual scene parts and their respective indicators. These integration strategies enable an efficient feedback loop (G2) between scene parametrization (T1), the assessment of the result (T2), and a comparison of multiple simulation runs in a spatial and an non-spatial visual context (T3).

4.4.2 Simulation Ranking View

The Simulation Ranking View (Fig. 4.5) represents one of the contributions of this work. We designed the view to match the requirements for efficient decision support (G4). The key idea is to rank multiple simulated solutions based on a weighted scoring of local and global result indicators. The user should be able to define an emphasis on spatial as well as non-spatial aspects of the simulation output, i.e., certain surfaces of the scene, as well as on certain global and local result indicators. Furthermore, the user should be able to explore different preference settings by interactively changing the weighting of indicators in order to see the effect on the ranking of the solutions. While fundamental aspects of the design were inspired by LineUp [GLG+13], the number of the involved result indicators as well as the hierarchical relation between them required some significant extensions.

Score Computation

Before explaining the design of the Simulation Ranking View, it is necessary to understand the computation of the scores per solution. In this context, we face three main issues: (1) The score must provide a combined assessment of global indicators as well as local indicators. Each measurement surface yields a value for each type of local indicator. (2) The number of objectives that contribute to the score of each simulation run is potentially large, as it is calculated by $n * m + o$, where n represents the number of measurement surfaces, m local indicators, and o global indicators. Each of the potentially large number of measurement surfaces as well as each of the local and global indicator types may be weighted differently. (3) Each local indicator type may have a different target value on each measurement surface type.

The total score of a solution is defined as the weighted sum of the scores per objective. Objectives can be classified into spatial and non-spatial. *Non-spatial objectives* refer to global indicators, such as the investment cost and the cost per month. For each indicator, a scoring function maps the measured indicator value to a score which ranges from zero to infinity. Our current implementation simply employs linear functions with a user-defined slope that map a cost of zero to a score value of zero. Smaller score values are therefore considered better. In other words, the score value of a particular objective represents the distance of the indicator from a user-defined target value. In the case of monetary costs, this target value is zero, since low monetary costs are desired.

We note that this definition of score values is different from the one in LineUp [GLG+13], which maps each indicator to a bounded score ranging from zero to one, with one being considered as the best score. The main motivation for inverting the scale of the scoring function and for avoiding an upper bound is to enable the expression of constraints. For example, mapping very high score values to investment costs above a certain threshold, will result in a severe penalty for the score of solutions where the indicator value exceeds the threshold.

Spatial objectives refer to the local indicators of measurement surfaces. These indicators are called "local", since they are defined per measurement surface ($n * m$). For example, 18 measurement surfaces and four local indicators define a set of 72 spatial objectives. For each spatial objective, the user may define a separate scoring function. In this case, the user can define linear as well as non-linear scoring functions. For example, a function could assign score values of zero for a certain target range while increasing exponentially with growing distance from that range.

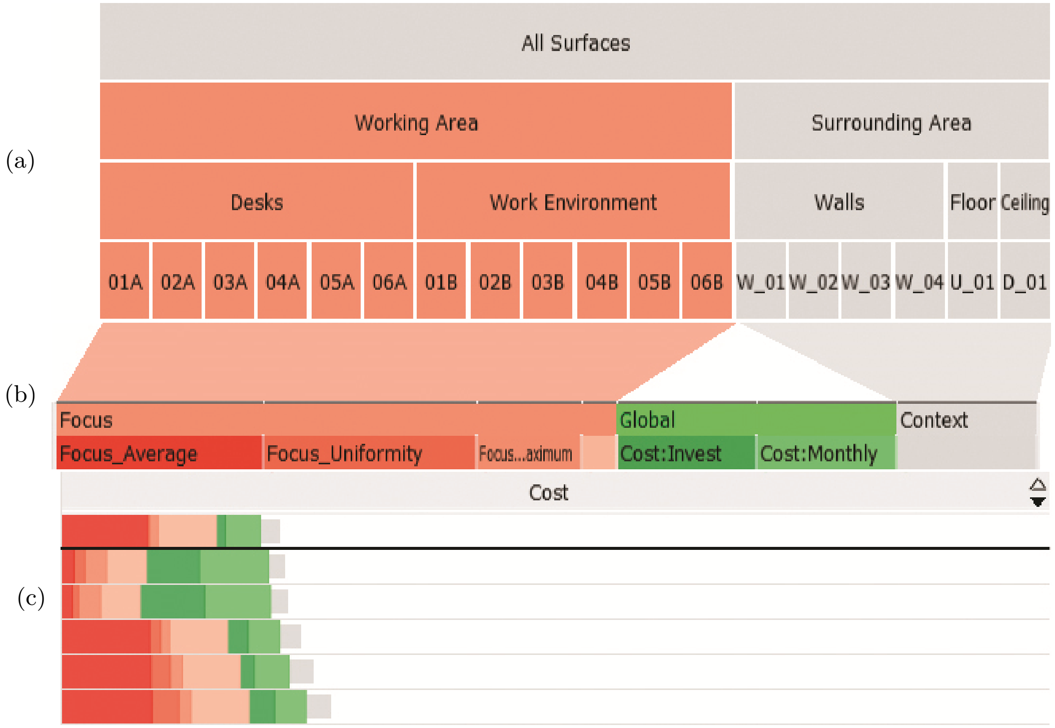


Figure 4.5: The Simulation Ranking View: The Spatial Hierarchy (a) allows the user to specify the individual importance per measurement surface and to select a (focus) subset of surfaces (colored in red) for detailed inspection. The remaining (context) subsets are colored in gray. The Indicator Bar (b) displays the weight per indicator type. It also discriminates between global indicators in green, local indicators (for spatial objectives) in red, and the accumulated weight of the context objectives outside the spatial focus (gray). The Ranked Table (c) displays the individual solutions as stacked bars corresponding to the sum of weighted scores per indicator type. Shorter bars correspond to indicator values that are closer to a user-defined target.

Weighting enables the user to explicitly specify importance factors for spatial and non-spatial objectives. Internally, a separate weight is maintained for each objective and all weights sum up to one. However, exposing several dozens of objectives and corresponding weights to the user is neither intuitive nor effective. We thus define the per-objective weights for spatial objectives implicitly as the product of a *per-indicator-type weight* and a *per-measurement surface weight*. For example, the user may specify a weight of 0.3 for uniformity across all surfaces. For the above mentioned scene with 18 measurement surfaces and four local indicators per surface, the user would have weighting control over $18 + 4 = 22$ objectives (instead of 72 as discussed above). All per-indicator-type weights sum up to one. The per-measurement surface weights sum up to one, as well. For example, a per-indicator-type weight of 0.3 for uniformity and a per-measurement surface weight of 0.1 for the surface of desk X would yield a weight of 0.03 for the spatial objective referring to maximizing the uniformity of surface X.

Visual Encoding and Interaction

The Simulation Ranking View consists of three integrated components: the Spatial Hierarchy, the Indicator Bar, and the Ranked Table. The *Spatial Hierarchy* (Fig. 4.5a) provides a hierarchically structured representation of measurement surfaces and their associated weights. The hierarchy is represented by an icicle plot [MR10] where each hierarchy level defines a semantic group of measurement surfaces. The root node corresponds to the entire scene and the leaf nodes represent the individual surfaces. The length of a node reflects the accumulated size of the underlying per-measurement surface weight(s). The user can modify these weights on any hierarchy level by dragging on the node borders. The weight change is evenly distributed between the child nodes. For example, reducing the size of the "Desks" node in Figure 4.5 would decrease the weights for the underlying measurement surfaces 01A to 06A, while increasing the weights for the surfaces that belong to the "Work Environment". By clicking on a node in any hierarchy level, the user can specify a *spatial focus*. Besides highlighting the geometric representations of the corresponding measurement surfaces in the linked Simulation View, the spatial focus enables the inspection of weighted scores for individual scene parts, as explained below.

As the second component of the Simulation Ranking View, the *Indicator Bar* (Fig. 4.5b) displays the weights of global and local result indicator types. The weights are encoded by the individual segments of a stacked bar. Vertically, the bar is subdivided into two levels. The lower level displays the proportions of the specific local and global indicators, such as uniformity, average illumination, etc. The upper level shows the aggregated weights for spatial (local) indicators in red and non-spatial (global) indicators in green. Hue is used to discriminate the individual indicators from each other. Just like in the Spatial Hierarchy, the user can modify weights by dragging on the borders of a bar segment on either level.

The relation between the per-measurement surface weights and the local per-indicator-type weights for spatial objectives is highlighted by the visual link between the Spatial Hierarchy and the Indicator Bar (Fig. 4.5b). If the Spatial Hierarchy is split into a focus and a context subset, the subdivision of spatial objectives is also reflected by the gray area in the Indicator Bar. The per-indicator-type weights, however, remain valid for the focus and the context subset of spatial objectives.

The *Ranked Table* (Fig. 4.5c) is the third component of the Simulation Ranking View. It displays the individual simulation runs as stacked bars, ordered in respect to their total weighted score. The score is encoded in the length of each stacked bar. The best solution according to the weighting specified in the Spatial Hierarchy and the Indicator Bar, is the one with the smallest weighted score, positioned on the top of the list of solutions. The individual segments of a stacked bar represent the weighted scores per result indicator type for the particular solution. The color coding is identical to the one in the Indicator Bar.

The subdivision of the stacked bars provides a visual decomposition of the total score. For each solution, this immediately reveals the indicators types that have the biggest impact on the score. This can reveal, for instance, that the quality of a particular solution is affected by cost-related rather than by illumination-related aspects. If the spatial objectives are split into a focus and a context subset, the stacked bars of each solution are split accordingly as well. This allows the user to compare the scores of local result indicators across different parts of the scene – for instance between the working area and the surrounding area. Changing the focus/context subset does not have any impact on the ranking – however, changing the weights of the focus/context subsets in the Spatial Hierarchy does.

Clicking on rows in the Ranked Table selects and highlights the corresponding solutions in all views of LiteVis. When a single solution is selected, the corrsponding luminaire parametrization and the resulting simulated illumination are loaded in the spatial Simulation View. Selecting multiple solutions enables the comparison of their illumination values in the Simulaiton View (see Sec. 4.4.3 and Sec. 4.4.4). Selected solutions are marked by a red border, and hovered solutions by a blue one.

In conclusion, the Simulation Ranking View is a key element of LiteVis for achieving all four of our specified design goals. Most importantly, the ability to assign importances separately for measurement surfaces *and* result indicators supports an explicit and reproducible decision making process (G4). Moreover, the Ranked Table enables an effective comparison of solutions in terms of their weight-based objective scores (G3) and enables an efficient selection of interesting solutions for a detailed inspection and further refinement in the Simulation View (G2). Integrating global and local indicators within one view furthermore supports a holistic overview of relevant aspects of the simulation output space (G1).

4.4.3 Measurement Surface Annotations

Non-spatial representations of a scene's illumination quality, such as in the Simulation Ranking View, is necessary to enable the comparison of multiple solutions. However, the analysis and comparison of such abstracted information still requires a spatial context to enable users to form an educated opinion about the investigated solutions. The measurement surface annotations (MSA) in LiteVis integrate visually abstracted representations of illumination quality of multiple selected solutions with the spatial context of the scene. The MSA enable the user to compare the distribution of illumination values on measurement surfaces from multiple simulation runs directly in the Simulation View, as displayed in Figure 4.6.

The distribution of illumination values for each measurement surface in the scene is thereby encoded in a binned histogram bar. Such a set of bars is generated for each solution that is currently selected in an Analysis View. The bars of selected solutions are grouped by the corresponding measurement surface and positioned in a floating overlay that is visually linked to the corresponding surface, as displayed in Figure 4.6. The rows

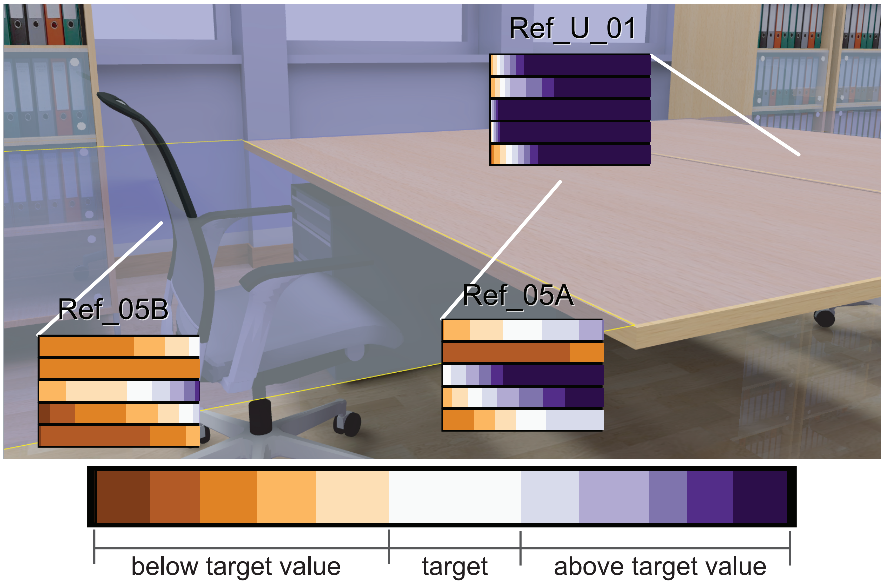


Figure 4.6: Measurement Surface Annotations displaying illumination distribution histograms for five selected simulation runs. The annotations enable the comparison of visually abstracted illumination values in a spatial context across multiple measurement surfaces and parametrizations.

in such a floating overlay correspond to the current selection of simulated solutions. This type of visual integration corresponds to *overloading* the Simulation View with the MSAs.

A single bar consists of eleven bins that are encoded with a brown/white/purple color scale. Depending on its illumination value, each texel on a surface is assigned to one of these bins. The central bin is colored white and represents the number of texels that match the respective lux target value of a surface. Values below/above the target are assigned to one of the brown/purple bins, depending on their distance from the target. The color coding that we employ is the same as in the false color render mode (as displayed in Fig. 4.4b). A bar implicitly encodes information about local result indicators for each surface and each selected run: apart from the distribution of illumination values around the target, the colors of the bins inform the user about the minimum and maximum illumination values on each surface. The uniformity of illumination is encoded in the size of the bins. An entirely white bar would be most desirable for each surface. Depending on the use case, values above the target value might also be acceptable. Values below the target are never desired.

The display of MSA is triggered as soon as more than two parametrizations are selected in an Analysis View. Mouse selection and hover states on the histogram bars are linked with the corresponding solutions in the Analysis Views. In order to avoid visual clutter, MSA are displayed only for surfaces that are part of the current spatial focus in the Spatial Hierarchy.

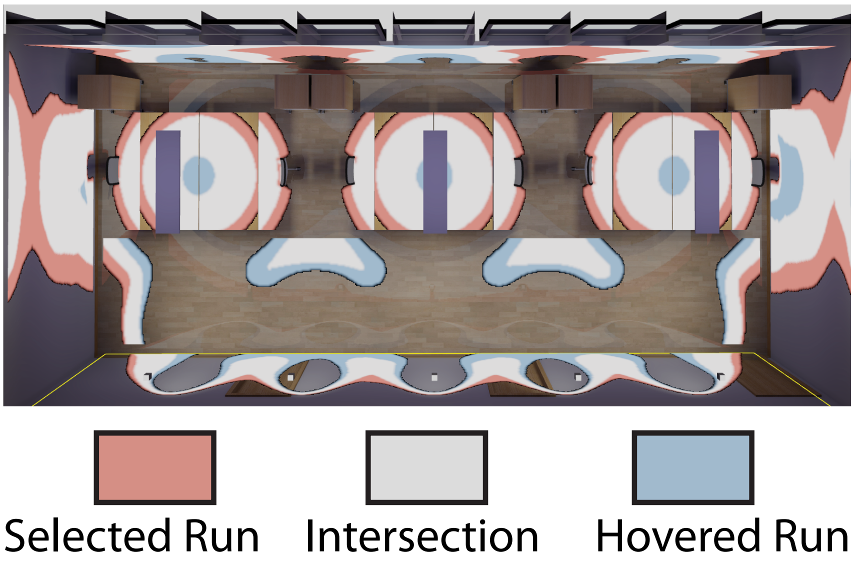


Figure 4.7: The False Color Comparison mode allows a detailed comparison of the optimally illuminated areas between a selected (red) and a hovered (blue) parametrization in a spatial context. Areas that are optimally illuminated in both parametrizations, are encoded with a white color.

4.4.4 False Color Comparison

If the illumination related, i.e., spatial, objectives of two simulation runs have the same score in the Simulation Ranking View, or even the same distribution of bins in an MSA, they might still differ in detailed qualitative illumination aspects. A purely non-spatial comparison of illumination results as in the Simulation Ranking View or the MSA is therefore not sufficient to form a decision. For a complete assessment of a solution, the analysis of illumination values on the actual scene geometry is therefore indispensable. For a single run, this is typically done with false color renderings that facilitate the estimation of illumination values by emphasizing the negative or positive distance to a specified lux target value of each point in a scene. The comparison of the spatial illumination distribution of multiple solutions is historically carried out in a side-by-side manner. The False Color Comparison (FCC) mode (Fig. 4.7) gives the lighting designer the means for a detailed comparison of the illumination quality between two simulation runs directly within the simulation environment. The areas where the illumination values of each run correspond to the respective target value of a surface are rendered in a false color. Red areas indicate ideal illumination in the currently selected simulation run (conforming to the red selection color), while blue areas indicate ideal illumination in the currently hovered run (conforming to the blue hover color). If both runs achieve ideal illumination on a texel, the texel is rendered in white. In compliance with the lighting designers' way of thinking, texels are considered as ideally illuminated, if they achieve at least the defined industry standard for the respective surface type. However, the acceptable range above the target value can be adjusted to adhere to special scenario requirements (e.g., when high illumination values should be avoided due to energy consumption reasons). The FCC is triggered when a single run is selected and a second run is hovered in an Analysis View or MSA.

The direct comparison through the FCC has two advantages for the user. On the one side, users do not have to switch their focus between multiple views when comparing two solutions. On the other side, the FCC allows for more efficient feedback on newly created parametrizations by enabling an instant and detailed comparison of a new simulation run to a reference parametrization.

4.5 Implementation

LiteVis is built as an extension of multiple frameworks that communicate with each other via a network protocol. The Simulation View is based on an interactive global illumination lighting software that was developed with our collaborating domain experts in the scope of another project [LTH$^+$13]. Its simulation kernel is based on a novel many-light simulation that allows the resulting illumination to be computed and visualized within a few seconds. This is achieved by using a GPU-based shadow mapping algorithm [Wil78] for the visibility calculation concerning both direct and indirect light sources. The results are collected and stored adaptively in so-called light-map textures that are mapped onto the scene geometry. After a scene modification, designers get immediate visual feedback, which continuously improves by converging to the physically accurate solution.

We extended the simulation with means to store, load, and export simulation data and added a web-overlay that handles rendering of and interactions with the MSA (implemented using the d3 toolkit [BOH11]). The Analysis Views are part of a versatile visual analysis framework [PTMB09] in which we implemented the Simulation Ranking View. In order to enable the functional integration of the simulation and the analysis frameworks, both were extended with a network communication interface that manages the exchange of simulation data and commands, e.g., for coordinating selection states.

4.6 Use Case Scenario

In this section, we demonstrate the applicability of LiteVis based on a real-life use case scenario from our collaborating lighting designers.

4.6.1 Scenario Description

The simulation setup consists of a 3D office scene with 18 measurement surfaces. These surfaces are arranged into three semantic groups with different target values according to industry standards (Fig. 4.2):

- Task area (desks): 500 lx, 0.6 uniformity
- Close surroundings (work environment): 300 lx, 0.4 uniformity
- Background (walls, floor, ceiling): 100 lx, 0.1 uniformity

Further, the lighting designers supplied us with a set of 107 parametrizations that refer to variations of four different lighting scenarios. A scenario is characterized by the deployed luminary types that are grouped into *light groups* (LGs). LG1 is positioned over the desks. LG2 is positioned over the hallway of the office.

- Scenario A: LG1: pendulum, LG2: downlight
- Scenario B: LG1: double floor lamp, LG2: downlight
- Scenario C: LG1: single floor lamp, LG2: downlight
- Scenario D: built-in ceiling luminaries

We demonstrate how LiteVis supports the decision making process based on a global-to-local exploration of the supplied parametrizations.

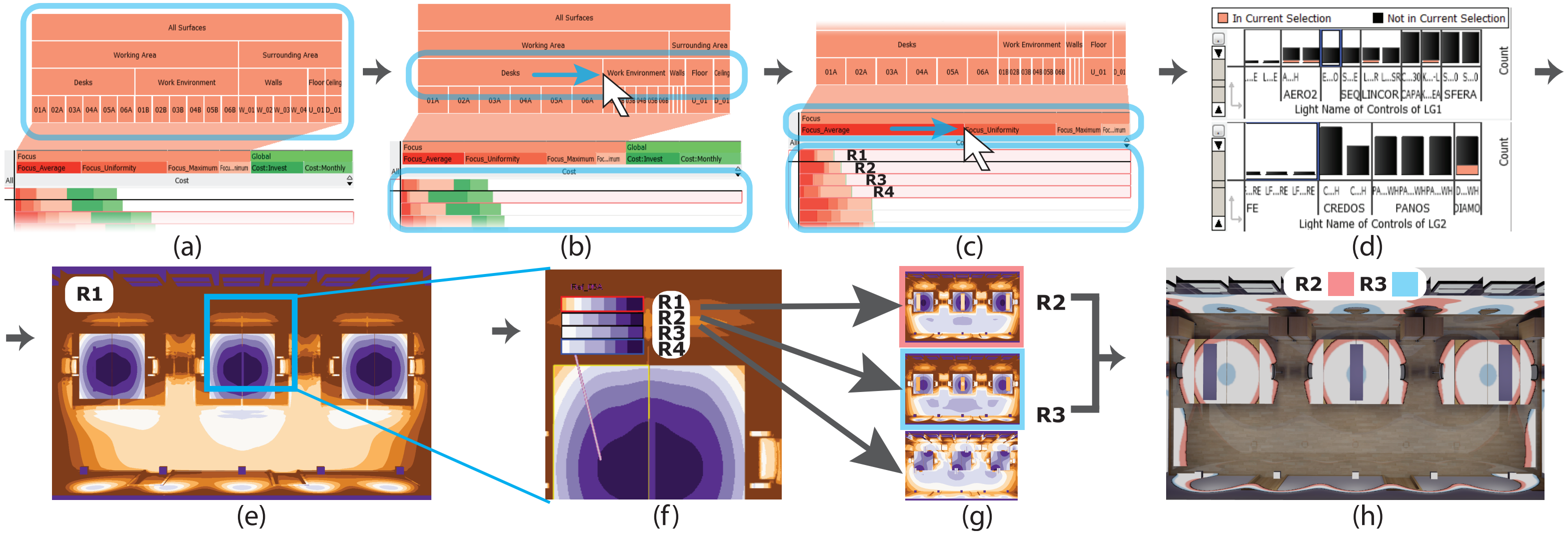


Figure 4.8: The schematic overview of the workflow in our use case scenario showing an exemplary path for decision making. a) Initial state of the weights for spatial and non-spatial objectives. b) Defining per-measurement surface weights in the Spatial Hierarchy. c) Changing the weights of global and local indicators in the Indicator Bar results in a ranking update. d) Highlighting the input parameters of the top four ranked solutions (R1-R4) in an Analysis View. e) Assessing the spatial illumination distribution of R1 using false color rendering. f) Comparing the R1-R4 with MSA. g) Assessing R2-R4 in the Simulation View. h) Triggering the FCC for a detailed comparison of R2 and R3 in a spatial context (white encodes ideal illumination in both R2 and R3).

