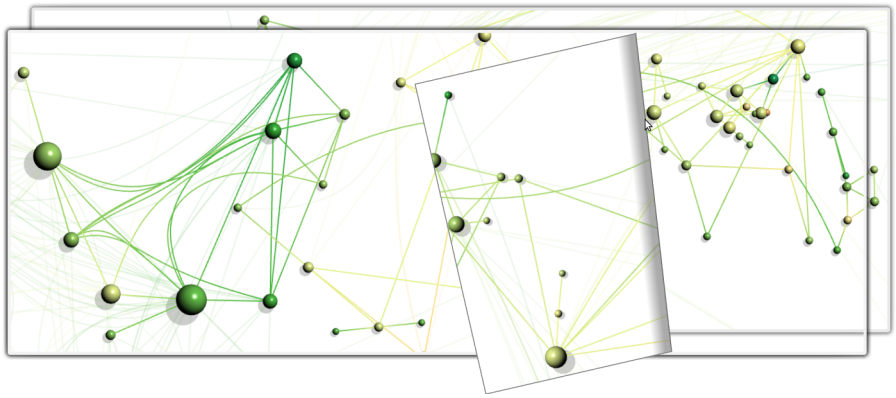
CONTRIBUTIONS TO THE CORNERSTONES OF INTERACTION IN VISUALIZATION

Strengthening the Interaction Side of Visualization



Habilitationsschrift
zur
Erlangung des akademischen Grades

Doktor-Ingenieur habilitatus (Dr.-Ing. habil.)

der Fakultät für Informatik und Elektrotechnik
der Universität Rostock

Vorgelegt von

Christian Tominski
Doktor-Ingenieur (Dr.-Ing.)

aus Rostock
geboren am 14. Juli 1977 in Greifswald

Rostock, 14. Juli 2014

urn:nbn:de:gbv:28-diss2015-0185-0

REVIEWERS:
Prof. Dr.-Ing. habil. Heidrun Schumann, Universität Rostock
prof.dr.ir. Jarke J. van Wijk, Technische Universiteit Eindhoven
Prof. Dipl.-Ing. Dr.techn. Helwig Hauser, Universitetet i Bergen
Prof. Dr.-Ing. Raimund Dachselt, Technische Universität Dresden

DATE OF DEFENSE:
October 6, 2015

DATE OF TRIAL LECTURE:
October 13, 2015

LOCATION:
Rostock

ABSTRACT

Visualization has become an accepted means for data exploration and analysis. Although interaction is an important component of visualization approaches, current visualization research pays less attention to interaction than to aspects of the graphical representation. Therefore, the goal of this work is to strengthen the interaction side of visualization. To this end, we establish a unified view on interaction in visualization. This unified view covers four cornerstones: the data, the tasks, the technology, and the human. Addressing challenges and questions related to these cornerstones, we develop novel concepts and techniques for interaction in the context of visualization. In addition to providing novel solutions for individual problems, a major contribution of this work is the comprehensive discussion of interaction in visualization as a whole. This also includes the formulation of research topics for future work with a focus on interaction.

ZUSAMMENFASSUNG

Visualisierung hat sich zu einem unverzichtbaren Werkzeug für die Exploration und Analyse von Daten entwickelt. Obwohl Interaktion ein wichtiger Bestandteil solcher Werkzeuge ist, wird der Interaktion in der aktuellen Visualisierungsforschung weniger Aufmerksamkeit gewidmet als Aspekten der graphischen Repräsentation. Daher ist es das Ziel dieser Arbeit, die Interaktion im Bereich der Visualisierung zu stärken. Hierzu wird eine einheitliche Sicht auf Interaktion in der Visualisierung entwickelt. Diese einheitliche Sicht stützt sich auf vier Eckpfeiler: die Daten, die Aufgaben, die Technologie und den Menschen. Die Herausforderungen und Fragestellungen, die mit diesen Eckpfeilern in Verbindung stehen, werden durch die Entwicklung neuer Interaktionskonzepte und -techniken adressiert. Neben der Bereitstellung neuer Lösungen für individuelle Probleme, liegt ein wichtiger Beitrag dieser Arbeit in der ganzheitlichen Diskussion von Interaktion im Visualisierungskontext. Hierzu gehört auch die Erarbeitung von Forschungsthemen für zukünftige Arbeiten mit Schwerpunkt auf Interaktion.

PREFACE

This thesis expresses my thoughts on interaction in visualization, a topic that has accompanied me since I have started doing research in the field of visualization. Visualization in general is concerned with generating visual depictions of some data, model, or concept for the purpose of helping humans to discover, to make sense, and to communicate. The idea is to utilize the enormous powers of the human visual system. Our perceptual capabilities enable us to process larger amounts of visual information and our cognitive skills allow us to spot interesting patterns as they appear.

Interaction deals with enabling the human to more actively take part in the process of gaining insight and extracting knowledge. Interaction helps us to understand the visual mapping behind a visual representation, to realize the effect of visualization parameters, and to get confident about the data and the features we crystallized from them. Interaction also provokes curiosity and allows people to experiment with different "what if" scenarios.

To underline the importance of interaction for human mental efforts I would like to reiterate a quote that many readers might already know:

> "Tell me and I will forget.
>
> Show me and I may remember.
>
> Involve me and I will understand."
>
> — Confucian proverb, around 300 – 200 B.C.

This Confucian proverb is definitely from an age that did not know electricity, not to mention computers or visualization. Yet it strikingly reflects the core motivation for interaction in visualization: Involve the users and they will understand the data more easily. When the interaction is done right, understanding will not only be developed for the data, but also for the process that generates the visual representation serving as a catalyst for knowledge crystallization. Understanding the visualization process will enable people to use visualization tools more efficiently by fine-tuning the available techniques to their needs and tasks.

Yet, interaction is not a universal cure. Interaction comes at a cost. In contrast to just looking at a visual representation, the user now acts upon the impression generated by the visualization. This implies cost for interpreting what has been seen and preparing and carrying out the action to respond. This prompts us to take a step back and

think more carefully about interaction and scenarios where alternative solutions might be a better choice.

Such alternatives can be analytic methods. Often we associate analytic methods with algorithms to analyze the data, for example, to extract clusters, to compute aggregations, or to identify patterns. Yet in the context of interactive scenarios, analysis also means keeping an eye on the user and the situation in which interaction is about to take place. Seemingly simple information gathered from observing the user can have a significant positive effect on how users experience the interaction.

These considerations led me to think of interaction as mediator between the user and the computer, but a mediator that has to be attuned to the specific characteristics of visualization. While I agree that many questions regarding interaction are studied well in the literature, I'm convinced that interaction in visualization is a relevant topic that deserves more attention. The goal is to reach a state where interaction and visualization are equally strong players in the concert of tools for people to live through the information age.

I will look at interaction in visualization from four perspectives: the data perspective, the task perspective, the technology perspective, and the human perspective. These perspectives direct our attention to the key cornerstones of interaction in visualization. Taken together, they constitute a unified view of that what needs to be considered when engineering interactive solutions in the context of visualization. I will address the particularities of the cornerstones by explaining two novel approaches for each of them. My explanations will focus on interaction, deliberately dedicating less space to the graphics involved in the visualization.

I hope that this work and the contributions made in relation to the individual cornerstones of interaction are valuable to the visualization community, provide new ideas and discussion to those who agree with me that thinking more deeply about interaction in visualization is vitally important, provoke thought in those who still doubt that, or maybe even convince them otherwise.

Christian Tominski Rostock, July 2014

SELECTED PUBLICATIONS

The insight delivered in this work is largely based on the following publications (in chronological order). A full list of the author's published work is available on page 317 at the end of this document and at: www.informatik.uni-rostock.de/~ct.

- C. Tominski, J. Abello, and H. Schumann. CGV – An Interactive Graph Visualization System. *Computers & Graphics*, 33(6):660–678, 2009.
- H. Piringer, C. Tominski, P. Muigg, and W. Berger. A Multi-Threading Architecture to Support Interactive Visual Exploration. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1113–1120, 2009.
- M. Spindler, C. Tominski, H. Schumann, and R. Dachselt. Tangible Views for Information Visualization. In *Proceedings of the International Conference on Interactive Tabletops and Surfaces (ITS)*, pages 157–166. ACM Press, 2010.
- C. Tominski. Event-Based Concepts for User-Driven Visualization. *Information Visualization*, 10(1):65–81, 2011.
- A. Lehmann, H. Schumann, O. Staadt, and C. Tominski. Physical Navigation to Support Graph Exploration on a Large High-Resolution Display. In G. Bebis et al., editors, *Advances in Visual Computing*, pages 496–507. Springer, 2011.
- C. Tominski, C. Forsell, and J. Johansson. Interaction Support for Visual Comparison Inspired by Natural Behavior. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2719–2728, 2012.
- C. Tominski, H. Schumann, G. Andrienko, and N. Andrienko. Stacking-Based Visualization of Trajectory Attribute Data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2565–2574, 2012.
- S. Gladisch, H. Schumann, and C. Tominski. Navigation Recommendations for Exploring Hierarchical Graphs. In G. Bebis et al., editors, *Advances in Visual Computing*, pages 36–47. Springer, 2013.
- S. Gladisch, H. Schumann, M. Ernst, G. Füllen, and C. Tominski. Semi-Automatic Editing of Graphs with Customized Layouts. *Computer Graphics Forum*, 33(3):381–390, 2014.

ACKNOWLEDGMENTS

"If I have seen further it is by standing on ye sholders of Giants."

— Isaac Newton, 1676

This work would not have been possible without the collaboration and support of many people. I'm deeply grateful for being given the opportunity to know them, to learn from them, and to work with them.

First and foremost, I thank Heidrun "Heidi" Schumann for being a great mentor, for maintaining an unparalleled working environment, for providing invaluable advice and motivation, for always seeing things positively, or simply for being – a dear friend!

I thank Jarke J. van Wijk, Helwig Hauser, and Raimund Dachselt for agreeing to review my habilitation thesis. Thank you very much, it is a great honor to have you as reviewers!

My habilitation thesis builds upon publications that I co-authored with several fellow researchers. James Abello worked with me on the CGV system. He has been a great mentor. I learned a lot from our discussions, but admittedly only a while later. With Harald Piringer and his colleagues I had the great pleasure to get my first InfoVis paper accepted. Our joint work on multi-threading for visualization purposes was a great experience. The collaboration on tangible views with Martin Spindler and Raimund Dachselt opened my view on what lies beyond mouse and keyboard. Their creativity was a great inspiration for my own work. Camilla Forsell and Jimmy Johansson were of great help in getting our work on interaction for visual comparison published. Especially their rich knowledge on carrying out and reporting on users studies was of immense value. Finally, I thank Natalia and Gennady Andrienko for collaborating with me on visualizing movement trajectories. Personally, I felt that we were an ideal team with inspiring ideas, creative designs, efficient engineering, and structured scientific thinking. To all my collaborators, thank you, I hope we do some joint work again soon.

Although not directly connected with the work presented here, I'm particularly thankful for having Wolfgang Aigner and Silvia Miksch as colleagues and friends. Our joint work on visualizing time-oriented data had much influence on me and my work. Co-authoring a book with you was an experience I would not like to miss.

I'm also grateful to my colleagues and collaborators at the University of Rostock. A special thank you goes to Hans-Jörg "Hansi"

Schulz, who provided valuable feedback on drafts of this work and with whom I enjoyed many discussions and delicious schnaps. Ordered by their date of appearance in my life as a visualization researcher, I'd also like to express my gratitude to: Petra Schulze-Wollgast, Thomas Nocke, René Rosenbaum, Georg Fuchs, Martin Luboschik, Falko Löffler, Malte Willert, Andrea Unger, Conrad Thiede, Stephan Ohl, Anke Lehmann, Axel Radloff, Steffen Hadlak, Quyen Nguyen, and most recently Christian Eichner and Stefan Gladisch. Thank you very much, it's been great working with you.

Finally, I thank those who make my life really count – my family. Anja, Vincent, and Arvid, I thank you for being what you are: a source of joy, a source of happiness, a source of love, a source of life.

CONTENTS

Part I

THE INTRO

This first part summarizes the key insights and results of this work. A common ground of understanding will be established by developing a unified, structured view on interaction in visualization. Under the umbrella of this view, novel interaction approaches addressing relevant open research questions will be presented.

1 INTRODUCTION

1.1 MOTIVATION AND BACKGROUND

Visual representations have been used for ages as a communication aid among human beings [244]. Nowadays, we live in a world full of data. Technological advances have led to a situation where we collect excessively far more data than we can make sense of. This problem has become known as *information overload* [317]. As early as in the 1980s, visualization pioneers recognized the enormous potential that modern computers would offer in terms of analytic power, graphics output, and interactive manipulation to address the information overload. McCormick et al. [250] formulated the key idea behind visualization as:

> "Visualization is a method of computing. It transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations. Visualization offers a method for seeing the unseen. It enriches the process of scientific discovery and fosters profound and unexpected insights."
>
> — McCormick et al. [250]

With this definition, McCormick and colleagues paved the way for visualization as a distinct field of computer science in general and computer graphics specifically. At its core, the definition brings together the capabilities of human perception and cognition and the computational abilities of computers as the key components. Accordingly, Card et al. [70] define visualization as the "use of computer-supported, interactive, visual representations of data to amplify cognition". Or as Ware [366] puts it plainly:

> "It is useful to think of the human and the computer together as a single cognitive entity, with the computer functioning as a kind of cognitive coprocessor to the human brain. [...] Each part of the system is doing what it does best. The computer can pre-process vast amounts of information. The human can do rapid pattern analysis and flexible decision making."
>
> — Ware [366]

Computer-generated visualization has always included the notion of interactivity. However, much of the literature on visualization focuses, in fact, on the visual part, not so much on the interaction part.

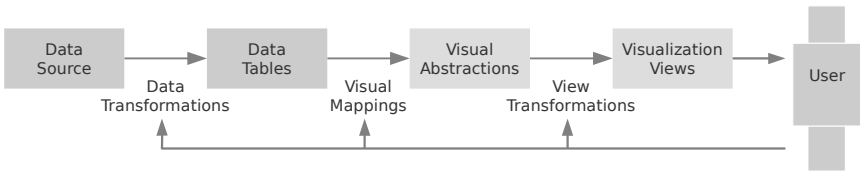


Figure 1.1: The visualization pipeline [70].

Many visualization publications describe in detail aspects of the visual representation, but less is reported about the design and the engineering of interaction in visualization. Several other researchers have taken note of this deficiency:

> "Even though interaction is an important part of information visualization (Infovis), it has garnered a relatively low level of attention from the Infovis community."
>
> — Yi et al. [385]

> "Until recently, the focus of InfoVis has been more on the graphical representation and less on the interaction."
>
> — Fekete [118]

> "Also, although interaction isn't yet a primary theme, the visualization research literature reflects an increasing focus on it."
>
> — Keefe [199]

> "Unfortunately, interaction is not discussed at all in graphic design, and even visualization textbooks tend to downplay this angle."
>
> — Elmqvist et al. [112]

A possible explanation for why interactive aspects are not on equal terms with visual aspects can be found in our model of visualization as manifested in the *visualization pipeline*, of which several variants exist [147, 70, 78, 100]. Let us take a closer look at the visualization pipeline by Card et al. [70] as depicted in Figure 1.1.

VISUAL ASPECTS In terms of visual aspects, the pipeline prescribes how data is to be transferred through several stages from a data source to data tables, to visual abstractions, and finally to visualization views. Data transformations, including filtering, clustering, error correction, can be found in the early stage of the pipeline. At the heart of the pipeline is the mapping of information to visual variables [56]. Finally, geometrical primitives and associated graphical attributes as generated in the mapping stage are rendered to visualization views, that is, to visual representations to be interpreted to gain insight.

The visualization literature describes a number of criteria that must be observed when data are transferred to visual representations. The economic model of visualization by van Wijk [354] demands that the benefits of using visualization as a means to gain insight outweigh the costs involved in carrying out the process (i.e., computation and interpretation). A necessary condition to achieve beneficial visual representations is to consider the two key visualization criteria *expressiveness* and *effectiveness* as identified by Mackinlay [243]. Visual representations are expressive if they actually do express the desired information. Effectiveness relates to exploiting the capabilities of the output medium and the human visual system.

Taken together, the visualization pipeline and related visualization criteria, provide widely accepted blueprints for generating meaningful visual representations of data.

INTERACTION ASPECTS Let us now consider the aspects of interaction and the level of detail with which the visualization pipeline illustrates how users are involved in the visualization process. The particular visualization pipeline in Figure 1.1 incorporates the user in two different roles. On the one hand, the user is the recipient of the information communicated via visualization views. On the other hand, the pipeline suggests that the user is an active participant controlling the different stages of the transformation from data to views.

However, in contrast to the description of visualization as a transformational process across several stages, the arrows indicating interaction in Figure 1.1 remain vague. Not much do we learn about the interaction itself, the interaction design, the principal interaction components, and how interaction can be implemented in a visualization infrastructure. This lack of details on interaction may be one reason for the imbalance of visual aspects and issues of interactivity.

1.2 GOAL AND CONTRIBUTION

The goal of this work is to address the discrepancy between visual aspects and aspects of interaction. To this end, we look at visualization from an interaction perspective. We develop an encompassing view of interaction in visualization bringing together the relevant concerns under a common hood. *Data* and *tasks* are key factors of visualization and likewise they are primary concerns to be considered for interaction. Our unified view is completed by the *human* user as the recipient of visual information and active participant in interactive data exploration and analysis, as well as by the *technology* providing the means for display, interaction, and computation. That said, the topic of research of this work is to investigate interaction in visualization comprehensively under a unified view along the four key factors data, tasks, human, and technology

Studying these factors individually, we contribute novel interaction approaches that, taken together, strengthen interaction in visualization as a whole. Addressing the *data*, we develop novel solutions taking into account both the structure of data as well as the spatio-temporal frame of reference in which data are usually given. With regard to *tasks*, we present novel interaction techniques for visual comparison and data editing, both of which being tasks that have not been investigated from an interaction perspective in previous studies. We introduce techniques that utilize different interaction *technologies*, including regular mouse and keyboard interaction, but also modern touch interfaces and physical interaction in front of large high-resolution displays. Focusing on the *human* user, we propose techniques for reducing interaction costs by considering inspiration from natural interaction, by following real-world workflows, and by integrating analytical methods.

The interaction side of visualization is also strengthened from an engineering perspective. We develop a novel multi-threading architecture that can serve as a general basis for engineering interactive visualization approaches. We explain how larger interactive visualization architectures can be designed by the example of a system for graph exploration. Our solutions incorporate modern display technology and interaction modalities to implement novel ways of interacting with visual representations of data.

Based on a summarizing reflection of the proposed interaction approaches, we identify directions for future research. With the unified view on interaction in visualization, with the novel interaction approaches taking into account the data, the tasks, the technology, and the human, as well as with the identification of open research questions, this work significantly contributes to lifting interaction to a level that corresponds to its undisputed importance.

1.3 OUTLINE

This thesis is divided into three parts. Part i, to which this introduction belongs, will review the basics of interaction, discuss the role of interaction in visualization, identify the challenges to be addressed in this work, and summarize our contributions accordingly. Chapter 2 starts with an introduction to the fundamental concepts of interaction in general and interaction in visualization specifically. This introduction will collect different definitions that are otherwise considered only separately. We further discuss the advantages and disadvantages of interaction in visualization and take a look at the issue of engineering interactive visualization solutions. Building upon the fundamentals and associated discussions, we identify challenges for interaction in visualization in Chapter 3.

In Chapter 4, we present our unified view on interaction in visualization with interaction engineering as the foundation upon which we build four cornerstones corresponding to the data, the tasks, the technology, and the human. Along this unified view, we introduce novel interaction approaches. The multi-threading architecture for interactive visualization proposed in Section 4.1 will address interaction engineering on a fundamental level. Section 4.2 will set the focus on the data aspect, introducing novel ways of interacting with graph structures and movement trajectories in space and time. The task aspect is taken up in Section 4.3, where we develop new interaction techniques for comparison and data editing tasks. In Section 4.4, we present tangible views and physical navigation in front of large displays as novel ways of interaction taking advantage of technological advance. Addressing the human user, we discuss the use of event-based methods and navigation recommendations as means to reduce interaction costs in Section 4.5. All approaches are described in a compact way, presenting the key messages with respect to interaction in visualization.

Chapters 5–13 of Part ii collect the original publications that this thesis builds upon. These chapters provide the details for the approaches that we summarize briefly in Sections 4.1–4.5. Except for the correction of typographical errors, the content presented in Part ii is identical to the original publications. The layout of figures and tables, the typesetting, and the citations have been harmonized to match the style of this thesis.

Part iii provides an overall summary and conclusion. Key concerns are to derive and discuss insights about the greater picture behind this work and to identify research topics for future work on interaction in visualization.

2 BASIC CONSIDERATIONS

This chapter takes a look at some fundamental aspects of interaction. We will first consider general human-computer interaction, before we shift our focus to visualization-specific questions of interaction.

2.1 HUMAN-COMPUTER INTERACTION

A primary source of scholarly literature on interaction is the realm of human-computer interaction (HCI) research [97, 87]. A general model for interaction, which also applies to human-computer interaction, is described by Norman [262], who conceptualizes interaction as a loop of two phases: the execution phase and the interpretation phase (see Figure 2.1). Starting with the human's goal, the execution phase is concerned with the formation of an intent to interact, the mental planning of the interaction, and the actual execution of the plan. Performing this first phase results in a response. Its interpretation is captured in the second phase of the loop. This includes the perception of the response, the mental interpretation of it, and its evaluation with regard to the intent that induced the interaction. As both phases occasion costs, Norman denotes them as *gulfs* of execution and evaluation.

Similar to the criteria expressiveness and effectiveness in visualization, there are certain criteria that interaction has to obey for the gulfs to be narrow. Usability [260] and user experience [154] are key categories which subsume criteria such as predictability, consistency, customizability, satisfaction, engagement, responsiveness, and task conformance, to name only a few. The rationales behind many of these criteria are also covered in a corresponding ISO standard [185].

Seeking to increase the level of conformance with these criteria, HCI researchers have proposed several themes of interaction. Early research focused on interaction using windows, icons, menus, and pointers, commonly known as the WIMP paradigm [97]. A classic

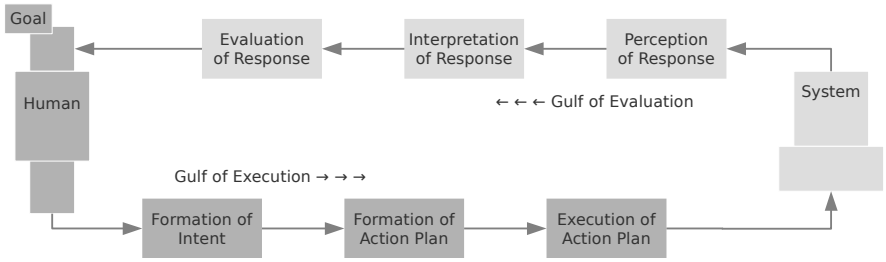


Figure 2.1: The human action cycle [262].

and most prevalent theme, also in visualization, is *direct manipulation* as proposed by Shneiderman [303]. With the advent of modern interaction devices, new ways of interacting became possible, the so-called post-WIMP era [349]. *Tangible interaction* [184] is based on interaction with tangible objects in the real world. Beaudouin-Lafon [50] coins the term *instrumental interaction* to capture the idea of using interaction instruments to manipulate domain objects. *Reality-based interaction* [187] and *natural interaction* [348] are the next steps in a line of recent developments that include aspects of greater awareness of the user and the environment in which the interaction takes place.

These models and studies from the HCI literature contribute largely to a better understanding of the role and needs of human beings interacting with computers in general. The specific aspects of interaction in the context of visualization deserve further elaboration.

2.2 INTERACTION IN VISUALIZATION

Spence [310] describes visualization as a tool to support humans in forming mental models of otherwise difficult-to-grasp complex phenomena. The fact that people *form* mental models suggests that interaction be a principal ingredient of visualization. In fact, the visualization pipeline by Card et al. [70] includes interaction in form of a user who controls the individual transformation stages and who receives responses via visual feedback (see Figure 1.1).

There are several reasons why users need to have interactive control. Given today's large and complex datasets it is usually impossible to encode all facets of all data into a single visual representation. Therefore, an iterative process has to take place during which different parts of the data are brought to the display with emphasis on different facets of the data. This exploration process generally follows the information seeking mantra by Shneiderman [305], where the user starts from an overview and then descends into the details on demand. Interaction is the key to enable the user to steer the exploration process, thus helping to gain a broad and comprehensive understanding of the underlying data.

Ware [367] explains interaction in visualization based on interaction loops at three interdependent levels. At the lowest level, interaction is concerned with basic operations of recognizing, pointing at, or manipulating objects (e.g., moving the mouse and performing a click to pinpoint a data item of interest). At an intermediate level, the user combines basic low-level operations to activities of exploring and navigating large visual data spaces (e.g., adjusting the visual encoding or visiting different parts of the data). The highest level captures processes of problem solving, which involves combining several activities to accomplish cognitively more demanding tasks (e.g., laying out different pieces of information for finding relations).

LOW-LEVEL INTERACTION From a conceptual point of view, different modes of interaction can be identified depending on how low-level actions are performed. Spence [310] distinguishes *stepped interaction* and *continuous interaction*. Stepped interaction is related to discrete, infrequent actions. In visualization, particularly for approaches that implement *direct manipulation* [303], the interaction-feedback loop is iterated at high frequency, continuously so to say. Continuous interaction is vitally important as it facilitates examining the visualized data with respect to multiple 'what if' scenarios in a short period of time. Apparently, continuous interaction requires easy-to-execute interactions and sufficient visual feedback, which must be provided quickly. Elmqvist et al. [112] expand on continuous interaction and propose the concept of *fluid interaction*, which integrates aspects of promoting flow, supporting direct manipulation, and minimizing the gulfs of execution and evaluation.

Heer and Shneiderman [161] underline the significance of the continuous and direct character of interaction in visualization: "To be most effective, visual analytics tools must support the fluent and flexible use of visualizations at rates resonant with the pace of human thought."

INTERMEDIATE-LEVEL INTERACTION In addition to thinking of interaction as step-wise or continuous manual operations, it makes sense to consider the activities carried out with actual interaction techniques. Spence [310], Ward et al. [365], and Ware [367] elaborate on classic and contemporary techniques for interacting with visual representations and the data behind them. The examples given by these authors document the versatility of the existing approaches. The concrete set of techniques to be made available to the users of a visualization solution mainly depends on what the users are expected to accomplish via interaction.

From an analysis of existing interaction techniques, Yi et al. [385] condense seven categories of user intents for interaction: *select* – mark something as interesting, *explore* – show me something else, *reconfigure* – show me a different arrangement, *encode* – show me a different representation, *abstract/elaborate* – show me more or less detail, *filter* – show me something conditionally, and finally *connect* – show me related items. For each intent, several interaction techniques are listed satisfying it in different ways.

Sedig and Parsons [300] expand upon these seven categories and suggest looking at action patterns, of which they list 28 exemplars, including unipolar actions such as selecting, navigating, searching, or comparing, as well as bipolar actions such as collapsing/expanding, composing/decomposing, inserting/removing, or animating/freezing. For each pattern identified, Sedig and Parsons [300] give examples of interaction techniques that instantiate the pattern.

HIGH-LEVEL INTERACTION Similar to combining low-level operations to realize intermediate-level activities, so are the activities a precursor to high-level problem solving, which involves setting data into relation, comparing pieces of information, or establishing and validating hypotheses. At this level, interaction is considered more broadly as a catalyst for analytic thinking and discovery.

The model by Pirolli and Card [273] describes the sensemaking process as two loops: the foraging loop and the sensemaking loop. The foraging loop is concerned with interactively gathering and extracting information. The subsequent sensemaking loop describes the development of mental models through schematization as well as hypothesis generation and validation. Liu et al. [238] take a closer look at the distributed cognitive processes being active when humans engage in a sensemaking dialog with visually represented information. They suggest that: "[...] cognition is more an emergent property of interactions between an individual and the environment through perception and action rather than a property bounded inside an individual." In a related context, Liu and Stasko [237] describe interaction as a tool for giving meaning to what is perceived, for collecting relevant information, and extracting and storing interesting findings.

So, at the highest level, interaction is more concerned with the dialog of the analyst and the knowledge artifacts extracted via interactive visual methods. This also involves interactively coordinating and organizing pieces of information in an analytic visual-interactive workspace.

In the range between low-level and high-level interaction, many visualization approaches provide standard interaction techniques employing standard mouse and keyboard interaction to satisfy common user tasks. Techniques such as *brushing* [52, 67] (or the enhanced *angular brushing* [155] or *compound brushing* [76]) have proved universally useful over the years. Yet there is more to interaction in visualization. Recent activities strive to take full advantage of interaction as a means to integrate the user more tightly into the visual analysis process. Corresponding challenges will be described next.

CHALLENGES OF INTERACTION The increasing importance of interaction for visually driven analytical methods has led researchers to start thinking more deeply about a "science of interaction", a topic that arose in the context of *visual analytics* – the science of analytical reasoning supported by interactive visual interfaces. Thomas and Cook [325] see interaction as "the fuel for analytic discourse" and convincingly explain why developing a new science of interaction is top priority. Several research agendas have been proposed by various researchers.