4.6.2 Scenario Workflow

As starting point, the provided parametrizations have been loaded in the Simulation Ranking View. The measurement surfaces in the Spatial Hierarchy are grouped by type and all surfaces are selected as spatial focus (Fig. 4.8a). The ranking is based on four local (average illumination, uniformity, maximum, minimum) and two global result indicators (investment cost, monthly cost).

A typical first step in the workflow is the specification of per-measurement surface weights in the Spatial Hierarchy. Since the desks and work environments are of primary importance, the analyst decides to adjust the weights in the following way: Desks 60%, Work Environment 20%, Floor 10%, Walls 5%, and Ceiling 5% (Fig. 4.8b). As a next step, the objectives for global and local result indicators are specified in the Indicator Bar. The analyst first focuses on runs with the best illumination quality, regardless of budgetary constraints. The analyst therefore sets the weights of the global (financial) indicators to zero. Among the four local indicators of illumination quality, the analyst considers average illumination and uniformity on a surface as the most important metrics. Thus, she adjusts the weights of the four indicators as follows: Average 50%, Uniformity 30%, Maximum 15%, and Minimum 5% (Fig. 4.8c). The Ranked Table now shows a ranking of the simulation runs with the best illumination quality according to the specified weighting (Fig. 4.8c). As a next question, the analyst investigates which input parameters are involved in the highest ranked simulation runs. She selects the four top-ranked simulation runs. She then creates an Analysis View with a bar chart for each light group. Each bar represents a luminary type and displays the number of occurrences of that type within a light group. The bar charts allow her to inspect which luminary types are used in the top-ranked solutions (Fig. 4.8d). This reveals that R1-R4 are each using a different luminary type in LG1, while the LG2 luminaries in all four solutions are parametrized with the same luminary type (highlighted in red in Fig. 4.8d).

The ranking also shows that the local indicator scores are very distinctly distributed in each of the four simulation runs (Fig. 4.8c). Run R4, for instance, has the best average illumination but a high cost regarding the minimal illumination value, which the analyst finds remarkable given the small weight assigned to this objective. She therefore decides to assess the illumination quality of these four runs directly in the Simulation View. First, she investigates the distribution of illumination values for R1 using the false color render mode (Fig. 4.8e). While R1 exhibits a good total score, the rendering exposes illumination values below the target value on the desk corners.

To compare the illumination distribution to the other top-ranked solutions, the analyst simultaneously selects the first four runs in the Ranked Table. This triggers the display of MSA for the selected solutions (Fig. 4.8f). For R1, the illumination distribution histogram in the MSA confirms that parts of the illumination are below the target value, while R2, R3 and R4 lie entirely on or above the target on the desks. In fact, their distributions are very similar. Therefore, the analyst decides to assess their illumination quality in a spatial context as well. By individually selecting the bars in the MSA, she inspects R2-R4 one after another in the Simulation View (Fig. 4.8g). R4 exhibits the most uniform illumination on the desks. However, as the analyst knows about the customer's preference for pendulum lights in this scenario, she does not want to discard R2 and R3 yet. To gain insights about eventual trade-offs between R2 and R3, she compares them directly in the 3D scene (Fig. 4.8h). The FCC reveals that the ideally illuminated area of R3 (blue) lies in the center between two desks, while the ideal illumination in R2 (red) is more evenly distributed across the desk surface. She therefore considers R2 as the preferred solution in terms of illumination quality.

A satisfactory luminary parametrization has been found. However, the analyst sees multiple directions for further investigations. For instance, Scenarios C and D were not present in the top four runs. She could investigate, where the runs are located in the ranking by selecting all associated runs based on their input parametrization in the Analysis Views (Fig. 4.8d). She could even create an individual ranking for each scenario by filtering runs, e.g., by the type of applied luminaries. Similarly, she could eliminate R1 from the ranking by filtering all parametrizations that violate a minimum illumination threshold on desk surfaces. She could also investigate how the chosen parametrizations compare against others with respect to financial goals. For this, she would mark the top-ranked runs as selected in order to track their new ranks after adjusting the weights of the financial (global) indicators. She could also directly compare multiple objective prioritizations by creating and comparing multiple Simulation Ranking Views, each with a different weight distribution. She could check, how the score of local indicators is distributed among the surfaces in the scene by setting, e.g., the working area as the spatial focus. Further, she could manipulate a run, e.g., by dimming the wattage and energy consumption, and see in which respect the resulting lower illumination values and runtime costs would influence the ranking.

4.7 Evaluation

4.7.1 Design Process

The design process of LiteVis can be described as an iterative cycle characterized by the three phases of (1) domain problem characterization, (2) data/operation abstraction, and (3) encoding/interaction technique design [Mun09].

For characterizing our collaborators' domain problem and refining our understanding of their workflow, we performed semi-structured interviews during small meetings. As we observed the lighting designers' accustomed workflow, contextual inquiries [HJ93] helped us to clarify tasks and challenges. The study of lighting design literature [DIN11] helped to increase our knowledge of the associated data. We presented the collective findings from this phase in Sections 4.3.1 and 4.3.2. While we were able to understand the overall goal of the lighting designers well, the most challenging part of this phase was to get a deep understanding of the nuances of each task. Such nuances regard questions like: what exactly are the independent degrees of freedom for parametrizing a scene, which particular aspects and metrics in the evaluation of a scene are considered relevant?

The resulting data abstraction can be found in the end of Section 4.3.1, the problem abstraction is the subject of Section 4.3.4.

Parallel prototyping [DGK$^+$10] of hand-drawn sketches enabled us to refine our understanding of the requirements and tasks, and to discuss and validate our abstractions with the experts. The Simulation Ranking View was subject to the most design iterations. The basic requirement of providing an overview of local result indicators on different aggregation levels was addressed already in the first visual prototype. However, the approach for ranking a large number of simulation runs by these indicators changed throughout the design process. An early version featured a transposable grid, but turned out to be cluttered and did not provide a sufficient overview. In the next iteration, we investigated an icicle-plot layout where each leaf corresponded to a parametrization. The hierarchy levels represented groupings of parametrizations by input parameters, and encoded aggregates of the contained result indicators. This structure was still too rigid due to the inherent hierarchical grouping by input parameters. Finally, we decoupled the leaves from the hierarchy. The leaves, i.e., parametrizations, were listed as stacked bars in the Ranked Table. The hierarchy was used to represent groups of measurement surfaces instead of parameters as in the final version of the Spatial Hierarchy.

Once we settled on the design, we iteratively refined a software prototype and discussed the progress in monthly intervals with two to three domain experts. The overall design and implementation process took approximately 15 months. Important feedback during the design process concerned the automatic grouping of surfaces by type. Depending on the task, the experts considered other groupings more effective, e.g., by spatial proximity. We thus added the option to group surfaces manually. Other feedback concerned the Spatial Hierarchy. The lighting designers had difficulties in understanding the relation between the surfaces in the Spatial Hierarchy and the local indicators in the Indicator Bar. The visual link that we subsequently added alleviated this issue. Concerning the FCC, an initial version encoded illumination values in a certain range above and below the target value. However, the lighting designers stated that a range that includes values below the target was not helpful to them. The FCC therefore now only considers values above the target by default. The range is adjustable as described in Section 4.4.4.

4.7.2 User Feedback

This section reports on user feedback collected during an evaluation workshop with three lighting design experts. Based on a protocol, we first gave an overview of the system and explained the individual components as well as how they are integrated with each other. The demonstration was based on a dataset provided by our collaborators (Sec. 4.6). After two hours of introduction including questions and discussion, the domain experts used the system for another two hours including time for feedback and discussion. To document the feedback, participants completed a questionnaire with qualitative questions regarding the individual features.

The overall feedback of the interactive trial session was very positive. All experts agreed that analysis and comparison of parametrizations are key problems in the lighting design workflow. All of them considered LiteVis as an effective approach to increase efficiency and confidence in identifying relevant lighting parametrizations. The most appreciated component of the visual design included the Simulation Ranking View. The experts approved the design decisions of summarizing parametrizations by a combined score of aggregated result indicators as well as the interactive specification of measurement surface importance. Specifically, the ranking gives them *"a never before envisioned overview of the quality of their simulation data"*. The specification of objectives concerning result indicators and measurement surface weights gives them *"the necessary means to control and explore the massive amount of information"*. Our collaborator's head of lighting solutions support pointed out: *"The system is very powerful and comprehensive. In particular the possibilities of an intuitive comparison of lighting solutions are a significant improvement as compared to other solutions that are currently deployed. It is exactly what we were looking for."*

More critical feedback concerned the MSA in the Simulation View. The experts commented that developing an intuition of the spatial illumination distribution based on the illumination distribution histograms requires some time for familiarization. However, the lighting designers were generally able to understand the encoded illumination metrics and appreciated the feature for comparing the illumination distribution on individual measurement surfaces. The FCC was appreciated as a useful way to compare the spatial distribution of illumination values of selected parametrizations, and was regarded as a good complement to the comparison of non-spatial representations in the Analysis Views and the MSA.

In summary, the discussions with the domain experts confirmed that we were able to meet our design goals. In fact, the lighting designers valued our approach so much that they intended to integrate LiteVis into their workflow.

4.8 Discussion and Future Work

4.8.1 Goals

By summarizing the key components of LiteVis we can illustrate how our design corresponds to the goals stated in Section 4.3.3. We enable a holistic overview of all relevant features of the parameter space by integrating the qualitative (Simulation View) and quantitative (Analysis Views) aspects of the simulation output (G1). Through this integrated approach, it is not only easier for a designer to understand the results, but also to incorporate additional semantic information, customer requirements, or prioritization concerns that are not included in the simulation itself. The tight integration of the scene parametrization with the evaluation of simulation results enables a feedback loop that allows users to iterate faster towards an optimal solution (G2). The effective comparison of multiple solutions (G3) is enabled through the MSA, the FCC and especially the integration with the Analysis Views that can be flexibly adjusted to depict input and output values of all sampled simulation runs. This detailed comparison also enables a sensitivity analysis to assess the impact of parameter variations. Especially our novel Simulation Ranking View provides explicit and reproducible decision support (G4) by clearly indicating the distributions of spatial and non-spatial objective weights and the resulting ranking scores.

4.8.2 Scalability

The scalability of our tool can be analyzed according to different aspects. The number of global and local result indicators in the Simulation Ranking View's Indicator Bar is limited by the horizontal screen space. Due to the aggregation of local indicators, their number is independent from the number of measurement surfaces. Local indicators could be further aggregated into categories if they would occupy more than the available horizontal space. The number of surfaces in the Spatial Hierarchy is limited by the available screen space, as well. However, since they are hierarchically groupable, they could be managed in a sort of zoomable icicle plot. In large office scenes, the surfaces could go into the hundreds. However, these scenes are typically uniformly structured, i.e., they consist of repeating patterns of furniture and lighting constellations. Measurement surfaces can therefore be grouped without losing valuable information. The number of solutions displayed in the Simulation Ranking View is not an issue concerning scalability as the most interesting ones (according to the specified objectives) are always ranked at the top. If an overview of hundreds or thousands of simulation runs is desired, a table lens could be used to gain a synopsis of all runs. Hierarchical grouping by parameters could be used to divide a large sample base of simulation runs. One part of LiteVis that does not scale well are the MSA. The annotations become cluttered when displaying more than 20 simulation runs within a floating overlay. However, feedback from lighting designers indicated that an in-depth comparison typically takes place on a smaller set of runs.

4.8.3 General Applicability

The general applicability of concepts from LiteVis is given especially in our Simulation Ranking View. The view enables the specification of objectives on two orthogonal levels. This concept has general applicability in decision making, i.e., in cases where objectives can be shared by hierarchically superordinate entities. In disaster management, for instance, a simulation run can represent the results of a parametrizable flooding simulation. Instead of measurement surfaces, the Spatial Hierarchy could represent the evacuation prioritization of hierarchically groupable areas. Local indicators could represent different properties of these areas, such as the degree of damage, or repair cost.

4.8.4 Future Work

Our evaluation indicates that the current version of LiteVis already has clear benefits compared to current decision making support tools in lighting design. In our discussions with domain experts, we identified additional topics that would be interesting to address in the scope of LiteVis:

- More user guidance: the Spatial Hierarchy could be used to supply the user with more information about the parameter space. The cells in the Spatial Hierarchy could be colored to encode how much they contribute to the aggregate of a local indicator. Another idea would be to use the cells to encode how many places in the ranking the current selection would change if the respective cell would be assigned an importance value of 100%.
- More control over spatial objectives: subdivision of measurement surfaces into sub-areas with individual importances could help the lighting designer to fine-tune the specification of spatial importance factors. On a desk, for instance, good illumination in the center is more important than at the corners. Such fine-tuning is actually already possible but only in the setup phase (T0), i.e., when the measurement surfaces are defined.
- Analysis of illumination over time: especially in out-door illumination scenarios, the sun plays an important role. Allowing users to analyze parametrizations at different times of day and to incorporate day time dependent dimming profiles into this process, would be an interesting application area to explore.

4.9 Conclusion

In this chapter, we describe the design study of LiteVis – a system aimed at assisting lighting designers in their decision making process in regard to finding an ideal lighting solution that adheres to competing qualitative and quantitative objectives. Visual and functional integration of parameter space representations, i.e., the simulation input, with representations of all relevant aspects of the simulation output is thereby the key in enabling an efficient exploration and comparison of lighting parametrizations. Further, we propose a novel ranking visualization that was inspired by multi-objective decision making solutions taking into account the special intricacies of the domain's complex parameter space. We characterize the problems in the application domain of lighting design and give a reflection on our design process. A use case illustrates how LiteVis supports decision making in the context of a database of lighting solutions that would have overwhelmed lighting designers using conventional lighting design solutions. Qualitative feedback from domain experts confirmed that we met our design goals and that lighting designers want to incorporate LiteVis into their daily workflow.

The integration techniques that we applied in this chapter were geared towards enabling an efficient exploration of multifaceted spatial data. For the visual integration of the individual data facets, we used juxtaposition for the Simulation and Analysis Views in order to keep the mapping and the assigned screen real estate of both view types independent. We chose to overload the Simulation View with the MSA in order to embed them in the spatial context of the measurement surfaces with which they are associated. For the comparison of solutions directly within the Simulation View, we adapted the spatial representation of the scene in order to enable the FCC. The Simulation Ranking View is actually composed of three visually integrated components: the Spatial Hierarchy, the Indicator Bar, and the Ranked Table. This allowed us to join the representations of the weights for the spatial and non-spatial aspects of the ranking, with the ranking itself. All three components are juxtaposed within the same view. To establish a relation between the three components, the Spatial Hierarchy and the Indicator bar share the same meaning for the color (focus and context) and for the size of the area of visual marks along the x-axis (amount of importance). The Indicator Bar and the Ranked Table share the same meaning for color (focus, context, and global objectives) as well as brightness (global and local indicators).

In terms of functional integration, we linked selection and highlighting states across spatial and non-spatial representations ($D_{s/a} \rightarrow V_{a/s}$). Further, we coordinated mapping colors for the representations of the MSA and the false color render mode, as well as for the FCC and the selection states ($V_{s/a} \rightarrow V_{a/s}$) for a better orientation between views. We enabled the loading of data into the Simulation View via the Analysis Views, and, vice versa, enabled the extension of the displayed data sets in the Analysis Views with new simulation runs created in the Simulation View ($D_{s/a} \rightarrow D_{a/s}$). Together the coordination between selection states, color mapping, and data exchange enable the iterative refinement of solutions within an integrated feedback loop. The functional integration between the components within the Simulation Ranking View can be described as $D_{SH} \rightarrow D_{IB} \rightarrow D_{RT}$, as the Spatial Hierarchy (SH) propagates weight changes to the Indicator Bar (IB) that, in turn, propagates weight changes to the Ranked Table (RT).

The scenes that we dealt with were small enough to be navigated manually and could even be assessed from an overview position that shows the entire setup like in Figure 4.8. We therefore did not see the need for guided navigation ($D_a \rightarrow N_s$, as introduced in Section 2.4) in the spatial view. However, for more complex scenes, such as large offices or sports stadiums, guided navigation would certainly be required to facilitate the lighting designers' workflow. Hereby we conclude the discussion of integration in the context of this exploration scenario. In the following two chapters, we will investigate how integration can be applied in presentation scenarios, where user interaction cannot be relied on in order to fulfill a task.

CHAPTER 5

Storytelling Templates For Temporal Integration

Creating Insights Via Animated Transitions

This chapter is based on the following publication:

Johannes Sorger, Peter Mindek, Peter Rautek, Eduard Gröller, Graham Johnson, Ivan Viola. *Metamorphers: Storytelling Templates For Illustrative Animated Transitions in Molecular Visualization.* In Proceedings of the Spring Conference on Computer Graphics 2017, pages 27-36. May 2017 [SMR$^+$17].

UNDERSTANDING the different facets of a spatial data set often requires different representation forms that are suitable for highlighting these aspects. Certain insights into a multifaceted spatial data set might be better conveyed within a non-spatial context. In molecular biology, an example for such insights can concern the quantities of protein types within a microorganism. For such quantitative information, a histogram would be a suitable representation form. For an untrained observer, the relation between the protein quantities in the histogram and the original spatial representation of the microorganism might not be clear anymore. An observer's mental model of a spatial data set is typically formed by real life experiences, or in the case of microbiology, from microscopy, or illustrations from scientific books. The mental model is therefore anchored in the same visualization space as the physical object. If no obvious relation between a mental model and a particular representation exists, strategies have to be employed that help the viewer to relate their mental model to a transformed representation, in order to better comprehend the intended message. In situations where interaction is possible or desired, users can engage directly with the data, in order to explore and clarify these relationships by themselves, provided that the required tools, such as brushing and linking between representations, are available.

In cases, where interaction is not feasible or very limited, such as in presentation scenarios, the relation has to be conveyed more explicitly. One way of establishing such relations can be achieved through annotations and visual links between corresponding parts of the data. Such representations might be found in info-graphics of scientific textbooks or in museum displays. If dynamic representation forms are an option, the encoding can be achieved through an animation that guides the viewer, for instance, through the continuous transformation of one representation to the other one. In both cases, visual integration is required to establish the relationship: for static, spatially integrated representations, for instance, by overloading the representations with visual links; in the case of dynamic representations, through a temporal integration of both representations.

In this chapter, we present a technique that is able to convey the relation between different representations of spatial data facets via animated transitions. An animated transition is thereby defined as a series of animation states, i.e., the individual representations, and the transitions, i.e., the temporal integration, between them. Our technique supports the definition of animation states and the transitions between them through the adaption of the original spatial representation of a molecular model. The original mapping to visual channels can thereby be neglected, modified, or re-assigned, as discussed in Section 2.2. The adaption is carried out through a set of operators, the so called *metamorphers*, that enable a wide range of visual abstractions or even the transformation of the visualization space, i.e., the spatial frame of reference. By defining the representation states and transitions between them based on adaptions of the original representation, metamorphers can be specified in a way that makes them modular, i.e., flexibly combinable, and re-usable. These properties yield many advantages in the context of authoring animations for complex molecular models, as we will discuss in the following.

5.1 Introduction

In recent years, we have seen a rapid increase in the communication of topics from molecular biology, such as through the work of Drew Berry [Ber], Graham Johnson [JNM], Gaël McGill [JM13], or Janett Iwasa [Iwa10]. These topics involve complex processes, such as the copying of DNA, the transport of oxygen in the blood stream, or the comparison

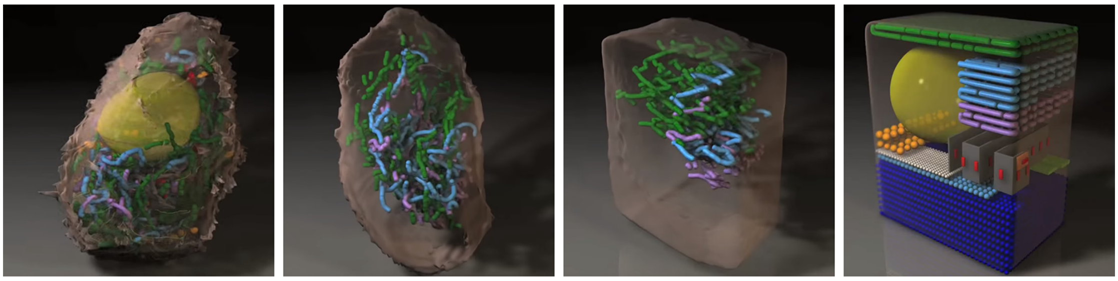
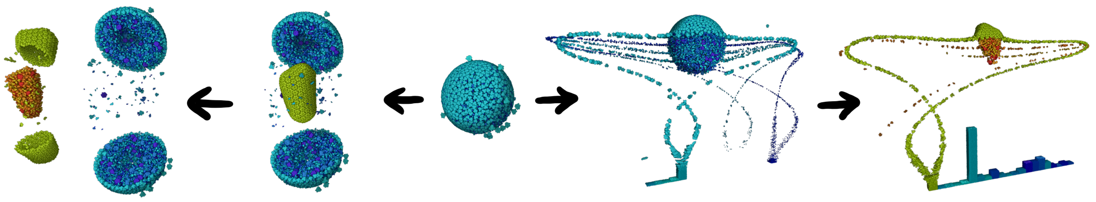


Figure 5.1: Transition of an organic representation of a cell to a more structured one that conveys the quantities of the cell's components [JNM].



(a) (b)

Figure 5.2: Metamorphers provide a flexible and re-usable interface for the creation of animated transitions of mesoscale data. Here, two different combinations of metamorphers are applied to create an exploded view (a) of an HIV model, and a bar chart (b) that represents protein quantities.

of molecular structures within a cell. Illustrators use animations to communicate these processes in an explicit and intuitive manner. An example sequence from such an animation is shown in Figure 5.1 where an organic cell structure is smoothly transformed into a more diagrammatic representation. The scientific animator [JNM] used this transformation to promote quantitative aspects of the data over its structural appearance. To achieve their goals, animators make use of well established visual abstraction techniques from scientific illustrations. *Exploded views*, for instance, are suitable for revealing how hierarchical structures, for example, the compartments of a microorganism, are layered (Fig. 5.2a). Such animations are typically handcrafted with 3D modeling and animation tools, such as Maya. These tools offer great flexibility in the variety of achievable results. However, the low-level approach that they apply, requires a high level of expertise in illustration, animation, and biology to create an animation that conveys a complex biological phenomenon. Such a manually key-framed animation cannot be transferred to other data sets.

Another problem of illustrative animation authoring on a low level arises in the context of molecular biology. In the visualization of molecular biology, we are dealing with very complex models, often containing tens of thousands of objects that represent individual cells, molecules, or atoms. Techniques routinely applied in hand-crafted scientific illustrations and animations are difficult to apply to such dense and large-scale data sets.

Besides manual low-level approaches, there are high-level approaches [LRA$^+$07, DMNV12] that semi-automatically generate a certain visual transformation for a supplied data set. These approaches relieve the user from manually creating key frames for the thousands of objects in complex molecular scenes. Further, they can be applied to arbitrary data sets that share a similar data structure. This advantage comes at the price of the flexibility that low-level techniques offer in terms of result variability.

Our goal is therefore to develop a technique for authoring animated transitions that unites the advantages of low-level and high-level approaches. We support *flexibility* and *re-usability*, while providing the means to mask the complexity of low-level approaches from inexperienced users. To achieve these goals, we propose modular animation operators, i.e., *metamorphers*, that act as combinable storytelling templates. Metamorphers belong to six different classes of operations (data restructuring, layout, morphing, trajectory, timing, and camera control) that collectively describe the intermediate or target states of an animation, as well as the transitions between these animation states. A single metamorpher modifies the properties of the molecular data in regard to one of these classes. *Re-usability* ensures that the metamorpher produces an intended result for any compatible input data. *Flexibility* ensures that arbitrary metamorphers can be combined in a modular way to create a wide range of results. Both of these properties allow the user the combination of multiple metamorphers into higher-level operations. Such high-level metamorphers are again re-usable and flexibly combineable. A high-level metamorpher serves as a hierarchical container that masks the complexity of the contained lower-level metamorphers from the user.

In the remainder of this paper, we describe the technical details that are required to understand and reproduce metamorphers for the application to the domain of molecular biology. We illustrate our technique by presenting exemplary metamorphers in the scope of a proof of concept implementation. Our proof of concept implementation features a visual scripting interface in the form of a hierarchical node editor that enables the parametrization and combination of metamorphers as well as the creation of re-usable high-level metamorphers. Based on our implementation, we demonstrate the flexibility and re-usability of metamorphers by creating two different animations and re-applying them to three different data sets from molecular biology. Finally, we reflect on our technique as well as give some informal feedback that we gathered from domain experts in illustration, animation, and biology.

5.2 Related Work

Animation and transitions between data representations serve as powerful tools for the dissemination of complex relations in space, time, and abstract dimensions. They are frequently used in visual storytelling as well as for the depiction of relations between different data representations. Kosara and Mackinlay [KM13] emphasize the importance of storytelling in visualization not only for presentation purposes but also in decision making and process analysis. They suggest that opening up an animation for interaction at the story's end, provides a convenient starting point for exploration and goes beyond a simple slideshow. Our implementation creates animations of the molecular data that can be paused and explored in real-time at any point during the transition. Wohlfart and Hauser [WH07] present a guided interactive volume-visualization approach in terms of camera path, transfer-function parameters, and annotations. Grimm et al. [GBKG04] introduce V-Objects as an abstraction of volume data. Their approach enables interactive specification of key-frames for animation paths, transfer-function fades, light movement, clipping planes, change of data sources, and enabling and disabling of objects. These approaches do not consider the re-usability of the animations for different data sets as they are based on the explicit authoring of the key-frames. Hyun et al. [HLL16] propose a more implicit approach in the context of storytelling, by using grammars to synthesize the animation of characters. They generate animation sequences for virtual players by employing a language that captures the behavioral structure of human movements in a basketball game. The grammar is used to check the validity of a sequence of movements and to suggest new plausible candidates for the next move. Karp and Feiner [KF93] present a scripted system that generates animations from communication goals. The system plans an animation by breaking it down into sequences, scenes, and shots down to the level of individual frames. Seligman and Feiner [SF91] present a system that generates technical illustrations using illustration rules. Annotations, highlights, and rendering styles are chosen to convey the intended information. Mühler et al. [MBP06] present a scripting language for the generation of illustrations and animations in the context of medical intervention planning. Mühler and Preim [MP10] introduce *keystates* that describe a visualization on an abstract level, including information about visibility, rendering style, etc. of each structure. Keystates are used to automatically generate animations and are re-usable for data sets that contain the same structures. Iserhardt-Bauer et al. [IBHT$^+$02] developed a system that generates video sequences for intra-cranial aneurysms. Visualization parameters and the camera path are automatically generated from a standardized predefined protocol.

Figure 5.3: Classes of metamorphers according to the six stages of a pipeline for animated transitions. The data restructuring, layout, and morphing stages define an animation state. Trajectory, timing, and camera control define how an animation state transitions to the next one, and thereby structure the presentation of the transition.

While these approaches convey purely spatial phenomena, other approaches reveal non-spatial relationships within the 3D spatial data. Hurter et al. [HTCT14] use animation to interpolate between different (non-spatial) projections of the original data dimensions in a 3D volume. Basch [Bas11] uses animated transitions of voxels to non-spatial volume representations, such as histograms of voxel intensity values. With metamorphers we propose a flexible technique that supports the depiction of spatial, as well as non-spatial relationships by transforming the spatial data into an non-spatial frame of reference.

5.3 Metamorphers

We describe the complete transformation of an object between two animation states by six stages of a pipeline for animated transitions (see Fig. 5.3). The first three stages of the pipeline (data restructuring, layout, and morphing) define, *what* the original model or data set should represent in an intermediate or target state of the animated transition. In the simplest case, an animation is created by linearly interpolating between the animation states (key frames) of the individual molecules in a scene at the same time. However, the simultaneous interpolation of tens of thousands of molecules, is hard to read due to clutter and occlusion. We refer to such an interpolation as an *unstructured interpolation*. By *spatially and temporally structuring* the transition, i.e., with trajectories, timings of the individual molecule transitions, and by controlling, which parts of the scene the camera presents to the viewer, a more pleasant animation can be created. The last three stages of the pipeline (trajectory, timing, and camera control) are thus responsible for defining, *how* a state should transform into the next one, and how the transition should be presented to the viewer.

In this section, we give a definition of how metamorphers can support re-usability and flexibility. We then describe the molecular data that metamorphers operate on, and we specify the interface that enables the modular combinations between them. Finally, we describe the different classes of metamorphers that are necessary to create all aspects of an animated transition.

5.3.1 Re-usability & Flexibility

In order to support *re-usability*, a metamorpher has to adhere to two rules to achieve intended results for arbitrary data sets:

1) A metamorpher has to define an animation state not in an absolute way but *relative* to the previous animation state. For instance, a new position Pos_{new} for an object A should not be defined as $Pos_{new}(A) = X$, but instead as $Pos_{new}(A) = Pos_{old}(A) + X$.

2) A metamorpher has to define an animation state not explicitly but *implicitly*, i.e., based on inherent properties of the scene and the data set. In the above example, X should not be an explicit value, e.g., $X = 500$, but should be given or derived from inherent properties, e.g., $X = width\ of\ object\ A's\ bounding\ box$. Exceptions to these rules occur if absolute and explicit values have the same semantic implication for all data sets, such as moving an object to the center of the scene.

In order to support *flexibility*, a metamorpher has to provide a modular interface that enables arbitrary combinations with other metamorphers. Such a modular interface is achieved by using the same data structure for the input as well as the output of each metamorpher. A metamorpher takes a data (sub)set as an input, modifies specific properties, such as the positions over time, and passes the modified data (sub)set as output to the next metamorpher.

Both of these qualities allow metamorphers to *mask* the complexity of authoring an animation. Multiple metamorphers can combine their lower-level operations into a single high-level operation with complex functionality. A high-level metamorpher serves as a hierarchic container that masks the combined lower-level metamorphers from the user. For instance, the steps involved in the creation of an exploded view can be summarized into a high-level metamorpher. A non-expert user can now apply the exploded view to an arbitrary data set without having to know about the individual operations that are involved. Flexibility assures that this high-level metamorpher can still be combined with other metamorphers. These characteristics turn metamorphers into re-usable storytelling templates that can be flexibly combined to convey diverse messages. At the same time, the low-level complexity of the operations involved in authoring illustrative animation sequences is masked.

5.3.2 Molecular Data: Inherent Properties

Our pipeline operates on biological data that is composed of individual proteins. These proteins collectively form the compartments and inner structures of the microorganisms that they depict. Proteins are therefore the building blocks of these data sets. The data set of the HIV virion that is depicted in Figure 5.2, for instance, contains approximately 20.000 molecules distributed across 42 protein types, with an atom count of 60 millions.

The molecular data is organized in a hierarchical structure. The smallest entity is an *atom a* of a certain *type*, i.e., chemical element. It is represented by a sphere that has a *position* in its local coordinate system, and a *scale* that describes its radius. A *molecule m* represents a certain protein *type* and has a *position*, *orientation*, and *scale* in the global coordinate system. A molecule is represented by a list of atoms $m = a_1, ..., a_n$ that yields an *atom count* and a *volume*. It is contained within the spatial extents of a *bounding box BB*. A *scene S* is comprised of a list of molecules $S = m_1, ..., m_j$ that can be partitioned into *hierarchical subsets M*, describing semantic structures of the microorganism. The leaf nodes are the individual molecules. Each node in the hierarchy thus contains a list of subsets or a list of molecules that yields a *molecule count* and *volume*, and is contained in a *bounding box*.

To support animation, the hierarchic data structure is extended by *key frames* and *time curves* for subsets and molecules. A key frame stores a node's properties, such as the position, and bounding box, for an animation state. Time curves determine the start time and speed at which the properties of a molecule are transformed between each pair of key frames. Metamorphers manipulate properties of subsets and molecules and automatically create new key frames and time curves that are interpolated to create an animation.

5.3.3 Metamorpher Modular Interface

The modular nature of the metamorpher input/output interface supports the flexible combination of metamorphers. This enables the user to create diverse animation sequences. The interface takes as input the data hierarchy and a pointer to the parent node of the subsets that should be processed. The metamorpher then accesses and modifies the respective properties of the node's children according to its functionality. The interface passes on the modified node as a single output node. Multiple outputs have to be used if the individual subsets of a hierarchy level should be forwarded to different metamorphers (as shown in Fig. 5.5). In this case, the pointers to the individual subsets are passed on to separate outputs. Since the separate outputs are explicitly linked to different metamorphers, the number of outputs has to be known, i.e., it cannot depend on the data set. A spatial data-restructuring metamorpher, for instance, creates always two subsets if it splits the space into two parts. If the number of subsets that a data restructuring metamorpher creates is unknown, the subsets have to be forwarded as the children of a single node.

5.3.4 Metamorpher Classes

In the following, we elaborate on the six metamorpher classes and we give examples for concrete implementations for each of them. We thereby use the exploded view and the transition to a bar chart from Figure 5.2 as guiding examples.

We use the following notation to describe a metamorpher *type(param, subset)*. *Type* describes the metamorpher operation, such as *rotate* for a layout metamorpher that triggers a rotation. *Param* describes the parametrization of the metamorpher, which can be a set of input values, such as the axis and the angle for a rotation. *Subset* contains the subset on which the metamorpher will be applied.

Data restructuring Metamorphers

Data-restructuring metamorphers partition the data structure to define, on which parts of the data, and in which granularity metamorphers from subsequent pipeline stages will operate, i.e., *they create subsets* from the list of molecules. The data hierarchy in a target representation can differ from the one in the initial representation, and therefore has to be adapted for subsequent operations. A data set can be partitioned *semantically*, according to data attributes, such as the protein type, or *spatially*, according to geometric properties, such as the bounding box extents. The partitioning creates a scene hierarchy for each animation state. The original topology is always preserved in the initial animation state.

For the exploded view in Figure 5.2a, for instance, the data first has to be partitioned semantically into the three subsets that represent the layered cell compartments. *Split(comp, S)* where $S = m_1, ...m_j$, partitions the input data into k cellular compartments so that $S = M_1, ..., M_k$. The compartments are in our case inherently given in the hierarchy of the data format. If no such hierarchy is present, segmentation algorithms are needed to determine semantic clusters.

In the exploded view in Fig. 5.2a, the compartment subsets are *spatially* partitioned into a lower and an upper half. *Split(spatial(plane), S)*, where $S = m_1, ..., m_j$ assigns all molecules to subsets M_{upper} and M_{lower} depending on which side of a plane they are located. To achieve a re-usable spatial partitioning, inherent geometric information, such as the bounding box, can be exploited, e.g, by placing a cutting plane at the center of the input subset's bounding box. By providing operations that work on semantic data properties, the user is not required to have an extensive knowledge of the data composition.

Layout Metamorphers

Layout metamorphers *spatially rearrange* subsets of molecules that were defined in the data restructuring stage, in order to create a desired target representation. Layout metamorphers define a target state with a sequence of transformations of the positions, rotations, and scales of the molecules in a given subset.

A $rotate(hinge(axis, degree), M)$ metamorpher needs an implicit axis, such as the axis of a scene's coordinate system or the edge of a bounding box, to support re-usable rotations. In the exploded view (Fig. 5.2a), for instance, we rotate the halves of a compartment around a common hinge where their bounding boxes align. The rotation angle can be explicit, since it yields the same results for arbitrary data sets. An angle of, e.g., 90 degrees between two planar surfaces provides a view into the interior, independent of the data. Similarly, for a *translate(direction, distance, M)* metamorpher the translation direction and distance have to be chosen so that they have the same impact on arbitrary input data. Such implicit layout operations can mask the complexity of low-level geometric transformations from the user.

Morphing Metamorphers

Metamorphers from the morphing stage change the visual appearance of individual molecules (Fig. 5.4a). Such operations are needed in scenarios where the representation of individual objects needs to be adapted in order to better convey an intended message. Molecules in mesoscale biological models can be displayed as space-filling models, stick models, or one of many other representations commonly used in molecular graphics [KKL$^+$15, Goo99, Ric81]. In the bar chart in Figure 5.2a, for instance, *morph(slab, S)* transforms proteins into square-shaped slabs that are then stacked upon each other by a layout metamorpher. In our implementation, a morphing metamorpher *morph(slab/stick/cube/sphere/scale, S)* re-assigns the atoms of a given protein to random positions within the extents of the intended shape. The random positions approximate the defined shape. Due to the random positioning within the specified geometric boundaries, the approximated shape appears as the same for arbitrary input molecules. The generated positions are stored within new key frames for atom positions in the molecule's local coordinate system. Morphing metamorphers mask the low-level complexity of molecular data structures that users would need to access and modify in order to change their representation.

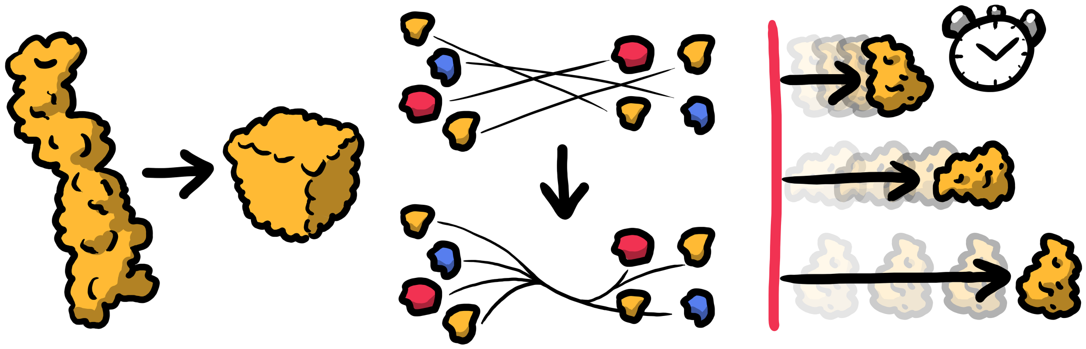

(a) (b) (c)

Figure 5.4: Schematic representations of the effects of different metamorphers from the a) morphing, b) trajectory bundling, c) timing stage.

Trajectory Metamorphers

The trajectory stage is the first of the three pipeline stages responsible for structuring the transition between two animation states (Fig. 5.3). Due to the size of the data (tens of thousands of molecules), the structure of the transition plays an especially important role, e.g., for visual clutter reduction, occlusion handling, and the guidance of the viewer's attention.

Trajectory metamorphers *spatially structure* the transition by defining the trajectories along which the individual molecules of a given subset will move. A common technique for spatially structuring the transition of particles is trajectory bundling. *Bundle(p, M)* reroutes the trajectories of the particles in M through a common point p, i.e., a *bundling point*, instead of directly interpolating between their initial and target position (see Fig. 5.4b). The position of a re-usable bundling point should be derived from the given data semantics. In the transition to a bar chart (Fig. 5.2b), we define bundling points with respect to the bounding boxes of the initial and final animation states.

Trajectory metamorphers mask the complexity of manually specifying explicit key frames for spatially structuring the transition of molecules.

Timing Metamorphers

Timing metamorphers are responsible for the *temporal structuring* of the transition between two animation states. Temporal structuring is achieved by manipulating the start time, duration, and speed of the transitions between key frames. These temporal manipulations are controlled via *time curves* that are associated with each pair of key frames.

Temporally structuring the transition of thousands of densely packed animated objects is essential to reduce the cognitive load on the viewer. A *staged* animation can reduce occlusion and clutter [HR07]. Staging determines the order and speed at which objects move, as well as the number of objects that move at the same time. Other temporal distortions that create more pleasant animations, such as slow-in/slow-out pacing [DBJ$^+$11], are handled in this stage as well. Timing metamorphers can also be used to convey information about the chronology of the illustrated events.

A *stage(distance(p), d, S)* metamorpher, for instance, delays the animation of molecules in respect to their distance to a certain point p. This distance-based delay temporally structures the transition so that molecules appear to be peeled away from their original position, layer by layer. A *stage(type, d, S)* metamorpher orders the time curves of the molecules in S by their protein type so that all molecules of the same type are animated synchronously. A delay d defines how long each molecule waits before starting its transition (Fig. 5.4c). The delay can be defined explicitly ($d = X$) or implicitly (for instance, $d = 1/j * Y$, where j represents the total number of molecules in a subset, and Y a scaling factor).

By relying on given semantic or geometric properties, timing metamorphers mask the complexity of sophisticated animation timing from the user. The resulting temporal structuring would be very difficult to achieve by manual authoring on the level of individual protein instances.

Camera Control Metamorphers

Camera control metamorphers govern the camera position and viewing direction at each time step during an animation sequence. Camera steering is an essential component of non-interactive explanatory visualizations. It effectively determines, *which parts* of a dynamic scene are presented to the viewers when guiding them through a sequence of events, as well as *how* these parts are presented, i.e., from which viewing angle and distance. A wide range of sophisticated semi- and fully-automatic camera control techniques [CON08] have been developed in the field of computer graphics. Re-usable camera control metamorphers that automatically react to dynamic changes in an animation, can rely on inherent data properties, such as molecule types and subsets, as well as their animation states (position, rotation, scale) at each time step.

A simple *camera(follow(BB), S)* metamorpher checks the extents of the input subset's bounding box and adjusts the camera zoom and look-at vector to keep the subset inside the view frustum during the animation. In more complex cases, a *camera(follow(salience), S)* metamorpher could derive semantic importances from salient features, such as objects that are currently in motion. Camera control metamorphers that react automatically to the dynamic changes in an animation, relieve the author from manually specifying camera paths in complex dynamic scenes.

5.4 Implementation

Our implementation is an extension of the cellVIEW framework [LAPV15], a tool for the real-time visualization of large mesoscale molecular models that is realized within Unity3D. The molecular models are loaded from files supplied in the cellPACK format [JAAA$^+$15]. We extend the cellVIEW data structure of atom and molecule types, positions, and orientations, with the means to create and store hierarchical subsets of molecule instances, as well as key frames and time curves (as described in Sec. 5.3.2).

The animation pipeline is realized as an application programming interface (API) in the C# programming language. All metamorphers are executed as parametrized function calls that operate on arbitrary subsets of the data hierarchy. On the C# level, new metamorphers can be defined by creating new functions that implement the interface that we defined in Section 5.3.3.

5.4.1 Interface

To allow users a more intuitive parametrization and combination of metamorphers, we implemented a node editor (Fig. 5.5) that acts as a visual scripting interface to our API. Each node in the editor represents a metamorpher. The inputs and outputs of nodes can be connected (2) to link two metamorphers. Metamorphers can be dragged from the node shelf on the left onto the canvas (1). Nodes are parametrized via the property window (4) that is accessed by clicking on a node. While the order of linked metamorphers and the applied timing operations create an implicit timing for the animation, the editor allows the user manual fine tuning. The start time and duration of a metamorpher is set via its anchor (5) to the time line (6) at the bottom of the editor. The time curve (7) for metamorphers that support timing can be accessed in a node's property window. The horizontal axis represents the time, and the vertical axis represents the progress of the respective operation. Data-restructuring metamorphers do not require timings.

Each node in the editor is associated with an *evaluate()* function that is called when the chain of connected metamorpher nodes is traversed upon execution. The *evaluate()* function calls the respective metamorpher and uses the parameters that are defined in the node properties. It passes the metamorpher output to the input of the next connected metamorpher node in line. For an additional level of control, experienced users can access and edit the C# code of *evaluate()* functions and the associated metamorphers by double clicking on a node. Upon saving, Unity3D will automatically re-compile and accommodate the changes.

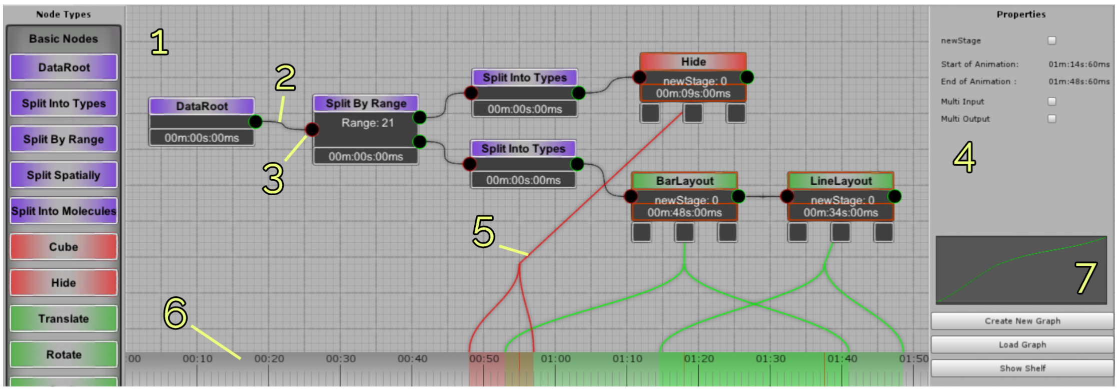
5.4.2 High-Level Metamorphers

High-level metamorphers mask the functionality of multiple metamorphers contained in a hierarchical group node. The node editor enables the user to create and store such group nodes on the fly. These high-level metamorphers are flexible and re-usable according to our definition from Section 5.3.1 as well. The contents of a high-level node still can be accessed and modified by users who have the required expertise and domain knowledge.

5.4.3 Producing the Animation

An animation is defined by a chain of linked metamorphers. The individual frames of an animation are produced by interpolating the animation states that are stored in the key frames of individual molecules in consideration of the specified time curves. The number of generated frames per second can thereby be specified by the user. To create the sequence of frames for each molecule, the data hierarchy is traversed. The transformations of higher-level nodes in the hierarchy are propagated to the transformations on lower levels, down to the individual leaf nodes, i.e., the molecules. For displaying the animation, the frame sequence is loaded onto the GPU, and the animation is played back in real-time within cellVIEW. The last state of a metamorpher setup can be inspected at any time during the authoring process by pressing the *evaluate* button.