Pike et al. [269] identify research challenges for interaction, covering ubiquitous, embodied interaction, capturing user intentionality,

knowledge-based interfaces, collaboration, principles of design and perception, interoperability, as well as interaction evaluation. Elmqvist et al. [112] envision future interaction research toward an interaction exemplar repository, visualization design patterns, and visualization criticism. Lee et al. [224] point our attention to the mismatch of available interaction technology and the level to which visualization research has utilized it so far. They define research challenges related to utilizing modern interaction modalities, providing freedom of expression, taking into account social aspects, breaking down barriers between humans and technology, and gaining a better understanding of human behavior.

These studies provide excellent analyses of the status quo of interaction in visualization and enthusiastic calls to action for more research on interaction in visualization. As such, they help strengthen the interaction side of visualization. The level to which interaction can be raised will depend on the concrete responses to the identified research challenges in form of new interaction concepts, models, and techniques for visualization.

The previous paragraphs indicate that interaction is employed extensively in visualization. The diversity of user intents [385] and action patterns [300] suggest that virtually every visualization approach offers at least basic interaction. Research agendas have been proposed to advance the topic of interaction in visualization. There are also more theoretical models of interaction, describing different modes and different levels of interacting. However, these models are often not considered when designing and implementing interactive visualization solutions. While the visual design is usually developed based on explicitly addressing graphics design rules, the rationale behind the interaction design at times remains vague at best, not to say it does not exist. Addressing this deficiency by making the design of interaction in visualization more explicit is a goal of this work. Further, we address some of the challenges mentioned in the recent research agendas.

Yet before we do so, we discuss in more detail why interaction in visualization is special and what advantages, disadvantages, and difficulties might be associated with interaction.

2.3 THE INTERACTION-VISUALIZATION GAP

Arguably there is still a discrepancy between interaction research and visualization research [104]. Zudilova-Seinstra et al. [393] even see "a major communication gap between HCI experts and those developing visualization algorithms and systems". As a consequence research results from one field do not always transfer smoothly to the respective other field. Why is this so?

Traditionally, interaction research is focused on the human being. Human-centered design methodologies, models of perception and cognition, accessible devices, as well as social aspects are among the contributions of interaction research. On the other hand, visualization research has the focus more on the computer. Automatic extraction of data features, algorithms for the visual mapping, and implementations of visualization systems, and high-performance graphics via GPU acceleration are topics that can be found in the visualization literature. This is not to say that either side neglects the user or the computer, but undeniably the foci are different.

Yet efforts have been made to bring both worlds closer to each other as documented by Fikkert et al. [122]. Keefe [199] states that the "momentum recently seems to be increasing toward integrating visualization research [...] with interaction research [...]." He further identifies two factors that make interaction in visualization different from classic interaction research:

- complex analysis tasks defined by a specific, highly motivated user population and
- complex data.

Looking from a visualization perspective there is another key difference regarding the use of the visual channel. In interaction research, the interface between human and computer is in the focus. Visual aspects play an important role mainly in terms of the interface design. In other words, the visual channel is almost exclusively reserved for the graphical interface between human and computer. In visualization however, the visual channel has to serve both the visual representation of the data and the graphical interface at the same time. This key difference and the resulting conflict over visual resources adds to the already complex endeavor of developing interactive visualization approaches.

2.4 INTERACTION – USEFUL OR HARMFUL?

Much has been done in recent years helping us to better understand how interaction actually works and how it affects the ability of human beings to extract knowledge from data. The many references cited in the previous sections are generally in favor of interaction and recognize it as a strong positive factor of visualization. But there are also critical voices. We shall discuss this matter very briefly in order to make the reader aware of the potential disadvantages of interaction. This is not to cast a poor light on interaction, but rather to underline the importance of thinking carefully about interaction in visualization.

USEFUL INTERACTION Let us start positively. The human-in-the-loop argument is often brought forward, putting the human in the position to make the final decisions, rather than leaving it to the computer [325]. Examples of useful interaction can be found en masse in the literature. Empirical evaluation in form of quantitative or qualitative studies testify to the benefit of interaction [292, 26].

Interaction enables the human to generate or influence results in a way that goes beyond what is computable. Wegner [372] explains why interaction is more powerful than algorithms. His discussion is based on a theoretical model of *Interaction machines*, which he demonstrates to be more powerful than *Turing machines*. Such theoretical considerations further strengthen the already positive picture that is generally drawn of interaction in visualization [385, 310, 367, 161, 200].

HARMFUL INTERACTION On the other hand, a recent definition of *interactivity* as the *quality of interaction* [301] suggests that there is a spectrum of interaction with positive and not-so-positive elements. Cooper et al. [87] identifies the mismatch between the *implementation model* and the *user model* as a primary source of interaction problems. In the context of visualization, the underlying models are often complex and matching them is not always easy. Failing in this regard most likely leads to bad interaction, of which several examples exist in working visualization software prototypes, but which are hardly reported in scientific publications.

In addition to concrete examples of bad interaction, there can be general reservations about interaction. Tominski et al. [334] reports on the results of a questionnaire that was carried out to assess the distribution and use of interactive visualization tools in climate research. An observation particularly related to interaction was that there can be a kind of mistrust in interaction in general. The participants feared the arbitrariness of visual representations that have been generated by interactive adjustments of thresholds or visualization parameters. The reason was that it is no longer clear if an artifact identified in the picture is an actual feature in the data or just a pattern-by-chance. Such statements point us to provide interaction in a balanced way, offering flexibility, but within reasonable boundaries to avoid arbitrariness.

There are even voices that openly challenge interaction in an attempt to break the myth of interaction as a universal cure. Victor [356] discusses interaction in the context of software to learn, to create, and to communicate, which are activities that are certainly addressed by visualization software as well. While interaction is indispensable for creating (e.g., constructing a CAD model), interaction for learning (e.g., accessing information from a visual representation) "is considered harmful". Victor argues that only as a last resort should input be solicited from the user. Considering the costs involved in interaction [221], this is a sensible argument. Victor's critique mainly relates to

the tendency of being quick with answering a particular user need with providing a way to satisfy it interactively. But it is the task of the system (and the designer of the system beforehand) to provide the information needed in a particular situation. This points us to think of interaction in a *less-is-more* way, and to infer, where possible, how interaction steps can be reduced.

Still, overall a positive image of interaction shall be maintained. To this end, we have to design and implement our approaches according to accepted criteria and models of interaction and under consideration of the requirements of visualization. In the context of large and complex data and comprehensive analytic tasks, this is not an easy endeavor.

2.5 ENGINEERING INTERACTIVE VISUALIZATION SOLUTIONS

So far, we have considered interaction as a means for the user to steer the visualization and to engage in an analytic dialog with the represented data. We also reflected on good and bad interaction, briefly touching the domain of interaction design. Yet, interaction must also be implemented as a workable solution in order to be actually usable and useful. The following paragraphs will therefore review some aspects of engineering interaction in the context of visualization.

Capturing the technical essence of interaction, Jankun-Kelly et al. [188] propose the *P-Set Model of visualization exploration*. This model formalizes user interaction as changes of visualization parameters. In other words, any concrete interaction is abstractly interpreted as specification of a visualization parameter, where concrete parameters can be manifold, e.g., the viewing angle into a 3D scene, the focus point of an interactive lens, thresholds of a dynamic query operation, or parameters that control a clustering algorithm.

For smooth and efficient interaction in the sense of fluid interaction, visual feedback has to be generated quickly, within 50 - 100 ms [304, 310] after a visualization parameter has been adjusted. However, even data of moderate size can pose computational challenges. Depending on the adjustment made, it might be necessary to re-compute the entire visualization pipeline, including analytical methods, visual abstraction, and rendering of potentially many graphical primitives. The risk for interaction is that visual feedback might lag, disrupting the interaction loop.

Another aspect adds to the time costs for presenting visual feedback. As interaction involves change, we have to take care that the users understand what is happening. Abrupt changes in the visual display may hurt the mental model that users are developing when exploring visual representations of data. There is evidence that interpolating the parameter change and applying animation to present

the visual feedback is a better solution [160, 276]. However, animations take time as well, and the potentially costly interpolation of parameter changes cannot be neglected.

As can be seen from the previous discussion, interaction engineers face a two-sided conflict. On the one hand, interaction needs *synchrony*, which comes down to all-time responsiveness of the visualization system and immediacy of the visual feedback. A system that is unresponsive and blocked while computing is a worst-case scenario for the user. On the other hand, interaction needs *asynchrony* for the computation of the visual feedback and for its animation. The difficulty is to integrate synchrony and asynchrony.

This technical issue is only one among many that need to be addressed when engineering interactive software. Letondal et al. [232] provide a more complete list of requirements for interaction-oriented development tools. Although their analysis is not specific to visualization, it can be taken for granted that the points raised hold true for the engineering of interaction in visualization as well. There are already several novel approaches to aid in implementing interaction. Examples include overcoming call-back spaghetti code via interaction state machines [37] or extensions of the model-view-controller (MVC) pattern to a model-display-picking-controller (MDPC) pattern that makes selection, which is so important in visualization, a prime component in the software infrastructure [85].

Despite the availability of such advanced concepts for developing interactive software, today's visualization solutions do not yet take full advantages of them. This calls for facilitating the engineering of interactive visualization through developing new methods and raising awareness of the already existing ones.

2.6 SUMMARIZING THOUGHTS

The previous sections looked at interaction from different points of view. We covered general aspects of human-computer interaction, different levels of interaction in visualization, the interaction-visualization gap, the question of good and not-so-good interaction, as well as the complexities of engineering interactive software. Figure 2.2 provides a summarizing view of these aspects and shall serve as a basis for the following discussion.

Interaction in visualization has to be approached from a human-centered angle. In Figure 2.2, this is illustrated by a human and a box labeled interaction design. The design of useful interaction must consider the human sensory and motor skills as well as the data exploration activities and sensemaking tasks that humans are engaged with.

Interaction in visualization has to be considered from a technology-oriented perspective as well. This is indicated by a system and a box

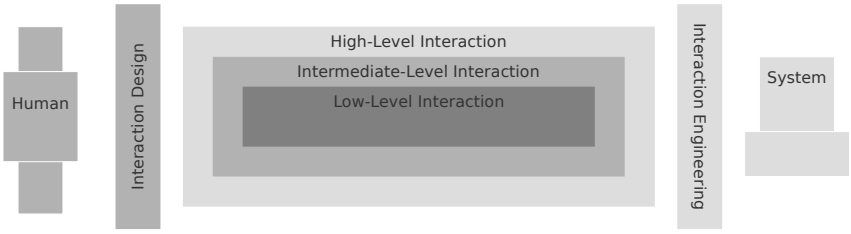


Figure 2.2: Summarizing model of interaction.

label interaction engineering in Figure 2.2. Engineering interaction means transferring designs into workable tools utilizing computer technology to manage, analyze, transform, and represent data by interactive visual means.

There is a mutual dependency between designing and engineering interaction. Well-designed interaction is rendered useless with insufficient engineering and the best architectures lie fallow without appropriate interaction designs.

In visualization scenarios, we have to consider multiple levels of interaction as illustrated in the center of Figure 2.2. These different levels add to the complexity of the interplay of design and engineering. Addressing low-level operations is fundamental because they have definite impact on the higher levels of interaction. But focusing on lower-level aspects alone is not sufficient. Intermediate-level activities must be taken into account and these must be put in the context of higher-level analytic thinking.

The interdependent design and engineering of interaction in visualization at multiple levels is the context to which this thesis contributes. To give only a few examples, Chapter 5 contributes to the engineering aspect and introduces a novel multi-threading visualization architecture addressing the computational requirements for fluid interactive visual exploration. The engineering of a larger interactive graph visualization system including several visualization techniques and novel interaction approaches is described in Chapter 6. An architecture for combining interactive visualization with event-based methods is described in Chapter 12.

Novel interaction designs are proposed for different visualization scenarios. We are concerned with low-level and intermediate-level interaction, and also touch aspects of high-level interaction. Examples for novel ways of how to fundamentally interact with visualizations are the tangible views from Chapter 10 or physical navigation as explained in Chapter 11. Novel methods for intermediate-level exploration and navigation will be introduced in Chapters 6, 7, and 13 in the context of graph data and in the context of spatio-temporal movement data. Chapter 8 introduces interaction support for more complex visual comparison tasks. Increasing the complexity further, Chapter 9 focuses on a semi-automatic approach for editing graphs

with customized layouts. Higher-level analytic working is also facilitated by interaction across multiple coordinated views as provided by the interactive graph visualization system CGV, which is described in Chapter 6.

Across all levels, our goal is to develop 'good' interaction approaches that are useful and usable. This implies to keep interaction costs low by designing according to humans' natural behavior as in Chapter 8 and by integrating assistive methods as in Chapter 13. That we achieved our goal is indicated by quite enthusiastic user feedback: "The software is nicely implemented, everything is very harmonic." from Chapter 8 or "The interaction is intuitive and works very much as expected." from Chapter 7. Being aware of the influencing factors and models of interaction as well as the potential down sides of interaction helped us to shape such positively received solutions.

A difficulty that persists is that the presented approaches can hardly be delineated according to the aspects illustrated in Figure 2.2. This is due to the mentioned mutual dependencies and the intertwined nature of the topic. To establish a clearer structure we next develop a unified view along four key aspects of interaction in visualization.

3

CORNERSTONES OF INTERACTION IN VISUALIZATION

The goal of this chapter is to define a structured view on the contributions made in this thesis and with this a unified view on interaction in visualization. Our view on interaction is centered on four cornerstones:

- Data
- Tasks
- Technology
- Human

The first cornerstone are the *data*. Data are a primary concern of visualization and so they are a key factor of interaction as well. The second cornerstone are the *tasks* that have to be accomplished for a given dataset. Data and task taken together can be considered the core of our view. This core is flanked on the one side by the *technology* as the third cornerstone. The flank on the other side is taken by the *human*. The human and the technology carry out the given tasks on the data in a cooperative effort. The human is recipient of visual representations generated by the technology. The technology provides interaction modalities allowing the human to act upon what is perceived from the visual representations. Sensemaking is usually left to the human, whereas complex computations are taken care of by the technology.

With these four cornerstones, we can better grasp the key factors of interaction in visualization. Next, these cornerstones and corresponding challenges shall be discussed in more detail.

3.1 THE DATA

Data characteristics have long since been identified as a factor that influences the visual design [305]. This holds true for the mapping of data features to visual variables as well as for distinction of classes of visualization techniques for different types of data.

In terms of the mapping as the core of the visualization pipeline it is general practice to use visual variables [56] that suit the characteristics of the data. Mackinlay [243] suggests good matches of data and visual variables based on empirical studies by Cleveland and McGill [82], which have been confirmed more recently by Heer and

Bostock [158]. These studies tell visualization designers that, for example, quantitative data are best visualized using the visual variables position and length, whereas density and color hue play increasingly important roles for qualitative data.

The distinction of classes of visualization techniques for different types of data need not even be confirmed by empirical studies because they are ubiquitous in the visualization community. Flow visualization, volume visualization, graph visualization, geo visualization, or time-series visualization all address a specific type of data and are distinctly represented by established books, journals, conferences, and workshops.

When we look at the interaction design of visualization approaches and how the data characteristics are considered there, no such data-centric view can be found. However, there is obviously a difference, for example, between interactively navigating along the dimension of time of time-series data and navigating in geographical space of a dataset of movement trajectories. For another example, selecting a subset of a numerical attribute (e.g., a closed interval around a point of interest) is different from selecting a subset of a graph (e.g., the k-neighborhood of a node of interest). Yet such differences are usually not addressed explicitly. Not much is reported on how the interaction design can be attuned to the character of the data.

There are examples of visualization approaches in the literature that illustrate how data-aware interaction design can support users. Techniques for interacting with visual representations of time-oriented data [170, 173, 390] demonstrate this quite well. Yet, more research is needed, not only for other types of data, but also for dealing with the complexity and multifaceted character of today's datasets.

Challenge: The interaction design of a visualization approach should consider the characteristics of the data, very much as the visual design already does.

3.2 THE TASKS

In addition to considering data characteristics, there is the issue of addressing the tasks to be accomplished with interactive visualization. Tasks as influencing factor for the visualization design have attracted more and more research in recent years. While classic distinctions of high-level tasks such as exploration (i.e., forming hypotheses), confirmation (i.e., falsifying hypotheses) and presentation (i.e., communicating findings) are still valid [365], more fine-grained and formal models such as proposed by Amar et al. [24], Andrienko and Andrienko [36], or Schulz et al. [297] help us to better understand the role of tasks in the visualization design process.

There are examples of visualization approaches that explicitly make the visual design dependent on the tasks. Tominski et al. [331] use dif-

ferent color scales for identification and localization tasks. Comparison tasks are supported through combining individual color scales into a global color scale. As a result, the user is presented with a visual representation that suits the nature of the task at hand.

Again, we do not have such a task-driven view on the interaction side of visualization. There are first studies that investigate which user intents are supported by existing interaction techniques. For example, Yi et al. [385] associate dynamic query controls [19, 304] with the user intent *filter – show me something conditionally*. Yet, these studies come as an afterthought. We still lack understanding of how to design the interaction according to given tasks. This does not mean that existing techniques do not work. In fact, there are many examples of interactive solutions that superbly support users in carrying out visualization tasks. But the design process, including addressed tasks, suitable concepts and alternatives, as well as implementation models, is often not disclosed. This leads to a situation, where interaction techniques appear to be the result of ad-hoc decisions of the visualization author.

A task-centric view on interaction in visualization is needed to better attune interaction to what the users actually aim to accomplish. Recent studies confirm this thinking. For instance, Sedig and Parsons [300] extend the work of Yi et al. [385] and describe a rich set of patterns of interaction for complex cognitive activities with visual representations. But still more research is needed to cover tasks more comprehensively. A particularly relevant aspect not included in current studies is research on combining the task of interactive visual exploration and the task of visual manipulation (or editing) of data [48, 197].

Challenge: Tasks of analytic data exploration and manipulation need to become a determining factor for the design of interaction in visualization.

3.3 THE TECHNOLOGY

When we look at the settings in which visualization tasks are primarily carried out these days, we will most certainly see the classic setup where a user is sitting at an off-the-shelf desktop computer. A regular display shows visual representations, while mouse and keyboard are used for interacting with the system. This setup has been predominant for years. Yet, new technologies have emerged in terms of both display devices and interaction modalities.

Modern display devices, such as large high-resolution displays or small hand-held displays, have been addressed in the visualization literature by means of adapting existing visualization approaches or devising new ones [150, 387, 124, 386]. Modern interaction technologies, such as multi-touch interaction or tangible interaction, create a

similar need for rethinking existing visualization solutions with regard to interaction. There are already approaches that address this need [360, 211, 388]. Yet novel interaction modalities only slowly find their way into the visualization domain, although they considerably broaden the spectrum of what is possible. For example, dissolving display boundaries and blending display and interaction into a single interactive-visual medium enables us to make Sheiderman's [303] direct manipulation truly direct. Forming mental models based on interactively exploring and manipulating data directly under one's fingertips appears to be quite a promising prospect.

However, as discussed by Tominski et al. [335] and Lee et al. [224] there is a gap in terms of promising new possibilities on the one hand, but only little integration of these possibilities into visualization research and applications on the other hand. Taking advantage of novel technologies is, however, not a straight-forward task. Developing interaction for classic devices is already a complex matter, and taking additional technologies into account further increases the demands in terms of both designing interactions and actually implementing them. Finding appropriate combinations of display technology and interaction modalities and seamlessly integrating the technology into visualization workflows will be key concerns.

Challenge: Interaction in visualization should take advantage of technological progress by integrating modern display devices and interaction modalities.

3.4 THE HUMAN

From the early beginnings, visualization has considered effectiveness vitally important. The effectiveness criterion demands that visual representations be designed so as to heed the characteristics of the human visual system [243]. Contemporary research continues to investigate human aspects as documented by Kerren et al. [204] and Huang [175] in the context of human-centered or human centric visualization.

From an interaction point of view, usability [260] and user experience [154] are key factors to be considered. In the context of interaction in visualization, this means that we have to keep an eye on the costs involved when users interact with visual representations of data. Lam [221] categorizes costs with respect to Norman's [262] seven stages of interaction. Her framework identifies costs to form goals, to form system operations, to form physical sequences, to execute sequences, to perceive state, to interpret perception, and to evaluate perception.

Elmqvist et al. [112] specify several requirements for interaction in visualization. These requirements demand that interaction be fluid in

terms of the performance of actions, the presentation of smoothly animated visual feedback, and in terms of switching seamlessly between different tasks as they occur in visualization applications.

Keefe and Isenberg [200] go one step further and argue for natural user interfaces in visualization. In doing so, they pick up a hot trend from HCI research in natural interaction [348, 375]. Although the term *natural* is still debated among researchers, the overall goal is generally agreed upon: Make interaction with the computer more akin to how humans interact naturally in the real world.

The essence of all these efforts is to make Norman's [262] cycle of action and interpretation of the result smooth and efficient. More concretely, the interaction has to be designed so that it can be executed effortlessly and the visual feedback must be easy to interpret, even with feature-rich data visualizations in the background. Keeping the costs low will enable us to keep the user in the flow, which increases efficiency in working with interactive visualizations, and which can also improve the user experience when carrying out data analysis or data exploration tasks.

Reducing costs, maintaining fluid interaction, and striving for naturalness are all challenges related to the human interacting with visualization tools. While we are now at a point were awareness of these issues is increasing with first results actually being published, there are still many unsolved problems to be addressed.

Challenge: Interaction in visualization needs to pay attention to interaction costs in order to narrow the gulfs of interaction and achieve fluidity and naturalness.

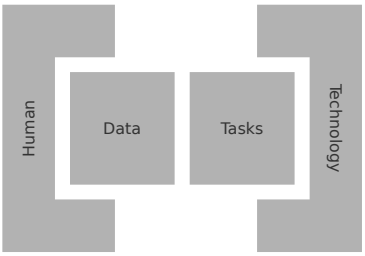


Figure 3.1: Unified view based on the cornerstones of interaction in visualization.

A summarizing illustration of the cornerstones constituting the unified view on interaction in visualization is presented in Figure 3.1. In the next section, we will further elaborate on these cornerstones as the structure for the contributions made in this thesis.

4 NOVEL APPROACHES OF INTERACTION

Structured according to the cornerstones of data, tasks, technology, and human, the following sections will introduce novel approaches of interaction in visualization. For each section, the focus will be set on a specific cornerstone. In addition to focusing on individual cornerstones, we will also establish connections across them, illustrating the overarching character of the proposed solutions.

The beneficiaries of these solutions are mostly the human users. They are the ones who have to work on their tasks related to their data using the technology available to them, and hopefully their work will be effective and effortless thanks to good interaction design and appropriate interaction engineering.

Yet, in another role, humans also act as the designers and the engineers. Addressing their needs and providing supportive approaches and infrastructures is a challenge in its own right [201, 86]. Still this thesis also contributes to easing the human engineer's life by providing a novel model for a multi-threading architecture. This architecture can serve as a foundation for interactive visualization solutions, from which in turn, end-users can profit as well.

Figure 4.1 provides an overview of the cornerstones of interaction in visualization and associated sections and chapters. The sections in this chapter offer brief summaries of the contributions to be explained in full detail in the later chapters based on the original publications underlying this work. We start in the next section with addressing interaction engineering as the basis for usable interaction in visualization.

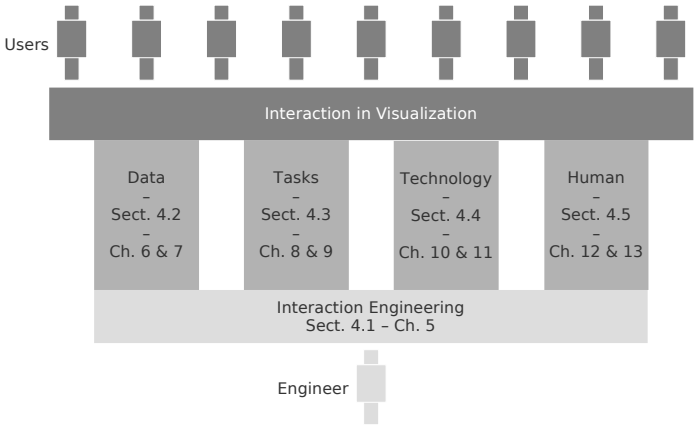


Figure 4.1: Overview of our contributions to interaction in visualization.

4.1 ARCHITECTURE FOR EFFICIENT INTERACTIVE VISUALIZATION

The basis for any useful interactive visualization is an architecture that realizes the interaction-feedback loop efficiently. Heer and Shneiderman [161] express the following concerns regarding the engineering of efficient visualization infrastructures:

> "Especially for large datasets, supporting real-time interactivity requires careful attention to system design and poses important research challenges ranging from low-latency architectures to intelligent sampling and aggregation methods."
>
> — Heer and Shneiderman [161]

Bringing in line *synchrony* (related to responsiveness and immediacy of the visual feedback) with *asynchrony* (related to the time cost involved to generate and present the visual feedback) has already been identified as a major challenge. A straight-forward implementation of the classic visualization pipeline will not be helpful in this regard. Any computation along the pipeline that fails to deliver results within interactive response time (ca. 50 - 100 ms for continuous interaction) will disrupt the interaction-feedback loop, and thus hinder fluid interactive analytic work. What is needed is an architecture that can cope with complex, time-consuming computation and is able to react to interactive user requests at any time, while providing visual feedback as rich as possible. Utilizing the advantages of modern multi-core processors would be one option. However, developing interactive multi-threading solutions is notoriously difficult and prone to manifold implementation errors [226].

CONTRIBUTION To avoid these difficulties and to make implementing multi-threading less error-prone, we designed a general multi-threading architecture around the concepts of *early thread termination* and *layered visualization*. Using multiple computing threads accommodates the need for asynchrony, and early thread termination accommodates the need for synchrony. Using multiple visualization layers makes it possible to scale the richness of the visual feedback according to the available computing resources.

The conceptual model of the general multi-threading architecture is illustrated in Figure 4.2. The architecture consists of four principal components: two storage components (the input storage and the output storage) plus two processing components (the event handler thread and the visualization threads).

The input storage consists of the data to be visualized and the visualization parameters, which is in line with the *P-Set Model* [188]. The output storage holds the visual representations. These include the visual feedback and the data visualization. The visual representations

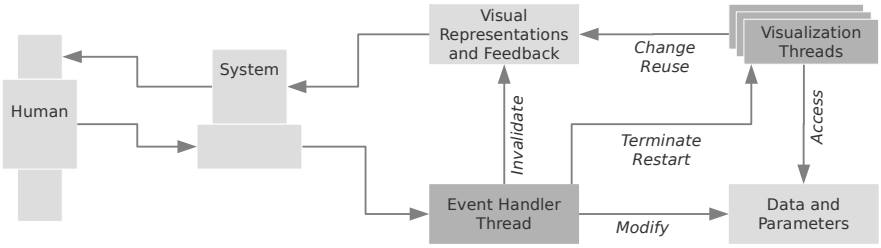


Figure 4.2: Architecture of multi-threaded visualization.

are not necessary complete, but can be partial if a computation had to be terminated early due to user interaction.

In terms of the processing components, the architecture is based on separations of concerns to be able to cope with synchrony and asynchrony. The key to responsiveness is the dedicated event handler thread. The only responsibilities of this thread are to receive interaction input events from the user and to perform signaling operations with respect to the other components of the architecture. The visualization threads are responsible for transforming input data and parameters into visual output. The visual output is subdivided into multiple layers according to different strategies (e.g., semantic layers, incremental layers, or level-of-detail layers). Using multiple layers enables the architecture to provide rich and scalable visual feedback, and to avoid redundant computations by reusing cached results that remain valid after a user interaction.

RESULTS AND DISCUSSION Empirical studies confirm that the architecture offers significant performance improvements, and hence provides a good basis for implementing interactive visualization solutions. The clear structure of the architecture further helps in avoiding typical programming errors that occur when multiple threads are involved. Successful installments of the architecture [329, 98, 271, 332] testify to its usefulness and general applicability across programming language boundaries.

The developed architecture is a contribution to the engineering side of interaction in visualization and affects interaction on all levels. More details on the architecture in general and the early thread termination and layered visualization in particular along with a discussion of design choices and empirical studies can be found in Chapter 5. An actual system instantiating the architecture is described in Chapter 6. The visualization tools mentioned in Chapter 12 have also been built on the multi-threaded approach.

The currency of the topic of efficient architectures for interactive visualization remains unbroken. The visualization research community continues to address this and related challenges via new concepts, data structures, and implementations [392, 239, 178, 235, 88].

4.2 DATA CHARACTERISTICS AND INTERACTION

The previous section laid out a foundation upon which one can consider interaction according to the four cornerstones of data, tasks, technology, and human.

The first cornerstone to look at is the interplay of data characteristics and interaction approaches in visualization. There are a number of standard interaction techniques that work well without considering the data characteristics. One such technique is *brushing* [52, 67, 155, 76], which is related to low-level selection. At intermediate and higher levels of interaction, it becomes more important to take the characteristics of the data into account.

Here we assume a data model similar to the one proposed by Kreuseler and Schumann [218]. Data are defined as data entities that are associated with quantitative or qualitative data values, and the data are characterized by:

1. the data's structure and
2. the data's frame of reference.

The structural aspect captures relations among data entities. Structures can be described as graphs in a most general sense. The data's frame of reference captures the spatial and temporal context in which the data have been collected or generated.