Our current implementation interpolates between key frames on the CPU. Linear interpolations between key frames can be achieved in real-time, i.e., the frame sequence does not have to be calculated before the animation is played back. If key frames require cubic interpolations, the sequence has to be calculated prior to playback. The generation of the frames for the videos in the supplemental material [SMR+] took between 10 and 20 seconds on current consumer hardware.



newStage ☐

Start of Animation: 01m:14s:60ms

End of Animation : 01m:48s:60ms

Multi Input ☐

Multi Output ☐

Figure 5.5: The node editor acts as a visual scripting interface to the metamorpher API function calls. 1: the canvas for arranging metamorpher nodes, 2: a link between input & output of two metamorphers, 3: input handle of a metamorpher node, 4: a node's property window, 5: a time anchor, 6: the time line, 7: a node's time curve.

5.5 Results

In this section, we showcase the authoring of the two animations illustrated in Figure 5.2, i.e., the transition to an exploded view and the transition to a bar chart. We thereby refer to the metamorphers that we introduced in Section 5.3.4. We demonstrate the re-usability of our technique by applying each metamorpher setup to three molecular data sets, i.e., to computational models of the immature HIV virion, the mature HIV virion, and the mycoplasma bacterium. The resulting animations can be viewed in the supplementary video [SMR+].

The first two data sets describe different development stages (immature and mature) of the structural model of the human immunodeficiency virus (HIV), which is built-up from more than 20.000 macromolecules. The HIV is surrounded by blood serum, and therefore not visible in the initial state of the data set (Fig. 5.6a). The mature virion features better defined hierarchical compartments and also differs in the types and spatial distribution of proteins, in comparison to the immature virion. Both data sets have an onion-like hierarchical structure. The mycoplasma bacterium, while spherical as well, has a less developed hierarchy. Most of its proteins are distributed loosely within its interior.

5.5.1 Exploded View

The three data sets that we present are all densely packed. Their outer shells obstruct the view to inner structures. In this example, we want to reveal the inner structures as well as the hierarchy in which they are arranged. We therefore create an exploded view that opens the individual compartments and places them side by side.

We start by creating the data structure that we need for the final transition state. First, we create molecule subsets that represent the individual compartments. We therefore feed our initial data S into a *split(comp, S)* metamorpher that yields i subsets M_i for S, where i is the number of compartments. Second, we want to split all compartments horizontally at their center. We achieve this by applying a *split(plane(*BB_{center}*), S)* metamorpher that uses a plane at the bounding-box center BB_{center} of each subset M_i.

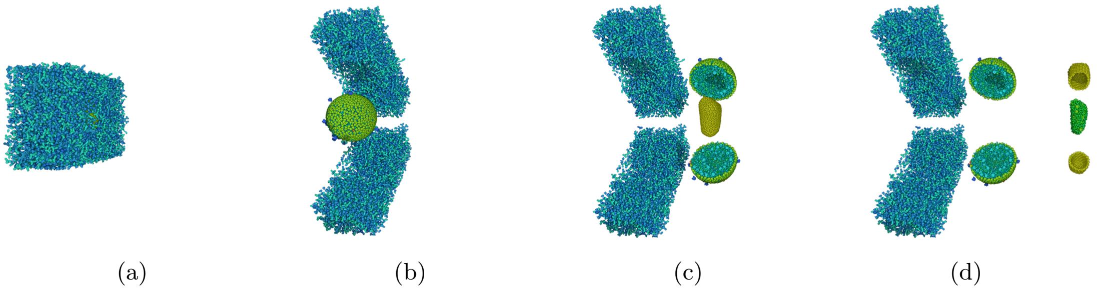

(a) (b) (c) (d)

Figure 5.6: Exploded view transition of the mature HIV dataset.

This will add an additional hierarchy layer to the data structure by splitting each M_i into an $M_{i_{upper}}$ and $M_{i_{lower}}$.

Next, we define the layout for the data structure that we created. To explode (or open up) the individual compartments, we rotate the lower and upper compartment halves by 45 and -45 degrees respectively. As rotation hinge we use the upper (back) and lower (back) edge of their bounding boxes. We pass the upper compartment halves to a *rotate(hinge(*BB_{lower}*, 45),* $M_{i_{upper}}$*)* metamorpher and the lower ones to a *rotate(hinge(*BB_{upper}*, -45),* $M_{i_{lower}}$*)* metamorpher.

To layout the exploded compartments side by side, we use a *translate(x-axis,* BB_{width}*, S)* metamorpher. The children of S, i.e., the compartments, will thereby be placed along the x-axis with respect to their bounding box width. We do not change the appearance of individual molecules, so no morphing metamorpher is needed.

After defining the target representation, we design the transition to it. We want the individual compartments to be revealed one by one. We simply apply a *stage(subset, S)* metamorpher that individually delays the transition of each compartment, i.e., the child nodes of S. Since we move individual compartments as a whole, no trajectory metamorpher is required. We use a *camera(maintain, S)* metamorpher to guarantee that the data stays inside the view frustum during the transition.

We group the final chain of metamorphers into a high-level *explode(S)* metamorpher that can now be applied to different data sets in order to create an exploded view of their hierarchical compartments. The resulting transitions can be seen in Figures 5.6a-d for the mature HIV. Figures 5.7a and 5.7b show the results of the same metamorpher sequence on the immature HIV and the mycoplasma data set. Even though the number of compartments and the associated molecule types differ in each data set, our chain

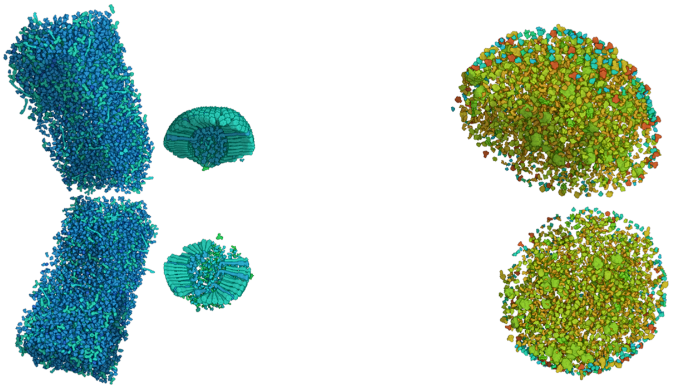

(a) (b)

Figure 5.7: The final animation state for the exploded view of the immature HIV dataset (a), and of the mycoplasma dataset (b).

of metamorphers creates an appropriate exploded view for each of them. Since the mycoplasma data set does not have a hierarchical structure, the application of an exploded view might not be required. Still, even here the result is semantically correct in regard to the desired target visualization.

5.5.2 Bar Chart

In the bar chart example, we convey knowledge about the quantities of proteins in a microorganism. We choose to represent these quantities in a bar chart. Each bar in the chart represents the volume that the respective molecule type occupies in a given structure.

Since we want to show the quantities per molecule type, we partition the data with a *split(type, S)* metamorpher into i subsets M_i, where i represents the number of molecule types. To create the individual bars, we apply a *translate(y-axis, length,* M_i*)* metamorpher that will stack the molecules of a specific type along the y-axis in intervals of the defined length. The length is defined by a user-specified maximum bar height, normalized by the maximal number of molecules among all molecule types. Next, to give the stacked molecules the appearance of a bar, we use a *morph(slabs, S)* metamorpher in order to change the original molecule shapes into square slabs of equal size. To align the bars side by side along the x-axis, we apply a *translate(x-axis,* BB_{width}*, S)* metamorpher. The bounding-box width corresponds to the width of the slabs in each stacked bar.

In order to increase the readability of the animation, we structure the transition spatially and temporally. For the spatial structure of the transition, we re-route the trajectories of the molecules so that they do not move though the virus on the way to their target positions. Instead, we want the molecules to move to trajectory-bundling points to the

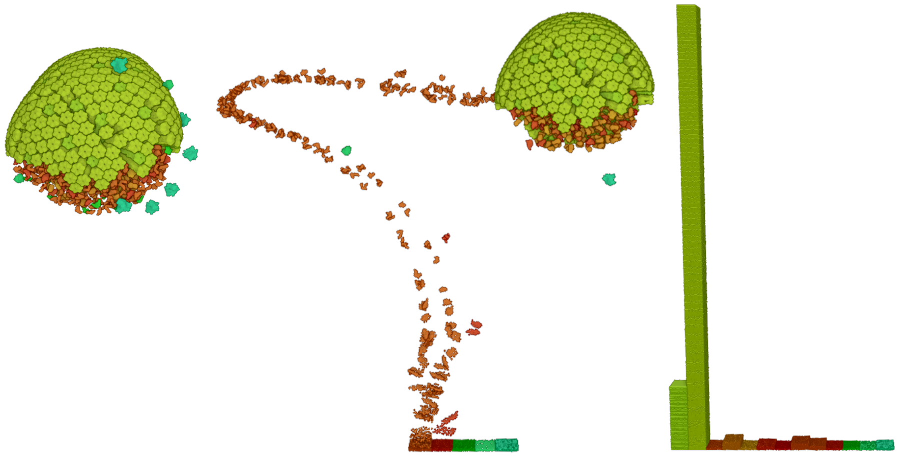

(a) (b) (c)

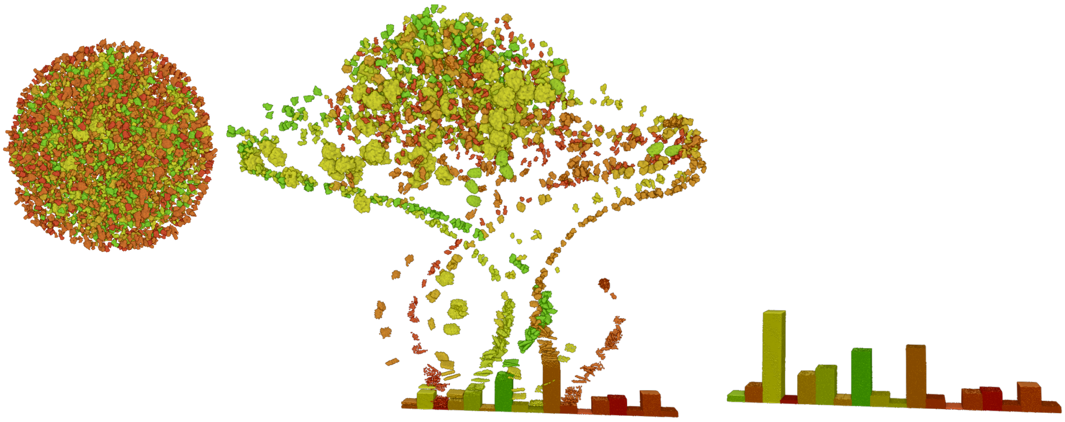
Figure 5.8: The bar chart metamorpher setup applied to the immature HIV data set.

right and left of the initial structure. We therefore partition the data further by applying a *split(plane(*$BB_{center.yz}$*, S)* metamorpher that splits each typed subset M_i at the yz-plane of the bounding-box center into $M_{i_{left}}$ and $M_{i_{right}}$ halves. The newly created subsets are now used by a *bundle(*BB_l*,* $M_{i_{left}}$*)* and *bundle(*BB_r*,* $M_{i_{right}}$*)* metamorpher to create bundling points on the left and right side of the virus structure's initial bounding-box.

We also want to create the impression of molecules falling into their respective bar from the top. We achieve this with a *bundle(*BB_{top}*, S)* metamorpher that creates a bundling point for each subset of S, at the top of each histogram bar. The trajectory-bundling metamorphers are inserted before (BB of the initial structure) and after the layouting (BBs of the bars), depending on which structural information they need to relate to.

In order to *temporally structure* the transition, we first stage the animation per molecule type (*stage(type, S)*). Then we apply a distance-based delay per molecule (*stage(distance(*BB_{center}*), S)*) to peel away the molecules layer by layer depending on their distance to the initial bounding-box center. Finally, we use a *camera(maintain, S)* metamorpher to guarantee that all molecules stays inside the view frustum during the transition.

The results of applying this sequence of metamorphers to two different data sets can be seen in Figures 5.8 for the immature HIV, and Figures 5.9 for the mycoplasma data set. The result for the mature HIV data set is displayed in Figure 5.2b. The intended result, i.e., a transition to a bar chart of protein quantities, has been created successfully for all three data sets. The principal requirement for this is the existence of typed entities (molecules) in the data.



(a) (b) (c)

Figure 5.9: The bar chart metamorpher setup applied to the mycoplasma data set.

5.6 Discussion

5.6.1 Expert Feedback

In order to judge the feasibility of our technique, we gathered informal feedback on metamorphers from three experienced biological illustrators. They commented that their work flow to create complex illustrations would greatly benefit from metamorphers. They consider the produced results as *"incredibly valuable"* to communicate knowledge. The current visual-scripting interface received some criticism for being not very intuitive. One expert suggested that the interface could benefit from *"icons (or animated icons) that show what each effect is roughly intended to perform"*.

According to one of the experts, setting up an illustration using his accustomed work flow *"would still take me a minimum of one hour to create a prototype, and then several hours or a couple of days to refine it"* – despite having 17 years of experience with 3D modeling software. With metamorphers, he claims, he could *"iterate on it in real-time in a matter of minutes"*. In addition, most of the illustrations that result from the accustomed work flow are not reusable or portable to other data without significant customization. The re-usability of animation setups was therefore regarded as highly valuable.

5.6.2 Applicability

While our technique is in principle applicable to different data types, it is especially valuable for large-scale molecular data. On the one side, the authoring of illustrative animations for large-scale molecular structures is a cumbersome task, as reported by domain experts. On the other side, large-scale molecular data bridges the disciplines of molecular visualization and cellular visualization, as microorganisms are depicted on molecular/atomic resolution. In terms of molecular visualization, the visual transformation of individual elements in a data set is common, as many different well established visualization techniques exist. In the visualization of complex biological structures, illustrative abstractions of the organic structures and compartments are commonly applied to convey specific information about an organism. Metamorphers are especially suitable for this type of data, as abstractions of both molecular and biological structures are supported by our approach, i.e., in the morphing and layout stages respectively.

Our technique supports in theory arbitrary animations of typed data with point cloud characteristics. It is especially well suited for the purpose of creating short illustrative animations, most notably, transitions between different representation forms. Re-usability is mainly a benefit in application scenarios where the same illustrative technique is frequently re-applied to different data sets. An example is to highlight a certain characteristic in a series of related data sets for the purpose of comparison. Stories with a more complex plot typically do not need to be re-usable, in the sense that the contained animations and transformations are applied to different data sets. For complex stories, the overhead of creating metamorphers would most likely outweigh the benefit if re-usability is not a concern.

The re-usability of our technique remains intact as long as the conditions in Section 5.3.1 are met. An explicit description of an animation state might require less effort than programmatically creating multiple metamorphers from scratch. However, once a metamorpher is defined, it can instantly be applied to other compatible data as well. The modular interface further broadens the scope of achievable results in combination with already defined metamorphers. The potential results are only limited by transformations that cannot be defined in relation to implicit semantic or geometric properties of the data.

5.6.3 Future Work

There are several directions that we consider as possible extensions to the concept of metamorphers. In terms of general applicability, our method could be adapted to be used with point clouds, or volumes in general, as the molecular data is represented by a set of points in 3D space. The topologies of objects that are formed through the spatial proximity of individual elements in point clouds and volumes are not changed, e.g., when spatially splitting a data set along a plane. For the application to polygonal data, however, our approach would have to be extended to support adaptions of the mesh topology.

Another interesting direction for further research is the transition between multiple scales at different orders of magnitude, such as the transition from the scale of a blood cell to the scale of the human body as presented by Hsu et al. [HMC11].

Regarding the categories in the design space for storytelling in visualization that Segel and Heer formulated [SH10], our technique gives the animation author great flexibility in terms of visual structuring, highlighting, transition guidance, and ordering of the presented content. Categories that our technique currently does not support explicitly are interaction and messaging (annotations). These two aspects are priorities for follow-up work, in order to further extend the storytelling capabilities of metamorphers. In theory, interaction with the animation is already possible, since the environment, in which we implemented our method [LAPV15], renders the animated transitions in real-time.

The continuous presentation of animations supports an intuitive understanding of the relationship between representation states. If carefully authored, such an animated transition can be both self-explanatory, and enjoyable to audiences [RFF⁺08]. However, the potential downsides of animations are characterized by the fleetingness of the displayed information, the potential sensory overload, and the distracting nature of badly designed animations [TMB02]. In consideration of these challenges, another way to extend our approach would therefore be to compensate the fleetingness of animations. The metamorpher output could thereby be sparsely sampled in order to produce a series of static images. This could be achieved, for instance, with object-based key-frame extraction techniques, as described in previous work [LLWC08, HCHY05]. Automatically generated glyphs and labels could thereby be integrated with the static images to compensate the information loss that the sampling of the animation caused, for instance by encoding directions, velocities and orderings.

Regarding the sensory overload of badly designed animations, an interesting direction for future work would be the incorporation of user guidance into the authoring tool, e.g., within the visual scripting interface. By determining, for instance, visibility in the intermediate animation states, a user could be notified about animation sequences that could potentially overwhelm or confuse a viewer.

The set of metamorphers that we presented in this paper serves as a proof of concept to illustrate our technique. As such, it represents by no means a complete feature set. Therefore, another exciting prospect would be the creation of a data base where users can share the metamorphers that they defined, as well as find and modify existing ones that were created by other users, in order to continuously increase the pool of available story telling templates.

Metamorphers represent a first step in the direction of a formalization of animated transitions for 3D spatial molecular data sets. As a next step, we intend to publish a formal grammar describing our animated transition pipeline in general and the individual metamorpher implementations in particular. The idea is to explore the solution space that can be created by a set of metamorphers by applying the grammar for the generative sampling of all potential metamorpher parametrizations.

5.7 Conclusion

Metamorphers allow users to create re-usable animations for molecular data sets. The re-usability has the advantage that representation states and transitions between them do not have to be manually re-modeled and key-framed when they are applied to different data sets. Instead, users can create animations by combining pre-made templates, i.e., metamorphers. The same chain of metamorphers can thus be used to generate animations for multiple data sets, while individual metamorphers can be re-used in different combinations. Domain experts from illustration and animation confirmed the feasibility of our approach. The set of introduced metamorphers already demonstrates the flexibility of our technique, but the presented list of operations is by no means complete. To enable the extension of the list of achievable animations, our interface provides access to the source code of individual metamorphers. It thereby allows users with programming expertise to modify existing metamorphers, or to create new ones. Users that are less experienced in programming, animation, or molecular data can access the list of existing metamorphers and combine them in the visual scripting editor. The grouping to high-level metamorphers thereby hides complex low-level animation concepts. In this way, metamorphers can be exchanged between users from different domains and of different levels of expertise.

Metamorphers as a visual integration technique enable the adaption of the visual channels of a spatial representation, as discussed in Section 2.2, as well as the definition of the temporal integration. The adaption of the visual channels is thereby handled by different types of metamorphers. The spatial channels are defined by the layout metamorphers. Morphing metamorphers handle the retinal channels. Morphing also governs shape transformations on the level of individual molecules. The motion channel is managed by timing metamorphers, as they can influence how two spatial or retinal channels are interpolated. The timing of the interpolations thereby temporally structures the transition. Trajectory metamorphers create intermediate states for the spatial channels in order to spatially structure the transition. The definition of a temporal integration between a pair of representations is thus supported by trajectory and timing metamorphers. Camera control does not describe the integration between visual channels, but rather the perspective from which the information is presented. As such, it corresponds to a guided navigation component, as described in Section 2.4.

When regarding the functional integration capabilities of metamorphers, the six animated transition pipeline stages, i.e., metamorpher types, can be related to the three visualization components that we defined in Section 3.3.4. The data component corresponds to data restructuring metamorphers. The visual component corresponds to layout, morphing, trajectory, and timing metamorphers. The navigation component corresponds to camera control metamorphers.

Functionally integrated interaction on the individual visualization components is currently not explicitly supported by metamorphers. The events in the presented scenarios are neither triggered through user interaction nor coordinated through functional integration, but rather through the scripted metamorphers. An essential exception to this are camera control metamorphers that can indeed be functionally integrated with scripted updates of the other types of metamorphers. They thus correspond to a $V \rightarrow N$ or $D \rightarrow N$ integration, depending on the type of scripted event that triggered the camera adjustment.

In this chapter, we explored an integration technique that is capable of presenting the relation between two representation forms of a single data set. Next, we will explore a related approach that concerns the presentation of relations between different states of a molecular data set.

CHAPTER 6

Masking Missing Information Via Visual Abstraction

In Molecular Visualization

This chapter is based on the following publication:

Johannes Sorger[1], Peter Mindek[1], Tobias Klein, Graham Johnson, Ivan Viola. *Illustrative Transitions in Molecular Visualization via Forward and Inverse Abstraction Transform.* In Eurographics Workshop on Visual Computing for Biology and Medicine (VCBM), pages 21-30. September 2016 [SMK+16].

Presenting the relationship between *different representations* of the same data set has strong parallels to conveying relationships between *different data sets*. In both cases, a relation between the displayed visual marks has to be established. There is one notable difference, however: the relationship between different representations of the same data is implicitly given through the data items. Each representation is based on a different visual encoding or on a different mapping of attributes to visual channels – but each mark, no matter how the mapping is done, still corresponds to the same data item across representations.

Between two data sets this relationship does not have to be implicitly given. In structural biology, for instance, detailed models of viruses or bacteria at different development stages are generated at a high level of detail. However, the processes that describe the relation from one stage to another one are harder to re-construct and are therefore often not clear. If the relation between two models can only be specified at a lower level of detail than the actual models themselves, the relationship cannot be visualized at the same visual level of detail as the models.

---

[1] Johannes Sorger and Peter Mindek contributed equally to this work.

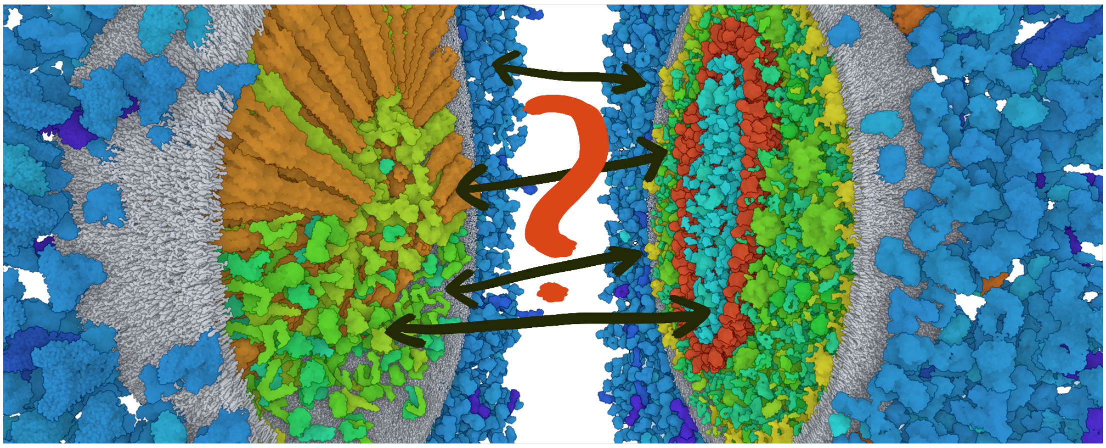


Figure 6.1: The chemical composition of HIV can already be accurately described at the level of individual molecules for the immature (left), as well as the mature state (right) of the virus. Comparatively little is known about the maturation process, i.e., how exactly the two states relate to each other.