We contribute novel interaction approaches that particularly consider the structure inherent in the data and the spatial and temporal dependencies of data. Our focus will be on novel navigation techniques and interactive lenses for graph data. Addressing complex spatio-temporal data, we present a novel approach to interactively exploring movement trajectories. Our approaches are both, designed according to the data characteristics and engineered so as to exploit the data characteristics.

4.2.1 *Interacting with Visual Representations of Graphs*

In recent years, graphs have gained increasing importance in many fields of study. A graph is a universal model that helps structuring and relating entities of interest, be it enzymes in biomedical networks, people and their social behavior, or simply pieces of information. The increased importance of graphs led to an increased demand of interactive visualization approaches for graph data, taking into account the many different applications scenarios in which graphs occur.

Classic graph drawing has been mainly concerned with generating layouts of graphs according to different aesthetic criteria [47, 319]. Aspects of interaction with graphs and their visual representations have attracted only little attention, but this topic is becoming increasingly relevant. McGuffin and Jurisica [252] state that:

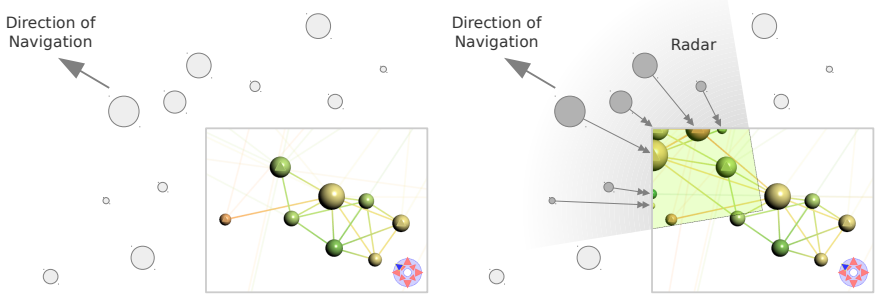
> "[...], there remains a significant need for users to be able to interactively manipulate such graphs, [...]"
>
> — McGuffin and Jurisica [252]

As documented in a recent survey by von Landesberger et al. [361], graph visualization approaches have begun incorporating more advanced interaction techniques. But still, supporting the navigation and exploration of complex graphs with appropriate interactive methods remains important.

CONTRIBUTION Addressing this need, we develop novel interaction techniques for navigating and exploring node-link graph layouts. What we call *radar view* and *edge-based traveling* are interactive visual tools for easy navigation with a preview of what can be expected to be found in the direction of traveling. Novel interactive lenses will be presented as tools for graph exploration. The *local-edge lens* and the *layout lens* can be used to tidy up edge clutter and to create local overviews of the neighborhood of nodes of interest on demand.

TECHNIQUES FOR NAVIGATING GRAPHS Large graph layouts are often presented in zoomable spaces [53], which implies that usually only a fraction of the data is visible at a time. When navigating in such space, users have to be supported in finding answers to the questions "Where can I usefully go?" and "What lies beyond?" [310]. The *radar view* provides an answer to these questions.

As illustrated in Figure 4.3, the radar view is a technique that provides a look-ahead in the direction of navigation. All off-screen nodes that fall into the radar are projected onto the view border to make them visible to the user. The radar automatically follows any change of direction initiated by the user. This way, the user can quickly acquire an overview and navigate to potentially interesting candidates to be visited next.

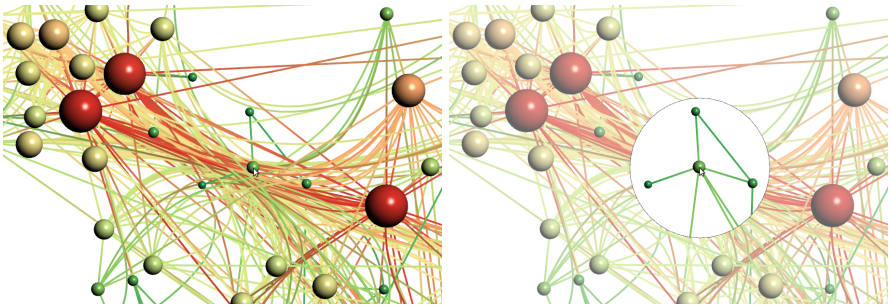


(a) Navigating without radar view. (b) Navigating with radar view.

Figure 4.3: The radar view.

Yet, covering larger distances can be cumbersome and time consuming. This is where *edge-based traveling* comes into play. The key idea is to utilize the structure of the graph as a kind of railway system across which the user can travel easily from one station to another. By simply clicking on edges, the user can cover larger distances in the layout effortlessly. Smooth and efficient animation [355] makes the navigation steps comprehensible.

LENS TECHNIQUES FOR GRAPH EXPLORATION Even sophisticated algorithms cannot guarantee that layouts of larger graphs are clutter-free and that adjacent nodes lie close together. So, users exploring a graph may encounter edge clutter and investigating a node's connectivity may be complicated due to widely distributed neighbors. The interactive graph lenses presented next enable users to overcome these difficulties.

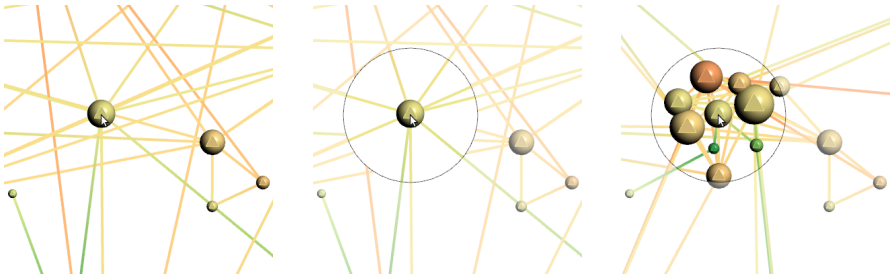


(a) Edge clutter. (b) Local-edge lens clears edge clutter.

Figure 4.4: The local-edge lens.

The *local-edge lens*, as illustrated in Figure 4.4, is designed so as to tidy up the lens area. Technically, the lens clips off all edges that pass the lens without actually connecting to a node within the lens. Such a local operation effectively clears the view, enabling the user to uncover and investigate edges within a local focus area. By dimming the visible context outside of the lens, the viewer's attention is further directed to the lens focus.

The *layout lens* supports the exploration of node neighborhoods, which are quite often not visible at a glance. To this end, the lens generates local neighborhood overviews by temporarily adjusting the graph layout based on the weighted distance between nodes and the center of the lens. Approaching a node with the lens and finally centering the lens on top of it results in the complete neighborhood of that node being visible within the lens. When the lens is deactivated, all nodes return to their original position. Figure 4.5 illustrates the effect of the layout lens in contrast to the local-edge lens. While the local-edge lens allows the user to see edges better, the layout lens further offers a better view on nodes.



(a) No lens: Connectivity remains unclear.

(b) Local-edge lens: Edges are visible, but neighbors are not.

(c) Layout lens: Neighborhood visible at a glance.

Figure 4.5: The layout lens.

RESULTS AND DISCUSSION In summary, our intermediate-level interaction techniques solve common problems occurring when navigating and exploring graphs and they do so by utilizing properties unique to graphs.

Edge-based traveling makes use of the graph structure to ease navigation between connected nodes that are far apart in the layout. The radar view augments navigation by providing a useful outlook on what lies beyond the currently visible part of a graph's layout. Both techniques are user-centric in the sense that the users change their point of view onto the graph. The proposed interactive lens techniques shift the graph elements (i.e., nodes and edges) into the focus of the interaction. By taking advantage of the connectivity information of graphs, the lenses generate locally adapted views that grant access to information that is otherwise not visible at a glance.

With these interaction techniques specifically designed for graph data, we already see a connection to the second cornerstone of this thesis, the tasks. For example, the edge-based traveling and the layout lens address tasks regarding path tracing and connectivity, which are relevant only in the context of graphs [223].

Chapter 6 will show how our techniques are put to use in a larger graph visualization system to facilitate higher-level exploratory and analytic tasks. Informal feedback of users of this system indicates that the proposed solutions are indeed helpful. Our techniques are also well received by other researchers as documented by their inclusion in a recent survey on graph visualization [361]. The same survey also states that interaction with graphs will be a topic of continued interest. This is confirmed by recent advances, such as context-aware graph navigation [135], interactive graph matching [153], or novel off-screen techniques [126].

For the next section we maintain a data-oriented view on interaction, but we make as switch from considering the data's structure to investigating the data's spatio-temporal frame of reference.

4.2.2 *Interacting with Spatio-Temporal Movement Trajectories*

A general goal of visualization is to show data in their frame of reference. Accepted methods exist for showing data in a spatial frame of reference [215] and for visualizing data according to time [21]. Spatio-temporal movement trajectories are data for which both space and time are relevant. Existing approaches already address the difficulty of showing space, time, and movements within a single interactive visual interface [220, 377, 176, 144].

A particular challenge that remains is to assist users in studying data attributes describing properties of movements (e.g., speed, acceleration, or sinuosity) in the context of where and when the movements took place. Addressing this challenge, we developed a hybrid 2D/3D representation of trajectories. Novel intermediate-level interaction techniques support spatial navigation and a novel interactive lens acts as a query tool for temporal information.

CONTRIBUTION As illustrated in Figure 4.6, the visual design of our solution is based on stacking individual 3D trajectories bands on top of each other along the third display dimension. This makes the trajectories individually distinguishable and data attributes can be color-coded along the trajectory bands. Trajectories are additionally represented as 2D trace lines rendered directly on a 2D map, which facilitates maintaining the spatial context.

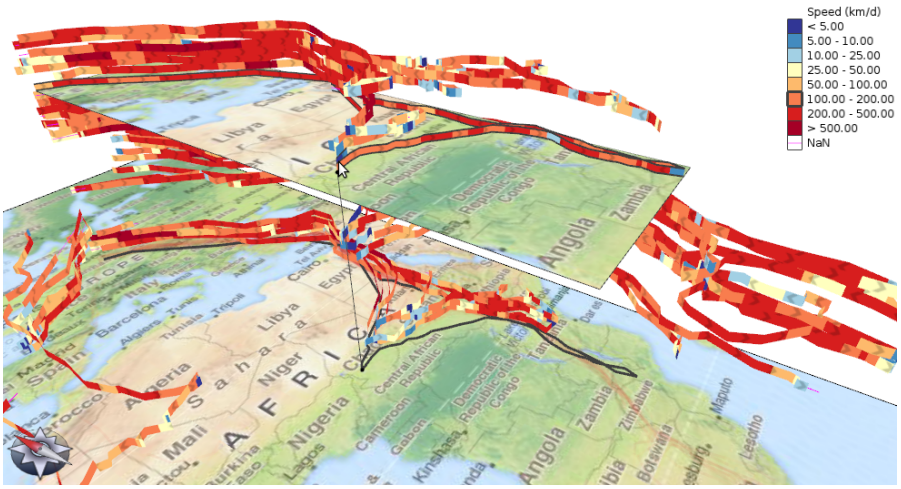


Figure 4.6: Trajectories as 3D color-coded bands stacked on top of a 2D map.

Balancing attribute visibility along 3D bands and visibility of the spatial context with respect to the 2D map as well as integrating temporal aspects are challenges to be addressed by appropriate interaction techniques. To this end, we designed dedicated context-dependent 2D/3D spatial navigation and an interactive time lens, which taken together account for the spatial and the temporal character of the data.

CONTEXT-DEPENDENT 2D/3D SPATIAL NAVIGATION We combine classic 2D drag and drop gestures with different types of 3D navigation, including orbiting, fly-through, and look-around. By user observation, we determined that an object-centric navigation of the scene made the most sense. This prompted us to promote *translate-world* and *orbiting* interactions to most simple drag operations. As a novel solution we devised the *elevator*, which takes the viewer to any level of the stack of trajectory bands by rotating the mouse wheel.

A key to easing spatial navigation is to consider the context in which it takes place: Is a particular location on the 2D map of interest, is a specific point in a 3D trajectory band of concern, or can we assume a less focused interaction intent? A simple way to determine the context is to consult the position of the cursor in the visualization scene. Depending on the identified context, automatic adjustments are made, including setting the center around which to orbit and correcting the speed with which to fly through the scene. Further, we enhance the elevator depending on the current context. When a user is investigating a 2D trace line on the 2D map, triggering the elevator is interpreted as the user's intent to visit the corresponding 3D trajectory band for closer inspection. Upon recognition of this intent, a shortcut takes the user directly to the position of the trajectory in the stack, no matter how high it may be.

The aforementioned techniques address the navigation of the spatial frame of reference. How temporal aspects can be integrated via an interactive lens will be explained next.

INTEGRATING TIME THROUGH THE TIME LENS To avoid overloading the visual display, we pick up the idea of interactive lenses [58], whose usefulness has already been demonstrated in the previous section in the context of graph data. Now we use an interactive lens, called the *time lens*, to provide on-demand access to temporal information.

As integrating the temporal aspect in full detail is not possible, we restrict it to a user-defined spatial query region. Condensing the temporal aspect further, we apply temporal aggregation according to the structure of time [21]. As illustrated in the lower right corner of Figure 4.7, the time lens is embedded into the visualization as a circular display. The center shows a scaled duplicate of the spatial query region and the outer ring shows the temporally aggregated information as color-coded histogram bins. Additional links establish a more direct connection between the spatial aspect (shown as dots in the interior) and the temporal aspects (shown as temporal scale at the ring).

Figure 4.7: The time lens represents temporally aggregated information.

By adjusting the query region and by switching between different temporal aggregations, the user can interactively control for which part of the data additional temporal information is displayed and how detailed the information will be.

The particular time lens in Figure 4.7 reveals that movements in the selected region occur only on weekdays, but not on weekends (empty bins). The highlighted trajectory represents a movement that took place later on a Friday as indicated by the links accumulating at the evening hours of the Friday bin.

RESULTS AND DISCUSSION Positive user feedback indicates that our interactive approach to exploring movement data in their spatio-temporal frame of reference is useful and usable. By combining different spatial navigation techniques and enhancing them with context-awareness, we are able to make navigation easy, even in a complex hybrid 2D/3D setting. The time lens as a dynamic interactive tool allows users to derive statements regarding the data's dependency on time, which is otherwise not possible. In summary, designing interaction techniques according to the particularities of the data's frame of reference allows the user to gain a balanced understanding of the complex interplay of space, time, and movement attributes.

Yet, for larger datasets our interaction techniques alone are not sufficient. Analytic methods are required to support the visual-interactive part. In their recent book on visual analytics of movement, Andrienko et al. [35] state that new methods of visualization need to be combined with new methods of algorithmic data analysis. The book picks up our visual-interactive solution and puts it to use in a larger visual analytics framework. Chapter 7 will provide more details demonstrating the benefits of combining analytic, visual, and interactive means for exploring and analyzing movement data.

4.2.3 *Summarizing Remarks*

In the previous sections, we elaborated on approaches taking into account the first cornerstone of interaction in visualization, the data. Looking at the data's structure and the data's frame of reference, we have seen how addressing the requirements imposed by the data (e.g., integrating space, time, and data attributes), on the one hand, and utilizing the characteristics of the data (e.g., navigating based on the structure of graphs), on the other hand, helped to design and engineer useful interaction techniques.

While focusing on the data aspect, we have also included aspects related to the other cornerstones considered in this thesis. In terms of the tasks, we addressed path-tracing and connectivity-related tasks, which are unique to graphs. Exploring movement trajectories, involves tasks such as behavior characterization, behavior search, and behavior comparison. Technology-wise the presented techniques address regular visualization workspaces. Yet the implementation of the techniques exploit modern multi-core processors via the architecture presented in Section 4.1 and GPU acceleration to provide visual feedback quickly. In terms of the cornerstone related to the human, the techniques for navigation based on a graph's structure, the context-aware 2D/3D spatial navigation, and the elevator shortcut demonstrate quite well the benefit of fluid and cost-efficient interaction.

An interesting observation that can be made is the recurring pattern of lenses as interactive tools to support the user in carrying out specific tasks in complex visualization settings [339]. In later sections of this work, we will see that lenses are indeed a kind of universal method applicable in different scenarios, for example, to assist in the task of editing graphs as described in Section 4.3.2.

4.3 TASK-SPECIFIC INTERACTION TECHNIQUES

Interaction in visualization is not only related to the data being visualized, but also to the reason that motivates the user to take actions. This brings us to the second cornerstone of interaction in visualization, the tasks.

Previous work on tasks in the context of visualization is mainly concerned with designing the visual representations in a task-dependent way [297]. However, not only the visual design is dependent on tasks, but so is the interaction design. Ware [367] declares the following:

> "The optimal navigation method depends on the exact nature of the task."
>
> — Ware [367]

There are already retrospective views that list different interaction techniques for different visualization tasks [385, 300]. These reviews, however, do not describe the design of the interaction according to a given task. What they indicate though is that there are still tasks that are under-researched from an interaction perspective. The goal of this section is to close this gap in the literature.

While Ware's above statement makes clear that interaction has to be task-specific at the intermediate level of navigation, we illustrate that designing for the task at hand is beneficial at higher levels of more complex interaction as well. For our discussion, we make a distinction between:

- interaction for *consuming* the data and
- interaction for *producing* the data.

A typical visualization task where data is consumed by the user in various ways is visual comparison. Yet, visual comparison is not specifically addressed by existing interaction techniques. To better support visual comparison tasks, we propose novel interaction designs inspired by natural comparison behavior. For the second part of this section, we study interaction for producing data via editing. In particular, we address the editing of graphs with customized layouts, a task for which practical interaction solutions are scarce. To fill this gap, we introduce a semi-automatic approach, called *EditLens*, combining interaction with automatic computation.

4.3.1 *Interaction for Comparison Tasks*

Visual comparison is among the most relevant visualization tasks. It encompasses comparing multiple data values, groups of values, and also spatial regions or temporal intervals where specific data occur [36]. Based on comparisons, users may formulate hypotheses about the data and draw corresponding conclusions, which indicates the high-level nature of the task.

A recent survey on this topic by Gleicher et al. [140] underlines the importance of comparison in visualization scenarios. Most of the solutions collected in that survey focus on supporting comparison by visual means, such as specific visual encoding or particular visual layouts. The survey also mentions interaction as an integral part, but often only rather basic interaction complements the customized visual solution. So far, no dedicated interaction techniques for comparison tasks are known in the literature. Next we introduce a specifically designed and generally applicable interaction approach for supporting visual comparison.

CONTRIBUTION Visual comparison can be considered higher-level interaction (see Section 2.2). It typically involves three phases. First,

pieces of information to be compared are identified and selected. Second, the selected pieces are arranged in a way suitable for the comparison. Third and finally, the actual comparison is carried out. These three phases must be supported by an appropriate interaction design.

Our solution is inspired by humans' natural comparison behavior. When a person compares information printed on paper, one can observe the following basic strategies:

- Side-by-side comparison: Sheets of paper are moved on a table until they are arranged side by side to facilitate comparison.
- Shine-through comparison: Sheets of paper are stacked and held against the light to let information shine through and blend.
- Folding comparison: Sheets of paper are stacked and individual sheets are folded back and forth to look at one or the other sheet in quick succession.

We design a novel interaction approach based on these natural comparison behaviors. To this end, the involved natural components and procedures have to be mimicked by virtual counterparts on the computer. In terms of the components, the workspace for the comparison is a zoomable *visualization space* based on the idea of zoomable user interfaces [53] and sheets of paper are replicated as *visualization views* that reside in this space.

In terms of the procedures, the first phase of comparison tasks is the selection of pieces to be compared. This is supported by enabling the user to mark relevant parts of a visualization view and to create new views of the marked parts. The newly created views are detached from their parent views, but their parent-child relationship is preserved in a view hierarchy. The second phase is the arrangement of the pieces to be compared. By moving views in the visualization space, the user can create any arrangement suitable for the later comparison. We integrate snapping methods to assist the user in arranging the views. The actual comparison is phase three. The assisted interactive arrangement already supports the natural side-by-side comparison. Shine-through comparison is realized via alpha-blending of stacked views. A specifically designed folding interaction enables the user to replicate the folding comparison behavior in the virtual visualization space. Figure 4.8 illustrates this with a visual representation of trajectory attribute data as discussed in the previous section.

Our interaction approach is completed with additional visual cues that indicate where views have been detached from their parent views and that explicitly visualize aggregated differences among overlapping views. Moreover, interaction shortcuts have been integrated to assist the user in covering larger distances when navigating or arranging views.

Figure 4.8: Folding for comparing color-coded trajectory attribute data.

RESULTS AND DISCUSSION With our approach, we are the first to develop a dedicated interaction design for the task of visual comparison. By analyzing this task and subdividing it into three phases, we were able to break down the complex problem into easier-to-solve smaller subproblems. Each phase is supported by corresponding interaction techniques that draw inspiration from human behavior. Given their natural origin, we hypothesized that the developed techniques are intuitive and easy to use. This was confirmed in a qualitative user study with 18 participants. Quite positive feedback suggests that our approach "feels realistic" and is even considered "better than natural comparison". More details about the user study and the obtained results are described in Chapter 8.

An important characteristic of the proposed solution is that it is engineered so as to be generally applicable for many types of data. At the same time it is customizable to the particularities of specific data. Generality is achieved by operating at the pixel stage of the visualization pipeline. Operating at the data stage allows us to provide dedicated interaction support for specific data. As illustrated in greater detail in Chapter 8, we did this for the example of data that come in tabular form (e.g., relational data or matrices). It is left for future work to further extend our solution with data-specific interaction techniques, for example with the ones for graphs or movement data described earlier in Section 4.2.

With the interaction approach for visual comparison, we have addressed a task for which the user *consumes* information in different ways. The next section will address a task during which the user acts as a *producer* of information.

4.3.2 *Interaction Support for Editing Tasks*

Visualization users U typically engage in tasks where they receive visual representations V of data D with the goal to gain insight [353], or short $D \rightarrow V \rightarrow U$. Yet, there are situations in which users have to manipulate data, for example, to insert missing items, update erroneous data values, or delete obvious outliers. Such data editing tasks should be supported through means of visualization as well [48], so that $D \leftrightarrow V \leftrightarrow U$.

However, there is a gap between visual data exploration and editing. Visual exploration focuses on interactive control of the visualization process $V \leftarrow U$, as documented several times in the previous sections. Data editing $D \leftarrow U$ is usually a manual, cumbersome, and time-consuming process [197]. In general, $D \leftarrow U$ and $V \leftarrow U$ are separate tasks that are carried out with separate tools. This not only disrupts the user's workflow, but also requires costly mental context switches, which make the whole procedure prone to human error.

To close the gap between exploration and editing, we have to consider the high-level character of editing tasks as well as the specifics of the data to be edited. To address the complexity of the task, we pursue a semi-automatic approach: Otherwise purely interactive editing is supported by automatic methods. As an example for the data to be edited, we focus on graphs with customized layouts, more specifically on the graph structure. This allows us to design our solution in a task-specific and data-specific manner.

CONTRIBUTION The visual basis of our approach is an orthogonal node-link representation. It shows the structure of the graph, textual labels for graph nodes, and optionally data attributes via color-coding. The node-link representation is embedded in a zoomable space that provides the necessary interaction techniques for exploring the graph.

To enable the user to edit the graph's structure and its layout, we developed an interactive lens, called *EditLens*. While previous work considered lenses mainly as a tool to provide alternative visual representations [339] (see also the lenses for graphs and trajectory data from Section 4.2), the EditLens is a tool to insert, update, or delete data items. In the context of graphs, the data items to be edited are nodes and edges.

Inserting a new node or edge into an already existing complex graph layout is usually a demanding task. In our particular scenario, the graph layout is custom-made and obeys certain application dependent constraints (e.g., specific types of nodes must be located in dedicated regions of the layout). Editing such graphs requires finding adequate positions for nodes to be inserted and may also involve routing edges through the already established complex layout.

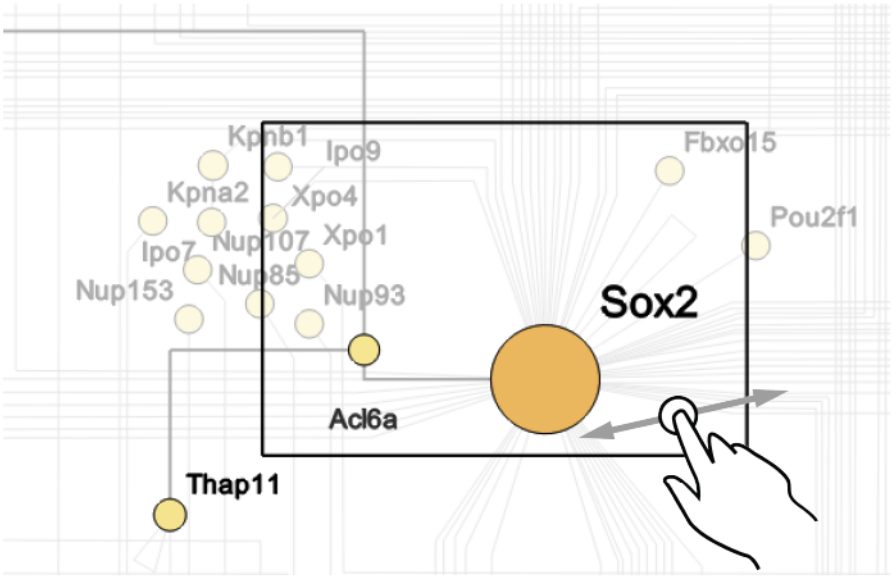


Figure 4.9: Using the EditLens for editing node "Acl6a".

Our idea is to ease editing tasks by relieving the user of defining precise points for nodes or edge routes. Instead, using the EditLens, the user specifies just the local region where an edit operation is to take effect. In other words, we relax *point-wise* manual editing to *region-wise* semi-automatic editing via the EditLens. While the interactively adjustable lens region acts as a coarse solution specified by the user, precise positions and edge routes are computed by automatic methods as described next.

First, we determine a suitable unoccupied area within the EditLens where an edited item can be placed. The precise spot within that area is computed based on different heuristics (see Chapter 9 for details). These heuristics prioritize different graph aesthetics criteria, for instance, maximum distance to other nodes, short overall edge length, or low number of edge bends. During the editing, the user can freely choose which heuristic to apply depending on the situation at hand. The EditLens will suggest suitable positions of nodes and edge routes accordingly. As illustrated in Figure 4.9, when the EditLens is in operation, only the nodes and edges being affected by the ongoing edit operation are shown fully saturated, whereas all other nodes and edges are dimmed. When the user agrees with a suggested solution, the result of the edit operation is committed to the data. In cases where the EditLens can find only insufficient or no solutions due to conflicting constraints, manual refinement is still possible with classic point-wise editing.

RESULTS AND DISCUSSION The EditLens has been applied to a real-world problem where bio-informatics scientists maintain a network of genes and biological relations among them. As new genes or

relations are discovered and reported in the literature, the network is edited to include the newly available information. Yet, as further described in Chapter 9, the scientists' manual editing workflow is cumbersome and time-consuming.

In a small qualitative study with two experts and two non-expert users, we compared the standard editing procedure with editing when using the EditLens. The collected feedback was generally positive and in favor of the EditLens. The participants expressed this in statements such as "the EditLens is very useful and can clearly reduce the editing effort" and "the automatic suggestion of node positions and edge routes is obviously beneficial."

With the EditLens, we significantly narrow the gap between data editing and data exploration in the sense of $D \leftrightarrow V \leftrightarrow U$. Our concrete implementation focuses on the structure of graphs with orthogonal layouts. This leaves two key aspects for future research. First, the EditLens could be extended to support the editing not only of the graph structure, but also of data values associated with nodes and edges. Second, editing based on alternative visual representations other than an orthogonal node-link diagram (e.g., bundled edges or matrix representation) could be investigated. Initial studies in this direction have already been carried out in the context of a master's thesis [362].

While the editing of node-link diagrams is a research challenge in its own right [125], broadening the scope of future investigations bears the potential of extending the EditLens to a general editing tool, not only for graphs, but for other types of data as well.

4.3.3 *Summarizing Remarks*

In this section, we have shown how interaction can be designed taking into account the second cornerstone of interaction in visualization, the tasks. We considered high-level interaction to support two fundamentally different types of tasks: tasks that involve the consumption of data and tasks that are related to the production of data. As concrete task instances we studied visual comparison and data editing. Paying attention to the procedures and workflows behind these tasks, we were able to design and implement interaction approaches that actually support users in carrying out their work. Positive user feedback testifies to our solutions' utility.

From the two approaches described, we can also see connections to the other cornerstones. Our EditLens has been designed specifically for graphs with customized layouts, which confirms the importance of the data as a cornerstone. While our solution for visual comparison is generally applicable in terms of the data to be compared, we yet have indicated that specific customization to particular data characteristics can be sensible and useful. An obvious link to the cornerstone

related to the human user is the interaction design based on natural comparison behavior. By closely following humans' natural comparison strategies, we obtained a solution that is intuitive and effective. Also the EditLens adheres to a user-centered design as it has been developed in accordance with the editing workflow and the needs of real users working on a real-world problem.

Speaking in terms of technology, we began to make a shift from classic mouse and keyboard interaction for the comparison approach to multi-touch interaction for the EditLens. In the latter case, touch interaction was positively received by the users, as they can directly manipulate the data under their fingertips. Thanks to the region-oriented EditLens, typical precision problems with touch interaction do not even surface in our solution, as precision is taken care of by automatic methods. This already hints at the importance of considering technological aspects when designing interaction solutions. In the next section, we further explore the utilization of modern interaction technology in the context of visualization.

4.4 UTILIZING MODERN TECHNOLOGY FOR INTERACTION

With data-specific and task-centric interaction as dealt with in the two previous sections, we have covered the core of interaction in visualization. As illustrated in Figure 3.1 back on page 25, this core is flanked by the technology on the one side and the human on the other side. Both of these flanks will be addressed in the following, the technology in this section and the human in the next section.

Undeniably, the technology plays a key role in interactive visualization, because it provides the facilities to display visual representations as well as the means to interact with them. Typically, the technical setup for visualization applications is a regular desktop computer to which a regular display as well as a mouse and a keyboard are connected. In this section, we go beyond mouse and keyboard interaction and regular displays, as called for in recent research agendas in the context of interaction technology and visualization [335, 224, 183].

The previous sections already indicated the immense size of the design space for interaction techniques in visualization scenarios. New display devices and modern interaction modalities add further possibilities with their own individual strengths and weaknesses. It is beyond the scope of this section to comprehensively discuss all possible options. Our considerations are focused on the two principal ways of establishing an interaction channel between the user and the computer:

- tracked objects – users manipulate objects that are tracked and
- tracked humans – users themselves are tracked.

While mouse and keyboard are the classic input devices, our goal is to illustrate alternative means of using tracked objects for interaction. To this end, we discuss *tangible views* as a novel way of interacting with visual representations of data. At the same time, tangible views are a novel form of lightweight displays that offer new possibilities for visualization applications. As an example where users are tracked, rather than objects, we look at interaction in front of a large high-resolution display wall. In particular, we investigate how tracking the user's position and viewing direction can support graph exploration on such large displays.

4.4.1 *Tangible Views for Interaction and Visualization*

Direct manipulation as advocated by Shneiderman [303] is a most prevalent theme of interaction in visualization. Many visualization approaches are designed so as to allow the user to directly manipulate the visual representation or the underlying data.

Yet, often the term *direct* just means that manipulations affect the visual object directly under the pointer, which is a quite limited interpretation of directness. In fact, standard interaction is rather indirect: the pointer is typically controlled with a mouse, whereas visual feedback is shown on the display. Therefore, researchers have started to explore how truly direct interaction with touch-enabled devices can effectively support visualization approaches [182]. Here, we expand upon basic touch-interaction *on* a display and propose a novel method for interaction *with* the display. By doing so, we extend the vocabulary for interaction with visual representations of data.

CONTRIBUTION The starting point for our approach is a horizontal touch-enabled tabletop device. The tabletop serves to visualize the data and to receive touch input from the user. In this setup, visualization and interaction remain in the horizontal 2D tabletop plane. Our idea is to utilize the 3D space above the tabletop to provide enhanced visualization and interaction functionality.

We extend the concept of the *PaperLens* [312] to what is called *tangible views*. Tangible views are lightweight "devices" that act as additional displays in the 3D space above the tabletop. In a most simple instantiation, a tangible view can be a piece of cardboard onto which a projector transmits visualization content, in which case the display is passive. A tangible view can also be active, that is, it is capable of displaying graphical content on its own without the help of an external projector, for example, a tablet device.

Irrespective of being passive or active, the key characteristic of tangible views is that they are spatially aware. Through constant tracking of the tangible views, the system is always aware of their position and orientation in 3D space. This opens up new possibilities for in-

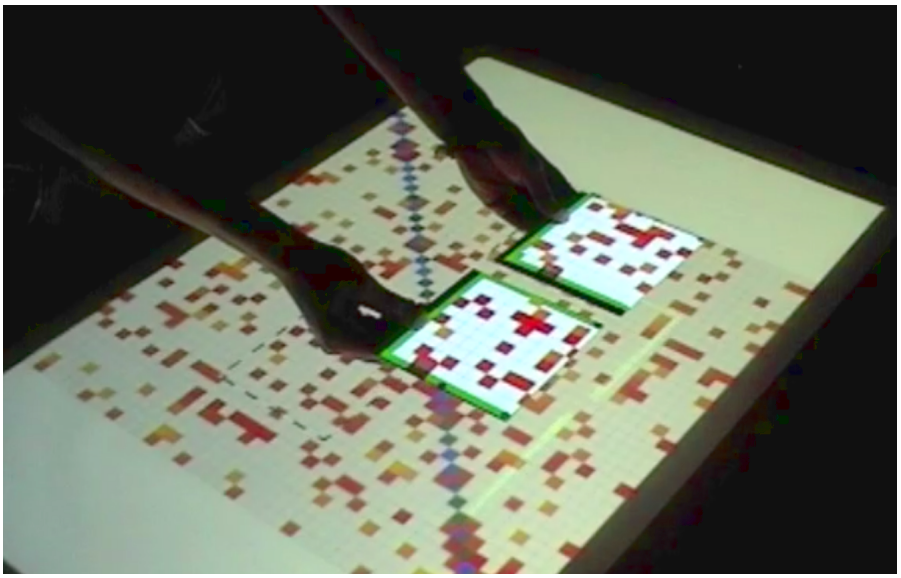


Figure 4.10: Tangible views applied for visual comparison of a matrix.

teraction. The interaction vocabulary of tangible views includes basic translation and rotation in 3D as well as gestures of flipping, tilting, and shaking. By providing tangible views that are distinguishable by shape or appearance it is possible to create an interaction toolbox, where users can infer interaction functionality from the look of a tangible view. Multiple tangible views can be applied simultaneously, each adds further display space for visualization purposes.

Developing an extended interaction vocabulary is only one part of the contribution. The second part is to appropriately map the vocabulary to typical interactions in visualization scenarios. Here, we take a brief look at an application of tangible views for visual comparison of graph data. We chose this example as it is interesting to see the novel way of tangible interaction applied to visual comparison in contrast to the visual comparison approach based on classic mouse interaction as introduced earlier in Section 4.3.1.

How visual comparison can be carried out using two tangible views is depicted in Figure 4.10. The tabletop shows as the background visualization a color-coded graph matrix. In order to select two sub-matrices to be compared, the user moves two tangible views horizontally above the tabletop. A freeze gesture is carried out to fix the selection. This allows the user to arrange both tangible views side by side for closer inspection and comparison. Once the views are sufficiently close to each other, the system recognizes the user's comparison intention and indicates by a colored frame (green in our case) the overall aggregated similarity of the sub-matrices. Shaking the tangible view releases the freeze and the user can select sub-matrices anew.

RESULTS AND DISCUSSION As illustrated by the visual comparison application, tangible views offer a novel and intuitive way of interacting with visual representations of data. This is true not only for comparison tasks and graph data, but for a broad range of visualization problems. In Chapter 10, we apply tangible views in a number of case studies including sampling in parallel coordinates, fisheye magnification in scatter plots, exploration of hierarchical graphs, and spatio-temporal data analysis with space-time cube. These case studies further explore and demonstrate the usefulness of the introduced extended interaction vocabulary.

From a conceptual point of view, with designing tangible views, we obtained three key results related to interaction and visualization. First, tangible views integrate display and interaction device. The user interacts directly *with* the tangible view to satisfy an interaction intent. Interaction *on* tangible views or the tabletop display can provide additional functionality. Second, tangible views enhance common 2D interaction with tangible 3D interaction above a tabletop display, thus extending the typical interaction vocabulary for visualization. Third, the enhanced interaction vocabulary and extended physical display space allow us to create a tangible experience of otherwise purely virtual visualization concepts, including overview+detail, focus+context, and coordinated multiple views as well as various visualization techniques for different types of data and different tasks.

Still, the approach of tracking tangible views in 3D also raises questions regarding the precision of interaction and fatigue of users operating multiple tangible views. Investigating these issues requires extensive user studies and is ongoing research. First results in this direction have been published by Spindler et al. [314]. They indicate that tangible views are indeed a promising alternative when interacting in and with layered information spaces, which are common in visualization scenarios.

By the example of tangible views, we illustrated the usefulness of novel interaction technology where users manipulate tracked objects. Next we describe how tracking the user, rather than objects, can support visualization on large high-resolution displays.

4.4.2 *Interaction for Visualization on Large High-Resolution Displays*

In the previous section, we used tangible views and a tabletop device to display visual representations of data. These and other display devices with conventional pixel resolution are typically limited in the amount of information that can be shown. In the era of big data, new solutions are needed to support the visualization on large high-resolution displays.

Thanks to technological advances, high-resolution displays are now becoming available to a broader range of users. The increased physi-

cal size and pixel resolution offered by such displays has obvious advantages for visualization applications, because much more information can be presented. Yet with increased size and number of pixels, there are also new challenges to be addressed in terms of visualization and interaction. Andrews et al. [28] point out that:

> "Replacing the conventional monitor with a large high-resolution display creates a fundamentally different environment that is no longer defined purely in terms of the technical limitations of the display, creating a new collection of design opportunities, issues, and challenges."
>
> — Andrews et al. [28]

In our work, we rely on standard visualization techniques, allowing us to fully concentrate on novel ways of interacting with information presented on a large high-resolution display. More concretely, we describe how the user's physical navigation in front of a display wall can be utilized to support the exploration of hierarchical graphs being displayed at different scales.

CONTRIBUTION We visualize graphs as standard node-link representations on a large high-resolution display wall. As shown in Figure 4.11, the wall consists of 24 individual displays covering an area of 3.7 m × 2.0 m with a total resolution of 11,520 × 4,800 which amounts to 55 million pixels. The graph we visualize is a hierarchical graph with different levels of abstraction. In standard visualization environments, individual levels of abstraction are typically accessed

Figure 4.11: Exploring a graph on a large high-resolution display wall.

via the mouse (e.g., by expanding or collapsing nodes). In our scenario, this is obviously impractical due to the large distances that would need to be covered with the mouse across multiple displays.

Therefore, we developed a solution where the exploration of the graph is steered by the user's physical movement in front of the large display. Head tracking is used to acquire information about the user's position and orientation (6 degrees of freedom). This information is utilized in two alternative approaches: the *zone technique* and the *lens technique*.

For the zone technique, the space in front of the display wall is subdivided into multiple zones with increasing distance to the display. Each zone corresponds to a level of abstraction of the graph. When the user moves toward the display, the graph is visualized at greater detail. This approach corresponds to natural human behavior. When interested in details, humans typically step up to the object of interest to study it in detail. In order to obtain an overview, the user can step back. Stepping backward into zones farther away from the display automatically adjusts the visualization to show higher levels of abstraction.

While the zone technique can be used to globally steer the level of abstraction, the lens technique has been designed to enable the user to access details for local parts of the graph. To this end, we utilize the tracking information about the orientation of the user's head to estimate where the user is looking. Using this estimation we position an interactive lens on top of the regular graph visualization. Nodes that are inside the lens are automatically expanded to reveal more detailed information. The lens technique enables the user to scan the graph in a focus+context fashion by simply moving the head around. Filtering the tracking input and smoothly animating node expand and collapse operations help to maintain a reasonably stable visualization and to avoid flickering caused by natural head tremor.

In addition to controlling the level of abstraction, we also evaluate the user's distance from the display to derive a suitable labeling of the graph items on the display. A user standing close to the display is presented with more and smaller labels. When looking at the display from a greater distance, the user will see fewer, but larger labels.

RESULTS AND DISCUSSION Preliminary user feedback has been collected in a pilot study. Eight participants explored a graph using the zone technique and the lens technique. Ease-of-use of the interaction and readability of the visualization was confirmed by all participants. The zone technique was reported as the approach that is easier to use, but on the other hand, the lens technique offered more control over where increased detail is to be shown. Suggestions for improvements included easier calibration and further stabilization of the lens technique.

Overall, we can conclude that physical movement in front of a large display can be a promising alternative in cases where classic means of interaction are impossible to apply. The large size of the display simply renders incremental mouse interaction infeasible. In contrast, physical movement better matches the scale of the display. Moreover, physical movement corresponds with natural interaction with real-world objects and hence it is intuitive and easy to carry out.

With our work, as further detailed in Chapter 11, we considered the relatively simple interaction task of adjusting the level of abstraction of a graph visualization. Yet, recent research results indicate that utilizing modern display and interaction technology can be beneficial also for higher-level sensemaking in large high-resolution display workspaces [113, 27].

4.4.3 *Summarizing Remarks*

As we have seen, addressing technological aspects typically involves studying interaction at a low level of basic ways of transmitting interaction intents from the human user to the computer. We presented two novel approaches to interaction in visualization, one based on tracking tangible views above a tabletop display, the other based on tracking the user in front of a large display wall. Using the low-level tracking information, we derived several techniques for intermediate-level interaction with visual representations of data.

With tangible views we significantly extend the interaction vocabulary for visualization applications and at the same time tangible views offer more space for displaying information. With physical navigation we match the scale of the interaction and the scale of the display in order to enable the exploration of mega-pixel visual representations of graphs at different levels of abstraction. Both approaches indicate that there is much potential in adopting modern technologies for visualization applications. With this thinking we are in line with recent efforts of the visualization community to go beyond regular displays and standard mouse and keyboard for visualization and interaction [28, 224, 183, 190].

Even when focusing our attention on the technological aspects of interaction, we see that the cornerstones of the data and the tasks are of high relevance. For the tangible views approach (see Chapter 10 for details), this is reflected by a number of case studies that take advantage of our extended interaction vocabulary for supporting different interaction tasks (e.g., parameter adjustment, navigation, comparison) on different types of data (e.g., multivariate data, spatio-temporal data). The case studies also include the exploration of hierarchical graphs, the very same data and task that we also addressed when we studied physical navigation in front of a large display wall. The CGV system described in Chapter 6 also supports the exploration

of hierarchical graphs, but uses classic mouse interaction for this purpose. An interesting question for future research would be to compare graph exploration using tangible views, physical navigation, and classic mouse interaction. Investigating this question will help us develop an understanding of which technologies are best suited to support the human user. In the next section, we take a closer look at approaches focusing on the human in the visualization and interaction loop.

4.5 THE HUMAN USER AND THE GULFS OF INTERACTION

The final cornerstone of interaction in visualization as considered in this work is the human. The human user is the source of interaction intents and at the same time the sink for visual information. As discussed in the previous section, interaction for different tasks and visualization of different data are mediated by different interaction and display technologies.

Developing interaction with a focus on the human user involves addressing aspects of fluidity, naturalness, and cost efficiency of the interaction. All these aspects have already been covered implicitly in the previous Sections 4.2–4.4. The easy navigation based on graph structures, the context-aware 3D navigation of trajectory data, the interaction for visual comparison inspired by natural behavior, the data editing approach that follows real-world users' workflows, the approach that makes interaction for visualization tangible, and the natural physical navigation in front of a large display have all been designed with the human user in mind, implicitly in one regard or the other. The multi-threading architecture from Section 4.1 provides a technical basis for fluid and smooth interaction.

In this section, we explicitly address the human user and we focus on reducing the costs involved when users interact. Lam [221] attributes interaction costs in visualization to the two gulfs in Norman's [262] interaction-feedback loop:

- gulf of execution – carrying out the interaction and
- gulf of evaluation – understanding the visual feedback.

The following paragraphs present approaches that aim to narrow these gulfs by considering analytic methods to assist in interactive visualization. In terms of the gulf of execution, we propose to use event-based concepts to automate certain actions that otherwise would need to be performed manually. In terms of the gulf of evaluation, we address typical questions arising when users navigate larger information spaces: "Where can I go now?" and "Where should I most usefully go?". We develop a degree-of-interest (DOI) approach that presents the user with navigation recommendations that help to make an informed decision on where to navigate next.

4.5.1 *Reducing Interaction with Automatic Event-Based Concepts*

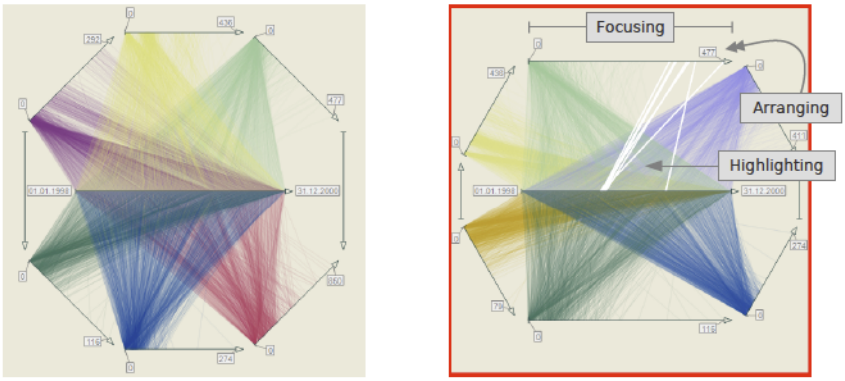
Interaction in visualization has been advocated throughout this work as a means to empower the human user to control the visualization as needed for the data and the tasks at hand. Yet in Section 2.4, we also indicated that interaction is not a universal cure and that interaction can be a burden to the human user. So it makes sense to critically question all interactive input that visualization tools solicit from the user [356]. After all, it is the task of the visualization to present relevant information effectively and expressively, and not primarily the task of the human user to parameterize the visualization appropriately to achieve this.

Thinking about reducing user interaction to a reasonable and useful minimum implies that we also have to look for alternative sources of input to be able to derive visual representation that still reflect the user's needs. One such source of information is the data itself and patterns of interest residing in the data. Knowing that certain parts of the data are of special interest to the user, we can automatically trigger adjustments of the visual representation to emphasize these interesting parts without the need of user interaction.

CONTRIBUTION In order to automate the adjustment of visual representations according to user interests, we developed a novel event-based approach to visualization. The approach comprises three stages: event specification, event detection, and event representation.

The event specification is concerned with defining event types that formalize the notion of "interesting parts of the data". We use predicate logic formulas to express three different kinds of event types. Addressing the relational data model, the user can specify tuple event types (e.g., tuple values exceed a certain threshold) and attribute event types (e.g., data attribute with the highest value). In order to address changes over time, we also support sequence event types (e.g., sequence of days with rising temperature). Composite event types can be compiled using set operators.

The second stage is the event detection. At this stage, the data under investigation are searched for matches of the interest expressed via event types. Different algorithms are involved at this stage, including query mechanisms for relational databases as well as search algorithms for sequence patterns. Efficiency of these algorithms is of importance, because the search has to be carried out upon every change of the data. This is particularly critical for dynamic data that undergo frequent changes. Once the event detection reports matches in the data, actual event instances are created. Event instances bear three important pieces of information: (1) the fact that something interesting has been found in the data, (2) where in the data interesting



(a) Default parameterization. (b) Improved parameterization.

Figure 4.12: Automatic event-based adjustment of the TimeWheel.

parts are located, and (3) what made these parts interesting (i.e., the event type). This information is the input to the event representation.

The goal for the event representation is to automatically adjust the visual representation of the data so as to (1) communicate the fact that something interesting is in the data, (2) emphasize interesting parts among the rest of the data, and (3) indicate the event type, where possible. These goals are to be achieved by re-parameterizing the visualization. Therefore, it is a necessary requirement that the visualization provides an appropriate set of visualization parameters. If this requirement is satisfied, our event-based approach automatically triggers the execution of instantaneous or gradual parameter changes upon the detection of events. These automatic parameter adjustments reduce the need of manual interaction.

RESULTS AND DISCUSSION Our approach has been implemented in a general system for event-based visualization. Chapter 12 describes the architecture of this system in greater detail. Several visualization techniques, including the Table Lens [280], the space-time cube [213], and the TimeWheel [329], have been enhanced with event-based automatic parameter adjustments.

Figure 4.12 illustrates an example using the TimeWheel technique. For this example, we assume that the user is interested in high numbers of cases of influenza in the depicted human health dataset. A default parameterization is generally unaware of this special interest and thus manual parameter tuning is required. With the event-based approach, on the other hand, we can make this interest known to the system and trigger automatic parameter adjustments once corresponding events (e.g., cases of influenza exceed a certain threshold) are found in the data. In our concrete example, the tuples that correspond with the user's interest are highlighted and the TimeWheel is rotated to create an arrangement where the axis representing influenza is on the top. Further magnification of the axis helps the user

to focus on the data of interest. A red frame signals the fact that events have been detected. The improved parameterization is the result of an automatic reaction to an event and as such does not require any user interaction. As a consequence, the gulf of execution is reduced.

It is important to understand that the gulf is reduced, not overcome. Still user input is needed: the description of interests in the form of event types. But this can be done in a one-time pre-process. Chapter 12 elaborates on three different levels of support for the event specification: direct specification of event types, parameterization of event type templates, and selection from an event type collection. In addition to describing event types, there is the need to define appropriate automatic parameter adjustments, which is typically a task for the visualization designer. Although this is a one-time pre-process as well, authoring parameter adjustments for various visualization techniques and for different event types poses a significant challenge calling for further investigation in future work. An interesting question to be studied would be to derive rules or guidelines on what visualization parameters are needed and how to adjust them in order to obtain visual effects that meet the requirements of the event representation.

We have seen that event-based concepts can be a useful complement to interaction in visualization. Automatic reaction to events of interest can reduce the gulf of execution. An approach to narrow the gulf of evaluation will be presented in the next section.

4.5.2 *Navigation Recommendations for Informed Interaction*

The gulf of evaluation relates to the costs arising when interpreting the visual feedback resulting from interacting with the visualization. Particularly during data exploration evaluation costs might accumulate, because exploring the data generally means carrying out a number of interactive navigation steps [131], which all ensue evaluation costs. Consider, for example, the techniques for navigating along the structure of graphs as presented in Section 4.2.1. After taking a navigation step the user has to evaluate the updated visual representation of the graph structure. Typical questions that users might ask themselves include: “Where am I in the structure?”, “What structural patterns can I see here?” or “How is what I see related to what I’ve seen before?”.

Spence [310] further lists the questions: “Where can I go?”, “What lies beyond?” and “Where can I usefully go?”. Our goal is to support the user in answering these questions. To this end, we compute and present *recommendations* to the user. Recommendations have already proved useful for supporting users in selecting visualization approaches [141]. We propose recommendations to reduce evaluation costs during interactive navigation in hierarchical graphs.

There are two types of navigation for hierarchical graphs. *Horizontal navigation* relates to navigation to different locations in the graph layout. Typical operations that support horizontal navigation are zooming and panning to adjust the view on the graph layout. *Vertical navigation* denotes navigation between different levels of abstraction of hierarchical graphs. As such, vertical navigation changes the degree of detail shown in the graph layout. Different levels of abstraction can be accessed level-wise, as for example for the zone-technique from Chapter 11 or by expanding or collapsing individual nodes, as realized in the CGV system described in Chapter 6.

CONTRIBUTION We propose a novel approach that supports horizontal and vertical navigation by recommending nodes that are worth visiting next. The general procedure is as follows. First, we determine a set of nodes as recommendation candidates. Second, the candidates are ranked and the top-ranked nodes are selected as actual recommendations. Finally, the selected recommendations are communicated visually to the user.

For building the set of recommendation candidates, we consider the "neighborhood" of the current exploration situation, that is, the context of the visualization content currently visible on the display. As illustrated in Figure 4.13, the neighborhood can be defined in different terms: structural neighborhood, spatial neighborhood, and data neighborhood. The structural neighborhood is based on the k-neighborhood of the graph. The spatial neighborhood is defined in terms of node positions in the graph layout. A data neighborhood can be determined based on similarity among the attribute values associated with the graph nodes. Restricting the recommendation candidates to a local neighborhood has two advantages. First, it is guaranteed that the candidates are related to the part of the graph currently visible. Second, the neighborhood is much smaller than the dataset as a whole, which eases the ranking of the candidates.

To determine where the user can *usefully* go, we need a definition of what *useful* means. An effective concept in this regard is the degree of interest (DOI) [350]. We use the common API (a-priori interest), UI (user interest), and DIST (distance to focus) components to define the degree of interest. As a fourth component, we add the new KNOW

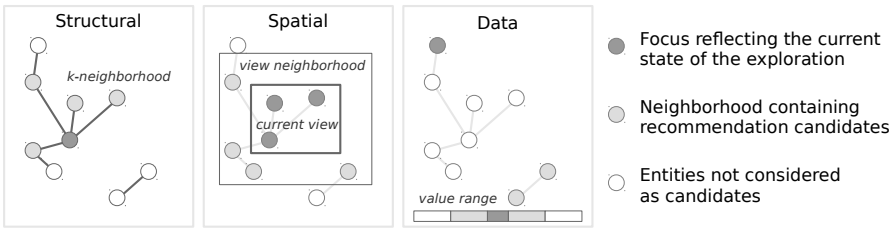


Figure 4.13: Neighborhoods for collecting recommendation candidates.

component, which models interest degradation for nodes that have already been visited. All components can be weighted to fine-tune the DOI computation. With an appropriate DOI specification, we can compute and assign DOI values to all recommendation candidates and rank them accordingly. The top-ranked candidates are selected as actual navigation recommendations to be presented to the user.

Navigation recommendations are additional pieces of information that need to be conveyed to the user, on top of the communication of the actual data. We designed visual cues that aim to subtly indicate the recommended target nodes. For horizontal navigation, target nodes can be on-screen or off-screen. On-screen targets are marked with rings, whereas off-screen destinations are hinted at by visual cues called *enriched wedges*. Both rings and enriched wedges can further visualize the DOI values assigned to the recommended nodes. For vertical navigation, targets are naturally at levels of abstraction different from the current level, and hence they are definitely not visible. Therefore, recommendations for vertical navigation are attached to anchor nodes whose expansion (or collapse) would bring the recommended target to the display. Subtly pulsing rings around anchors suggest that an expand (outward pulsing) or a collapse (inward pulsing) operation will uncover a target of interest.

RESULTS AND DISCUSSION The navigation recommendations assist the user in making informed navigation decisions. Thanks to the DOI concept, we are able to recommend targets that reflect the user's interest, provided that the DOI components are appropriately defined and weighted. The design of the visual cues that hint at the recommended targets follows a defensive strategy in order to only minimally interfere with regular data exploration. Only when users have difficulties in determining a good next navigation target on their own should their attention shift to the navigation recommendations.

A proof-of-concept implementation of our approach has been incorporated into the graph visualization system CGV (see Chapters 6 and 13 for more details). The implementation has been tested with different hierarchical graphs of moderate size (hundreds of nodes and thousands of edges). The tests indicate that the approach in general is also technically feasible. Navigation recommendations could be extracted from the graphs on-the-fly, while the computations were not hindering any regular exploration activities of the user.

Still a difficulty is to appropriately outfit the DOI function. The components API, UI, and DIST can be defined in accordance with existing work [350]. However, our initial definition of the KNOW component to capture what parts of a graph have already been explored is rather rudimentary. A promising alternative could be to use gaze-controlled approaches, for example, based on the work recently published by Okoe et al. [266].

4.5.3 *Summarizing Remarks*

In this section, we addressed the human as the fourth cornerstone of interaction in visualization. With event-based visualization and navigation recommendations, we presented two approaches to reduce the gulfs of execution and evaluation of Norman's [262] interaction-feedback loop. Interestingly, both approaches are, in a sense, in contrast to our previous focus on interactive solutions, because at their core they are analytic methods: finding interesting events in the data and determining interesting navigation targets. What this contrast illustrates quite well is that it definitely makes sense to critically question interaction and investigate alternative methods to improve visualization. So, the proposed solutions close the circle to the discussion on useful and harmful interaction from Section 2.4.

Of course complementing interaction with our solutions raises questions related to balancing automatic and interactive methods. Such questions have to be answered individually depending on the application scenario and user expertise. For example, Cooper et al. [87] state that casual users need basic functionality, that experienced users tend to explore enhanced functionality, and that expert users seek ways to automate tasks. Although these statements provide some indication, it is left for future work and longitudinal studies to more thoroughly investigate guidelines on balancing interaction and automatisms.

A second concern, not only relevant to our solutions, is the need to appropriately estimate user interest. Both approaches presented here depend on the existence of suitable definitions of user interest: event types and DOI specification. Yet in real-world applications, these definitions might turn out to be difficult to set up. Therefore, it makes sense to continue researching alternative means for estimating user interest, where gaze-based approaches appear particularly promising.

Overall, we can conclude that addressing the human is vitally important for interaction in visualization. Here we focused on the human as the user of interactive visualization tools. On the other hand, we mentioned that the human can also be in the role of the visualization engineer. With the multi-threading architecture from Section 4.1, we already provide support for the human engineer. Our event-based approach to visualization adds to this support, because it provides a conceptual model according to which visualization engineers can enhance other visualization techniques with automatic parameter adjustments.

For the future, we see an increased relevance of complementing interactive visualization with assistive methods. Under the umbrella of guidance in visualization, ongoing research investigates how the human can be supported at different levels, including guidance to visualize data effectively, guidance to assign tasks to the right user, and guidance to employ suitable technologies [298].

4.6 SUMMARY

In this chapter, we presented 1+8 novel approaches related to interaction in visualization. We started out with a multi-threading architecture laying out a technical foundation for interactive visualization. Then the four cornerstones of interaction, the data, the tasks, the technology, and the human, were addressed with two novel approaches each. We presented techniques for intermediate-level interaction with graph data and movement data. Higher-level interaction support was discussed in relation to visual comparison and data editing tasks. Addressing modern technology, we investigated low-level tracking of tangible views above a tabletop and tracking of users in front of large displays. Finally, we proposed two approaches to reduce the gulfs of interaction for the benefit of the human user.

All approaches were described in a compact fashion focusing on key aspects and on establishing connections among the cornerstones of interaction. The following Part ii of this work contains the original publications that provide the details behind the approaches discussed briefly in this chapter. An overall conclusion and discussion of the topics covered in this thesis along with an outlook on future work in the context of interaction in visualization will be given in Part iii, which starts on page 273.