In this chapter, we address the challenge of presenting the relation between development states of molecular models where the actual biochemical process that describes the relation is not exactly known or it is too complex to model. Our solution proposes the application of visual abstraction to mask details that cannot be related across two data sets due to missing information in the description of their relationship. Temporal integration is thereby used to establish the relationship between the abstracted and the non-abstracted representation, as well as the relationship between the two data sets. Instead of directly interpolating between two models that show different states of an organism, we gradually transform the models into a level of visual abstraction that matches the level of detail of the modeled relation between them. At this level, the models can be interpolated without conveying false information.

6.1 Introduction

Biologists often utilize illustrations and 3D animations to communicate their knowledge to different audiences, such as students or the general public. However, the acquired scientific knowledge about biological phenomena is often sparse and the processes can usually not be modeled with absolute accuracy. While a lot is known about a particular biological system, there is usually still even more unknown. Illustrators therefore have to deal with uncertainties when utilizing these biological models. In order to avoid the conveying of false information, they have to respect the information incompleteness when visualizing biological phenomena.

For instance, we have a relatively good understanding of the chemical composition of HIV at various stages of its development, such as the immature and mature stages of the

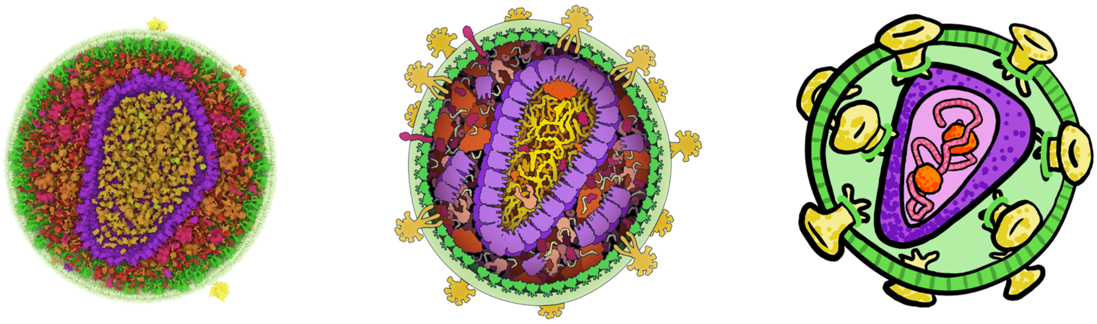


Figure 6.2: Visually abstracted representations of the mature HIV virion. (a) The computational model at atomic detail. (b) A schematic illustration of the virus depicting the silhouettes of individual proteins. The organic shapes and an impression of the depth and layered structures are still retained. (c) A stylized illustration of the structural virus composition. Only certain proteins that are responsible for the characteristic visual impression of the virus are shown.

virus. We know which proteins from an earlier stage build up structures in later stages. However, it is not exactly known in which order the chemical reactions are triggered. Structural biologists are able to build computational models of the immature and mature HIV, which are relatively accurate. However, they cannot model the process of HIV maturation, which is the transition between these two stages, with the same level of accuracy because of the missing information about this process, as depicted in Figure 6.1. This has to be taken into account when these models are used to illustrate the process of the transition of the HIV between its development stages.

When the specifics about certain developments are unknown, scientific illustrators can only show these processes in abstracted ways, e.g., by choosing a viewpoint or representation where the unknown is not shown. Illustrators reduce the level of detail of the modeled states to match their knowledge about the process, so that the created animation does not convey false information. These abstracted representations are created manually, according to the current knowledge in biology and the illustrator's artistic expressions. Figure 6.2 illustrates the sequential visual abstraction of an HIV virion to a more simplified schematic representation.

The creation and the maintenance of these animations in regard to new research results is a challenging task. When the current knowledge in biology changes due to new scientific discoveries, the illustrations and animations may become out of date and they need to be recreated. This is a time consuming task, since, depending on its complexity, it can take days or weeks weeks to create a 3D animation describing a biological process.

To avoid the manual re-creation of illustrations and animations when new knowledge about a phenomenon is gathered, biologists produce complex computational models and simulations, which describe the biological processes according to their current knowledge. The communication of these phenomena is then carried out by the visualization of these models. When the biology knowledge changes, the computational models are simply updated, while the visualization pipeline remains the same [MKS^{+}17]. This enables much shorter turnaround times for the communication of scientific results with illustrations and animations.

In this chapter, we propose a method for the automatic creation of visual transitions between different molecular models. Our method deals with the challenge of conveying missing information about the relation between two models. The relation describes the biological process that transforms one model to another one. Our approach supports the depiction of those processes at multiple levels of visual abstraction, depending on the degree of information that is available.

To enable the interpolation between models at different levels of visual abstraction, we automatically generate several representations with varying level of detail, which we refer to as *levels of visual abstraction*. It is possible to smoothly interpolate between those representations in order to continuously reduce the level of detail of a visualized model. Before transitioning between two models, we therefore first interpolate to a higher level of visual abstraction, where the interpolation between the models would be meaningful and would correspond to our knowledge about the process. We then display the transition between the models at this visual abstraction level, and apply the inverse transformation back to the original representation of the model, i.e., the lowest level of visual abstraction, which corresponds to the highest level of detail. The process is depicted in Figure 6.3. In this way, we are able to perform a meaningful interpolation between two models, which sparsely represents a biological process or phenomenon. This ensures that the generated animation does not convey false information about the illustrated biological process.

The main contribution of this chapter is a method for creating illustrative transitions between molecular models, such as different development stages of a virus, where the actual process of the transition is not exactly known or it is too complex to model. We specifically address the following challenges in our design:

- The visual abstraction of multi-scale molecular data.
- The transition between different levels of visual abstraction.
- The transition between different multi-scale molecular models at different levels of abstraction.

6.2 Related Work

Since our method comprises elements from different visualization areas, the discussion about related work is split into schematic illustrations of molecular data, and visualization approaches that illustrate or compare change.

6.2.1 Schematic Illustrations of Molecules

Molecular data and corresponding processes on a molecular level are highly complex. Schematic representations are therefore often utilized to convey knowledge about these processes on a visually more abstract level in order to allow a focus on the information that is relevant in the corresponding context. Goodsell [Goo99] separates molecular representations into atomistic and continuous models. The schematization of atomistic models usually shows either chemical bonds or the surface of a molecule, whereas illustrations of continuous models visualize derived properties of the underlying molecules. Falk et al. [FKRE09] especially emphasize the analysis of signal transduction and therefore propose a schematic visualization that depicts individual molecules and their tracks, or reactions. With the introduction of a cartoon representation for molecules [Ric81] more abstract illustrations [HOF04, WB11] were proposed, especially to schematize the structure of molecules. Cipriano and Gleicher [CG07] propose a simplification of the molecular structure that preserves significant shape features. They utilize symbols placed on the surface to visually encode additional properties. Closely related to our work is the approach by Zwan et al. [VDZLBI11], which is able to gradually transform the visualization of one molecule between different degrees of visual abstraction. In contrast to their approach, we focus our visual abstraction approach on the quantitative and structural aspects of complete models of viruses and bacteria.

6.2.2 Visualization and Understanding of Change

Comparative Visualization. The comparison and the establishment of relationships between complex objects is often a challenging task. Comparative visualization provides support in fulfilling those tasks. The design of comparative visualizations is traditionally categorized into juxtaposition, superimposition, or explicit encoding of relationships [GAW$^+$11]. Vivek et. al [VP04] argue that side-by-side comparisons impose the task of finding and quantifying differences on the user instead of on automation. Since we are directly comparing two models to each other, we can use a temporal integration that conveys the relation of organic structures and quantities in an intuitive way.

Small multiples [Tuf90], coordinated multiple views (CMV), and static images in conjunction with traces and glyphs are typically used to visualize transitions, trends, correspondences or sequences [RFF$^+$08]. Hsu et al. [HMC11] present an automated method for the illustrative visualization of multi-scale phenomena. They generate an image that contains multiple levels of detail of one subject by blending the renderings of multiple pin hole cameras at different zoom levels. VisLink [CC07] establishes relationships between two 2D data representations by placing them in 3D space and drawing edges between the corresponding visual marks. Such a visual linking of related objects could also be feasible for 3D molecular data. However, due to the inherent occlusion in 3D spatial representations, the linking would need to be established between visible parts of cross-sections of the individual models.

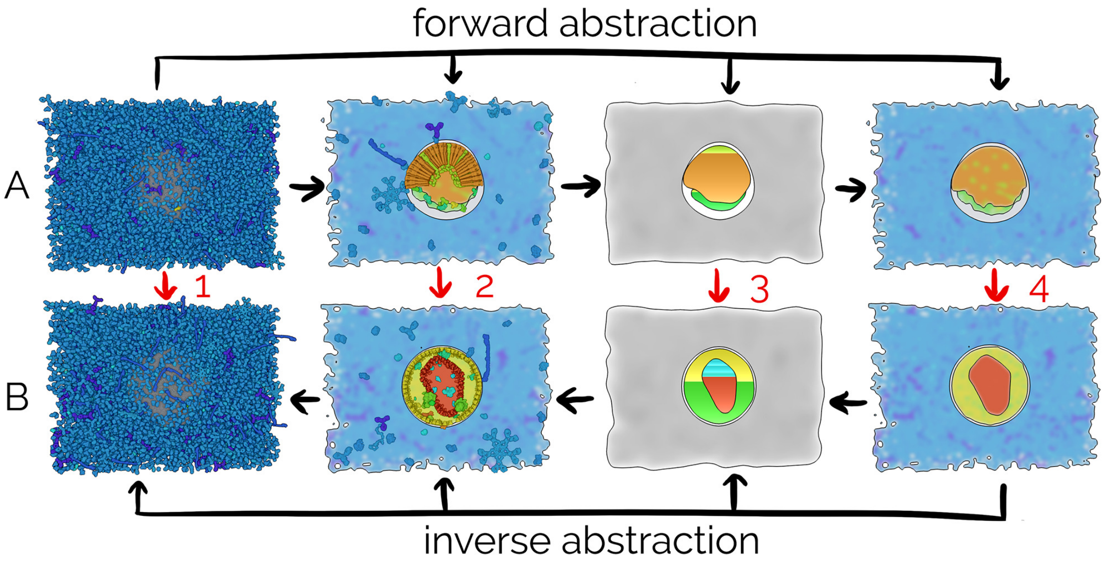


Figure 6.3: Four levels of visual abstraction used to illustrate a transition between the model of an immature HIV virion (A) and a mature HIV virion (B). The level of information that is available about the transition between these two states determines, at which level of visual abstraction the interpolation between the models should take place. We can smoothly interpolate between the levels of visual abstraction (black arrows), in order to visually abstract the models to a level, at which the interpolation meaningfully illustrates our level of knowledge about the transition (red arrows).

Animated Transitions between Visual Representations. Tversky et al. [TMB02] summarize cognitive studies on the benefits of animation. Although, they conclude that animation alone has not been convincingly demonstrated to be superior to static illustrations, other findings of their study suggest a direction for research on animations in visualization. They report that animation together with basic interaction methods like pausing, partial re-playing, zooming, and change of perspective might be the key to enhancing the effectiveness of animations. Robertson et al. [RFF+08] state that animated transitions of data is more enjoyable and exciting to users. They also found that it was the fastest technique for conveying trends in presentation scenarios but less exact and less effective for analyzing data. We give a detailed discussion of animated transition approaches in Section 5.2.

6.3 Forward & Inverse Abstraction

Illustrators want to show molecular models at the highest level of structural detail that is available to them. At the same time they want to avoid conveying false information about the relations between models. Our method addresses this challenge with *forward* and *inverse* abstraction. By abstracting a highly detailed structural model down to the level of detail, on which the mapping information between development states is

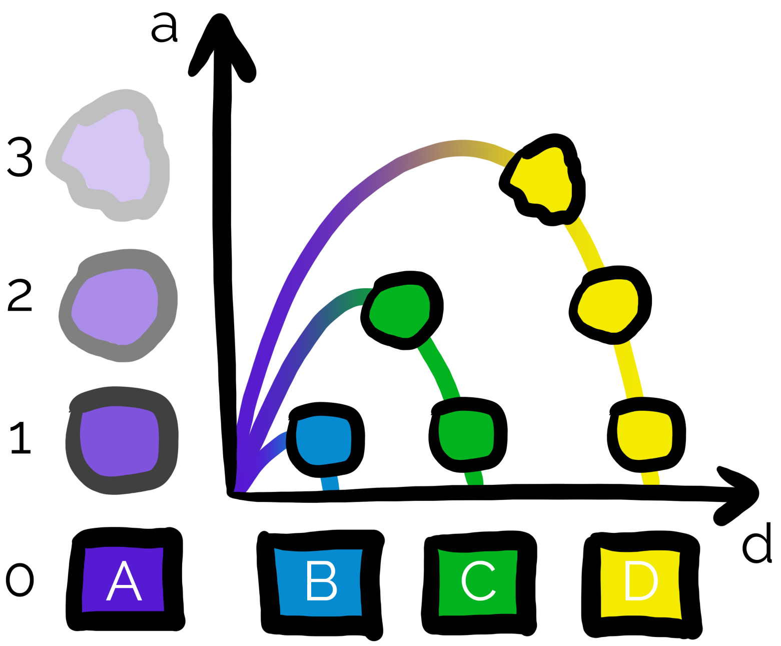


Figure 6.4: Transition between model *A* and models *B*, *C*, *D*. Axis *d* represents the incompleteness in the mapping between the models. The more incomplete the mapping is, the farther away both models are in mapping space. Axis *a* represents the level of visual abstraction. The higher the distance in mapping space between two models, the higher is the level of visual abstraction that is needed to interpolate between them.

available, an interpolation from one state to another one is possible without conveying false information. We refer to this step as the *forward abstraction*. After the transition between the two states is completed on the required level of visual abstraction, we decrease the degree of abstraction to the highest level of detail available by applying the *inverse abstraction*. The procedure is schematically depicted in Figure 6.3. With this approach we can display both models as well as the relations between the models at the highest level of detail available.

This strategy is depicted in Figure 6.4. Axis *d* describes the degree of incompleteness, in which the relation (or mapping) between the two models is described. The further away two models are on this axis, the less detail is available in terms of how one model relates to the other one. We denote this as the distance in *mapping space*. A distance of zero in mapping space corresponds to a complete description of the relation between two models (independent of how detailed the models are). If the models are described on a higher level of detail than the relations between them, the distance in mapping space increases. Axis *a* describes the levels of visual abstraction of a model. The further two models are apart in mapping space, the higher we have to abstract both models before we can interpolate them. We have to abstract the models to the level of visual detail that corresponds to the level of detail, at which the relation between the models is described.

This relation between the distance in mapping space and the required level of visual abstraction is a symmetrical one. We depict this symmetry by the curves that are drawn between two models in Figure 6.4.

The number of visual abstraction levels, as well as the criteria for measuring distances in the mapping space between two models depends on the type and complexity of the presented data. In the case of molecular models at the scale of complex viruses and simple bacteria, we define four levels of abstraction on Axis a, as well as four discrete distances in mapping space on Axis d.

6.3.1 Molecular Data

The molecular data, which we demonstrate our method on, describes biological organisms at atomic resolution. Biologists model their mesoscale data according to so called recipes [JAAA+15] that describe the molecular and structural composition of a model. These recipes are executed by a packing algorithm that populates the space with the specified macro molecules. The result is a 3D molecular model of an organism that consists of tens of thousands of molecular instances, each comprised of hundreds or even thousands of atoms. This comparatively large number of molecular instances originates from a relatively small number of protein types. In the case of the HIV virion, 20.000 molecular instances are distributed across 42 different types of molecules. The molecules are densely packed within their respective compartments of the model and also constitute the walls of the compartments.

For this type of molecular data we devised four levels of visual abstraction, as displayed in Figure 6.3 for the immature and mature models of the HIV virion models. The corresponding four discrete distances in mapping space that we identified are depicted in Figure 6.5. In the following, we describe the four levels of visual abstraction that are necessary to match two models at different distances in mapping sapce.

6.3.2 Level 0 - Implicit Relations

Distance in Mapping Space: The cellular data is depicted at atomic resolution. A transition between two models at this level of detail requires a definition of relations between the individual protein molecule instances in each model. This corresponds to a distance of zero in mapping space at the lowest level of abstraction. Processes at this level of detail describe, for instance, how individual proteins split up and merge to form new proteins and structures, e.g., in the life cycle stages of a cell. The presentation of processes that transform a biological data set from one state to another one at this level of detail is typically handled by simulations, i.e., at an implicitly described or procedural level, as manual matching for up to hundreds of thousands of molecules is virtually impossible.

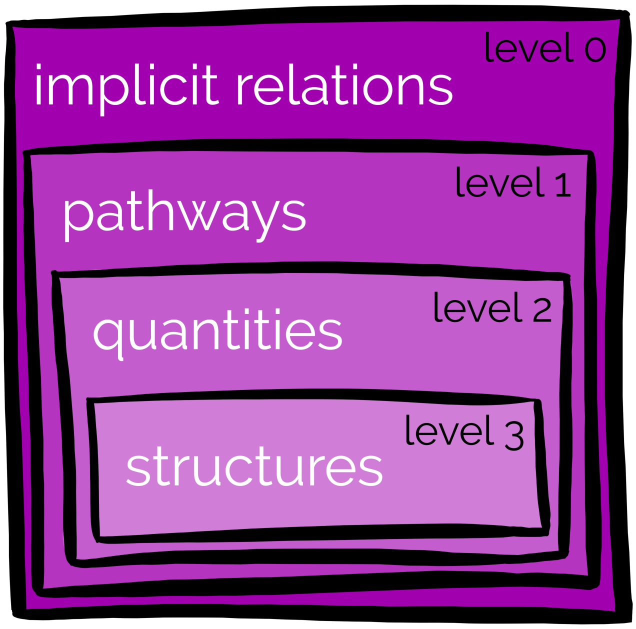


Figure 6.5: Distance in mapping space: Depending on the amount of available mapping information, the corresponding level of visual abstraction (levels 0-3) has to be used for a transition between two models. If only information about high level structure correspondence is available, we have to perform the transition at visual abstraction level 3. If we also have information about the relation of protein quantities per structure across models, we can perform the transition at level 2. If we also know about the protein pathways that connect the chemical composition of one model to the composition of the other one, we only have to abstract to level 1 before the transition. If the implicit relations between individual molecules are given, we do not need to abstract the model any further to avoid conveying false information.

Forward Abstraction: The lowest level of visual abstraction corresponds to the highest level of detail that a molecular model is represented in. We therefore do not need to abstract the model representation. Nevertheless, this level is still represents abstraction from reality, as the data is not measured but computationally modeled and simulated. Typically, only macro molecules are represented while smaller, more common molecules, such as water molecules, are omitted from the presentation.

6.3.3 Level 1 - Pathways

Distance in Mapping Space: If the relation between individual molecule instances cannot be described accurately, we have to visually abstract a molecular model to level 1. At this level, the relationship can still be described in terms of *protein pathways*. This implies that knowledge about the relation between protein types in each model, as well as the relation between *protein quantities* is available. For instance, 50% of protein A combine with 30% of protein B to result in protein C in the other model. This further implies that the relation between *protein shapes*, as well as between *high level structures* in the biological organism, such as cell membranes, is known.

Protein pathway relations thus require the matching of protein types across models, for instance, in terms of how they split up in one model to merge into new protein types in the other model.

Forward Abstraction: This degree of incompleteness of mapping information corresponds to a distance of one in mapping space in respect to the lowest level of abstraction (level 0). By visually abstracting the model representation to this level, the distance in mapping space becomes zero and a smooth transition between the two models that avoids false information is possible.

In order to visually abstract a model from level 0 to level 1, we reduce the number of displayed molecules to only a few representatives for each type. The representatives thereby can be used to encode the protein pathway information between models. Through the reduction of molecules, the implicitly encoded information about protein quantities is lost. We therefore choose the number of representatives so that it still conveys the relative quantities between protein types. To convey the relation between two models at this level of visual abstraction, we interpolate between the shapes of matching protein types and the shapes of high level structures, as well as between the number and the localizations of protein type representatives across compartments.

6.3.4 Level 2 - Quantities

Distance in Mapping Space: If protein pathway information is not available, we cannot relate individual protein types to each other anymore. The molecular models thus have to be further abstracted before conveying relationships between them. At visual abstraction level 2, we assume that the relation between protein type quantities and high level structures across models is known.

This degree of incompleteness of mapping information corresponds to a distance of two in mapping space with respect to the lowest level of abstraction. We therefore have to transform the models to the second visual abstraction level before we can interpolate between them properly.

Forward Abstraction: Since we cannot show the relation between protein types due to the missing pathway information, we have to remove the representations of individual proteins from the molecular model in order to visually abstract a model from level 1 to level 2. What is left is the information about high level structures, i.e., compartments, in the model. Again we have to substitute the information about protein quantities that was conveyed by the number of protein representatives in the previous level of abstraction. Since at this level, we are only left with representations of the structural compartments of a molecular model, we have to *visually integrate* this information with the compartment representations. One feasible strategy for the visual integration of the quantitative information would be to *nest* the non-spatial quantitative information within the spatial context of the cellular compartments. The concrete strategy that we applied in our implementation will be discussed in the following Section 6.4.4.

6.3.5 Level 3 - Structures

Distance in Mapping Space: If the relation between protein type quantities across molecular models is unknown, we also have to remove this information from the visual encoding in order to relate two models to each other. At the highest level of visual abstraction therefore only information about high level structure correspondences remain.

Forward Abstraction: To further increase the level of visual abstraction, we simply remove the visually integrated information of protein quantities that we introduced in the previous level to the model. This leaves us with the bare representation of high level structures that can be interpolated with each other to convey their relation.

6.4 Implementation of Visual Abstraction Levels

In this Section, we describe the implementation of the concept of forward and inverse abstraction that we introduced in the previous section. The four levels of visual abstraction applied to two distinct models is displayed in Figure 6.3 for reference. The transitions between different levels of visual abstractions were scripted in an extended version of the pipeline for animated transitions that we introduced in Chapter 5. For implementation details on the pipeline, we refer to Section 5.4.

6.4.1 Level 0 - Implicit Relations

On level 0, the *cellVIEW* [LAPV15] is used to display the molecular data. No visual abstraction is yet required. The cellVIEW output is also used as the basis for the subsequent levels of visual abstraction.

6.4.2 Schematic Representation of Molecular Structures

Visual abstraction levels 1-3 are based on the display of the high level structures that form the molecular compartments of a microorganim. We create these compartment shapes in these levels with a special visual abstraction approach. We thereby merge the 3D information about the individual molecules into aggregated geometrical shapes representing individual compartments and the compartment hierarchy within the model.

To create the schematic representation, we blur the rendering of the molecular model with a Gaussian filter including the alpha channel in a post processing step (see Fig. 6.6b). The radius of the filter specifies, to which degree the compartment shapes will be simplified. The blurring serves as a fast and easy way to create a smooth structure from the noisy patterns of the densely packed macro molecules. The coloring of the resulting shapes results from the blurring of the colors of the individual proteins within each compartment.

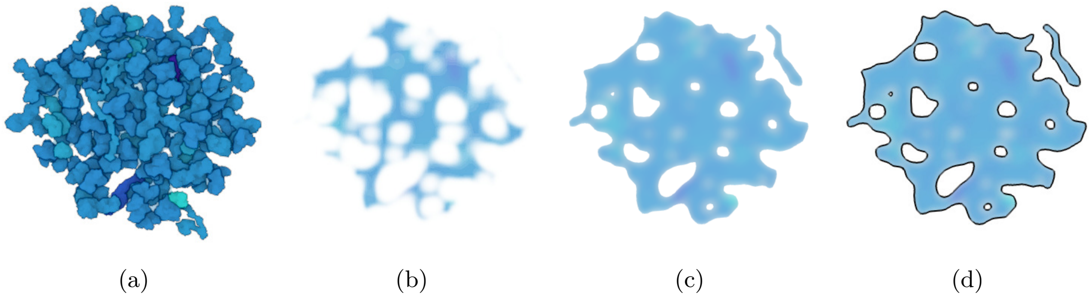

(a) (b) (c) (d)

Figure 6.6: The steps involved in rendering the schematic representation for a given molecular subset: the transition from (a) to (b) shows the blurring with a Gaussian filter. (c) The result after a steep ramp function is applied to the alpha channel to achieve hard edges. (d) A Sobel operator is applied on the alpha channel to create contours for the generated shape.

In order to create hard edges for the blurred surfaces, a steep ramp function is applied to the alpha channel (see Fig. 6.6c). Finally, we apply a Sobel operator on the alpha channel and use the output to display contours of the generated shape (see Fig. 6.6d). These contours allow us to distinguish the blurred compartment shapes from each other.

The schematic rendering is applied to all compartments individually, i.e., only the molecules of a given compartment are rendered in the same pass. The individual compartments are thereby rendered to separate layers. The layers are subsequently composed through alpha-blending, as illustrated in Figure 6.7. For the composition the layers are ordered from outermost to innermost. This ensures that the hierarchy of the compartments is visible in the schematic representation from any viewing direction, provided that the underlying model has an onion-like composition of the individual compartments. If there is too much overlap between compartments, the clarity of the schematic representation is reduced.

6.4.3 Level 1 - Pathways

In order to enable the display of protein pathway relations across two model states, we display representative specimen of proteins in the context of their structural compartment hierarchy. These representative protein specimen serve as examples for the chemical composition of the individual compartments within the model. We therefore show a cross-section view of the entire model and we significantly reduce the number of individual molecules.

By significantly reducing the number of proteins, we free up screen space to enable the display of pathway information. The remaining protein representatives can now be used to demonstrate chemical processes, such as the splitting and merging of proteins across model states.

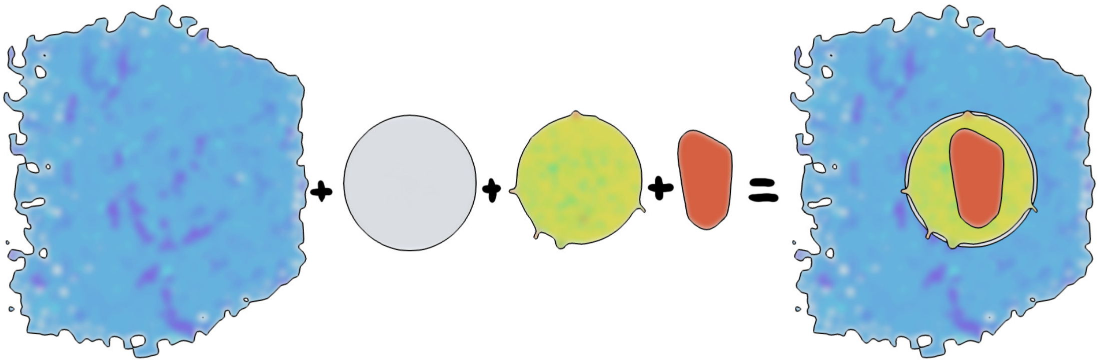


Figure 6.7: The schematic representation of the molecular structure is created for all compartments individually. The compartments are then rendered in layers and subsequently composed through alpha-blending.

As we see in Figure 6.8d, the schematic representation of the model is used as background, on top of which we display selected protein representatives. The representatives are split into two categories: matrix proteins and membrane proteins. While membrane proteins compose the hulls (membranes) of the individual compartments, matrix proteins are the ones encapsulated within these structures.

In order to convey the relative quantities of the proteins, the number of displayed representatives is proportional to the total number of molecules per protein type and compartment. Representatives are chosen randomly from all protein instances, in a way that approximates a uniform distribution within a compartment.

To better convey the shapes of the selected specimen, their orientation is modified so that they face the current viewpoint. We find the smallest dimension of the minimal bounding box of each protein type (x, y, or z). Each selected specimen is then rotated so that the smallest dimension of its bounding box is parallel to the viewing direction. Additionally, we scale the representative specimen by a constant factor (in our implementation, we use a factor of two) so that their shapes are better visible (Fig. 6.8d).

Membrane proteins form boundaries between individual compartments. We select representatives of membrane proteins so that they convey the shape and molecular composition of these compartments within the schematic representation. The representatives of the membrane proteins therefore keep their original orientation and scale. Membrane proteins are selected as representatives if their principal direction is perpendicular to the viewing direction.

Since all the selected specimen remain on their original positions, and only their orientation and scale are changed, it is possible to smoothly interpolate from the first level of abstraction to the second one without introducing unnecessary visual clutter. The transition is initiated by displaying the original molecular rendering of the model on top of the schematic representation with a blurring radius of zero. Subsequently, the orientations and the scales of the selected specimen are interpolated to the desired values, while the scales of all the other molecules are interpolated towards zero. We use spherical linear interpolation (SLERP) for interpolating orientations, which are specified by quaternions. For interpolating the scales, linear interpolation is used. We achieve an incremental abstraction of the compartment shapes by continuously increasing the blurring radius of the schematic representation, concurrently with the orientation and scale interpolations.

The time offset for starting the interpolation of the scales and orientations of the individual molecules can be made dependent on a distance field, so that not all the molecules are interpolated at the same time. The interpolation can be performed sequentially for the individual compartments. This peel-away effect in the resulting transition is depicted in Figure 6.8.

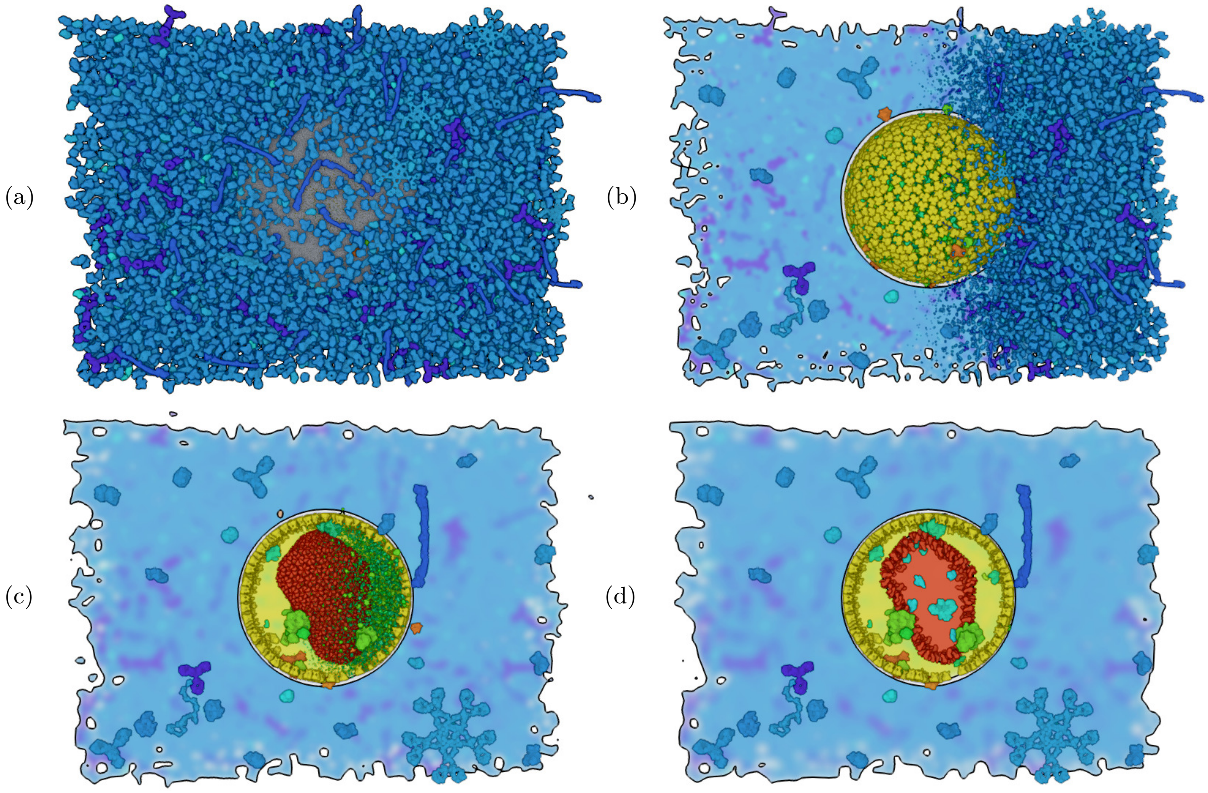


Figure 6.8: The sequence of images shows the transition between the visual abstraction levels one and two of the mature HIV virion. The noisy and dense high resolution molecule information is reduced in a delayed peel-away fashion from left to right and for each compartment. The dense structure makes way for the schematic representation of the model – the inner structures are revealed one by one. Representatives of each protein type are enlarged and rotated toward the viewer for better visibility.

6.4.4 Level 2 - Quantities

In visual abstraction level 2, we have to establish the relation between protein quantities and compartment structures. Since pathway information is not available, the representative protein specimen are removed from the display by scaling their size to zero. Due to the absence of protein representatives, the implicit encoding of quantities through protein occurrences has to be replaced with an explicit encoding. We chose to encode the quantities of protein types per compartment within stacked bars that are nested within the spatial context of the schematic representation (see Fig. 6.9c and d). The quantity of a certain protein type is encoded in the height of the bar that corresponds with that type. The total bar height is scaled to the height of the respective compartment. The bars thus fill up the entire height of the compartment. As alternatives to the bars, also other representation forms of the quantities, such as a tree-map or voronoi diagram, could be nested within the compartments.

The individual bars are colored with each protein type's given color. Additionally, a color gradient is applied to each bar, so that the color luminance slightly increases with decreasing y coordinates. This ensures that there is a visual separation between consecutive bars, which can be of similar colors, since currently, the protein coloring is randomized, as there is still no standardized color scheme for these types of macro molecules. To transform the representation to this level of visual abstraction from either of the previous levels, the molecules displayed on top of the simplified rendering are continuously scaled down until they disappear. Simultaneously, the stacked bars are faded-in through alpha blending. In compartments where the quantitative information is not of interest, the color saturation is continuously decreased to steer the viewers' attention towards the stacked bars (see Figure 6.9c and d).

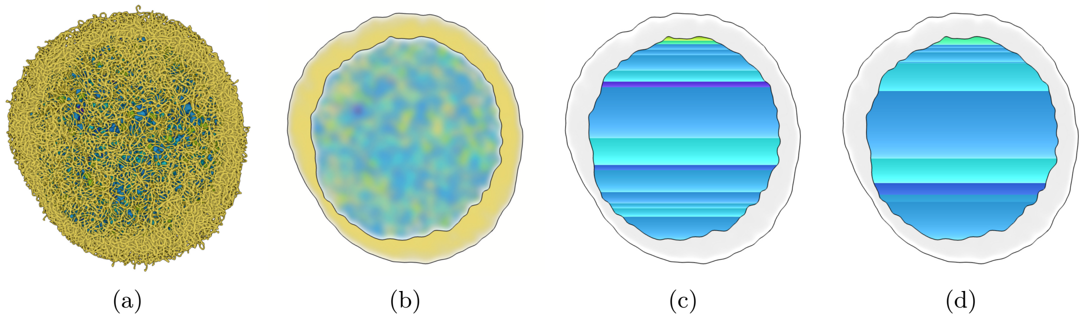

(a) (b) (c) (d)

Figure 6.9: The transition of mycoplasma. (a), (b) and (c) show the transition from visual abstraction level 0 to level 2. (c) and (d) show a transition at visual abstraction level 2 between two different model states of the mycoplasma (indicated by the change in the distribution of different protein quantities).

6.4.5 Level 3 - Structures

In visual abstraction level 3, only the relation between the high level structures of the models remains. The high level structures correspond to the schematic representation of the model without any overlays, as described in Section 6.4.2. To transform the representation to this level of abstraction, the molecules displayed on top of the schematic representation are removed by continuously scaling them down. Stacked bars from the previous level of abstraction are faded away through alpha blending.

6.5 Results

In this section, we demonstrate the results that we achieved with our method by applying it to two molecular data sets. The first result depicts the transition of an HIV virion from its immature to its mature state. The second result depicts the transition between two simulation states of a mycoplasma bacterium.

6.5.1 HIV Data Set

In this example, we show the relation between the immature and the mature state of an HIV virion. Partial knowledge about the relation of proteins exists: the large proteins in the immature model split up into multiple smaller proteins that correspond to three different compartments in the mature virus: the virus membrane, the capsid, and the capsid interior. However, we do not have the actual pathway information that describes the relations between proteins in the different maturity states. While the quantities of protein types in each model are known, the direct relation between the quantities is not clear. Many intermediate development steps between the immature and mature states are unknown. We therefore also cannot depict the relation of quantities accurately.

The relation between both models can therefore only be accurately described on the level of individual compartments. The immature HIV model thus has to be abstracted to the highest level of visual abstraction (level 3) before we can show its relation to the mature HIV model without conveying false information.

The transition between both models at visual abstraction level 3 is shown in Figure 6.10. The orange, green, and gray structures in (a) correspond to the red, yellow and gray compartments in (d). We achieve the transition between both models at this level of visual abstraction simply by scaling the blurred structures in the simplified rendering of the immature HIV virion to zero, while simultaneously scaling the structures of the mature HIV virion from zero to their original size. After the transition, we apply the inverse abstraction on the mature data set back to the lowest level of visual abstraction, as can be seen in Figure 6.3 B. The transition can be regarded in motion in the supplemental video [SMK$^+$].

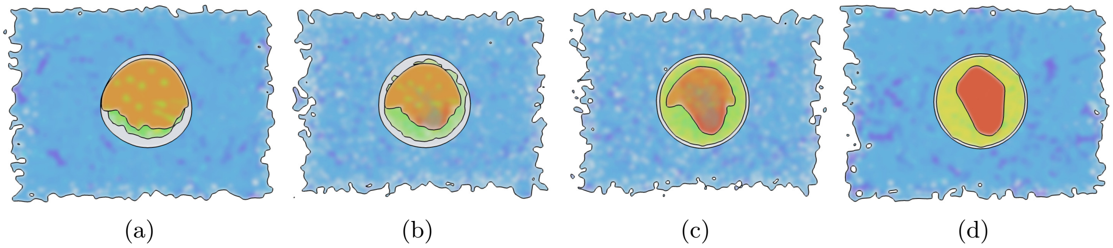


Figure 6.10: This sequence of images shows the transition between the immature and mature HIV virion at visual abstraction level 3. We abstract both models to this level, since we only have information about the compartment to compartment relations between them. The sequence shows how the orange, green, and gray structures in (a) correspond to the red, yellow and gray compartments in (d).

6.5.2 Mycoplasma Data Set

In the data set of mycoplasma mycoides (Fig. 6.9a), we have the opposite situation as in the previous example. Many high resolution simulations of the mycoplasma life cycle stages are publicly available [KPC14], while the mycoplasma 3D model is still a work in progress. Biologists have been developing this model for more than one year at the time of the publication [SMK+16] of this chapter. We therefore only have access to one incomplete model but to many detailed quantitative simulation results.

In contrast to the previous example, the quantitative information between development states has been sampled at a high frequency. This means, that interpolating the quantities between two time steps of the simulated model would not convey false information about their relationship. The development of protein type quantities across simulation states can therefore be conveyed at visual abstraction level 2. Even if the detailed protein pathway information between models would be known, we could not perform the interpolation at level 1, since there is only a model that corresponds to a single state of the mycoplasma bacterium.

Once the model is abstracted to visual abstraction level 2, the transition between the time steps in the development cycle of the mycoplasma is achieved by simply interpolating between the quantities of proteins. This results in the growing and shrinking of the corresponding bars that fill the respective compartments. Two time steps are displayed in Figures 6.9c and 6.9d. The animation can be observed in the supplemental material video [SMK+]. After the transition, we do not have to apply the inverse abstraction of the model as we did with the HIV virion, as the singular model does not correspond to the information that is encoded in the nested stacked bars.

6.6 Discussion and Expert Feedback

In this section, we first discuss design choices that are involved in our method and subsequently present feedback, which we received from two domain experts.

6.6.1 Design Choices

Using animated transitions to demonstrate the relationship between two different model states provides an implicit way of showing the link between related structures. In some cases, explicit encoding of these relations might be more desirable, e.g., by drawing edges that link related objects [WPL$^+$10, CC07]. For a presentation scenario, however, an animation is the more engaging approach [RFF$^+$08].

The choice of organizing the visualization in four different levels of visual abstraction results from the characteristics of the molecular data. The data comprises molecular structures that are organized in different hierarchically ordered compartments. The four levels result from the different levels of details at which protein-protein relations can be modeled: on a per-instance level (level 0), on a pathway level (level 1), on a quantitative level (level 2), and on the level of higher level structures (level 3).

The applicability of our approach is not conceptually limited to molecular data. In theory, our approach is applicable to any domain that exhibits data with the same characteristics, namely a possible discrepancy between the modeled or measured detail in the data items and the relation between these data items. However, in order to apply our method to data that features different characteristics, the definition and number of abstraction levels may have to be adjusted.

6.6.2 Domain Expert Feedback

We received feedback from two domain experts, a researcher that specializes in molecular illustrations and animations, and a researcher with a focus on molecular modeling. Both were excited about the concept of automatic transitions between molecular data sets, as they typically have to create such results manually in 3D modeling and animation software, such as Maya. In fact, the idea and inspiration for our method came from discussions with one of the researchers during another project. The concept of visually abstracting two models to matching levels of abstraction based on the relationship information that is available was well received. One domain expert stated that the abstraction successfully conveyed the compartmentalizations and compartment-to-compartment relationships in the virus. Also the "painting style" that the image-space blending achieved was complemented. The illustrator stated, *"This transition does a great job of taking a densely packed model and simplifying it to reveal the composition of molecules that are isolated to each compartment, while still retaining a simplified rendition of the compartment."* The transition was stated to give a good impression *"where molecular color schemes match the simpler fills [i.e., the schematic representation] that they become."*. The experts further appreciated that the transition between models conveyed relations more intuitively than a simple juxtaposition of the model states. The transition *"does help show what the different types of molecules look like, because the animation takes away the layers that obstruct the deeper molecules."*

Visual abstraction level 2 received some criticism for the readability of protein quantities. While the stacked bars enable an efficient comparison between quantities, it is difficult to associate a protein type with a certain stacked bar solely by its color. We agree that the identification of the stacked bars is an issue. However, we regard the chosen representation as a proof of concept for the principle idea of integrating the non-spatial / quantitative information within the spatial context of the molecular model. Nevertheless, the generation of annotations and labels in such a dynamic setting are an important aspect for future work on animated transitions of molecular data, as we also discussed in Section 5.6.3.

6.7 Conclusion and Outlook

In this chapter, we propose a method that deals with the creation of illustrative transitions between molecular models, where the actual process of the transition is not or only partially known. Our method utilizes visual abstraction to achieve smooth transitions between different models while respecting the known relational information. Instead of directly interpolating between two different models, we gradually forward transform the models into a level of visual abstraction that matches the preciseness of information about the relationship. These transitions provide intuitive connections between the structural and quantitative characteristics of the two dissimilar models. We exemplary demonstrate the flexibility of our approach on the basis of data sets of the HIV virion and the mycoplasma bacterium. We have received positive feedback from domain experts in the field of biological modeling and animation. The concept of our approach is applicable to all types of data that exhibit the same characteristics, i.e., where the knowledge about the model states might not match the knowledge about the relation between those states.

With the proposed method, we started to explore the topic of integrating different states of molecular data with varying levels of relationship knowledge. The support of transitions between data sets where the relationship knowledge varies between individual *sub-regions* or categories across model states would be a natural advancement of our technique. For instance, if only the relationship between certain types of molecules is known but not between others, different levels of abstractions could be blended together.

We demonstrated our results in Section 6.5 on molecular models that are structurally very similar. An essential next step to test and improve the flexibility of our method in conveying relations between models is thus the application of our technique to models that exhibit stronger structural differences. At the time of writing, such models were not available.

The transitions between different visual abstraction levels is handled with an extension of the pipeline for animated transitions that we introduced in the previous chapter. The types of transitions required for conveying the relations between data sets in Section 6.5 were chosen based on the four distances in mapping space that we defined (Section 6.3). An interesting extension of our approach would be the automatic identification of the distance in mapping space between two models in order to automatically select the appropriate level of visual abstraction for the transition between them. Such an extension would require the specification of a data format for storing relationship information at various levels of detail. We hope our approach inspires further advancements in dealing with missing information and uncertainty in bio-medical data visualization.

Concerning the relation of this chapter to the topic of visual integration, the statements from Section 5.7 apply also here, since both approaches are based on the same pipeline for animated transitions. A notable addition to this discussion concerns the schematic representation (Section 6.4.2) as well as visual abstraction level 2. The schematic representation corresponds to the *adaption of the spatial representation* where the shape channel of individual molecules is *neglected* (as discussed in Section 2.2). Molecules cannot be discriminated from each other anymore based on this visual channel, as they are blurred into one high level structure. The shapes of the higher level structures, i.e., the hierarchical compartments, are thereby preserved. In visual abstraction level 2, we use *nesting* to visually integrate the quantitative information of protein type occurrences. Therefore also the retinal channels of the molecules are neglected in order to provide a canvas for the nesting of the stacked bar representation.

No functional integration mechanisms are applied in the technique that we presented in this chapter, as it does not allow for any type of interaction. However, in the context of functional integration, we could envision interesting future extensions of our technique. For instance, the user could mark specific protein types or high level structures in one model so that they are also indicated during and after the transition in the respective other model (as in Visual Indication $D \rightarrow V$, Section 3.4.2) – or the user could trigger the transformation to the other model state only for the selected subset of proteins / high level structures (as in Data Manipulation $D \rightarrow D$, Section 3.4.1).

With this, we come to the end of Part II. In the following chapter, we conclude this thesis with a reflection on the presented work in the context of integration.