Part II

THE CORE

This part collects the original publications underlying this work. The following chapters complement the compact descriptions from Sections 4.1–4.5 by providing additional details and discussions. Each chapter will start with a brief summary of the contribution, the original abstract, and the reference to the original publication. The chapters' contents are identical to the published articles, except for layout of figures, typesetting, correction of typographical errors, and harmonization of the citations.

5

A MULTI-THREADING ARCHITECTURE TO SUPPORT INTERACTIVE VISUAL EXPLORATION

CONTRIBUTION This chapter makes a contribution to support the engineering of interaction in visualization. A multi-threading architecture for visual exploration is developed. The goal is to exploit the capabilities of modern multi-core CPUs to provide rich visual feedback in a timely manner. As such the architecture addresses the conflict of synchrony and asynchrony, which is clearly a low-level concern with impact on all higher levels of interaction.

ABSTRACT During continuous user interaction, it is hard to provide rich visual feedback at interactive rates for datasets containing millions of entries. The contribution of this paper is a generic architecture that ensures responsiveness of the application even when dealing with large data and that is applicable to most types of information visualizations. Our architecture builds on the separation of the main application thread and the visualization thread, which can be cancelled early due to user interaction. In combination with a layer mechanism, our architecture facilitates generating previews incrementally to provide rich visual feedback quickly. To help avoiding common pitfalls of multi-threading, we discuss synchronization and communication in detail. We explicitly denote design choices to control trade-offs. A quantitative evaluation based on the system VISPLORE shows fast visual feedback during continuous interaction even for millions of entries. We describe instantiations of our architecture in additional tools.

ORIGINAL PUBLICATION [272] — H. Piringer, C. Tominski, P. Muigg, and W. Berger. A Multi-Threading Architecture to Support Interactive Visual Exploration. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1113–1120, 2009.

5.1 INTRODUCTION

Exploration of unknown data is an important task in the context of information visualization. Explorative tasks are different from presentation tasks in that they require frequent changes of the view on the data. This includes both, navigation between different data subsets and adjustment of parameters that control the visual mapping. Multiple coordinated views [363], dynamic queries [304], and direct manipulation [303] are key concepts to support visual exploration.

For smooth and efficient exploration, the ensemble of analytical, visual, and interaction methods has to generate results in a timely manner (within 50 – 100 ms [304, 310]). However, even moderately sized data can pose computational challenges. Computing a graph layout of a few hundered nodes or rendering a data set with a few thousand data records as a parallel coordinates plot may take a few seconds on a desktop computer. For *discrete interaction* (e.g., a single click on a button) delays or temporary loss of responsiveness might be acceptable, because interaction occurs at low frequency.

However, research in human-computer-interaction has long been emphasizing the significance of *continuous interaction* as a requirement of interactive systems to support native human behavior [116]. This is in particular true for information visualization, because examining multiple 'what if' scenarios is a key aspect of exploratory data analysis [310]. A scenario could, for example, refer to setting a model parameter to a certain value. For discrete interaction, the user has to explicitly specify scenarios of interest in a successive manner. This approach provides no information about properties between two scenarios and it requires much time to explore parameter ranges. Continuous interaction, on the other hand, allows the user to explore any range in any speed and reduces the risk of losing interesting scenarios. During continuous interaction, two important requirements are to keep the application responsive and to provide a sufficient amount of visual feedback. What 'sufficient visual feedback' refers to depends on the visualization and the purpose, but definitely involves showing a representation of the data.

Many approaches provide a fixed amount of feedback during a continuous user interaction. However, as the available computation time per update can hardly be predicted generically and may vary due to caching and scheduling effects, such approaches suffer from one of two drawbacks: 1) time is left unused and less visual feedback is provided than possible or 2) single updates take longer than the time between consecutive user events. In the second case, the application responsiveness may degrade severely if visualization generation happens in the same thread that is responsible for receiving events.

Therefore, some systems (e.g., IMPROVISE [370]) parallelize these tasks using multi-threading. Although multi-threading makes use

of commonplace multi-core technology and is thus desirable, implementing multi-threaded programs is difficult [226, 253] and involves many potential pitfalls which have not sufficiently been addressed in the context of interactive visualization so far. Moreover, multi-threading by itself neither guarantees responsiveness due to potential blocks caused by thread synchronization, nor does it ensure rich visual feedback at interactive rates.

As contribution of this paper, we propose a generic multi-threaded visualization architecture that should help to avoid pitfalls related to multi-threading. It has been designed to meet the following goals:

- Guarantee responsiveness to the user at all times, i.e., avoid perceivable delays of the GUI
- Provide visual feedback as quickly as possible, i.e., keep the latency between interaction and visual feedback below 100 ms [304]
- Provide as much visual feedback as possible
- Scale to data sets with several millions of data items
- Scale with regard to multiple views
- Support most common types of visualizations
- Be applicable regardless of environment or language

Where goals are conflicting, we explicitly outline and discuss particular design choices. This architecture has been shaped based on experiences in implementing several visualization systems and tools, including SimVis (C++) [99], Visplore (C++) [271], CGV (Java) [332], and VisAxes (C#) [329].

In the next section, we take a look at related work. Section 5.3 describes our architecture, including details related to multi-threading. We present a quantitative evaluation based on the system Visplore in Section 5.4. We close with a discussion about design choices, further instantiations of our architecture, and ideas for future work in Section 5.5, and a conclusion in the last section.

5.2 RELATED WORK

We structure the discussion of related work into non-parallel techniques for achieving rapid visual response, concurrency and parallel programming in general, and multi-threading in interactive visualization in particular.

5.2.1 *Non-Parallel Techniques for Rapid Visual Response*

Without parallelizing event handling and the generation of visual results, constantly updating the entire visualization during interaction

does not scale for large data as both the update frequency and the application responsiveness degrade significantly. Therefore, many systems provide only a fixed (usually minimalistic) amount of feedback during continuous interaction to ensure responsiveness. For example, the commercial system TABLEAU shows only an elastic rectangle during dynamic query operations, whereas the query evaluation is triggered only after releasing the mouse button.

Tanin et al. [321] describe optimizations to dynamic queries. They pre-compute the set of affected items for each pixel position of a slider. During slider movement, newly selected data items are displayed on top of the visualization, whereas removed items are drawn with the background color. Several visualization systems implement this approach (including SPOTFIRE and TREEMAP4). However, as noted by Fekete [117], the restriction to pixel precision is often not tolerable. Fekete also points out that query optimizations alone can not guarantee responsiveness, because the limiting factor is usually the rendering.

One way to speed up rendering is to use abstraction methods, which can operate in data space to reduce data size (e.g., sampling [106]) or in view space to accelerate the rendering (e.g., binning [265]). However, performing costly computations (e.g., clustering) for large data may cause a temporary loss of application responsiveness. Moreover, while abstraction methods can emphasize important information better as compared to indiscriminately showing all items, they necessarily imply a loss of details, which is not always acceptable.

5.2.2 *Concurrency and Parallel Programming*

Many real-time graphics applications (e.g., games) exploit the parallelism of modern graphics hardware to achieve interactivity when transforming geometric or volumetric data into images. In information visualization, Fekete and Plaisant [119] investigated methods based on hardware acceleration to interactively visualize a million data items in scatter plots and treemap visualizations. Besides rendering performance, non-standard visual attribute mappings support perception, and appropriate interaction methods are integrated. However, while definitely useful for particular visualization and interaction techniques, transferring all steps of the visualization pipeline to the GPU is not always possible.

Chan et al. [74] developed a client-server system for exploring massive time series. Interactivity is maintained by delegating data queries to eight multi-processor database servers and by applying caching and pre-fetching mechanisms. To guarantee smooth interaction, constraints are derived from the capabilities of the employed hardware and software, and limit the distance that a user is allowed to travel per exploration step. It remains unclear how far such large-scale ar-

chitectures downscale to desktop PCs. Moreover, concurrency is not mentioned with regard to mapping and rendering steps. Chan et al. argue that the time required to map and render the data is negligible compared to query computation time, which contradicts the aforementioned claim by Fekete [117]. Obviously, the position of the bottleneck depends on the platform, the data size, and the type of both visualization and user interaction. Approaches that assume any of these factors as given can not solve the problem of guaranteeing responsiveness and maximizing feedback in general.

Parallelism and concurrency in a general sense are key topics of computer science and subject to ongoing research. There are numerous highly non-trivial related issues involving synchronization, communication, scheduling, consistency, deadlock prevention, data and task parallelism, performance, and scalability. In case of multi-threading, the advantages like utilizing commonplace multi-core architectures come at the expense of increased system complexity and higher implementation costs [226]. Automatic support (e.g., OpenMP or Intel Threading Building Blocks) provides help for exploiting parallelism for particular computations, but does not scale to parallelizing application-wide tasks like separating user input from generating visualizations. This problem has recently been termed as the *Multicore's Programmability Gap* [253].

Defining design patterns for particular problems has proven a good approach to cope with this complexity. Schmidt et al. [294] describe 17 patterns for concurrent and networked objects, covering event handling, synchronization, and concurrency. Similarly, Mattson et al. [248] define a pattern language for parallel programming, which is structured as dealing with finding concurrency, algorithm structure, supporting structures, and implementation mechanisms. More recently, Herlihy and Shavit [167] summarize the theory when programming for multiple processors and describe practical implementations for concurrent data structures. Many of the patterns and topics described in these books are applicable to systems for visual data analysis. Some patterns are partly related to the architecture as proposed in this paper (e.g., the *Active Object* design pattern [294]). However, the scope of most patterns is very general and none of these books addresses the requirements regarding responses to user interaction nor visualization aspects.

5.2.3 *Multi-Threading in Interactive Visualization*

Parallel algorithms and systems play an important role in scientific visualization. Besides approaches tailored towards dedicated graphics hardware or supercomputing environments, multi-threading is frequently used. However, many techniques focus on exploiting data parallelism by parallelizing the processing of data blocks [222]. On

a task level, computations in SciRun [194] are multi-threaded and do not block the GUI, but are typically not designed for early cancellation due to new input. The system ParaView [73] separates the VTK-based processing engine from the user interface by running both in different processes, and it relies on Tcl scripts for inter-process communication. Due to the design of ParaView to scale to client/server environments and batch processing, it supports only two static levels-of-detail – one during interaction and one for still images, and does not address early termination due to frequent user interaction. While there are also numerous approaches for progressive visualization, most of them focus on a dedicated visualization technique like volume rendering [69]. For this purpose, most approaches specifically tune the internal representation of the data to maximize performance.

In contrast, information visualization tools typically can not make as many assumptions about the data while offering the user many options to control the visualization pipeline. Unfortunately, little attention has been paid to multi-threading in information visualization literature so far. Heer and Agrawala [157] note that an important issue in implementing the *Scheduler* pattern is to handle concurrency, but no information concerning communication and synchronization is given. Their framework prefuse [162] offers a scheduler mechanism to execute costly computations in a separate thread, e.g., to drive animations. The XmdvTool uses multi-threading only for asynchronous data pre-fetching [101]. Our review of open source visualization software showed that The InfoVis Toolkit [117], Processing [129], and Mondrian [323] do not employ multi-threading at all.

The visualization system Improvise [369, 370] focuses on a generic approach for coordinating multiple views. It uses shared objects (*Live Properties*), a visual abstraction language (*Coordinated Queries*), and other coordination patterns including containment patterns that are related to semantic layers which will be discussed in Section 5.3.2. Improvise implements asynchronous displays based on retarding worker threads to allocate as much resources as necessary to the user interface thread (called *throttling* [371]). The authors also propose caching of visualization tiles and other enhancements to improve performance and interactivity during exploration. However, most aspects related to multi-threading are specific to Java. No details are provided on thread synchronization, early termination of updates, or on exploiting multi-threading for maximizing visual feedback. Moreover, the scalability to millions of data records remains unclear as "Interactive Performance" has been listed as future work [370].

To the best of our knowledge, there exists no generic architecture for inherently multi-threaded information visualization of large data, as many details about multi-threading have been left unpublished for information visualization systems. However, we believe that such an architecture could significantly facilitate the development of highly

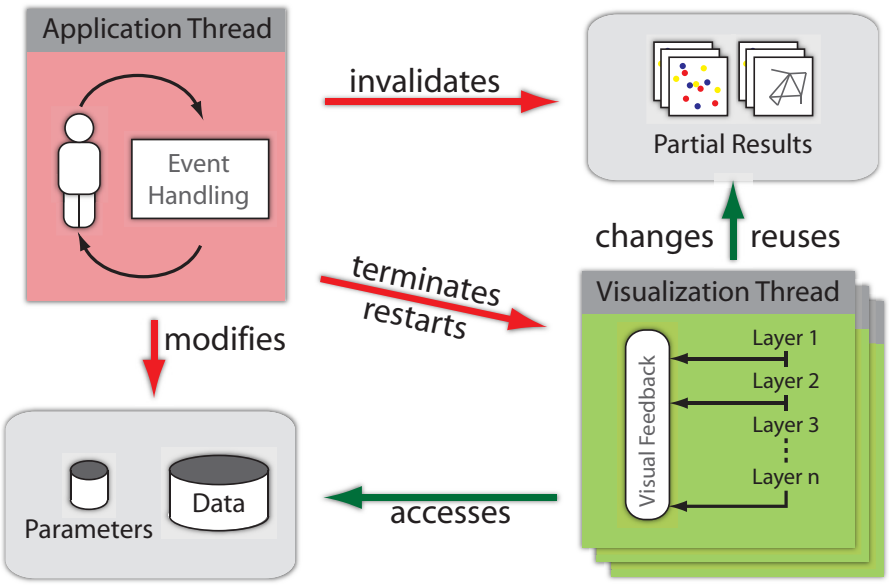


Figure 5.1: Overview of our architecture. It shows involved threads and data, how threads access this data, and how the application thread controls the visualization threads.

interactive information visualization tools, which combine responsiveness and rich visual feedback even during continuous user interactions. The importance and the current need for reusable architectures for visual data exploration are also documented by the fact that *Visual Analytics Infrastructures* has been established as a dedicated working package in the ongoing European project *VisMaster* [10].

5.3 MULTI-THREADING VISUALIZATION ARCHITECTURE

We first provide an overview of our architecture before discussing its details in Sections 5.3.1 and 5.3.2. The architecture builds on the separation of the main application thread and visualization threads (see Figure 5.1). The application thread is responsible for managing user requests in the event loop using event handlers. To keep this loop alive, event handlers are restricted to perform inexpensive tasks only, i.e., changing visualization parameters and triggering updates. Costly computations are delegated to visualization threads. In a multiple view environment, each view has its own visualization thread.

Especially during continuous interaction, updates in progress will frequently become irrelevant due to the arrival of new events. Therefore, the visualization thread checks repeatedly if it may proceed or should terminate early. For this purpose, we use a thread state object that serves as central point of communication. Depending on the semantics of the event, the execution of event handlers may be concurrent to the execution of the visualization thread (asynchronous), or mutually exclusive (synchronous).

The visualization is subdivided in image space into layers, and the visualization pipeline is processed separately for each layer. We will see later on that the term "layer" is used in a broader sense. Layers serve as partial visual results and can – in addition to partial results in data space – be reused across multiple executions of the visualization thread. Upon (early) thread termination, layers that have been validated so far can be displayed to provide as much visual feedback as possible and as early as possible.

5.3.1 *Early Thread Termination*

Our approach to support continuous interaction is to provide dynamic visual feedback by adapting the amount of detail to the available computation time. In general, this time is known only a posteriori, i.e., when it has elapsed due to receiving new input. Receiving this input, however, must be possible and not hindered by generating the feedback itself, which implies performing both tasks in parallel. It thus requires a multi-threaded architecture of each visualization.

According to the *Active Object* design pattern [294], invocation on an object should occur in the client's thread of control, whereas execution should occur in a separate thread. In our context, the 'object' is an interactive visualization, 'invocation' refers to event handlers for processing change notifications which are typically triggered by user input, and 'execution' means processing the visualization pipeline as widely accepted reference model [78] to generate visual results. Consequently, our visualization architecture maintains a single dedicated visualization thread T per view (maintaining multiple threads per view is discussed in Section 5.5). Changes of parameters along the pipeline affect the final image and thus need to trigger a new execution of the pipeline. In this case, T must abort its current execution (if running) and eventually start processing the pipeline anew. We call this paradigm Early Thread Termination (ETT), as an execution may be aborted before T has finished the final image. During execution, T must repeatedly check for the permission to proceed. Besides necessary clean ups like freeing resources, it must abort once this permission is no longer granted.

The time between requested and actual thread termination incurs a certain latency L. Minimizing L is a central aspect of ETT and requires checking for abort at a high frequency. It is therefore an important requirement that checking is inexpensive, which is generally possible as explained below. When accessing data sequentially, performing a check after every few thousand entries is usually sufficient. In general, T should check at least 10 to 20 times per second to achieve interactive response rates [304, 310], but preferably even much more often. However, it can become impossible to guarantee a high frequency when calling to foreign APIs, which is admittedly a potential limitation of

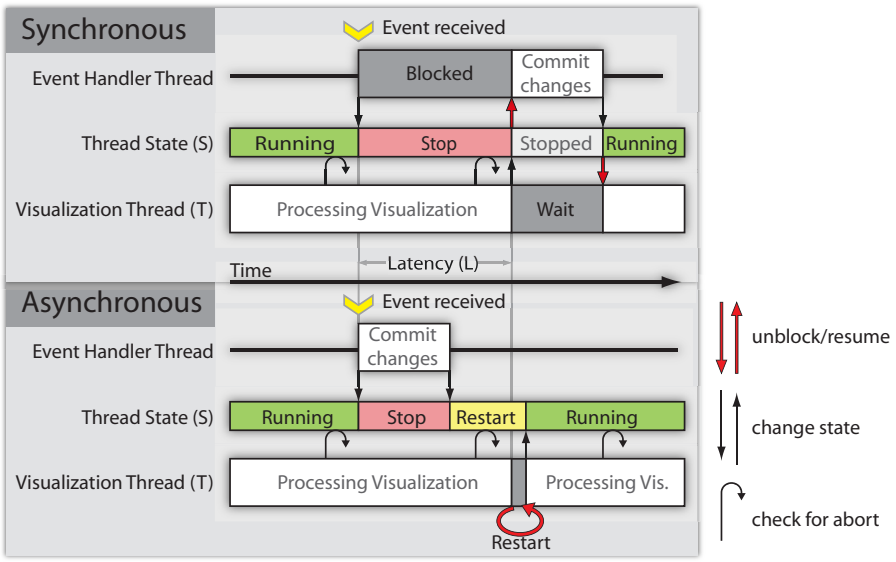


Figure 5.2: Comparison of synchronous and asynchronous event handling. Threads communicate by changing the thread state S.

ETT. In order to lessen the practical impact of this problem in particular and to make the responsiveness of the application less dependent on L in general, an important observation is that changes (i.e., events) are critical with a different degree. Some changes require an ordered communication between the handler and T while others do not. We distinguish synchronous and asynchronous handling.

Synchronous event handling (see Figure 5.2) enforces a mutually exclusive execution of the handler and T. This implies that handlers need to stop T, and must wait for this stop to occur before proceeding and eventually re-starting T. Synchronous event handling ensures that any subsequent execution of T is aware of the change.

Asynchronous event handling (see Figure 5.2) also tells T to stop execution, but does not wait for this to occur. After committing the change, which potentially involves modifying parameters, the handler states that T needs to be restarted as soon as possible and returns. A current execution of T may notice the effects some time afterwards.

Basically, all changes could be handled synchronously. However, the performance of a synchronous handler – and thus the responsiveness of the application – depends directly on L, whereas asynchronous handlers are independent of L and typically do not block the event-handler thread. With regard to responsiveness, asynchronous handlers are therefore preferable and should be used for uncritical changes like modified parameter values. On the other hand, some events require synchronous handling, for example, when objects or data must no longer be accessed (e.g., due to deletion). In practice, visualizations will need both synchronous and asynchronous event handling.

It is a potential problem of ETT, that if an execution is constantly aborted before completing any result, no result will be delivered at all. In general, redundant computation across multiple executions of T should be avoided. It is therefore an important issue to:

1. identify *partial results* along the visualization pipeline, which can be cached and potentially reused across multiple executions,
2. maintain a *state of validity* V[1..n], one for each partial result,
3. minimize the *impact of changes* by invalidating only those elements of V, where the respective result directly or indirectly depends on changed parameters.

Section 5.3.2 discusses this concept in detail in the context of interactive visualizations. For now, it is important that V is part of the communication between event handlers and T. Moreover, the communication involves the requested state of T, referred to as S. Figure 5.2 illustrates, how S is accessed and modified by involved threads over time for both synchronous and asynchronous changes. As a fundamental idea of ETT, T repeatedly checks the state of S. STOP tells T to terminate execution. RESTART also tells T to terminate its current execution, but to immediately restart a new one. Figure 5.2 also shows, how L directly affects the duration of synchronous handlers, which are blocked until T has reached the state STOPPED. In order to prevent deadlocks and livelocks, it is generally not recommendable for T to directly or indirectly trigger events itself.

As for all parallel systems, synchronization is important for ETT in order to avoid race conditions. The following points of synchronization can be identified:

- Between event handlers and T, as discussed above.
- Between different event handlers. If changes may occur in more than one client thread, event handlers themselves must be mutually exclusive in order to provide a predictable communication between each handler and T.
- Access to S between all handlers and T. As an important exception, if access to S is atomic (i.e., S is always accessed in one piece as is typically the case for basic data types), checking S for abort – i.e., read access – does not need synchronization, unless S is subsequently written in dependence of the result. This explains why checks for thread termination are usually cheap, meeting a requirement of ETT.
- Access to V between asynchronous handlers and T. For synchronous handlers, V is implicitly synchronized and thus does not require explicit synchronization.

- Access to *local* (i.e., view-specific) parameters along the visualization pipeline which are written by asynchronous handlers and read by T. However, synchronization of access is not sufficient to guarantee that the same state of parameters is used throughout one execution of T. To ensure this, T must maintain a local copy of those parameters which are potentially modified by asynchronous handlers. This is a major disadvantage of asynchronous handlers. Local parameters modified only by synchronous handlers are implicitly synchronized by the mutually exclusive execution. T does therefore not need to maintain a local copy of them. For this reason, modifications of memory-intensive local parameters (e.g., local derived data or a local selection state) typically require synchronous handling.

- Access to *global* (i.e., application-wide) parameters. Such parameters may change outside the execution of handlers of the particular visualization. In a multi-view environment, global parameters refer to the very information linking the views and thus include the data to be visualized itself. However, concurrent read access to global parameters by multiple views is necessary, because a synchronization of read-access to data would otherwise prevent concurrent processing of multiple visualizations, blocking all but one. It would thus eliminate responsiveness. Maintaining a local copy for each view is not practicable for large data. As a solution, changes to global parameters require two notifications: One synchronous notification preceding any modification, which forbids access, and one asynchronous notification permitting access when the modification is finished.

Finally, it is worth mentioning that although ETT is discussed in the context of visualizations in this paper, it is not limited to them. ETT can be applied to the design of any kind of objects that need to combine expensive computations with potentially frequent state changes due to interaction (e.g., ad-hoc queries or derived data columns).

5.3.2 *Layered Visualization*

As explained in Section 5.3.1, identifying and reusing partial results during the execution of the visualization thread is necessary to avoid redundant computation. This section discusses potential approaches to identify such partial results in the context of interactive visualizations, and how partial results help to display a dynamic amount of detail during continuous interaction.

A key idea is to subdivide the final image into separate passes through the visualization pipeline (referred to as "layer"), and to process one layer after the other. Each layer provides additional information and thus increases the amount of detail. It is important that the

processing order may be chosen independently of the display order to prioritize important information for previews, as discussed below. In contrast to decomposing work in data space, which is often not possible in information visualization (e.g., computing graph layouts), layering is thus a concept for decomposing results in view space. For most visualizations, it is possible to identify one or more types of layers:

- *Semantic layers* are semantically different parts of the visualization. Typical examples include the background (e.g., an image, a map, a grid, etc.), all visible data items, those items selected by an ad-hoc query, and overlays providing detail-on-demand like labels or precise values [370]. It is reasonable to process semantic layers by decreasing relevance or increasing effort. For example, processing the layer of selected items ("focus") first will typically be less effort than considering all items ("context") and may already provide the most important information.

- *Incremental layers* can be identified in item-based visualizations (like scatter plots or parallel coordinates) by subdividing the data into disjunctive subsets and treating each subset as layer. Each incremental layer contains a sampled version of the data and the accumulation of all layers represents the entire dataset. A desirable feature is to ensure a sampling distribution that conserves important properties of the final image as soon as possible, i.e., in the layers being processed early. Desirable properties could be a size or a relative distribution similar to the final image. This aspect boils down to determining an index that specifies the order in which data entries are to be dealt with.

- *Level-of-detail (LoD) layers* provide visual representations of the same data with different complexity and rendering cost. In contrast to incremental layers, more detailed LoD layers may replace coarser layers, which are consequently not part of the final image. For example, a tree-map showing a hierarchy depth of four might be used instead of one showing only two hierarchy levels [60]. The design space for level-of-detail layers is large and includes abstraction in both view and data space. A *view space-based approach* could be to reduce the rendering quality for early layers, possibly in addition to displaying a sampled version of the data. Examples include disabling anti-aliasing and reducing geometric resolution. As example of a *data space-based approach*, lower levels of details might display features of the data like major trends, clusters, and outliers, or may use aggregation (e.g., bin maps) to reduce the rendering effort [265]. As a special case of LoD layers, *iterative layers* refer to visualizing intermediate results of an iterative algorithm, as for example the

computation of a graph layout. In this case, each new layer (i.e., each iteration) typically replaces any previous iteration.

Once the final image could be completed, it is shown to the user. According to the ETT paradigm, the work is aborted whenever relevant parameters have changed. However, it is an important design choice, how visual feedback can be provided even in cases when the visualization thread could not complete.

DESIGN CHOICE 1: IMMEDIATE FEEDBACK VS. FEEDBACK ON TERMINATION. *Immediate feedback* updates the display whenever a layer could be completed. As advantage, feedback is given early and is guaranteed to be up-to-date. As disadvantage, the composition of each image is exposed to the user and produces potentially disturbing flicker, which could be misinterpreted as data artifacts in extreme cases. In contrast, *feedback on termination* updates the display just before thread termination to show all valid layers, i.e., the highest amount of detail that could be dealt with in between two consecutive user interactions. The advantage is that only one image is generated per execution of the visualization thread, which reduces flicker significantly. As disadvantage, it might take longer until feedback is provided – in particular, if the execution is not aborted. The number and the type of layers and the effort for generating the final image are critical factors in the decision for one approach.

DESIGN CHOICE 2: TYPE, NUMBER, AND ORDERING OF LAYERS. In general, the number of visual layers increases with the complexity of a visualization. A single layer is most likely sufficient for basic bar charts, whereas a subdivision of parallel coordinates discriminating multiple selections and providing overlays could involve several semantic layers, which could in turn consist of LoD layers. Layers can thus be organized hierarchically. In this case, it is a design decision whether to prioritize level-of-details over semantic layers or vice versa. Apart from semantic dependencies, a processing order of layers may also be implied by internal dependencies between layers. For example, layers showing data items may depend on the layer showing the grid to determine the ranges of all displayed data dimensions.

An important decision for item-based visualizations is whether to provide fine-grained incremental visualization (i.e., a large number of incremental layers), or a fixed – typically small – number of LoD layers. The first case maximizes the average amount of provided detail (e.g., the number of shown items), yet it also increases the variation in the amount of details over time. This might create the impression of flicker even if a single image is shown per execution of the visualization thread. The second case is more stable with respect to the visual feedback, yet also reduces the possibility to adapt the amount

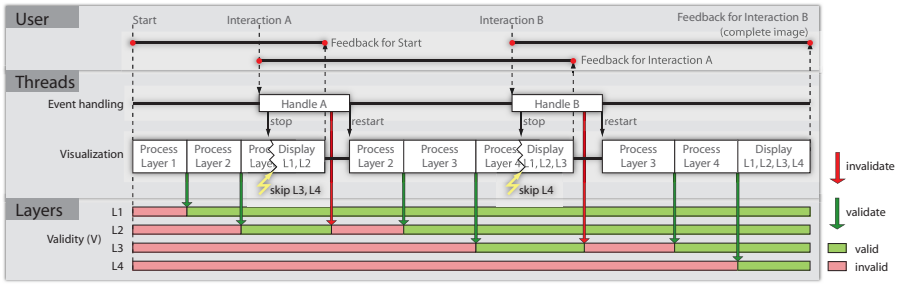


Figure 5.3: Caching and early feedback of layers. Two user events (handled synchronously) interrupt the computation and invalidate layers. Visual feedback is provided on thread termination.

of detail to the available computation time. This shows a trade-off between the amount of detail and stability.

DESIGN CHOICE 3: CACHING CONCEPTS. In order to avoid redundant computations, layers also represent reusable partial results. According to the model as proposed by Chi [78], different parameter adjustments affect different stages of the visualization pipeline. For example, changing a color could just require a redraw of already filtered, projected and possibly aggregated data. Performing just the rendering may thus be magnitudes faster than processing the entire visualization pipeline. We refer to this type of reuse as *caching results in data space*, which is related to lazy evaluation and demand-driven pipelines in visualization literature [222]. It is particularly useful for types of visualizations where computing internal representations of the data is relatively expensive as compared to the rendering itself, and where these representations consume a limited amount of memory. Examples include pivoted values of categorical data, aggregated representations as generated by binning continuous data, and the state of iterative algorithms (e.g., for graph-layout and clustering).