CHAPTER 7

Conclusion

7.1 Summary

This thesis addresses the topic of visualization integration, both from a theoretical point of view, and in the context of concrete applications for exploration and presentation tasks. The taxonomy describes the visual and functional aspects of integrating the given and chosen visual mappings of data attributes across multiple representation forms. The topic of integration is thereby discussed specifically in regard to the special characteristics that emerge in the context of multifaceted spatial data. A discussion on this level addresses a clear gap in the literature, since, to the best of the author's knowledge, both aspects of integration are so far only treated on a basic level and not in consideration of the special challenges and opportunities that the context of spatial data representations generate.

High level discussions of visual integration approaches that go beyond juxtaposition of multiple views can be found in the works of Javed and Elmqvist [JE12], and Balabanian [Bal10]. The taxonomy in Chapter 2 goes one step further in the exploration of the concept of visual integration, by dissecting the design space on the level of individual visual channels. An analysis of the design space at this level of detail enables the classification of each type of visual integration in terms of three aspects. Firstly, it enables the differentiation of techniques based on which visual channels are restricted through the nature of the technique, and which ones remain available for the visualization designer to choose the mapping for. Secondly, it enables the description of the relationship types that an integration method helps to establish between the involved representations. Finally, it enables the decoding of a representation's visual encoding by transcribing the individual operations on visual channels, i.e., whether the mapping to a channel is preserved (given), neglected, modified, or re-assigned (chosen). An analysis on this level of detail thus enables an unprecedented way of comparing visualization designs.

In Chapter 3, the same attention to detail is given for dissecting the design space of functional integration. Previous approaches in this regard described functional integration based on the coordination of visualization pipeline stages, and offered general guidelines for coordination between views [Rob07, BRR03, WBWK00] – but did not explore the resulting design space. A major aim of the presented taxonomy [SOP^{+}15] is to illustrate the range of potential applications for functional integration by regarding all potential combinations of the three high level components of the visualization pipeline, i.e., the Data, Visual, and Navigation components – and thus to create a deeper understanding of the solution space in functional integration. Even though the taxonomy is discussed in the context of multifaceted spatial data, the general principles of visual and functional integration are not restricted to this data type, and therefore hold general validity.

Part II of this thesis introduces novel visualization concepts in concrete application scenarios. Integration thereby plays a fundamental role in overcoming obstacles that impede exploration and presentation tasks.

Chapter 4 presents an application of integration in a decision making scenario in lighting design [SOL^{+}16]. The proposed solution demonstrates how visual and functional integration of the different facets of lighting simulation data can enable an efficient exploration, refinement, and decision making process when compared to the conventional isolated solutions that are typically employed in this domain. Integration is thereby a key factor in enabling the comparison of lighting solutions in a spatial context. Another important aspect of this solution is a novel ranking visualization that integrates the user controls for assigning importances to spatial parts of the scene and to non-spatial result indicators within a single representation.

Chapters 5 and 6 address the application of integration strategies in presentation scenarios for bio-molecular data visualization. Each chapter addresses a specific challenge in the context of the presentation of insights via temporal integration, i.e, animated transitions. The goal in Chapter 5 is to intuitively convey relationships between different representations of the molecular structure and composition of a microorganism. A temporal integration should thereby establish the relation between two representation forms by continuously converting one into the other. The addressed challenge in this context is the creation of transitions that can be re-applied to different molecular data sets, while achieving the same intended result, such as a certain type of illustrative or quantitative rendition of the data. This challenge is successfully handled by a technique featuring six types of modular operators (metamorphers) that describe the stages of a pipeline for animated transitions [SMR^{+}17]. By adhering to a set of rules, each operator can thereby be defined in a way that creates the same semantic output for an arbitrary input.

The goal in Chapter 6 is to convey relationships between a pair of molecular models that represent, for instance, different states in the development cycle of a microorganism. The addressed challenge in this context is caused due to incomplete information in the description of the relation between the individual elements, i.e., the molecules, between the model states. The proposed technique [SMK$^+$16] addresses this challenge with a set of four levels of visual abstraction that correspond to different degrees of incompleteness in the relationship descriptions. Temporal integration is thereby the means to create a continuous transformation between the different levels as well as to establish the relationship between two molecular models at a certain abstraction level. Nesting is applied at a certain level of abstraction to re-introduce inherent information into the spatial representation that was lost due to the visual abstraction.

7.2 Outlook

This thesis represents a next step for extending the basic research on the topic of integration that should give the reader a clearer understanding of the range of possibilities that this topic has to offer for improving visualization effectiveness and expressiveness. With the disclosure of the design space for visual and functional integration, many interesting directions for future research arise, as will be discussed in the following.

While the concept of animated transitions is not a popular one in the visualization community, its application is important for knowledge dissemination in molecular biology and other life science fields. In Chapters 5 and 6, temporal integration is the key to establish relationships between representations for the purpose of knowledge dissemination. While the presented works do not explicitly address functional integration in this context (as the techniques were designed to support presentation tasks), considering interaction for a potential functional integration of temporally integrated representations could be an interesting area for further research. A central question thereby would be, how to apply functional integration to the animation in a feasible way. Since the visually integrated representations are only displayed in intermediate (transitioning) states at the same time, conventional integration mechanisms that work on persistent representations, such as brushing and linking, cannot be simply applied to such a dynamic setting. A re-imagining of functional integration approaches for this setting would thus be necessary. A potential application could be beneficial in the context of the incorporation of labels and annotations (i.e., messaging in animation according to Segel and Heer [SH10]). Labels and annotations could persist throughout the animation. Interaction with them could trigger, for instance, branching animation paths that would allow a user to further explore or experience a specific aspect of an animated data set.

In the scope of this thesis, the application of temporal integration was exclusively discussed in the context of presentation scenarios. The application of animated transitions in exploratory visualization is so far only established to a certain degree for purely non-spatial representation forms [HR07, BOH11]. Interesting follow-up work in this regard could explore potential benefits of temporal integration between 3D spatial representations in an exploratory context. For instance, a bio-medical visual analysis framework that supports the exploration of neuronal brain data could allow the user to switch between different 3D spatial representation forms of the brain, such as the ones depicted in Figures 2.3a and 2.13, depending on which kinds of information they seek. In upcoming work, we employed this concept in an exploratory context for bridging multiple representation forms of DNA nano structures [MLS+17].

Another interesting application area for integration that was not explicitly addressed in the scope of this thesis is time-dependent data. While successful applications of integration in this context already exist [CKS+15, WFR+10, RWF+13], no explicit basic research on the potential benefits of integration in this area has yet been conducted. Also the exploration of cause-and-effect relationships, such as in parameter space exploration, can benefit from novel integration techniques, as we saw in Chapter 4. In this regard, another promising application field for exploring the potential benefits of integration is the visualization of neural networks, an area that recently gained a lot of traction in visualization research.

An aspect that the individual works covered in Part II of this thesis have in common is the way in which the techniques were evaluated. Each technique was validated by its demonstrative application in use case scenarios, as well as in terms of design reflections and informal feedback from domain experts. This form of evaluation is well established in the context of applications and design studies [Mun09], in the sense that it validates whether the requirements of the intended target users are fulfilled. Since the feedback in these works was retrieved from very specific audiences, i.e., domain experts, a good counterpart to this form of validation and an interesting future research direction would be an evaluation with a broader, more general validity – for instance, in terms of determining, which data types and tasks are generally best supported by which types of visual and functional integration. Such results could be retrieved through thorough quantitative evaluation of various aspects of integration in the scope of formal experiments and user studies. The taxonomy from Part I would thereby serve as a framework for these experiments. Studies in this regard would represent an essential next step for basic visualization research in the context of integration.

The ultimate goal for future efforts concerning the basic and applied research on the topic of visualization integration would be the creation of a visualization framework that supports all types of visual and functional integration that were covered in this thesis. Such a framework could be designed to offer guidance to users in the creation of an optimal visualization setup for a particular task and data set. A setup wizard could lead users through an interface that lets them assign different representation forms to different parts of the data, down to the level of mapping attributes to individual visual channels. For inherently mapped attributes in spatial data representations, the adaption of the spatial representation in terms of the visual abstraction and the re-assignment of channels could be supported as well. In a next step, users could be guided through the selection of visual integration strategies for the chosen representations in order to find the optimal spatial and temporal composition. Guidance could thereby notify the user about the types of relations that the individual visual integration strategies promote between representations. In a final setup step, the system could guide the user in selecting appropriate functional integration forms for improving the efficiency of interactions between the composed representations. Such a system would offer the ideal environment for the aforementioned user studies in order to gain insights into the suitability of the various techniques. The results of those studies could flow back into the framework, in order to allow the system to learn and automatically suggest the most appropriate representation and integration forms for a specific data/task combination.

With the highlighting of interesting future prospects of visualization integration, this work is now concluded. This thesis hopefully contributes to strengthening the basic research foundation on the topic of visualization integration – and as such hopefully supports future works that will build on this foundation in interesting and meaningful ways.

Bibliography

[AA01] Gennady Andrienko and Natalia Andrienko. Constructing parallel coordinates plot for problem solving. In *1st International Symposium on Smart Graphics*, pages 9–14, 2001.

[AAB+10] Gennady Andrienko, Natalia Andrienko, Sebastian Bremm, Tobias Schreck, Tatiana Von Landesberger, Peter Bak, and Daniel Keim. Space-in-time and time-in-space self-organizing maps for exploring spatiotemporal patterns. In *Computer Graphics Forum*, volume 29, pages 913–922. Wiley Online Library, 2010.

[AABS+14] Ali K Al-Awami, Justus Beyer, Hendrik Strobelt, Narayanan Kasthuri, Jeff W Lichtman, Hanspeter Pfister, and Markus Hadwiger. Neurolines: A subway map metaphor for visualizing nanoscale neuronal connectivity. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):2369–2378, 2014.

[Bal10] Jean-paul Balabanian. *Multi-Aspect Visualization: Going from Linked Views to Integrated Views.* PhD thesis, Dept. of Informatics, Univ. of Bergen, Norway, 2010.

[Bar97] Lyn Bartram. Perceptual and interpretative properties of motion for information visualization. In *Proceedings of the 1997 workshop on New paradigms in information visualization and manipulation*, pages 3–7. ACM, 1997.

[Bas11] Christian Basch. Animated transitions across multiple dimensions for volumetric data. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, Oct. 2011.

[BC03] Miguel Bagajewicz and Enmanuel Cabrera. Pareto optimal solutions visualization techniques for multiobjective design and upgrade of instrumentation networks. *Industrial & engineering chemistry research*, 42(21):5195–5203, 2003.

[BDW$^+$08] T. Butkiewicz, Wenwen Dou, Z. Wartell, W. Ribarsky, and R. Chang. Multi-Focused Geospatial Analysis Using Probes. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1165–1172, 2008.

[Ber] Drew Berry. Molecular animations. http://www.molecularmovies.com/showcase/. Accessed: 2017-09-16.

[Ber83] Jacques Bertin. Semiology of graphics: diagrams, networks, maps. 1983.

[BG06] Stefan Bruckner and M. Eduard Gröller. Exploded views for volume data. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):1077–1084, September 2006.

[BHGK14] Michael Beham, Wolfgang Herzner, M Eduard Gröller, and Johannes Kehrer. Cupid: Cluster-based exploration of geometry generators with parallel coordinates and radial trees. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):1693–1702, November 2014.

[BL04] Michel Beaudouin-Lafon. Designing interaction, not interfaces. In *Proceedings of the working conference on Advanced visual interfaces*, pages 15–22, New York, NY, USA, 2004.

[BM10] Stefan Bruckner and Torsten Möller. Result-driven exploration of simulation parameter spaces for visual effects design. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1468–1476, 2010.

[BM13] M. Brehmer and T. Munzner. A multi-level typology of abstract visualization tasks. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2376–2385, Dec 2013.

[BOH11] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301–2309, 2011.

[BRR03] N. Boukhelifa, J.C. Roberts, and P.J. Rodgers. A coordination model for exploratory multiview visualization. In *Coordinated and Multiple Views in Exploratory Visualization, International Conference on*, pages 76–85, 2003.

[BSG$^+$09] S. Bruckner, V. Solteszova, E. Gröller, J. Hladuvka, K. Buhler, J.Y. Yu, and B.J. Dickson. BrainGazer - Visual Queries for Neurobiology Research. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1497–1504, 2009.

[BVG10] Jean-Paul Balabanian, Ivan Viola, and Eduard Gröller. Interactive illustrative visualization of hierarchical volume data. In *Proceedings of Graphics Interface 2010*, pages 137–144. Canadian Information Processing Society, 2010.

[CAS$^+$13] Shahar Chen, David Amid, Ofer M Shir, Lior Limonad, David Boaz, Ateret Anaby-Tavor, and Tobias Schreck. Self-organizing maps for multi-objective pareto frontiers. In *Proceedings of IEEE Pacific Visualization Symposium (PacificVis 2013)*, pages 153–160. IEEE, 2013.

[CC07] Christopher M Collins and Sheelagh Carpendale. Vislink: Revealing relationships amongst visualizations. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1192–1199, 2007.

[CG07] Gregory Cipriano and Michael Gleicher. Molecular surface abstraction. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1608–1615, 2007.

[CKS$^+$15] Daniel Cornel, Artem Konev, Bernhard Sadransky, Zsolt Horvath, Eduard Gröller, and Jürgen Waser. Visualization of object-centered vulnerability to possible flood hazards. In *Computer Graphics Forum*, volume 34, pages 331–340. Wiley Online Library, 2015.

[CLEK13] Dane Coffey, Chi-Lun Lin, Arthur G Erdman, and Daniel F Keefe. Design by dragging: An interface for creative forward and inverse design with simulation ensembles. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2783–2791, 2013.

[CMS99] Stuart K Card, Jock D Mackinlay, and Ben Schneiderman. *Readings in information visualization: using vision to think.* Morgan Kaufmann, 1999.

[CON08] Marc Christie, Patrick Olivier, and Jean-Marie Normand. Camera control in computer graphics. In *Computer Graphics Forum*, volume 27, pages 2197–2218. Wiley Online Library, 2008.

[CR98] Ed Huai-Hsin Chi and J.T. Riedl. An operator interaction framework for visualization systems. In *Information Visualization, 1998. Proceedings. IEEE Symposium on*, pages 63–70, 1998.

[CWK$^+$07] R. Chang, G. Wessel, R. Kosara, E. Sauda, and W. Ribarsky. Legible cities: Focus-dependent multi-resolution visualization of urban relationships. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1169–1175, 2007.

[DBJ$^+$11] Pierre Dragicevic, Anastasia Bezerianos, Waqas Javed, Niklas Elmqvist, and Jean-Daniel Fekete. Temporal distortion for animated transitions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2009–2018. ACM, 2011.

[DGK$^+$10] Steven P. Dow, Alana Glassco, Jonathan Kass, Melissa Schwarz, Daniel L. Schwartz, and Scott R. Klemmer. Parallel Prototyping Leads to Better Design Results, More Divergence, and Increased Self-efficacy. *ACM Trans. Comput.-Hum. Interact.*, 17(4):pages 18:1–18:24, December 2010.

[DIN11] DIN Standard EN 12464-1: Light and lighting - Lighting of work places - Part 1: Indoor work places, August 2011.

[DMNV12] J. Díaz, E. Monclús, I. Navazo, and P. Vázquez. Adaptive cross-sections of anatomical models. *Computer Graphics Forum*, 31(7):2155–2164, 2012.

[Dol07] Helmut Doleisch. SIMVIS: interactive visual analysis of large and time-dependent 3D simulation data. In *Proceedings of the 39th conference on Winter simulation*, pages 712–720, 2007.

[ET08] Niklas Elmqvist and Philippas Tsigas. A taxonomy of 3d occlusion management for visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 14(5):1095–1109, 2008.

[FH09] R. Fuchs and H. Hauser. Visualization of Multi-Variate Scientific Data. *Computer Graphics Forum*, 28(6):1670–1690, 2009.

[FKRE09] Martin Falk, Michael Klann, Matthias Reuss, and Thomas Ertl. Visualization of signal transduction processes in the crowded environment of the cell. In *IEEE Pacific Visualization Symposium*, pages 169–176. IEEE, 2009.

[GAW$^+$11] Michael Gleicher, Danielle Albers, Rick Walker, Ilir Jusufi, Charles D. Hansen, and Jonathan C. Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4):289–309, 2011.

[GBKG04] Sören Grimm, Stefan Bruckner, Armin Kanitsar, and M. Eduard Gröller. Flexible direct multi-volume rendering in interactive scenes. In *Vision, Modeling, and Visualization*, pages 386–379, 2004.

[GLG$^+$13] Samuel Gratzl, Alexander Lex, Nils Gehlenborg, Hanspeter Pfister, and Marc Streit. Lineup: Visual analysis of multi-attribute rankings. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2277–2286, 2013.

[Goo99] David S Goodsell. Atomistic vs. continuous representations in molecular biology. In *Visual Representations and Interpretations*, pages 146–155. Springer, 1999.

[GRPF16] Henning Gruendl, Patrick Riehmann, Yves Pausch, and Bernd Froehlich. Time-series plots integrated in parallel-coordinates displays. In *Computer Graphics Forum*, volume 35, pages 321–330. Wiley Online Library, 2016.

[GRW$^+$00] D. L. Gresh, B. E. Rogowitz, R. L. Winslow, D. F. Scollan, and C. K. Yung. WEAVE: a system for visually linking 3-D and statistical visualizations, applied to cardiac simulation and measurement data. In *Proceedings of the conference on Visualization '00*, pages 489–492, 2000.

[GTCD03] Daniel C. Glaser, Roger Tan, John Canny, and Ellen Yi-Luen Do. Developing architectural lighting representations. In *Information Visualization, 2003. INFOVIS 2003. IEEE Symposium on*, pages 241–248, Washington, DC, USA, 2003. IEEE Computer Society.

[HCHY05] Ke-Sen Huang, Chun-Fa Chang, Yu-Yao Hsu, and Shi-Nine Yang. Key probe: a technique for animation keyframe extraction. *The Visual Computer*, 21(8):532–541, 2005.

[HJ93] Karen Holtzblatt and Sandra Jones. *Contextual Inquiry: A Participatory Technique for System Design*, pages 177–210. Lawrence Erlbaum Associates, Hillsdale, 1993.

[HLL16] Kyunglyul Hyun, Kyungho Lee, and Jehee Lee. Motion grammars for character animation. *Computer Graphics Forum*, 35(2):103–113, 2016.

[HM90] Robert B Haber and David A McNabb. Visualization idioms: A conceptual model for scientific visualization systems. *Visualization in scientific computing*, 74:93, 1990.

[HMC11] Wei-Hsien Hsu, Kwan-Liu Ma, and Carlos Correa. A rendering framework for multi-scale views of 3d models. *ACM Transactions on Graphics*, 30(6), December 2011.

[HMP$^+$12] Paul-Corneliu Herghelegiu, V Manta, Radu Perin, Stefan Bruckner, and Eduard Gröller. Biopsy planner–visual analysis for needle pathway planning in deep seated brain tumor biopsy. In *Computer Graphics Forum*, volume 31, pages 1085–1094. Wiley Online Library, 2012.

[HOF04] Andreas Halm, Lars Offen, and Dieter Fellner. Visualization of complex molecular ribbon structures at interactive rates. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 737–744. IEEE, 2004.

[HR07] J. Heer and G. Robertson. Animated transitions in statistical data graphics. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1240–1247, Nov 2007.

[HTCT14] C. Hurter, R. Taylor, S. Carpendale, and A. Telea. Color tunneling: Interactive exploration and selection in volumetric datasets. In *Proceedings of IEEE Pacific Visualization Symposium (PacificVis 2014)*, pages 225–232, March 2014.

[HWM$^+$06] Helwig Hauser, Daniel Weiskopf, Kwan-Liu Ma, Jarke J van Wijk, and Robert Kosara. Scivis, infovis bridging the community divide. *IEEE Visualization (Panel Proceedings)*, pages 52–55, 2006.

[IBHT+02] S. Iserhardt-Bauer, P. Hastreiter, B. Tomandl, N. Köstner, M. Schempershofe, U. Nissen, and T. Ertl. *Standardized Analysis of Intracranial Aneurysms Using Digital Video Sequences*, pages 411–418. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[Inc] Google Inc. Google maps. https://www.google.at/maps/. Accessed: 2017-09-16.

[Iwa10] J. H. Iwasa. Animating the model figure. *Trends Cell Biology*, 20(12):699–704, Dec 2010.

[JAAA+15] Graham T Johnson, Ludovic Autin, Mostafa Al-Alusi, David S Goodsell, Michel F Sanner, and Arthur J Olson. cellPACK: a virtual mesoscope to model and visualize structural systems biology. *Nature methods*, 12(1):85–91, 2015.

[JDL09] Radu Jianu, Cagatay Demiralp, and David H Laidlaw. Exploring 3d dti fiber tracts with linked 2d representations. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1449–1456, 2009.

[JE12] Waqas Javed and Niklas Elmqvist. Exploring the design space of composite visualization. In *Proceedings of IEEE Pacific Visualization Symposium (PacificVis 2012)*, pages 1–8, 2012.

[JJJF07] Mikael Jern, Sara Johansson, Jimmy Johansson, and Johan Franzen. The gav toolkit for multiple linked views. In *Coordinated and Multiple Views in Exploratory Visualization, 2007. CMV'07. Fifth International Conference on*, pages 85–97. IEEE, 2007.

[JM13] Jodie Jenkinson and Gaël McGill. Using 3D animation in biology education: Examining the effects of visual complexity in the representation of dynamic molecular events. *Journal of Biocommunication*, 39(2):E42–E49, 2013.

[JNM] Graham T. Johnson, Andrew Noske, and Bradley Marsh. Rapid visual inventory and comparison of complex 3d structures. https://www.youtube.com/watch?v=Dl1ufW3cj4g. Accessed: 2017-09-16.

[KF93] Peter Karp and Steven Feiner. Automated presentation planning of animation using task decomposition with heuristic reasoning. In *Proceedings of Graphics Interface '93*, GI '93, pages 118–127, Toronto, Ontario, Canada, 1993. Canadian Human-Computer Communications Society.

[KH13] J. Kehrer and H. Hauser. Visualization and Visual Analysis of Multifaceted Scientific Data: A Survey. *Visualization and Computer Graphics, IEEE Transactions on*, 19(3):495–513, 2013.

[KKL+15] Barbora Kozlikova, Michael Krone, Norbert Lindow, Martin Falk, Marc Baaden, Daniel Baum, Ivan Viola, Julius Parulek, Hans-Christian Hege, et al. Visualization of biomolecular structures: State of the art. In *Eurographics Conference on Visualization (EuroVis)-STARs*, pages 061–081. The Eurographics Association, 2015.

[KM13] Robert Kosara and Jock Mackinlay. Storytelling: The next step for visualization. *IEEE Computer*, 46:44—-50, 2013.

[KPC14] Jonathan R Karr, Nolan C Phillips, and Markus W Covert. WholeCellSimDB: a hybrid relational/hdf database for whole-cell model predictions. *Database*, 2014.

[KW08] Pekka Korhonen and Jyrki Wallenius. Visualization in the multiple objective decision-making framework. In *Multiobjective optimization*, pages 195–212. Springer, 2008.

[KWZ11] Murat Koksalan, Jyrki Wallenius, and Stanley Zionts. *Multiple criteria decision making: from early history to the 21st century*. World Scientific, 2011.