On the other hand, some changes affect the entire visualization pipeline, but only for a particular (semantic) layer. For example, ad-hoc queries may require a frequent re-processing of the selected data ("focus"), but may have no impact on the visualization of the entire data ("context") or other visual elements like the grid. In this case, it is advantageous to *cache results in view space* for each layer independently. Figure 5.3 illustrates caching and reuse of layers from the point of view of the user, the involved threads, and the layers as well as their validity. In this example, events are handled synchronously, feedback is provided on abort, and the validity is assumed on a per-layer basis, i.e., not taking partial results along the pipeline into account.

The additional complexity for implementing item-based visualizations using layers as compared to naive implementations can be summarized as:

- Invalidate affected layers instead of redrawing everything.
- Support multiple iterations through arbitrary subsets of the data instead of processing all items in one pass. In the case of multiple selections, for example, iterate through the data once for each selection (and once for all entries), instead of mapping the selection state of each entry to visual attributes like color or size within a single pass. As data records may appear in multiple layers, more significant layers must be shown on top of less significant ones. In particular, it is often desirable – though not required – that the visual representation of a selected item occludes its representation as non-selected item.
- Render layers to off-screen buffers and blend them together instead of drawing directly to screen. In practice, this is more easy to implement for 2D visualizations. In 3D, a composition in view space is generally harder to realize due to the additional depth-information necessary for correct occlusion handling.
- Check for thread termination regularly.

In our experience, these issues apply to all types of item-based visualization (scatter plots, parallel coordinates, time-series views, etc.). Sorting the items by their selection state or grouping them by identical rendering parameters is usually even necessary without explicit layering. The additional complexity imposed by *semantic layers* is thus usually (much) less than 20% in terms of lines of code. For *incremental layers*, the main effort lies in identifying an index for fair sampling (i.e., shuffling rows appropriately). Implementations of incremental layers typically require a single off-screen buffer where new visual output is added. For *LoD-layers*, the additional complexity may range from negligible (e.g., just disabling anti-aliasing) to considerable for cases that require the computation of features of the data like clusters.

5.4 EVALUATION

This section evaluates the proposed architecture. The goal is to demonstrate its applicability and its possibilities to support visual exploration of large data. All tests have been conducted on consumer hardware: Intel Core 2 Quad CPU with four cores at 2.4 GHz, 4 GB of main memory, and an NVidia Geforce 8800 GTS graphics card. Windows XP Professional x64 Edition was used as operating system.

As test dataset, we used a multivariate CFD-simulation of a two-stroke engine. The data table consists of 14.589.282 rows and 50 columns (approx. 5.3 GB), which are mostly physical properties like temperature or pressure. One row in the data table represents one cell of the model geometry at one particular discrete time-step of the simulation. Previous analyses of the dataset have been conducted using

the SimVis system [98], which also implements the proposed architecture (see Section 5.5). The focus of this evaluation is on performance issues with respect to maximizing visual feedback during continuous interaction for data of such non-trivial size.

We performed all tests in a system for visual exploration, termed Visplore. It provides more than 10 different visualizations, which are partly standard (e.g., 2D and 3D scatter plots, parallel coordinates, histograms, etc.) and partly specific to certain application tasks [271]. All views implement the proposed architecture to support continuous interaction and early visual feedback. Multiple views are linked by ad-hoc selections and derived data columns, whose evaluation also utilizes the ETT paradigm. Visplore is written in C++, it uses GTK+ as GUI library and OpenGL for rendering. The system has successfully been applied to analyze data of numerous application domains and is going to be released as part of the software suite of our company partner AVL List GmbH in 2009. However, the focus of this evaluation is to demonstrate the possibilities of our architecture, not to compare Visplore as such against any other system.

We discuss two examples of continuous interaction, which cover important cases: (1) The interaction concerns a single view, yet entails performing the mapping and the rendering stage of the visualization pipeline for the entire data. (2) The interaction concerns multiple views, but affects a single semantic layer. The implementations of the involved views also cover different options for the design choices 1 and 2, as explained below.

For the first example, we drag a slider to restrict the value range displayed on the X-axis of a 2D scatter plot. For the evaluation, we stored the interaction sequence as a macro (which takes 12 seconds) and replayed it with four different implementations to highlight trade-offs in the design space.

- *Case 1.1* A single-threaded implementation as example of a naive approach, i.e., each change entails a redraw of the entire visualization in the same thread as used for handling events.

- *Case 1.2* An ETT-based implementation providing immediate visual feedback for two static LoD-layers as example of a common case in many visualization systems. The first layer consists of a sampled subset of 32.768 data items without point smoothing and without transparency, the second layer is the entire dataset using point smoothing and transparency for visualizing density.

- *Case 1.3* An ETT-based implementation providing early visual feedback of fine-grained incremental layers as example of maximizing visual detail. The visualization pipeline is processed separately in blocks of 4096 rows and the visualization thread checks for abort after each block. The view provides feedback

	Case 1.1	Case 1.2	Case 1.3	Case 1.4
avg. # events handled / s	4.1	36.3	35.9	35.6
avg. # visual updates / s	4.1	13.2	35.9	0
min. # visual updates / s	3	9	18	0
min. items shown / update	100%	0.2%	0%	0%
q25 of items shown / update	100%	0.2%	3.5%	0%
avg. items shown / update	100%	0.2%	8.8%	0%
q75 of items shown / update	100%	0.2%	10.8%	0%
max. items shown / update	100%	0.2%	97.1%	0%

Table 5.1: Results for example 1 – restricting a slider.

on termination, displaying all data handled so far, and on completion of the entire data set.

- *Case 1.4* An ETT-based implementation without preview visualization, i.e., visual results are only shown if the thread completed the entire visualization. This case has been chosen as example of evaluating the effect of multi-threading without layering.

We use several indicators. Responsiveness is quantified by the average number of user events that could be handled per second during the interaction. The frequency of visual feedback is given by the average and the minimal rate at which the visualization is updated per second. The amount of feedback and its variation – indicating flicker – is given by the minimal, average, and maximal percentage of shown data per update as well as the percentage of data that could at most be shown for 25% and for 75% of the frames (i.e., quantiles). Table 5.1 shows the results for the time between the start and the end of the interaction.

In case 1.1, feedback is given at a very slow rate. Even worse, the application is hardly responsive during the interaction. All other cases show that multi-threading ensures responsiveness of the application. Comparing case 1.2 to case 1.3 highlights the trade-off between minimizing flicker and maximizing visual feedback. In case 1.2, flicker does not occur at all because the visualization is updated only if the first LoD-layer could finish while the entire visualization (i.e., the second LoD-layer) could never complete. However, both the frequency and the average amount of visual feedback are significantly lower than in case 1.3, where even the minimal update rate of 18 is clearly faster than the desirable frequency of 10 (= 100 ms per update), and where a considerable percentage of the data (8.8%, i.e., 1.2 million items) is displayed on average – the best values are close to showing the entire dataset. On the other hand, the feedback sometimes drops

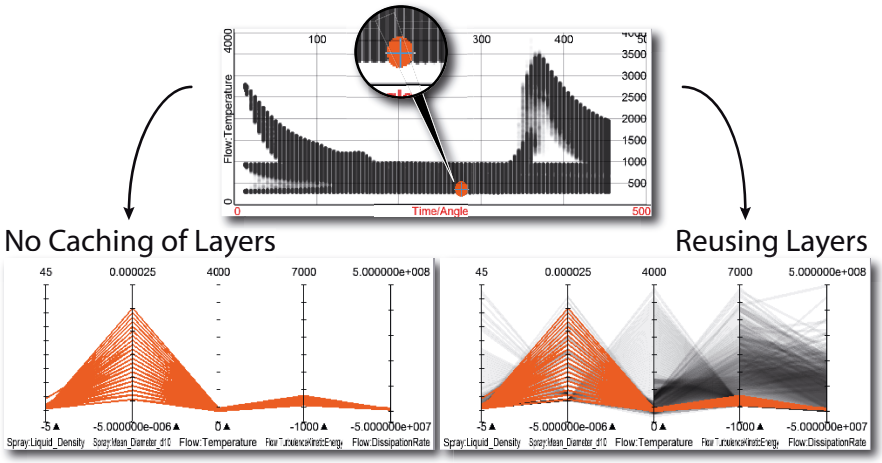


Figure 5.4: Example 2: comparison of an ad-hoc selection of entries beneath the mouse cursor in a multiple view setup for two implementations of parallel coordinates. The response time is equally low in both cases, but the amount of detail is much higher when caching and reusing layers.

to displaying the grid without data and flicker is generally high in case 1.3. Case 1.4 does not provide any feedback on the data, because at no point during the 12 seconds of interaction, the visualization thread is able to process the entire data in between two consecutive user events. This highlights the importance of early visual feedback. However, even case 1.4 is arguably superior to case 1.1, as it ensures responsiveness (i.e., the slider is updated continuously) and pausing the slider movement without releasing the mouse button would give the visualization thread the time to generate visual feedback. In practice, VISPLORE uses case 1.3.

For the second example, we drag an ad-hoc selection in a 2D scatter plot and highlight the selected data items in a linked parallel coordinates view showing 5 axes (see Figure 5.4). Besides other types of queries, VISPLORE offers an instant ad-hoc query (referred to as *Focus*) that always selects all data items under the mouse cursor. The *Focus* is pre-computed for all possible mouse-positions of a view, which reduces its evaluation to a look-up operation. However, each view needs to update frequently (i.e., on every mouse move) to reflect *Focus* changes. For the evaluation, we again stored an interaction sequence as a macro, this time a continuous mouse movement of 23 seconds, which causes frequent *Focus* updates. The macro has been tested against the following four implementations of parallel coordinates.

- *Case 2.1* A single-threaded implementation without caching any partial results as example of a naive approach where each change necessitates processing the entire visualization pipeline in the application thread

	Case 2.1	Case 2.2	Case 2.3	Case 2.4
avg. events handled / sec.	0.2	20.1	13.5	8.8
min. response time (sec.)	6.7	0.03	0.03	0.03
avg. response time (sec.)	12.1	0.07	0.09	0.09
max. response time (sec.)	13.4	0.23	0.25	0.14
average data shown	100%	0.05%	100%	100%

Table 5.2: Results for example 2 – linked ad-hoc selection.

- *Case 2.2* An ETT-based implementation without caching any partial results. However, the *Focus* is processed first and is immediately displayed to provide visual feedback.
- *Case 2.3* An ETT-based implementation caching the image of the *Context* layer, i.e., the semantic layer displaying all data items, and reusing this images as long as the layer stays valid. The comparison of case 2.2 to case 2.3 is intended to emphasize the effect of caching.
- *Case 2.4* Same as case 2.3, but single-threaded to evaluate the effect of caching separately

The average number of events that could be handled per second during the interaction quantifies the responsiveness of the application. The minimum, maximum, and average response time indicate the latency between changing the *Focus* and providing visual feedback. The average amount of shown data refers to the number of visualized items. In contrast to example 1 where continuous movement triggers updates constantly, the frequency of visual feedback is not a reasonable indicator in example 2, as moving the mouse cursor through empty space does not trigger updates. Table 5.2 shows the results.

For case 2.1, interaction is practically impossible as the system blocks for several seconds at each mouse move. For the cases 2.2 and 2.3, the system stays responsive and visual feedback is provided quickly. However, case 2.2 only displays the *Focus* most of the time, as illustrated by the left image in Figure 5.4 while the entire visualization is only shown when the *Focus* is not updated for some time. Case 2.3 on the other hand always displays the entire visualization due to reusing the cached image of the *Context* as shown by the right image in Figure 5.4. The results of case 2.4 are similar to those of case 2.3. This is not surprising considering that by re-using the image of the *Context*, not much work is left to be done. However, interactions invalidating the *Context* degrade the responsiveness as badly as shown for case 2.1.

Concluding, this evaluation demonstrates that the proposed architecture successfully preserves responsiveness of the application while

providing visual feedback during continuous user interactions even for a dataset of 14.5 million items. It also shows that ETT, previews, and caching must work together to achieve this goal. Although not shown in this evaluation, our architecture also scales with respect to a large number of views. For informal evidence we refer to publications related to the systems implementing the architecture (for example [99]).

5.5 DISCUSSION AND FUTURE WORK

Three important yet contradicting objectives of our architecture are:

- to *minimize the latency* between interaction and visual feedback, which is equivalent to maximizing the frequency of updates
- to *maximize the amount of detail* shown upon ETT
- to *minimize the variation* of the amount of shown detail in order to provide a stable image.

The design choices 1 and 2 have been explicitly denoted, because they allow for trading off these objectives against each other: (1) Providing immediate feedback for each completed layer minimizes latency while it maximizes flicker – especially in the case of fine-grained layering. (2) Utilizing many layers allows for minimizing latency and maximizing detail, but the flicker is usually significant.

Another option for trading off latency against preview details concerns the handling of asynchronous events. As requests by asynchronous handlers are less critical than those issued by synchronous handlers, they can be ignored for some time. For example, it seems reasonable to finish and to display costly visual results which are almost complete when receiving a request for thread termination.

Design choice 3 is essential for adapting the architecture to a wide range of visualizations. Caching of results in view space is important where rendering is expensive, as for item-based visualizations. Caching results in data space is suitable in case of expensive computations yet potentially cheap rendering. For example, visualizing a large data warehouse may require aggregating billions of data rows for generating little output like a bar chart. Although less obvious than for costly rendering, early visual feedback is also possible in this case: the final results could repeatedly be estimated and displayed during the computation based on already considered data.

The most important limitation of our architecture is the need to frequently check for termination. As already mentioned, this may become impossible when passing control over to foreign APIs for a long time. This is particularly critical for synchronous changes as it compromises responsiveness much like a single-threaded architec-

ture. Asynchronous changes preserve responsiveness, but the latency of visual feedback may still be disturbing.

It may seem reasonable to spawn a new visualization thread for each asynchronous event (synchronous changes must wait for thread termination anyway). However, we decided against this option, because our practice has shown that gains are small compared to a significant increase in complexity. While synchronization is complex for a single visualization thread, it becomes worse for multiple threads. Redundancy increases too, as each thread requires a copy of all local view parameters. Furthermore, it is not reasonably possible for graphics APIs that do not support concurrent access to the same rendering context (e.g., OpenGL). In such cases, maintaining a single thread per view avoids the significant overhead caused by context switches, which is incurred when using a common thread pool for multiple views.

The proposed architecture has been implemented in several systems besides VISPLORE (see Section 5.4). The SIMVIS visualization framework [99] is mainly used in the context of 3D or 4D simulation data. Multiple linked views are provided to the user, allowing for interactively selecting and viewing data in different attribute spaces. Multiple of these views implement ETT to maintain responsiveness even when visualizing hundreds of millions of data entries. Attribute views such as scatter plots or time series visualizations [257] all support asynchronous as well as synchronous thread termination and use cached background layers to provide feedback to the user during continuous interaction. The 3D visualization capabilities of SIMVIS also rely on concepts presented in this work to perform progressive rendering as well as level of detail rendering during continuous interaction using a multi-resolution approach. When dealing with very large data, a common approach is to access data in blocks [222] which are guaranteed to be in memory while the rest may be swapped to disk (known as out-of-core visualization). Both VISPLORE and SIMVIS perform active memory management, which shows that out-of-core visualization is compatible with our architecture. Switching blocks even provides a dedicated point to check for thread termination.

CGV is a system for interactive exploration of graphs [332]. It uses multiple linked views to show different aspects of clustered graphs. Early thread termination is used for instance in the graph splatting view. Because the performance of graph splatting depends not only on the number of data items, but also on pixel resolution, the view uses a level-of-detail layering and renders the splat progressively at increasing resolutions. Continuous interactions as for instance dragging a noise level slider or a threshold slider are guaranteed to stay responsive and feedback is provided quickly.

Axes-based visualizations map data values to positions relative to well-arranged visual axes. The TIME WHEEL [329], for example, allows

among other interactions for continuous rotation of data axes around a central time axis. Interactive visual feedback is crucial in this case to help users maintain the mental map. Therefore, ETT and layering are applied for the TIME WHEEL. It is subdivided into semantic layers: axes layer, labels layer, preview layer, and data layer, which are drawn in this order. As a result, the basic shape of the visualization (i.e., the axes), labels, and a sampled version of the data are visualized early, while the entire data set is processed in the background.

We see multiple directions for future work. First, the design space potentially involves more than three design choices as discussed in this paper, and a systematic coverage would be very helpful. Second, the aspect of flickering needs more thorough research, including a user-evaluation about how much flickering is considered acceptable. New approaches could strive for minimizing flickering while still providing much visual feedback, e.g., by ignoring asynchronous changes for some time or by fading the images of consecutive updates. Third, it still remains a challenge to achieve rich visual feedback during continuous interaction in a distributed environment.

5.6 CONCLUSION

Continuous user interaction is important in information visualization to support smooth data exploration. A key concern is to preserve responsiveness and to provide rich visual feedback at the same time. Realizing this in practice is difficult, however, as it requires parallelism of application tasks which involves many non-trivial details.

We proposed a generic multi-threaded architecture to support continuous interaction and to help avoiding pitfalls. As illustrated by the evaluation, our architecture scales with respect to data size and the number of views. It is applicable to many types of visualizations regardless of a particular platform, programming language or graphics API, as instantiations in several visual analysis systems and tools show. GPU-based rendering is supported, but not required.

We identified and discussed three major design choices to allow others to adapt the architecture to particular visualization needs and to trade off latency against the amount of detail and the stability of visual feedback during continuous interaction. We also discussed in detail communication and synchronization aspects of our architecture as key issues of any multi-threaded program. We believe that our architecture will facilitate the development of highly interactive information visualization tools, and that it will help to promote rich visual feedback during continuous user interactions.

6

CGV – AN INTERACTIVE GRAPH VISUALIZATION SYSTEM

CONTRIBUTION This chapter makes a contribution to the design of interaction by introducing novel techniques for interacting with graph data. The techniques address intermediate-level exploratory tasks on graphs. This chapter also contributes to the engineering of interaction by implementing the architecture presented in Chapter 5 as a larger system for interactive graph visualization. With its multiple views, the system supports even higher-level analytic activities.

ABSTRACT Previous work on graph visualization has yielded a wealth of efficient graph analysis algorithms and expressive visual mappings. To support the visual exploration of graph structures, a high degree of interactivity is required as well.

We present a fully implemented graph visualization system, called CGV (**C**oordinated **G**raph **V**isualization), whose particular emphasis is on interaction. The system incorporates several interactive views that address different aspects of graph visualization. To support different visualization tasks, view ensembles can be created dynamically with the help of a flexible docking framework. Several novel techniques, including enhanced dynamic filtering, graph lenses, and edge-based navigation are presented. The main graph canvas interactions are augmented with several visual cues, among which the infinite grid and the radar view are novel. CGV provides a history mechanism that allows for undo/redo of interaction.

CGV is a general system with potential application in many scenarios. It has been designed as a dual-use system that can run as a stand-alone application or as an applet in a web browser. CGV has been used to evaluate graph clustering results, to navigate topological structures of neuronal systems, and to perform analysis of some time-varying graphs.

ORIGINAL PUBLICATION [332] — C. Tominski, J. Abello, and H. Schumann. CGV – An Interactive Graph Visualization System. *Computers & Graphics*, 33(6):660–678, 2009.

6.1 INTRODUCTION

In many application fields, visualization techniques have been recognized as a potential tool to order, manage, and understand large data. Interaction has long since been an important aspect in information visualization [303, 19]. More recent work confirm its relevance [212, 364, 385]. It is becoming apparent as stated by Thomas and Cook [325] that: "Visual representations alone cannot satisfy analytical needs. Interaction techniques are required to support the dialogue between the analyst and the data."

Interaction in general can be thought of as being driven by some computational serving process that is able to produce within a "small" time frame a "verifiable" answer to a client's query that is formulated in a language common to both the server and the client. Following even this very limited thinking, it is clear that designing interactions in information visualization is a challenging task in several respects:

- First, there is no common "query language" between a visualization system and the human user.
- Second, the time frame used to judge the quality of interactivity is very small. Due to the high involvement of the human visual system what this time frame should be is not really well understood.
- Third, what the data encoding (i.e., the visual representation) shall be is a current research endeavor.
- Fourth, verification of the quality of visual feedback provided by a visualization system to a human is still a largely subjective endeavor that is highly domain dependent.

In the case of graph visualization systems the challenges are very unique but not as challenging as those of general visualization systems or even those of general information visualization. The reasons are the following: The data model is well specified (i.e., graphs) [96]; there are fundamentally not too many visual representations to choose from (i.e., some flavor of low dimensional node-link diagrams or matrix-tensor representations) [47]; the fundamental objects of discourse are nodes, edges, and patterns obtainable from them as sets or sequences (e.g., subgraphs, induced subgraphs, paths, cycles) [223]; and finally graphs are amenable to filtering and aggregation operations that preserve certain properties at large scale [168, 13, 12].

Our focus is on interactive system building rather than on promoting any particular set of visual tools. The fundamental problem we address is how to visually interact with a graph that is too large to fit on the available screen. We present a graph visualization system called CGV (Coordinated Graph Visualization) whose primary focus

is on interactivity. CGV is based on our personal experiences developing tools to extract information from a variety of "large" graphs arising in industrial and academic settings, including telecommunications, search engine, and biomedical data. We have been facing disk resident multi-attribute labeled graphs that do not fit in the available RAM (i.e., semi-external or fully external graphs [14]). The graphs have been typically sparse, low diameter, and with high clustering coefficients. Typical questions have revolved around informal notions like "global graph views", "high density graph regions", "induced sub-graphs on subsets of vertices whose associated attribute values satisfy a Boolean formula", and visual access to the neighborhood of a given vertex.

Fortunately, global graph views are well encapsulated by the notion of *maximal antichains* in a *hierarchy tree* (see [12] and Section 6.3); each node in a maximal antichain corresponds to a "graph region", and induced subgraphs can be viewed as the result of specialized range filters. This suggested to base the system on hierarchy tree computations and methods to extract graph macro-views that are small enough to fit on the available screen [15], where they are shown as multi-linked visual representations together with mechanisms to filter nodes and edges via range sliders.

To achieve scalability two special RAM resident macro-views (Top and Bottom) are used in conjunction with a fast disk index that is triggered by the Bottom macro-view. The Top and Bottom macro-views are parameterized by the amount of RAM available. Their views can be panned, zoomed and tagged for future reference. Panning has been enhanced by a special radar view introduced in [333]. Visual access to the neighborhood of a point required the introduction of special lenses and edge-driven navigation (first introduced in [330, 16] and considered later on in [256]). To ensure interactivity, multi-threading became a necessity. To reach a wide user base and to facilitate future system evaluation CGV has been designed as a dual-use system that can run as a stand-alone application or as an applet in a web browser. To our knowledge, there is no single turn key system that offers to an unsophisticated user the "large" graph visualization and navigation facilities offered by CGV.

In summary, any system for visualizing large graphs has to perform the following three tasks:

1. Build a graph hierarchy of logarithmic depth and bounded degree and extract from it graph macro-views that fit on the available resources.

2. Come up with multiple linked visual representations of a given hierarchy tree and its associated macro-views at a multi-scale level.

3. Devise visual interaction mechanisms that allow users to pan a large macro-view and to navigate among macro-views in a smooth fashion.

We describe how each of the three tasks is implemented in CGV, compare our system to a selection of existing systems, and provide feedback from early adopters. CGV's predecessor is the system ASK-GraphView [15]. To achieve flexibility, we opted for a modular multiple view design instead of a monolithic architecture.

6.1.1 *Main Contributions*

CGV's design allows for integration of synchronous and asynchronous computations. This is a necessary requirement for interactive systems to be able to generate and present users with feedback across multiple views in a consistent, uniform and timely manner.

Besides standard visual graph representations (i.e., node-link diagrams, matrix views, graph splatting), CGV uses focus+context hierarchy views to drive overall navigation. These views include a textual tree representation, two color-coded representations, and a projection of the hierarchy onto a 3D hemisphere. These views are complementary.

One of the system's novelties rests on the tools provided to interact in a coordinated fashion with these visual representations of "large" multivariate graphs at different levels of abstraction. Commonly accepted pan and zoom operations, are enhanced with pan-wheel interaction and edge-based navigation. These are augmented with visual cues that include smooth viewport animation together with a novel use of an infinite grid and a look ahead radar view. At the most refined level of granularity, CGV offers a variety of lenses that help reduce node and edge clutter. This is a feature we have not seen in existing graph visualization systems. Data filtering in CGV is aided by an interface to compose complex filters out of basic range sliders and textual filters.

CGV provides a history mechanism to allow for undo/redo of interactions. This is a feature rarely seen in current graph visualization systems.

In summary, CGV incorporates traditional and non-traditional interaction techniques to support the exploration of graphs at different levels of abstraction. One of CGV's strengths is its extensibility and ease of maintenance. CGV is implemented in Java and can be operated as a stand-alone desktop application across multiple platforms or as a visualization client in a web browser.

6.1.2 *Paper Overview*

In Section 6.2, we describe how we follow the Model-View-Controller design to meet our specific interactivity needs. CGV's data model is discussed in Section 6.3. The different views provided by the system are the subject of Section 6.4. Section 6.5 details the set of user tools that the system offers to interact in a coordinated fashion with multi-scale visual representation of large graphs. It also discusses visual augmentation of interactions, the user interface, and the history mechanism provided by CGV. Section 6.6 compiles a list of common interaction mechanisms and details the extent at which they are supported by CGV and four other graph visualization systems. Section 6.7 describes how early adopters feedback helped improve CGV. In Section 6.8, we conclude and point out some directions of further research. Sections 6.9 and 6.10 illustrate the use of the system in two case studies.

6.2 CGV'S DESIGN AND ARCHITECTURE

6.2.1 *Overall Goals*

The literature offers a wealth of sophisticated algorithms for graph visualization. When such algorithms have to work in concert with each other and with different interaction methods, architectural aspects become important. In this section, we discuss related challenges and give an overview of how CGV addresses them.

One of the major challenges in graph visualization stems from the sheer size of readily available multi-attributed data. We refer to the size of a data set as the number of objects that it contains, e.g., the overall number of nodes, edges and labels for the case of string labeled graphs. The problem of visualizing large graphs (without explicit consideration of node or edge attributes) has been addressed in previous work (see [168, 13, 12] for a review). The basic idea is to transfer the big and hard to solve visualization problem to one that is scalable and amenable for interaction at different levels of abstraction. This is achieved by computing a hierarchy on top of the raw graph data (see Section 6.3). The hierarchy provides access to specially selected macro-views of the graph and is the backbone for interactive information drill-down.

In the case of graph data where the elements have associated a vector of attributes, multiple view approaches enable users to look at the data from different perspectives (see Section 6.4). The use of multiple coordinate views in a more general setting has been documented in [363, 370]. For exploratory analysis, users need to switch the perspective frequently. Therefore, it is crucial to provide means to navigate between different size data subsets and different data attributes.

Current graph visualization systems offer standard interaction (e.g., brushing, zoom & pan) to support basic data navigation. Large and more complex data demand for better support.

Our goal has been to enhance existing interaction mechanisms to aid visual graph exploration. This is partially achieved by providing flexible interaction access to different data views. A critical aspect we have dealt with is the integration of synchronous interaction with otherwise asynchronous computations.

In summary, at a very high level the major driving issues behind our design decisions were:

- Flexible access to different views of the data,
- Enhanced interaction methods for graph exploration,
- Visual augmentation of interaction methods, and
- Integration of synchronous interaction with asynchronous computations.

Further decisions were made based on [320, 157].

6.2.2 *System Design*

To achieve the aforementioned goals, we follow the Model-View-Controller (MVC) design [217] as the basis for CGV. It dictates a strict separation of the *data model*, the *views* on the model, and the *controllers* that regulate interaction.

The basic elements of CGV's *data model* are attributed nodes and edges that constitute a graph hierarchy (see Section 6.3). This graph hierarchy allows for multi-scale access to large graphs. It is the fundamental data structure for interactive navigation between different data subgraphs. We use the *decorator pattern* [133] to equip nodes and edges at runtime with view-specific information depending on the visualizations used. Since not only the data determine the visual output, but also the visualization parameters, CGV integrates them in the data model. Making visualization parameters first-class objects becomes a necessity for the development of flexible, enhanced interaction.

CGV provides several dedicated *views*. Each view is designed as a monolithic interactive component that implements its own visual mapping and rendering as well as a common interface to access and alter data and visualization parameters (see Section 6.4). This allows us to utilize different graphics engines simultaneously to drive the visualization. We use Java2D for 2D views and rely on Java OpenGL Bindings (JOGL) for 3D views. To be flexible in choosing perspectives on the data, we opted for a design that is based on dynamic view composition, rather than on a hard-wired layout of visual components. In

the default setup, six different views offer a broader perspective on a single data model. Users can create alternative view arrangements for different visualization tasks and store them for later reuse. To support visual comparison of two or more data models, several instances of CGV can be executed.

Controllers regulate interaction with the data model and its views (see Section 6.5). In information visualization, interaction is modeled as adjustment of the data model, which includes the raw data and its visualization parameters [188]. Adjustments can be conducted in two different ways: by direct manipulation and via a dedicated graphical user interface. We further distinguish between interactions that are coordinated and, hence, have global effect (e.g., dynamic filtering), and interactions that are not coordinated, i.e., have local effect (e.g., changing the zoom level of a view). To achieve a high degree of consistency among interactions, especially for those with global effect, the system follows a common interaction policy that ensures that a physical interaction (e.g., double left click) results in the same effect (e.g., expansion of a node) in all views.

To assist the user in interactively exploring the data and the parameter space of a visualization, CGV incorporates a history mechanism for undo and redo of interactions. Our implementation is based on the *command pattern* [133], which avoids holding duplicates of the data model in memory.

To ensure system responsiveness, CGV implements a threading component that is responsible for handling asynchronous computations. It has been designed as a worker queue that allows for the necessary level of control to handle synchronous interaction requests. However, there is no general rule to decide this. It is the task of the

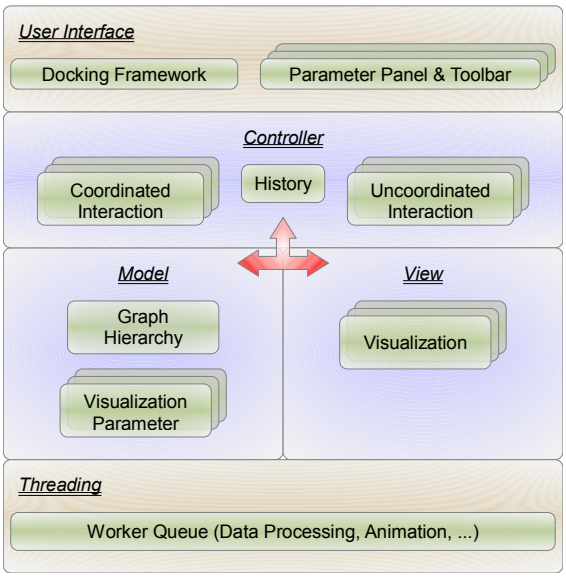


Figure 6.1: The Model-View-Controller architecture of CGV.

developer to incorporate proper synchronization points to allow for intermediate feedback and cancelation of long running computations. Careful implementation is necessary to avoid race conditions and deadlocks. This non-trivial task is currently not well supported [226]. The class `SwingWorker` of Java 1.6 provides some assistance – this is the main reason for using this most recent version of the Java language.

To target a broad user base, we developed a dual-use system that can be run as a stand-alone application or as an applet from a web browser. As recent developments like for instance ManyEyes [357] have shown, web-based systems are a good choice to reach users and foster collaboration. Figure 6.1 summarizes the architecture of CGV. Its key elements are:

- Threading (worker queue for asynchronous computations)
- Model (graph hierarchy and visualization parameters)
- View (various visualization techniques)
- Controller (data and views interaction)
- User interface (docking framework, parameter panels, and toolbars)

6.3 DATA MODEL

CGV operates on data organized as a *multivariate graph* $G = (V, E)$ where $V \subseteq \{A_V^1 \times \ldots \times A_V^n\}$ is a set of attributed nodes and $E \subseteq (V \times V) \times \{A_E^1 \times \ldots \times A_E^m\}$ is a set of attributed edges. $A_V^i : 1 \leqslant i \leqslant n$ and $A_E^i : 1 \leqslant i \leqslant m$ are domains of *node attributes* and *edge attributes*, respectively. The attributes can encode quantitative or qualitative values. Some of them may be computed by the system itself (i.e., degree of a node or flow on an edge) or may come from external sources (i.e., a textual label). Conceptually one can extend attributes to subsets of nodes and edges. In the case of numerical attributes some form of aggregation function provides a way to associate a meaningful value to a node or edge set (i.e., sum, max, min, average, mean, centrality, eccentricity, etc.). In the case of qualitative string labels the computation required to obtain such group labels varies in complexity depending on the application.

Since interactivity is at the essence we rely on methods presented in [13] and [15] to handle computationally and visually large graphs. The main data structure used is a *graph hierarchy* H over the nodes of G, i.e., a rooted tree whose set of leaves is in one to one correspondence with the nodes of G. Non-leaves of the graph hierarchy are sometimes called macro-nodes or cluster nodes. Central substructures in a graph hierarchy are *maximal antichains*, which are "cuts"

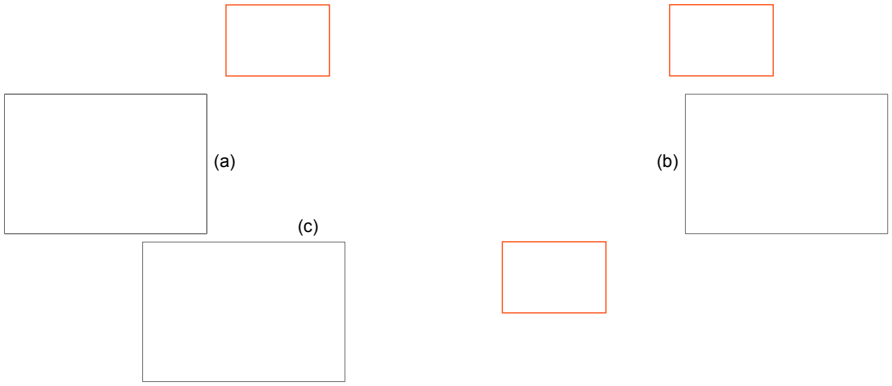


Figure 6.2: Different edge representations in the main graph view – (a) Plain edges; (b) Proportional edges; (c) Bundled edges.

through the hierarchy tree, or put differently, they are maximal collections of hierarchy nodes such that no pair is a descendant of the other. Each maximal antichain represents a partition $< V_1, \ldots V_k >$ of V. It determines a *macro-view* of G with k nodes where node i represents the set $V_i \subseteq V$. Details on the construction of H and on determining "good" macro-views can be found in [15]. The system can work with pre-computed graph hierarchies as well, which is important in certain application scenarios.

6.4 VISUAL METHODS

The different aspects of graph hierarchies we address here require multiple views on the data. We need views to communicate the general macro structure of G, dedicated tree visualization techniques to represent the graph hierarchy H, and overviews to preserve the user's mental map. Moreover, quantitative and qualitative attributes, such as weights, statistical meta data, or labels need to be displayed. In this section, we briefly describe purpose and characteristics of the proposed views. Most of them are instantiations of known approaches, so we refer interested readers to the original publications for more details.

6.4.1 *Main Graph View*

At any point in time CGV displays in its main canvas a node-link representation of a macro-view of the graph G (see Figure 6.2). Currently, we use classic spring embedders [47], squarified treemap layouts [65], and LinLog layouts [261] depending on the characteristics of the graph region being examined. To reduce computation time, layouts are lazily created, i.e., layout information is computed only for

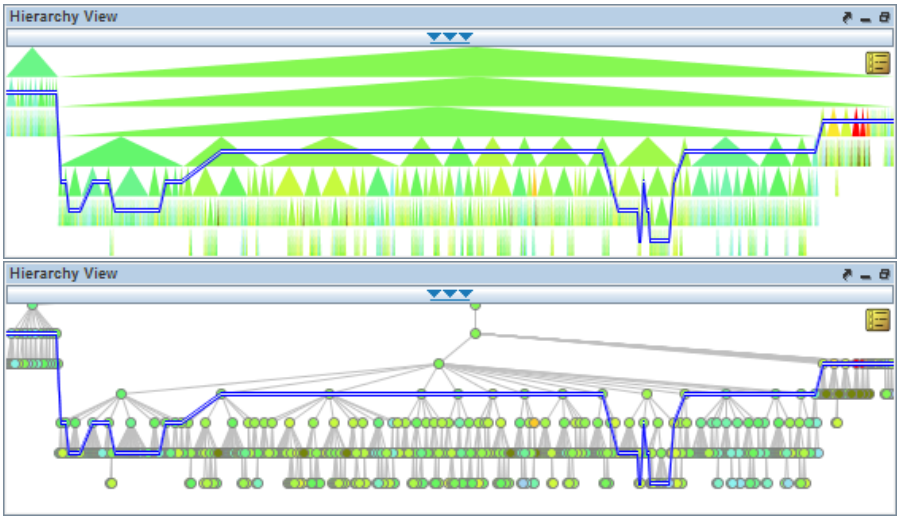


Figure 6.3: Two alternative representations of the hierarchy view.

those portions of the graph that are currently being explored. Users can reposition the displayed nodes to improve the layout interactively.

In this view, nodes are represented as spheres. Color and size of the spheres are used to encode numerical attributes on demand according to user specifications. Additional iconic shapes can be attached to the spheres to convey structural properties of the sub-graph induced by a node. Edges can be represented in three ways (see Figure 6.2). First, a classic straight line representation can be used to visualize connectivity. Second, CGV represents an edge as a line whose thickness at the source node and at the target node is proportional to the relative amount of edge weight that the edge contributes to the sum of all edge weights at the source node and the target node, respectively. This *proportional encoding* results in tapered edge shapes that indicate the "information flow" in a network. Third, edge bundling and convex hulls are used to de-clutter the view and to emphasize nodes cluster affiliation. String labels provide textual information (see Section 6.5.6). A novelty is that the displayed graph layout can be easily tiled for individual printing. These tiles can be manually assembled as a large poster for presentations or off-screen collaborative data analysis.

6.4.2 *Hierarchy Representations*

CGV uses the following coordinated views to represent the graph hierarchy H, which is the backbone for the navigation between different macro-views of G. These views give an overview of H, represent the current level of abstraction (i.e., maximal antichain), and visualize selected node attributes (qualitative or quantitative).

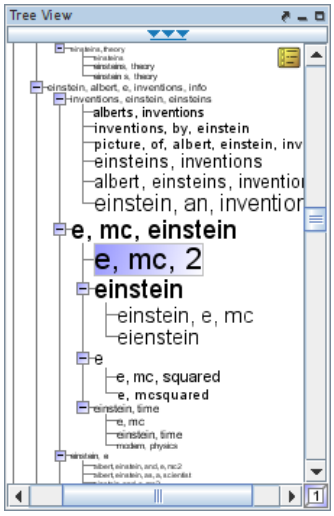


Figure 6.4: The textual tree view shows labels that can be optionally magnified by a fisheye transformation.

HIERARCHY VIEWS Two alternative representations of the overall topology of H are offered that follow Reingold-Tilford layouts [283]. One representation uses triangles the other dots and lines (see Figure 6.3). In both cases, a superimposed polyline indicates the current antichain. Color coding is applied to visualize a selected node attribute. While the triangle-based representation makes it easier to recognize colors and, hence, to grasp an overview of the value distribution, the mapping to dots and lines is better suited to convey structure.

TEXTUAL TREE VIEW This view (see Figure 6.4) complements the hierarchy view in that it presents textual labels of the nodes of H (which is not possible in the hierarchy view). The expanded and collapsed nodes in the view represent the current maximal antichain.

MAGIC EYE VIEW The magic eye view is based on a Walker layout [66] that is projected onto a 3D hemisphere. It was originally developed as a focus+context technique for hierarchies [218]. We extended the original design to visualize selected cross-edges among nodes of the hierarchy by representing them as arcs spanning around the hemisphere surface (see Figure 6.5, left). With this extension, the magic eye view represents both the graph hierarchy H and selected edges of a macro-view of the underlying graph G. As an alternative, users can also switch to a 2D representation of the radial tree layout (see Figure 6.5, right).

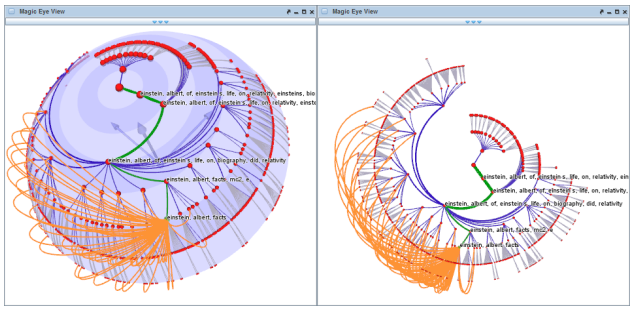


Figure 6.5: The magic eye view represents cross-edges in addition to the edges of the graph hierarchy in 3D or 2D.

6.4.3 *Complementary Overviews*

The aforementioned views are all suited to see details of nodes and edges of the hierarchy H and the underlying graph G. To help users maintain an overall mental map of the data two additional representations are incorporated:

SPLAT VIEW This view represents the layout of a macro-view of the underlying graph G in condensed form as a color-coded map [352]. It is helpful in spotting quickly dense layout regions, even if the available screen real estate is limited (see Figure 6.6, left). As an alternative to the default rainbow color scale, users can opt to use a gray scale encoding to avoid the disadvantages inherent in rainbow color scales [62].

MATRIX VIEW Alternatively, CGV uses a matrix representation to give an overview of the edge weight distribution of a macro-view of G (see Figure 6.6, right). The weights are color-coded in the matrix cells using a yellow to red color scale; gray is used to color those cells where no corresponding edges exist. The main diagonal cells are used to color-code node characteristics, which in the case of clustered graphs may be taken to represent edge density in the cluster. This helps users in spotting dense clusters that may be interesting to explore next.

6.4.4 *Remarks on Views*

This section provided an overview of the different views available in CGV. In addition to these views, which all represent connectivity and edge/node attributes the system provides several meta views. They contain meta information like statistics about the data set, detailed textual descriptions of node properties, search results, or the current system status. Meta views provide users with assorted context information that assists in the exploration process.

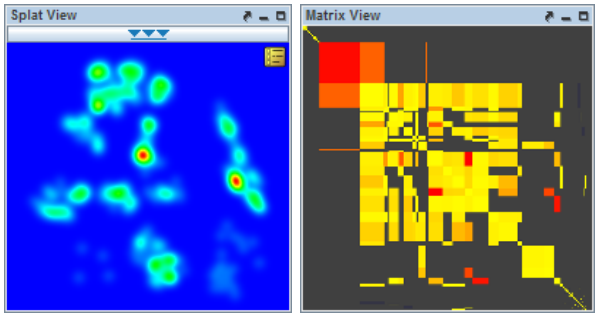


Figure 6.6: The splat view (left) provides a condensed overview of the graph layout. The matrix view (right) visualizes edge weights and edge density.

It is worth mentioning that the graph view, the hierarchy view and the textual tree view are endowed with fisheye-based focus+context representations that allow users to access detailed information quickly without loosing the overall context [330]. This capability greatly enhances readability of colors and labels and it is particularly useful for browsing visually large data sets. The magic eye view is by design a focus+context technique.

The architecture of CGV enables different assemblies of views (see Section 6.5.7). This mechanism can be used to provide customized view arrangements depending on the application context and the visualization task.

6.5 INTERACTION

For visual data exploration, interaction is a necessity. Since explorative analysis tasks can be many-faceted, an a-priori decision for a particular visual mapping is impractical. Well-designed interaction helps users choose relevant data subsets and adjust the visual mapping to suit a particular course of visual exploration. To that end, CGV provides several ways of interacting with the data and their visual representations.

6.5.1 *Coordinated vs. Uncoordinated Interaction*

Our system distinguishes between coordinated and uncoordinated interaction. Uncoordinated interactions are those that are local to a particular view. For example, there is no need to coordinate changes of visualization parameters (e.g., color scale or font size) across views, because each view is monolithic and implements its own mapping strategy and has visualization parameters that are independent of other views.

In contrast, coordinated interactions are those that change the global perspective on the data, that is, all views are consistent in what they

represent (but not how). Operations on the data model like filtering graph elements or choosing a different macro-view are examples of coordinated interactions. CGV implements coordinated interactions as follows. When a user performs an interaction in a view, the request is propagated to a controller that in turn notifies all other views of the pending operation. This gives all views the opportunity to take any actions required to prepare for the pending interaction. After that, the controller performs the interaction on the data model, which might include scheduling costly calculations on the worker queue (e.g., layout computations). Once the work is done, the controller informs all views about the particular change, the view having initiated the interaction being informed first. Each notified view updates itself to react to the change in the data model. Again, this can involve computations that need to be asynchronous (e.g., layout generation or animation). Given the multitude of views that can initiate this procedure, it is necessary to follow a consistent common interaction policy. For instance, a double left click on a node should result in expansion of the node, no matter in which view the double click was performed.

6.5.2 *Basic Interactions*

In any visual interactive system we can think of, interaction presupposes some form of object selection [378], together with mechanisms to temporarily lock a view focus and to coordinate data and visual operation updates across multiple views. Next we present a more or less high level description of fundamental CGV interactions. To convey an impression of how these basic interactions can be applied by users, they are presented in an order that resembles a real usage scenario, rather than in an order that relates to interaction complexity.

ZOOMING, PANNING, AND FISHEYE MAGNIFICATION One common interaction is to adjust the viewport to a level of graphical abstraction that suits the task at hand. This includes zooming and panning operations as well as scrolling. In situations where only a quick view on details is sufficient, users can shortcut viewport adjustments by using dynamic fisheye magnification techniques (in the hierarchy view, the textual tree view, and the graph view).

IDENTIFY Once users have found suitable viewports they usually want to *identify* the visualized data objects. This can be accomplished by hovering the mouse cursor over visual elements in any view. The identified object is highlighted in all visualizations and its detailed information is shown in a dedicated meta view.

LOCATE Making an identified data object visible in all views is cumbersome if this has to be done by adjusting all viewports sep-

arately. As a novel alternative to manual viewport adjustment, we implement the *locate* interaction, which is activated by a left click. Its purpose is to broadcast to all the views the fact that a data object has become the focus of interest in a particular view. When a view receives notification of this type of interaction it is obliged to adjust itself so that the involved data object becomes visible. This operation is particularly useful in cases when users spot an "interesting" element in one view and want all other views to automatically focus on the same element.

LOCK/UNLOCK To help users stay focused on an identified data object, it is possible to *lock* it (CTRL+left click). In lock mode the focus is fixed and any request to change the focus is neglected (i.e., hover is deactivated) until the lock is released by an *unlock* operation (CTRL+right click). While being in lock-mode, CGV can provide further interactions that work only with the fractional size of the focus, but would be hard to apply to all or even a subset of the data objects. One such interaction is edge-based traveling, which we describe later.

BRUSHING Setting a primary focus on a single data object via locking is complemented by *brushing*, which can be used to set a secondary focus on multiple data objects by holding SHIFT and using the left mouse button (click or drag). Since there is no restriction on the number of brushed elements, CGV highlights them sparingly to avoid cluttering (i.e., using little screen space and few visual attributes).

EXPAND/COLLAPSE Since CGV's basic data structure is a graph hierarchy, *expand* and *collapse* can be used to interactively change the macro-view of the graph, i.e., to control the information drill-down. The expand operation (double left click) provides access to more details by replacing a selected node with a layout of its induced subgraph. Collapse is the inverse operation (double right click) and can be used to get back to an overview.

VISUALIZATION PARAMETER ADJUSTMENT All views provide some mechanism to adjust visualization parameters either by direct manipulation (e.g., mouse wheel rotation) or by entering values in a user interface. In Section 6.5.7, we discuss the interface facilities that CGV offers in this regard.

In contrast to visualization parameter adjustments (which are uncoordinated), identify, locate, lock/unlock, brushing, and expand/collapse are coordinated interactions. All these basic interactions are supported by all views. In the previous descriptions, we also indicated the common interaction policy that all views should follow: The left mouse button is associated with "positive" interactions (e.g., identify, locate, lock, brush, expand), while the right mouse button handles

"negative" interactions (e.g., unlock, de-brush, collapse). This distinction has been made to help users apply the interactions consistently, even though the views are quite different.

6.5.3 *Dynamic Filtering*

When exploring a graph with respect to certain attributes, it can be very helpful to dynamically *filter* out irrelevant nodes and edges [19]. Depending on view characteristics and visualization tasks, two alternatives exist to display filtering results: filtered objects can be dimmed or they can be made invisible. Dimming objects is useful in views that maintain an overview, where all information needs to be displayed at all times, but filtered objects need only to be indicated. Making objects invisible is useful in views that notoriously suffer from cluttering.

BASIC DYNAMIC FILTERING Elementary filters are commonly applied to describe conditions that must be satisfied for an object to pass through. Range sliders are an effective mechanism to filter by any particular numerical attribute. These include exogenous data or numerical attributes computed by the system like degrees, peeling numbers, or weights. In addition to range sliders, CGV provides a textual filter that extracts objects with specified labels. Both basic filters allow users to specify manifold filter conditions by simply moving a slider or typing letter strings. Further elementary filters can be integrated if required.

ENHANCED DYNAMIC FILTERING For complex data sets where nodes and edges are associated with a multitude of attributes, relying solely on elementary filters is not sufficient. The next natural step is to combine elementary filters to provide some form of multidimensional data reduction. The usefulness of this approach is discussed in [99]. In CGV, composite filters can be created by logically combining elementary filters. A logical AND combination generates a filter that can be passed only if an object obeys all conditions. An object passes a logical OR filter if it satisfies any of the composed filter conditions.

The question that needs to be answered is how to enable the user to create composite filters dynamically during runtime. While other systems offer only fixed filter combinations or require users to enter syntactic constructs of some filter language, CGV implements a visual interface where the user can visually specify logical combinations of filters. The interface can be operated in two modes – basic and advanced. In basic mode, only the elementary filters can be adjusted, whereas the structure of the filter (i.e., the logical combinations) is fixed (see Figure 6.7 (a)). In advanced mode, the structure of a filter

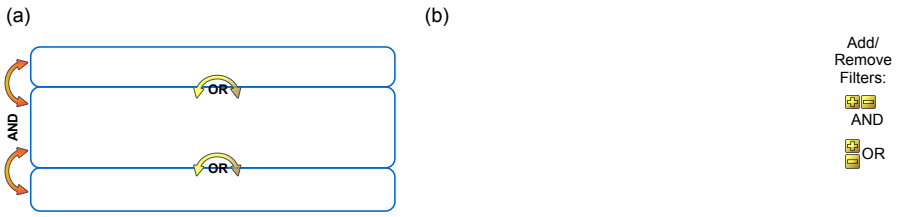


Figure 6.7: Enhanced dynamic filtering via a logical combination of five elementary filters – (a) Basic mode; (b) Advanced mode.

is subject to change (see Figure 6.7 (b)). This allows users to compile more complex filters, which can be conserved for later reuse once they have proven useful. The advanced mode is kept at a steerable level by following the sieve metaphor, according to which objects have to pass the filters from top to bottom. The sieve is modeled as a single logical AND combination of an arbitrary number of logical OR combinations, which in turn contain arbitrary elementary filters. The elementary filters involved in an OR combination are arranged horizontally, indicating that an object can pass any of them. The AND filter is represented as vertical arrangement of the participating OR combinations, indicating that an object has to pass all of them. Note that the restriction to the sieve metaphor only affects the user interface. Internally any logical combination of filters is possible.

Figure 6.7 shows an example of a filter that can only be passed by nodes that have an attribute value named mrv in the interval $[-1.5; -1.0]$, and that contain in their labels the terms "bohr" or "einstein", and that have a $degree$ between 11 - 17 or a node weight (wv) between 14 - 34:

$$\begin{gathered} mrv \in [-1.5; -1.0] \wedge \\ \text{"bohr"} \subset label \vee \text{"einstein"} \subset label \wedge \\ degree \in [11; 17] \vee wv \in [14; 34]. \end{gathered}$$

As an illustration of the flexibility of these filters consider the exploration of time varying multi-graphs. Browsing with respect to time is a typical task when exploring such data. By modeling time as an attribute one can associate a range filter to it and use a corresponding time slider to obtain a sequence of time varying snapshots of the graph's evolution (an implementation of a dedicated time slider is not necessary).

6.5.4 *Graph Lenses*

Dynamic filtering provides a means to globally adjust what is being displayed in CGV's views. For more local interaction, we apply interactive lenses. Although lenses are recognized as useful tools to

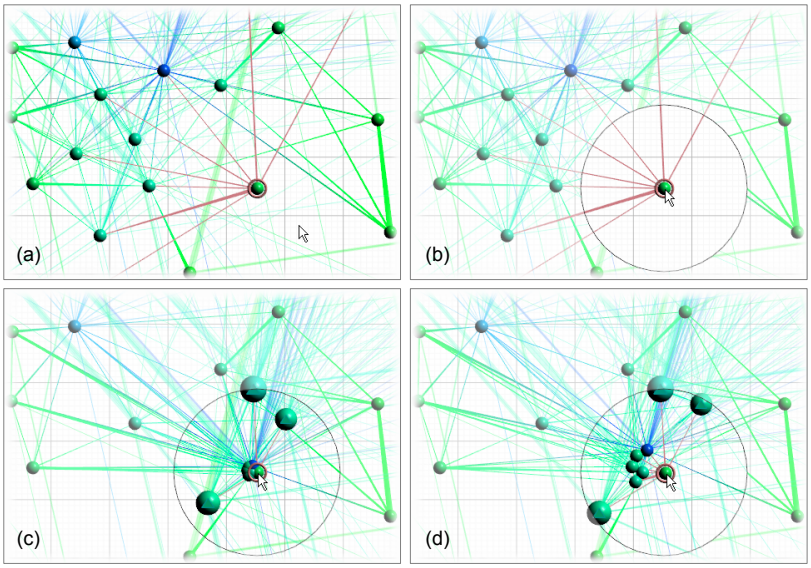


Figure 6.8: Interactive graph lenses – (a) View with a focused node; (b) The local edge lens removes edge clutter; (c) The layout lens gathers nodes that are adjacent to the focus node, but might be scattered in the layout; (d) The composite lens combines (b), (c), and a fisheye lens to tidy up the lens area, gathers relevant nodes, and spreads those that get accumulated in the lens center in a single operation.

support visual exploration [58, 324], they have been considered only recently for graph visualization [330]. In CGV, lenses are used in the main graph view to locally tidy up layout regions that are cluttered and to generate local overviews. They can be switched on/off by a single click and can be moved around using drag and drop.

LOCAL EDGE LENS A lens that operates on the rendering stage is the *local edge lens* [330]. It is used to unveil the connectivity of nodes when the visual representation is cluttered with edges. To tidy up a lens perimeter, we consider as relevant only those edges that are adjacent to nodes inside the layout area covered by the lens. During rendering we apply stencil buffering to allow only relevant edges to be displayed in the lens view. Compare Figure 6.8 (b) to Figure 6.8 (a) to see the effect of this lens.

LAYOUT LENS Consider the example depicted in Figure 6.8 (b). The application of the local edge lens clearly reveals that the focused node is connected to eight neighbors. However, only five of them are visible in the current viewport. To discern the characteristics of the other three neighbors, the user would have to navigate to each of them manually – a costly procedure that is common in known graph visualization systems. To address this and similar exploration tasks, CGV provides the *layout lens*, which is a generalization of the bring-

neighbors-lens introduced in [330]. Its purpose is to adapt the graph layout to gather nodes that exhibit certain similar characteristics, but that might be scattered in the layout (e.g., as the set of neighbors in Figure 6.8 (b)). When applying this lens, nodes change their position according to the lens position and the lens perimeter. While moving the lens towards a focus node, the affected nodes are attracted towards the lens. When the lens reaches the position directly above the focus node, all affected nodes become located within the lens perimeter, with the node originally farthest away now on the lens perimeter. Figure 6.8 (c) shows that this generates a local overview of the focus node and the affected nodes (including the three previously unknown neighbors). While the focus node is simply defined as the node nearest to the center of the lens, affected nodes can be determined in different ways. Currently, we limit ourselves to affect nodes that are neighbors of the focus node in the current macro-view. In this sense, our current implementation can be termed topological, but one could think of extending the current capabilities to provide semantic lenses based on similarity of nodes as long as smooth interactivity is maintained.

FISHEYE LENS A lens that is useful to temporarily separate local accumulations of nodes and to reduce node clutter is the *fisheye lens*. It operates on the node positions only, not on node sizes.

COMPOSITE LENS The layout lens can sometimes accumulate too many nodes in a local region (see Figure 6.8 (c)). The *composite lens* resolves this problem by superimposing the effects of the local edge lens, the layout lens, and the fisheye lens in one operation. It tidies up the lens area, gathers relevant nodes, and spreads those that get accumulated in the lens center as depicted in Figure 6.8 (d).

The on-demand character of the described lenses make them an interesting alternative to resolve visibility issues (i.e., edge clutter, out-of-view objects, and node accumulation) in cases where classic zoom and pan are impractical or cumbersome to apply. CGV's architecture allows to programmatically place lenses in the graph layout, for instance to emphasize interesting data elements.

6.5.5 *View Space and Data Space Navigation*

Navigation of the view space is an essential task during visual exploration. A fairly standard approach not only in graph visualization is to let users grab a view and drag it as necessary. Though being a very natural and intuitive interaction, it implies higher physical costs in terms of mouse mileage [151]. CGV alleviates these costs by providing alternative navigation methods: *pan-wheel navigation* for the view

space and *edge-based traveling* as a dedicated data space navigation method.

PAN-WHEEL NAVIGATION In contrast to common pan or scroll operations, for which users know the specific position to where they want to go, the *pan-wheel* interaction allows users to travel the view space freely when no particular graph region needs to be reached. This opens up the possibility of spotting "interesting" areas for further inspection via zooming into more specific regions. Pan-wheel navigation is activated by simply holding the left mouse button and dragging the mouse to specify direction and speed of the navigation. The pan-wheel navigation can optionally be augmented with a radar view as described in Section 6.5.6. As interesting aspect for future work will be to integrate speed-dependent zooming, which has shown to be effective for larger view spaces [83].