[LAPV15] Mathieu Le Muzic, Ludovic Autin, Julius Parulek, and Ivan Viola. cellVIEW: a Tool for Illustrative and Multi-Scale Rendering of Large Biomolecular Datasets. In *Eurographics Workshop on Visual Computing for Biology and Medicine*, pages 61–70, September 2015.

[LBK04] Alexander V Lotov, Vladimir A Bushenkov, and Georgy K Kamenev. *Interactive decision maps: Approximation and visualization of Pareto frontier*, volume 89. Springer Science & Business Media, 2004.

[LHH+13] Wen-Chieh Lin, Tsung-Shian Huang, Tan-Chi Ho, Yueh-Tse Chen, and Jung-Hong Chuang. Interactive lighting design with hierarchical light representation. *Computer Graphics Forum*, 32(4):133–142, 2013.

[Lig] Lighting Analysts, Inc. AGi32. http://www.agi32.com/. Accessed: 2015-03-30.

[LLWC08] T. Y. Lee, C. H. Lin, Y. S. Wang, and T. G. Chen. Animation key-frame extraction and simplification using deformation analysis. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(4):478–486, April 2008.

[LMMS+16] Mathieu Le Muzic, Peter Mindek, Johannes Sorger, Ludovic Autin, David Goodsell, and Ivan Viola. Visibility equalizer: Cutaway visualization of mesoscopic biological models. *Computer Graphics Forum*, 35(3), 2016.

[LRA+07] Wilmot Li, Lincoln Ritter, Maneesh Agrawala, Brian Curless, and David Salesin. Interactive cutaway illustrations of complex 3d models. *ACM Trans. Graph.*, 26(3), July 2007.

[LTH+13] Christian Luksch, Robert F. Tobler, Ralf Habel, Michael Schwärzler, and Michael Wimmer. Fast light-map computation with virtual polygon lights. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 87–94, 2013.

[Mac86] Jock Mackinlay. Automating the design of graphical presentations of relational information. *Acm Transactions On Graphics (Tog)*, 5(2):110–141, 1986.

[MBP06] Konrad Muehler, Ragnar Bade, and Bernhard Preim. Adaptive script based animations for intervention planning. In *Proceedings of the 9th International Conference on Medical Image Computing and Computer-Assisted Intervention - Volume Part I*, MICCAI'06, pages 478–485, Berlin, Heidelberg, 2006. Springer-Verlag.

[MFNF01] Isabel Harb Manssour, Sérgio Shiguemi Furuie, Luciana Porcher Nedel, and Carla Maria Dal Sasso Freitas. A framework to interact and visualize with multimodal medical images (st). In *Volume Graphics*, 2001.

[MKS+17] Peter Mindek, David Kouřil, Johannes Sorger, David Toloudis, Blair Lyons, Graham Johnson, Meister Eduard Gröller, and Ivan Viola. Visualization multi-pipeline for communicating biology. *Visualization and Computer Graphics, IEEE Transactions on*, PP(99), 2017.

[MLS+17] Haichao Miao, Elisa De Llano, Johannes Sorger, Yasaman Ahmadi, Tadija Kekic, Tobias Isenberg, Meister Eduard Gröller, Ivan Barisic, and Ivan Viola. Multiscale visualization and scale-adaptive modification of dna nanostructures. *Visualization and Computer Graphics, IEEE Transactions on*, PP(99):1–1, 2017.

[MM06] Kaisa Miettinen and Marko M Mäkelä. Synchronous approach in interactive multiobjective optimization. *European Journal of Operational Research*, 170(3):909–922, 2006.

[MP10] Konrad Mühler and Bernhard Preim. Reusable Visualizations and Animations for Surgery Planning. *Computer Graphics Forum*, 2010.

[MR10] Michael J McGuffin and Jean-Marc Robert. Quantifying the space-efficiency of 2d graphical representations of trees. *Information Visualization*, 9(2):115–140, 2010.

[Mun09] T. Munzner. A Nested Model for Visualization Design and Validation. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):921–928, Nov.-Dec. 2009.

[Mun14] Tamara Munzner. *Visualization analysis and design.* CRC press, 2014.

[OMSI07] Makoto Okabe, Yasuyuki Matsushita, Li Shen, and Takeo Igarashi. Illumination brush: Interactive design of all-frequency lighting. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, PG '07, pages 171–180, Washington, DC, USA, 2007. IEEE Computer Society.

[OSP$^+$16] Thomas Ortner, Johannes Sorger, Harald Piringer, Gerd Hesina, and Eduard Gröller. Visual analytics and rendering for tunnel crack analysis. *The Visual Computer*, pages 1–11, 2016.

[OSS$^+$16] T. Ortner, J. Sorger, H. Steinlechner, G. Hesina, H. Piringer, and E. Gröller. Vis-a-ware: Integrating spatial and non-spatial visualization for visibility-aware urban planning. *Visualization and Computer Graphics, IEEE Transactions on*, PP(99):1–1, 2016.

[PBMF07] Fabio Pellacini, Frank Battaglia, R. Keith Morley, and Adam Finkelstein. Lighting with paint. *ACM Trans. Graph.*, 26(2), June 2007.

[PF92] Pierre Poulin and Alain Fournier. Lights from highlights and shadows. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, I3D '92, pages 31–38, New York, NY, USA, 1992. ACM.

[PKH04] H. Piringer, R. Kosara, and H. Hauser. Interactive focus+context visualization with linked 2D/3D scatterplots. In *Coordinated and Multiple Views in Exploratory Visualization*, pages 49–60, 2004.

[PTG02] Fabio Pellacini, Parag Tole, and Donald P. Greenberg. A user interface for interactive cinematic shadow design. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 563–566, New York, NY, USA, 2002. ACM.

[PTMB09] Harald Piringer, Christian Tominski, Philipp Muigg, and Wolfgang Berger. A multi-threading architecture to support interactive visual exploration. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1113–1120, 2009.

[Rela] Relux Informatik AG. DIAL GmbH. http://www.dial.de/DIAL/en/dialux/. Accessed: 2015-03-30.

[Relb] Relux Informatik AG. ReluxSuite. http://www.relux.info/. Accessed: 2015-03-30.

[RFF$^+$08] G. Robertson, R. Fernandez, D. Fisher, B. Lee, and J. Stasko. Effectiveness of animation in trend visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1325–1332, Nov 2008.

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

[Rob98] Jonathan C Roberts. On encouraging multiple views for visualization. In *Information Visualization, 1998. Proceedings. 1998 IEEE Conference on*, pages 8–14. IEEE, 1998.

[Rob07] J.C. Roberts. State of the Art: Coordinated Multiple Views in Exploratory Visualization. In *Coordinated and Multiple Views in Exploratory Visualization*, pages 61–71, 2007.

[RTM$^+$03] Theresa-Marie Rhyne, Melanie Tory, Tamara Munzner, Matthew O Ward, Chris Johnson, and David H Laidlaw. Panel: Information and Scientific Visualization: Separate but Equal or Happy Together at Last. *IEEE Visualization (Panel Proceedings)*, pages 611–614, 2003.

[RWF$^+$13] H. Ribicic, J. Waser, R. Fuchs, G. Bloschl, and E. Gröller. Visual Analysis and Steering of Flooding Simulations. *Visualization and Computer Graphics, IEEE Transactions on*, 19(6):1062–1075, 2013.

[SBS$^+$13] Johannes Sorger, Katja Bühler, Florian Schulze, Tianxiao Liu, and Barry Dickson. neuroMap - Interactive Graph Visualization of the Fruit Fly's Neural Circuit. In *IEEE Symposium on Biological Data Visualization*, pages 73–80, 2013.

[SC07] Amit Shesh and Baoquan Chen. Crayon lighting: Sketch-guided illumination of models. In *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*, GRAPHITE '07, pages 95–102, New York, NY, USA, 2007. ACM.

[SDS$^+$93] Chris Schoeneman, Julie Dorsey, Brian Smits, James Arvo, and Donald Greenberg. Painting with light. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 143–146, New York, NY, USA, 1993. ACM.

[SF91] Dorée Duncan Seligmann and Steven Feiner. Automated generation of intent-based 3d illustrations. *SIGGRAPH Comput. Graph.*, 25(4):123–132, July 1991.

[SH10] Edward Segel and Jeffrey Heer. Narrative visualization: Telling stories with data. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1139–1148, Nov 2010.

[SHB$^+$14] Michael Sedlmair, Christoph Heinzl, Stefan Bruckner, Harald Piringer, and Torsten Möller. Visual parameter space analysis: A conceptual framework. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):2161–2170, 2014.

[Shn92] Ben Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on graphics (TOG)*, 11(1):92–99, 1992.

[SMB$^+$14] Nicolas Swoboda, Judith Moosburner, Stefan Bruckner, Y Yu Jai, Barry J Dickson, and Katja Bühler. Visual and quantitative analysis of higher order arborization overlaps for neural circuit research. In *Eurographics Workshop on Visual Computing for Biology and Medicine*, pages 107–116, 2014.

[SMK$^+$] Johannes Sorger, Peter Mindek, Tobias Klein, Graham Johnson, and Ivan Viola. Forward & inverse abstractions - supplemental video. https://goo.gl/1Y7Pvf. Accessed: 2017-09-16.

[SMK$^+$16] Johannes Sorger, Peter Mindek, Tobias Klein, Graham Johnson, and Ivan Viola. Illustrative Transitions in Molecular Visualization via Forward and Inverse Abstraction Transform. In *Eurographics Workshop on Visual Computing for Biology and Medicine*. The Eurographics Association, 2016.

[SMR$^+$] Johannes Sorger, Peter Mindek, Peter Rautek, Meister Eduard Gröller, Graham Johnson, and Ivan Viola. Metamorphers - supplemental video. https://goo.gl/iMo5x8. Accessed: 2017-09-16.

[SMR$^+$17] Johannes Sorger, Peter Mindek, Peter Rautek, Meister Eduard Gröller, Graham Johnson, and Ivan Viola. Metamorphers: Storytelling templates for illustrative animated transitions in molecular visualization. In *Proceedings of the Spring Conference on Computer Graphics 2017*, pages 27–36, May 2017.

[SOL$^+$] Johannes Sorger, Thomas Ortner, Christian Luksch, Michael Schwärzler, Eduard Gröller, and Harald Piringer. Litevis - supplemental video. https://goo.gl/DvLSZ6. Accessed: 2017-09-16.

[SOL$^+$16] Johannes Sorger, Thomas Ortner, Christian Luksch, Michael Schwärzler, Eduard Gröller, and Harald Piringer. Litevis: Integrated visualization for simulation-based decision support in lighting design. *Visualization and Computer Graphics, IEEE Transactions on*, 22(1):290–299, 2016.

[SOP$^+$15] Johannes Sorger, Thomas Ortner, Harald Piringer, Gerd Hesina, and Eduard Gröller. A Taxonomy of Integration Techniques for Spatial and Non-Spatial Visualizations. In David Bommes, Tobias Ritschel, and Thomas Schultz, editors, *Vision, Modeling & Visualization*. The Eurographics Association, 2015.

[SRH$^+$09] Michael Sedlmair, Kerstin Ruhland, Fabian Hennecke, Andreas Butz, Susan Bioletti, and Carol O'Sullivan. Towards the Big Picture: Enriching 3D Models with Information Visualisation and Vice Versa. In *Smart Graphics*, volume 5531, pages 27–39. Springer Berlin Heidelberg, 2009.

[SW14] Michael Schwarz and Peter Wonka. Procedural design of exterior lighting for buildings with complex constraints. *ACM Trans. Graph.*, 33(5):166:1–166:16, September 2014.

[SWS^{+}11] Markus Steinberger, Manuela Waldner, Marc Streit, Alexander Lex, and Dieter Schmalstieg. Context-preserving visual links. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2249–2258, 2011.

[SzBBKN14] Christian Schulte zu Berge, Maximilian Baust, Ajay Kapoor, and Nassir Navab. Predicate-based focus-and-context visualization for 3d ultrasound. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):2379–2387, 2014.

[TM04] Melanie Tory and Torsten Möller. Rethinking Visualization: A High-Level Taxonomy. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 151–158, Washington, DC, USA, 2004. IEEE Computer Society.

[TMB02] Barbara Tversky, Julie Bauer Morrison, and Mireille Betrancourt. Animation: Can it facilitate? *International Journal of Human-Computer Studies*, 57(4):247–262, October 2002.

[TSH^{+}14] Cagatay Turkay, Aidan Slingsby, Helwig Hauser, Jo Wood, and Jason Dykes. Attribute signatures: dynamic visual summaries for analyzing multivariate geographical data. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):2033–2042, 2014.

[Tuf90] Edward Tufte. *Envisioning Information.* Graphics Press, Cheshire, CT, USA, 1990.

[VDZLBI11] Matthew Van Der Zwan, Wouter Lueks, Henk Bekker, and Tobias Isenberg. Illustrative molecular visualization with continuous abstraction. In *Computer Graphics Forum*, volume 30, pages 683–690. Wiley Online Library, 2011.

[VFSG06] I. Viola, M. Feixas, M. Sbert, and M.E. Gröller. Importance-Driven Focus of Attention. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):933–940, 2006.

[VKG04] Ivan Viola, Armin Kanitsar, and Meister Eduard Gröller. Importance-driven volume rendering. In *IEEE Visualization 2004*, pages 139–146, Oct 2004.

[VP04] Vivek Verma and Alex Pang. Comparative flow visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 10(6):609–624, 2004.

[vPGL$^+$14] Roy van Pelt, Rocco Gasteiger, Kai Lawonn, Monique Meuschke, and Bernhard Preim. Comparative blood flow visualization for cerebral aneurysm treatment assessment. In *Computer Graphics Forum*, volume 33, pages 131–140, 2014.

[War00] Colin Ware. *Information visualization*, volume 2. Morgan Kaufmann San Francisco, 2000.

[WB11] Manuel Wahle and Stefan Birmanns. Gpu-accelerated visualization of protein dynamics in ribbon mode. In *IS&T/SPIE Electronic Imaging*, pages 786805–786805. International Society for Optics and Photonics, 2011.

[WBWK00] Michelle Q. Wang Baldonado, Allison Woodruff, and Allan Kuchinsky. Guidelines for using multiple views in information visualization. In *Conference on Advanced visual interfaces*, pages 110–119, 2000.

[WFR$^+$10] Jürgen Waser, Raphael Fuchs, Hrvoje Ribicic, Benjamin Schindler, Gunther Blöschl, and Eduard Gröller. World Lines. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1458–1467, 2010.

[WGK10] Matthew Ward, Georges Grinstein, and Daniel Keim. *Interactive Data Visualization: Foundations, Techniques, and Applications.* A. K. Peters, Ltd., Natick, MA, USA, 2010.

[WH07] Michael Wohlfart and Helwig Hauser. Story telling for presentation in volume visualization. In *Proceedings of the 9th Joint Eurographics / IEEE VGTC Conference on Visualization*, EUROVIS'07, pages 91–98. Eurographics Association, 2007.

[Wil78] Lance Williams. Casting curved shadows on curved surfaces. In *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '78, pages 270–274, New York, NY, USA, 1978. ACM.

[WPL$^+$10] Manuela Waldner, Werner Puff, Alexander Lex, Marc Streit, and Dieter Schmalstieg. Visual links across applications. In *Proceedings of Graphics Interface 2010*, pages 129–136. Canadian Information Processing Society, 2010.

[WYL$^+$14] Zuchao Wang, Tangzhi Ye, Min Lu, Xiaoru Yuan, Huamin Qu, Jiaxin Yuan, and Qianliang Wu. Visual exploration of sparse traffic trajectory data. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):1813–1822, 2014.

[YaKSJ07] Ji Soo Yi, Youn ah Kang, J.T. Stasko, and J.A. Jacko. Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1224–1231, 2007.

[ZFA$^+$14] Wei Zeng, Chi-Wing Fu, Stefan Muller Arisona, Alexander Erath, and Huamin Qu. Visualizing mobility of public transportation system. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):1833–1842, 2014.

[Zum] Zumtobel Lighting GmbH. ecoCALC. http://www.zumtobel.com/com-en/ecoCALC.html. Accessed: 2015-03-30.

[ZYM$^+$14] Jiawan Zhang, E Yanli, Jing Ma, Yahui Zhao, Binghan Xu, Liting Sun, Jinyan Chen, and Xiaoru Yuan. Visual analysis of public utility service problems in a metropolis. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):1843–1852, 2014.

Curriculum Vitae

Name: Johannes Sorger, Dipl.Ing.
Address: Wasagasse 31/22, 1090 Vienna, Austria
Date of Birth: April 21st, 1983
Nationality: Austrian
Phone: +43(0) 699 190 939 09
Email: johannessorger@gmail.com

Education

03/2013 – 10/2017 (estimated):	Doctoral program in Engineering and Computer Sciences Institute of Computer Graphics and Algorithms TU Wien (Technical University of Vienna) Dissertation: “Integration Strategies in the Visualization of Multifaceted Spatial Data” Advisors: Assoc.Prof. Dipl.Ing. Dr.techn. Ivan Viola, Ao.Univ.Prof. Dipl.Ing. Dr.techn. Eduard Gröller
10/2009 – 03/2013:	Master’s program in Visual Computing Specialization: Real Time Graphics and Visualization TU Wien (Technical University of Vienna) Master Thesis: "Interactive Graph-Visualization of the Fruit Fly’s Neural Circuit" Advisors: Ao.Univ.Prof. Dipl.Ing. Dr.techn. Eduard Gröller, TU Wien Dipl.Ing. Dr.techn. Katja Bühler, VRVis Research Company
2003 – 2009:	Bachelor’s program in Media & Computer Science Specialization: Design TU Wien (Technical University of Vienna) Bachelor Project: “Audio-Visual Perception in Interactive Virtual Environments” (in collaboration with INRIA, France), Advisor: Matthias Bernhard, PhD

Work Experience

01/2016 – 10/2017:	Project Assistant at the Institute of Computer Graphics and Algorithms, TU Wien working on basic and applied research in the fields of illustrative and molecular visualization
10/2012 – 01/2016:	Researcher at the VRVis Research Company, working on basic and applied research in the field of visual analytics in spatial and abstract data visualization
02/2011 – 10/2012:	Student researcher at the VRVis Research Company, working on biological data visualization in cooperation with the Institute of Molecular Pathology, Vienna
01/2007 – 09/2011:	Sponsorship monitoring at United Synergies, agency for analyzing and appraising presence and value of advertising
10/2005 – 12/2005:	Volunteer at the NGO Poder Ciudadano in Buenos Aires: translations (English/German/Spanish), research

Awards

Best Overall Concept Award 2017, in the BootCamp for Sciencepreneurs, innovation incubation center (i2c), TU Wien

Austrian Computer Graphics Award 2016 for best technical solution (as a team member of the cellVIEW project)

OCG Incentive Award 2014 (OCG Förderpreis 2014), Austrian Computer Society (OCG)

Best Paper Award, at the 3rd IEEE Symposium on Biological Data Visualization (BioVis), Atlanta, Georgia, USA, 2013

Reviewing History

TVCG (2014), IEEE VIS (2013, 2014, 2015), EuroVis (2014, 2015, 2017), CGI (2013), CHI (2013)

Teaching

Computer Animation (lecture unit on behavioral animation), supervision of bachelor theses and seminar works

Selected Publications and Talks

“Multiscale Visualization and Scale-adaptive Modification of DNA Nanostructures”
Haichao Miao, Elisa De Llano, Johannes Sorger, Yasaman Ahmadi, Tadija Kekic, Tobias Isenberg, Meister Eduard Gröller, Ivan Barisic, Ivan Viola
in Visualization and Computer Graphics, IEEE Transactions on, 24(1). January 2018

“Visualization Multi-Pipeline for Communicating Biology”
Peter Mindek, David Kouřil, Johannes Sorger, David Toloudis, Blair Lyons, Graham Johnson, Meister Eduard Gröller, Ivan Viola
in Visualization and Computer Graphics, IEEE Transactions on, 24(1). January 2018

“Metamorphers: Storytelling Templates For Illustrative Animated Transitions in Molecular Visualization”
(with talk at SCCG 2017, Mikulov, Czech Republic)
Johannes Sorger, Peter Mindek, Peter Rautek, Eduard Gröller, Graham Johnson, Ivan Viola
In 33rd Spring Conference on Computer Graphics (SCCG), pages 27-36. May 2017

“Illustrative Transitions in Molecular Visualization via Forward and Inverse Abstraction Transform”
(with talk at VCBM 2016, Bergen, Norway)
Johannes Sorger, Peter Mindek, Tobias Klein, Graham Johnson, Ivan Viola
In Eurographics Workshop on Visual Computing for Biology and Medicine (VCBM), pages 21-30. September 2016.

“LiteVis: Integrated Visualization for Simulation-Based Decision Support in Lighting Design”
(with talk at VIS 2015, Chicago)
Johannes Sorger, Thomas Ortner, Christian Luksch, Michael Schwärzler, Eduard Gröller, Harald Piringer
In Visualization and Computer Graphics, IEEE Transactions on, 22(1):290-299, January 2016.

“Vis-A-Ware: Integrating Spatial and Non-Spatial Visualization for Visibility-Aware Urban Planning”
Thomas Ortner, Johannes Sorger, Harald Steinlechner, Gerd Hesina, Harald Piringer, Eduard Gröller
In Visualization and Computer Graphics, IEEE Transactions on, 2016.

“Visibility Equalizer: Cutaway Visualization of Mesoscopic Biological Models”
Mathieu LeMuzic, Peter Mindek, Johannes Sorger, Ludovic Autin, David Goodsell, Ivan Viola
In Computer Graphics Forum Volume 35 (2016), Number 3

“A Taxonomy of Integration Techniques for Spatial and Non-Spatial Visualizations”
(with talk at VMV 2015, Aachen)
Johannes Sorger, Thomas Ortner, Harald Piringer, Gerd Hesina, Eduard Gröller
In 20th International Symposium on Vision, Modeling and Visualization (VMV 2015). October 2015.

"*neuroMap - Interactive Graph-Visualization of the Fruit Fly's Neural Circuit*"
(with talk at BioVis 2013, Atlanta)
Johannes Sorger, Katja Bühler, Florian Schulze, Tianxiao Liu, Barry Dickson
In *Biological Data Visualization (BioVis), 2013 IEEE Symposium on* , pages 73-80. October 2013.
(Best Paper Award)

Skills

Research Interests:	Visual Analytics, Information Visualization, and Scientific Visualization – in the domains of biological/molecular visualization, geographic information systems, and simulations
Languages:	German: first language French: Berlitz Language Certificate: grade A English: Berlitz Language Certificate: grade A Spanish: Berlitz Language Certificate: grade A Japanese: beginner level
IT Skills:	(applied in 2D and 3D visualization, render engine-, GUI-, and game- programming)
proficient:	C++, C#, JAVA, JavaScript, LaTeX, yFiles
advanced:	Unity3D, database modeling (SQL), OpenGL 1.x – 3.x, ES, GLSL/HLSL (shader programming), Server/client web applications (AJAX), Web design, audio editing, video editing, graphic design (Photoshop, Illustrator), Qt, jquery, d3.js
beginner:	Image processing (MATLAB), GTK, Adobe Flash, Mobile platform programming (Android 2.x), 3D modeling and animation (Maya)
Driving License:	B
Personal Interests:	Traveling, photography, music, literature, outdoor sports, drawing.