EDGE-BASED TRAVELING Current graph visualization systems offer only limited support for data space navigation. Most systems map the task of data space navigation to one of view space navigation. But graphs offer a well-defined structure, which should be utilized for data space navigation. CGV follows this idea and provides a navigation method called *edge-based traveling* [16]. It operates on the structure of the currently explored macro-view of G and allows users to explore the macro-view's structure by performing a series of simple mouse clicks. To activate edge-based traveling, the user has to lock on a node u. After that it is possible to click any edge adjacent to u. From then on there are two options to follow: travel or preview. The user can left click an edge to travel to the node v that is connected via the clicked edge (u, v). This is implemented as a composition of the following basic operations: unlock u, locate v, and lock v. The first unlock operation is a prerequisite. The locate operation causes all views to center on v. The final lock operation ensures that edge-based traveling can proceed further from the just focused node v. Since not all nodes that can be reached via edge-based traveling are necessarily visible in the graph view, users sometimes do not know where they will be traveling when clicking an edge. In such cases, the preview mode helps. Users can right click an edge (u, v) to center it in the view so that both u and v become visible. In the preview, users can decide to actually perform edge-based traveling via a left click or to abort the preview and return to the original view by right clicking a second time. Both preview and traveling are augmented with smooth viewport animations (see Section 6.5.6). This edge-driven navigation is a novel interaction mode that to our knowledge has not been incorporated in previous systems. In combination with appropriately set filters and lenses, it is a useful tool to explore large graph layouts in concert with very localized data characteristics.

6.5.6 *Visual Augmentation*

Interactions can encapsulate complex semantics, which are not immediately apparent to users. To make CGV's interactions more understandable and efficient to use, they are augmented with visual cues. The goal of this section is to illustrate the potential of such visual cues.

USE OF ANIMATION Animation is a helpful tool to support the user's mental map [160, 276]. We apply two kinds of animations. One is view-centric and augments the navigation in the view space; the other is data-centric and augments the transition between different levels of abstraction.

Viewport switches are caused among others by zoom and pan operations, locate interactions, and edge-based traveling. When interacting with large graph layouts, viewport switches can be difficult to comprehend. CGV's graph view applies smooth viewport animation following the method described in [355]. During animation the viewport is parametrically interpolated between origin and destination. It is simultaneously slightly enlarged to facilitate the overview. The length of the animation is determined by the distance between origin and destination.

The data-centric expand and collapse interactions change the level of abstraction by exposing new nodes or removing some of them. Since this changes the layout of nodes and edges significantly, smooth animation is applied in the textual tree view and the graph view. This is implemented as an interpolation between the position of a newly exposed node and the position of its parent node. Expand animations appear as if nodes emanate from their parent, whereas collapse animations work the opposite way. These animations are time-bound, that is, they are finished within an assured time frame. We figured 500ms and 2s to be good time frames for the textual tree view and the graph view, respectively. However, these values heavily depend on user preferences, and thus, are subject to parameterization.

Both kinds of animation can be interrupted by the user at any point (recall synchronous interaction vs. asynchronous animation). While expand and collapse animations immediately switch to the final result upon interruption, the viewport animation just stops. This gives users the option to pause if they spot something interesting during the animation.

While animation in a single view is a well-investigated solution, a question remains for multi-view systems: Can animation really be useful if we animate multiple views simultaneously? Users could hardly follow all changes, and animation could lead to confusion rather than support [344]. Therefore, all animations in CGV are queued and played one after the other. The view that has been the source

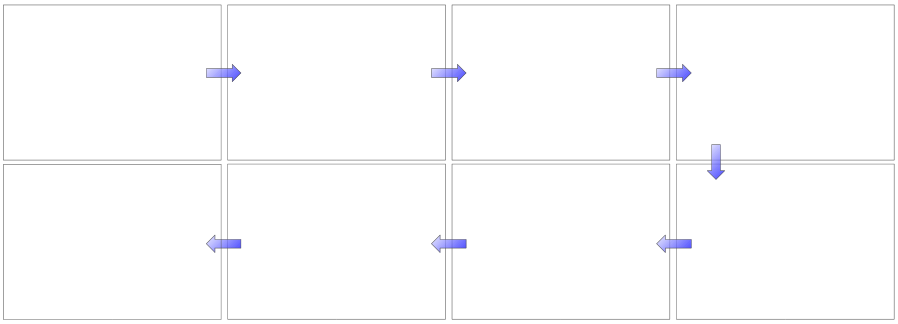


Figure 6.9: The infinite grid – The image series shows how the grid changes when zooming into a cluster of nodes.

of an interaction (i.e., is in the user's focus) is animated first. Possible extensions of this very basic animation scheduling could account for proximity and sizes of views to prioritize larger neighbors of the source view and neglect animations of distant views, which can be necessary for a larger number of views.

INFINITE GRID CGV's main graph view visualizes the general structure of a macro-view of G and uses node size to encode a selected node attribute. To maintain expressiveness, node sizes are kept fixed during zooming operations. However, this impairs perception of closeness and distance of nodes in the layout during zooming, because size is an important visual cue to judge them. To overcome this disadvantage, we use a novel grid display termed *infinite grid* that provides a form of visual reference for the perception of distances. An ordinary grid is not sufficient, since the layout coordinates span several magnitudes (due to incremental layout computation). The infinite grid shows a primary grid for the magnitude of the current zoom level. A secondary grid is sketched in a dimmed color to indicate the next lower magnitude. When the zoom level is increased the secondary grid smoothly takes on the fully saturated color of the primary grid. At a certain point, the secondary grid fully resembles the primary grid and replaces it, and a new very dimmed secondary grid appears (see Figure 6.9). This procedure works analogous when decreasing the zoom level. Our current users found the infinite grid particularly helpful to judge distances during viewport animations, which perform a smooth change of the zoom level.

RADAR VIEW We introduced the pan-wheel navigation as a method for navigating in CGV's main graph view. A weak spot of this approach is that users are unaware of what they might discover during the course of the navigation. To address this concern, pan-wheel navigation is augmented by providing users a novel look ahead guide of what is coming next in the current travel direction. Look ahead

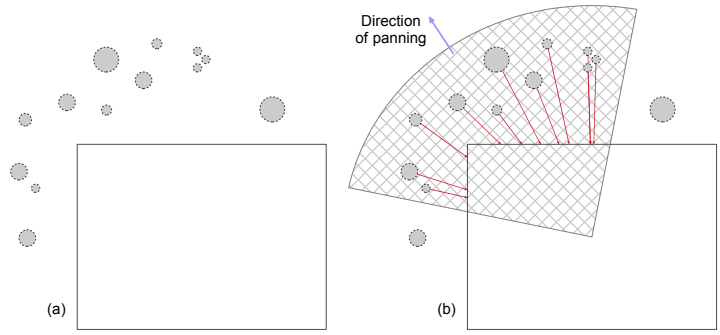


Figure 6.10: The radar view is designed as to project off-screen nodes to the viewport. This provides guidance during pan-wheel interaction toward possibly interesting spots in the data. (a) Navigation without radar view; (b) Navigation with radar view.

guidance has recently been suggested for visualization on mobile devices [146], but has not yet been proposed for graph visualization.

What we call the *radar view* [333] shows a semi-transparent circular sector that emanates from the center of the view to infinity and that is aligned with the selected navigation direction of the pan-wheel. The sectors's central angle can be parameterized. We currently use a fixed angle of 90°. Dynamically changing the angle depending on traveling speed is also feasible, but it is not currently implemented. During pan-wheel navigation, all nodes that fall into the circular sector, but are currently not within the viewport, are projected to the borders of the viewport so that they become visible. The different sizes and colors of the projected nodes provide a look ahead mechanism of what will be found in the current navigation direction. Users can use this as a guide to adjust their travel direction with the pan-wheel. The radar view then adjusts itself accordingly. Figure 6.10 illustrates pan-wheel interaction with and without the radar view.

LABELING Labeling is an important visual cue. It helps users establish connections between visualized data characteristics and particular data objects identified by some textual labels (e.g., the larger red-colored node (characteristics) represents "Einstein's work" (object label)). This is one of the reasons for treating string labels as first-class objects in the data model and in the system views.

In particular, labels are always embedded explicitly in a visual representation if they do not occlude the view. The textual tree view representation of the graph hierarchy H can be seen as a complete taxonomy of the underlying graph data in label space. Coordinated linking of any view with the textual tree view together with the text search facility makes the connection between the displayed data objects and their string labels quite apparent. CGV strictly adheres to the policy of always highlighting node labels when corresponding nodes are activated in any view.

Because avoiding occlusion among labels and between labels and other visual features is a computationally complex task, the main graph view and the magic eye view display labels in a selective manner only. The magic eye view shows labels for the currently highlighted node, and only optionally for its ancestors or for nodes connected via cross-edges. This keeps labeling costs at a manageable level. The graph view follows a global labeling strategy, that is, it tries to label as many nodes as possible. If "enough" display space is available, labels get attached to nodes directly. Otherwise, it attempts to label ancestors with aggregated labels (visually differentiated by font type and size, see Figure 6.2). This simple strategy accounts for label density, i.e., the number of labels per available screen space. The net effect is that overviews show few aggregated labels only. During zooming, the labeling is refined and labels are displayed in an incremental fashion according to the zooming level.

6.5.7 *User Interface*

Interactive visualization using multiple views requires a well designed user interface. This includes both, the arrangement of views on the display and the interface to allow users to parameterize views.

The integration of visualization parameters in CGV's data model allows us to export them via two different user interfaces (see Figure 6.11). One is a panel that lists all available parameters using standard widgets (e.g., buttons, sliders, etc.). Users can adjust parameters to their needs and optionally apply the changes to get visual feedback. We refer to this as asynchronous parameterization. But there are certainly parameters that are used more often than others during interactive visual exploration. To ease the adjustment of important parameters, we provide instant access via a toolbar that processes requests for parameter changes immediately; we call this synchronous parameterization. Recently, it has been shown that duplicating important functionality from an all-encompassing control panel to an exposed position (e.g., a toolbar) is a useful way to drive adaptable user interfaces [132]. In CGV, the toolbar can be considered to be adapted to the common user, whereas the panel for setting all visualization parameters is for advanced and expert users only.

When it comes to arranging multiple views on the display, there are two extremal positions one can take. One is to use a fixed arrangement that has been designed by an expert and has been proved to be efficient. The other is to provide users with the full flexibility of windowing systems, allowing them to move and resize views dynamically. Both extremes have their pros and cons and are actually applied in the literature, e.g., [15] for fixed arrangement and [40] for full flexibility. We do not go for either extreme, but instead make use

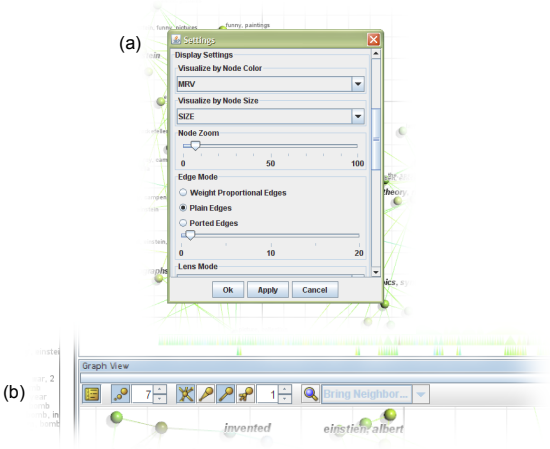


Figure 6.11: User interface for visualization parameters – (a) All-encompassing settings panel; (b) Toolbar for quick access to important parameters.

of a docking framework[1]. The main reason is that docking maintains flexibility, but imposes certain order in terms of what arrangements are possible. Preferably, views should not overlap partially; a view should either be visible or not. To achieve this, the available screen space is partitioned into regions, each containing one or more views. The regions can be resized and moved as long as the arrangement remains a partition, i.e., remains overlap-free. A region that contains multiple views provides an interface to switch between them (usually tab-based). Partially visible views can thus be avoided. Figure 6.12, 6.13, and 6.14 (see last pages of this chapter) show possible arrangements of the views provided in CGV. The docking framework we use also allows for predesigned arrangements. This is a requirement for user-centered and task-oriented visualization. By default CGV uses an arrangement of views that is suited for visual exploration, which is similar to that in [15]. We have also experimented with other view arrangements for specific tasks or data sets, as for instance for browsing through time-varying graphs. Besides relying on pre-designed view arrangements, experts or users can design their own customized ones and store them for later reuse. This is a useful feature that allows for easy adaptation of CGV to domain- and task-specific needs.

6.5.8 *Interaction History*

As described in [305], keeping a history of user interactions supports iterative visual exploration. This allows users to undo operations that were performed by accident or that yielded unsatisfactory results. Even though history mechanisms have long since been recognized as important in the HCI community (e.g., [17]) and are gaining im-

1 See `www.infonode.net` for details.

portance in the visualization community (e.g., [95, 219, 188]), they are rarely present in current graph visualization systems.

We provide for basic undo and redo to close this gap. A history manager enables us to keep track of interactions and to undo them. Undone interactions can be redone, unless a new interaction is executed, which causes the history manager to dispose of all previously undone interactions. This behavior resembles a linear history as provided in many computer applications (e.g., web browsers).

Currently, CGV's history mechanism supports commands that are deterministic and do not require a copy of the whole data model to undo them (e.g., brushing as well as expand and collapse operations (see Section 6.5.2). Synchronous parameterization performed via the toolbars of CGV's views (see Section 6.5.7) is logged and can be undone. This helps users tune different parameter settings that best suit their visualization task.

In summary, CGV supports visual graph exploration by integrating standard and enhanced interaction techniques. Some of the interactions introduced in this section can be applied in other visualization contexts (e.g., the user interface to create advanced filter queries during runtime, the radar view as a look-ahead guide for off-screen objects, or the infinite grid as a spatial reference). We compare CGV to selected graph visualization systems in the next section.

6.6 RELATED WORK

A wide range of graph visualization software has been and is still being developed. These include *general graph libraries* like JUNG (Java Universal Network/Graph Framework) [6] or JGraphT [5], which focus on algorithms and data structures, but consider visualization more or less as an add-on. Dedicated *graph visualization tools* like Walrus [11], HyperTree [4], or Vizster [159] focus on the implementation of a single visualization approach. *Multiple-view systems* like Improvise [370] or Tableau [9] integrate several visualization approaches, but are often focused on data visualization, rather than graph visualization. More general *information visualization frameworks* like the InfoVis Toolkit [117] or Prefuse [162] (and recently VTK [381]) provide the platform to develop graph visualization software, but they are not self-contained ready-to-use systems.

Given the wealth of software out there, the reader might ask what makes CGV special or in what aspects does CGV differ from other systems. It is beyond the scope of this work to compare CGV with an exhaustive list of visualization software. Instead, we elaborate on the strengths and weaknesses of CGV compared to selected approaches keeping our focus on graph visualization systems and interactivity.

We decided to compare CGV to four selected systems: ASK-GraphView, Tulip, Pajek, and GUESS. All these systems support the visual

exploration of graphs. However, they are very different in terms of system architecture (ASK-GraphView and Pajek: monolithic architecture; Tulip: plug-in system; GUESS: scripting language), implemented views (single view vs. multiple views), and offered interaction.

ASK-GraphView [15] is the direct predecessor of CGV. The system has been developed with a focus on large graphs and, hence, employs sophisticated data structures and data handling mechanisms. A fixed arrangement of several linked views allows for easy graph exploration. Basic search, navigation, and filtering functions are offered to the user.

Tulip [40] is an open source visualization system that is capable of showing different representations of graphs. The system is comprehensive in terms of provided layout and analysis algorithms. It is suited for experts to compare research results or to test new algorithms. Extensions can be easily integrated thanks to Tulip's plug-in mechanism. Tulip is not restricted to exploration tasks, but can also be used to edit a graph (i.e., add/remove nodes and edges).

Pajek [93] is similar to Tulip in that it provides several different layout algorithms. From the number of algorithms implemented in Pajek it is apparent that the system has a clear focus on graph analysis, including clustering and graph partitioning. The visualization facilities include node-link diagrams and simple view navigation.

The GUESS system [18] in its basic version provides a single node-link view of a graph and a command prompt to allow users to manipulate the visual representation in manifold ways. Behind the command prompt, GUESS employs a scripting language called Gython to drive the interaction. The embedded language is the core innovation of the system as it provides a very flexible and capable means to interact with the graph and its representation.

Since visual interaction is the major subject of our work, we will compare the aforemention systems to CGV from that perspective. To that end, we consider a list of interactions, which we derived from [385]:

1. *Select*: Mark something as interesting.
2. *Explore*: Show me something else.
3. *Encode/Reconfigure*: Show me a different representation/arrangement.
4. *Abstract/Elaborate*: Show me more or less detail.
5. *Filter*: Show me something conditionally.
6. *Connect*: Show me related items.
7. *Undo/Redo*: Let me go where I have been already.
8. *Change configuration*: Let me adjust the interface.

These interactions are supported differently by the systems Pajek, Tulip, ASK-GraphView, GUESS, and CGV, as we will see next.

1. SELECT All systems provide means for selection of nodes and edges, which are basically implemented as point-and-click interactions.

2. EXPLORE Pajek, Tulip, ASK-GraphView, and GUESS offer exploration in terms of navigation of the presentation space. Standard zoom and pan interaction are offered. In addition, CGV enhances view space navigation and, additionally, supports navigation in the data space. The pan-wheel in conjunction with the radar view or the locate interaction are useful for exploring the view space, whereas CGV's edge-based traveling is a novel way of navigating the data space, which is not possible in the other systems.

3. ENCODE/RECONFIGURE Pajek offers different layouts and allows for linking of data attributes to visual variables. As mentioned before, Tulip is comprehensive with regard to layouts and graphical encoding. Similarly powerful is GUESS, whose users can flexibly script the visual encoding by using the embedded language. A list of different layout algorithms is offered as well. ASK-GraphView and CGV are not that flexible. They only allow users to choose the node attribute to be visualized via color coding. Nonetheless, alternative arrangements of the graph are offered in the different views of both systems, but CGV provides more choices than ASK-GraphView.

4. ABSTRACT/ELABORATE The GUESS systems allows users to cluster nodes based on different metrics. Clusters can then be made distinguishable by applying different visual encodings (incl. convex hulls). The concept of super nodes that can be expanded or collapsed is not available. It is available in Pajek and Tulip. However, expand and collapse cannot be conducted in the main view by means of direct interaction, but must be performed in separate views. Nodes are not literally expanded or collapsed, but are rather made visible or invisible. In contrast to that, ASK-GraphView and CGV are based on the philosophy of expanding and collapsing nodes on demand. Expand/collapse are visually augmented in both systems by means of animation. Moreover, any view provided by CGV can be used to perform this type of interaction, which is a usability plus compared to the other systems.

5. FILTER Interestingly, neither Pajek nor Tulip offer filtering facilities. ASK-GraphView offers a basic filter slider to focus on nodes with specific properties. GUESS allows for filtering via its scripting language. A programming language allows for very complex filtering

conditions and manifold visual encodings of the filter results, which, however, must be typed at the command prompt. Yet, by extending the system, helpful user interface elements (e.g., sliders) can be incorporated. CGV provides an advanced filter mechanism that is based on logical AND and OR combinations of basic filters and an intuitive user interface. Filtered data elements are dimmed or omitted automatically.

6. CONNECT The connect interaction, which shows the user related items, is more difficult to grasp. Basically, all systems allow users to relate items via their clustering and filtering facilities. Tulip, ASK-GraphView, and CGV have multiple views that, through appropriate linking, can show related views of the same data. CGV provides additional methods: The layout lens is suited to bring related nodes to the viewport and edge-based traveling supports navigation between topologically neighboring nodes that appear distant in the layout.

7. UNDO/REDO Tulip, and ASK-GraphView do not provide a history mechanism. Pajek's log files and GUESS's command prompt offer a way to reuse or edit previously issued commands, which however are not history mechanisms in terms of undoing or redoing interaction. CGV is the only system that offers undo/redo in the classic sense.

8. CHANGE CONFIGURATION The views of Pajek, ASK-GraphView, and GUESS are subject to basic reconfiguration such as expand or resize. Tulip uses a mix of docking and freely adjustable windows. The docking framework applied in CGV is more flexible as it allows for arbitrary arrangements, including grouping of views (within the constraint of the docking paradigm) or task-based arrangements of views, which can be stored.

The previously made statements are summarized in Table 6.1. It is worth mentioning that Pajek, Tulip, and GUESS support an additional aspect of interaction, namely graph editing. In this sense, they are not only graph visualization systems, but visual graph editors. This is an aspect that would be interesting to pursue in the future development of CGV. In general, we found that other systems and tools described in the literature do not focus on interaction as much as CGV does. In particular, advanced navigation methods, history mechanisms, view arrangements (e.g., via docking), or combinations of 2D and 3D views are scarce. Moreover, the lack of web-accessibility is a limiting factor of some of the existing software.

	Pajek	Tulip	ASK-Gr.V.	GUESS	CGV
Select	+	+	+	+	+
Explore	○	○	○	○	+
Encode/Reconfigure	○	+	○	+	○
Abstract/Elaborate	+	○	+	○	+
Filter	–	–	○	+	+
Connect	–	○	○	○	+
Undo/Redo	–	–	–	–	+
Change configuration	–	○	○	○	+
Edit	○	○	–	○	–

Table 6.1: Comparison of different graph visualization systems with respect to offered interaction facilities (+, ○, and – stand for advanced, basic, and limited/no support).

6.7 EVALUATION DESIDERATA

Evaluation of visualization and interaction techniques in information visualization is a challenging task [29, 105]. For systems that incorporate many different visualization and interaction techniques, the challenge is even bigger. This is due to the multitude of parameters that influence the evaluation, a fact that raises questions like: Which parameterization should be tested for which visualization?, Which combinations of visual and interaction methods should be evaluated?, Which tasks and user groups should be addressed?, and so forth. These are questions that we are not ready to address in this writing.

Our current approach is very rudimentary, i.e., "Deploy the system and get feedback". For the continuous evaluation of CGV we are employing a three-fold strategy. (1) Early focus on users, empirical measurements, and iterative design [142]. (2) Expert interviews are conducted to evaluate CGV in the context of real world application scenarios [340]. (3) We provide access to CGV over the Internet and are preparing to collect feedback from a broader audience. This type of evaluation can yield valuable results [209].

6.7.1 *Users Feedback*

Next, we discuss feedback collected from early adopters of CGV mainly concerning the user interface and interaction techniques. The different visualizations were not a major issue since they are mainly well-accepted approaches from the literature. The feedback of users influenced the development of the system in several aspects.

VIEWS ARRANGEMENT In the early stages, users complained about the fixed arrangement of views. Even though they were able to resize the views independently, instant maximization of a view or rearrangements were not possible. To overcome these difficulties we incorporated the docking framework as described in Section 6.5.7.

PARAMETER SETTINGS Another aspect was the setting of visualization parameters. As the views in CGV became more and more complex, the number of visualization parameters increased and the panel to set the parameters grew. Users found it difficult to locate important parameters and demanded easy access. This prompted us to incorporate tool bars to allow instant adjustment of the most important parameters on a per view basis (see Section 6.5.7).

ZOOM DIRECTION AND CENTER Some design flaws were noticed regarding some of the interaction mechanisms offered by CGV. An example is the zoom operation performed via the mouse wheel in the graph view. The direction of zooming was opposite to the intuition of some users. Interestingly, other users had no problems with the zoom function. Only recently, we found evidence that both directions of zooming are appropriate, depending on the mental model of the user [143]. So we added a parameter that allows users to control the direction of zooming operations. Moreover, when zooming in, we considered the current location of the mouse cursor as the center of the zoom operation. Since this was counterintuitive for some users, we introduced a zooming mode that takes the center of the viewport as the center for the zoom operation, rather than the mouse cursor. In the textual tree view, initially we used the mouse wheel for zooming as in the graph view. However, this differs from the standard behavior in classic tree views. Now, users can scroll this view via the mouse wheel, while for zooming the CTRL key has to be held down in addition to rotating the wheel.

Further users feedback led to the implementation of the infinitive grid, the pan wheel and the radar view, the preview mode for edge-based traveling, the search box, and the history mechanism mentioned earlier (see Section 6.5). In summary, user feedback has been incorporated continuously during CGV's development. This supports the thesis that formative testing is indeed: "a valuable technique during the development of an information visualization to gain design feedback and fix bugs." [29].

6.7.2 *Expert Interviews*

On a second evaluation stage, we provided CGV to domain experts and asked for their criticism. Besides the general feedback, two specific needs were raised by the experts.

"dIEM oSiRiS" is a joint effort of several disciplines to investigate regenerative systems with the goal to assist the research for a cure of Parkinson's disease and other neuronal defects [345]. The need to store complex filters for later re-use was pointed out to us by researchers from this group. As we described earlier, such functionality is now available in CGV.

Investigators who conduct research related to neuro-mapping and development atlases of neuronal systems [296] pointed out that filtering based on node and edge attributes is not enough in their domain. They need methods to filter their data according to structural graph properties. These suggestions motivated us to start a project to introduce new tools in CGV for graph motifs finding. Though we were able to generate some initial results, this project is still in its early stage, and we hope to report this in the future.

6.7.3 *Web-Based Evaluation*

The third stage in our evaluation strategy is to deploy CGV on the Web and to collect feedback from a broader audience. Currently, we provide access to CGV and a medium size data set for experimentation [326]. We encourage the readers to go to the web site, test our tools, and send us their general feedback. The flexibility of CGV also allows for more controlled experiments. For that purpose we have begun to identify specific aspects of CGV for user testing. We derived, for instance, an applet that focuses on the use of lenses and an applet that solely consists of the Magic eye view. One could also think of two applets that employ the same visual and interaction concepts, but use different parameterizations. We plan to embed these applets in web pages that automatically collect timings and/or success rates and provide an interface to rate user satisfaction. Even though we are aware of the risks of web-based questionnaires, we hope that the feedback collected over the internet reveals hidden issues that otherwise would have remained undiscovered.

6.8 CONCLUSIONS AND FUTURE WORK

We presented the interactive graph visualization system CGV. We described its views and how they support the visual exploration of graph hierarchies. A particular focus of this work is on interaction. We introduced several novel interaction techniques. They include filter composition, graph lenses, a pan wheel, edge-driven navigation, adjustable parameter settings, and a flexible docking framework. Novel visual cues are used to augment the provided interaction techniques (i.e., the infinite grid and the radar view). A history mechanism is available to allow for undoing and redoing of interaction, which is usually not available in current graph visualization systems.

CGV is a general modular system that is applicable in a variety of scenarios. First real world applications have been in the context of evaluating graph clustering algorithms (see Figure 6.2) and the exploration of topological structures of neuronal systems (see Section 6.9 and Figure 6.12). We are currently experimenting with CGV's filtering and history mechanisms to explore time evolving graph data. The flexibility of CGV has paid off significantly during adaptation to different application scenarios.

When interactivity is the focus of system development, the separation of visual interface and backing computations becomes a necessity. Interaction techniques and their visual augmentation require tight integration with the visualization. Well designed models for coordination and history are required to keep the system consistent.

General questions we want to address in future research are: Is it possible to come up with common interaction patterns? Can user interfaces for visualization parameters be generated automatically from a proper description of the main parameter characteristics? How can the flexible docking mechanism be used to achieve task-based adaptation of the system?

A more specific future objective is the integration of motif detection as indicated in Section 6.7.2. This requires investigation of aspects related to the specification of motif patterns, the detection of motif patterns in the graph hierarchy, and the visualization and interaction with detected pattern matches.

Improving the history mechanism to support more complex interactions is another aim for future work. An approach is to keep track of parameter adjustments performed via the settings panel (asynchronous parameterization) and interactive changes applied to the dynamic filtering mechanism. To accomplish this we have to extend our implementation in two aspects. First, we need composite commands to encapsulate multiple basic commands. Secondly, we have to face continuous interactions. CGV's current history mechanism reaches its limits when users apply continuous parameter adjustments via sliders or continuous browsing in the view space via the pan-wheel. In these cases, continuous interaction is mapped to a series of discrete commands, all being tracked in the history, which is rather impractical. One possibility to overcome this shortcoming is to aggregate a series of related commands into a single command based on well-chosen timing. In other words, during continuous interaction commands are aggregated. Only if the user pauses for a certain time period during interaction (e.g., to check an intermediate result more closely), we do issue a separate command to the history and begin aggregating anew.

Finally, extension of the proposed interaction mechanisms to navigate large directed graphs requires major breakthroughs in hierarchy generation. This is due to the fact that current methods for undirected

graphs are hindered by their inability to handle a sense of topological direction.

6.9 CASE STUDY: TOPOLOGY OF NEURONAL SYSTEMS

CGV has been used to explore the topological structure of neuronal systems. In particular, the researchers goal is to create atlases of rat brains. For that purpose, rat brains are dissected and mapped, for instance with the help of computer vision methods [296]. The atlases are organized in a hierarchical fashion, starting with the whole brain, splitting into its left and right parts, and going further down to more and more specific areas of the neuronal system. This hierarchy is constantly updated according to new findings published in medicine articles. Additionally, this hierarchical structure of rat brains is manually annotated with knowledge about functional dependencies between regions of the brain. That is, if an experiment reveals a relationship between regions A and B, then an edge (A, B) is inserted. These edges can be directed or undirected, depending on the experimental results. It is also possible that an edge connects a rather coarse brain region (in upper parts of the hierarchy) with a quite specific one (in lower parts in the hierarchy). Such edges give researchers hints as to where further experiments could reveal more interesting and more specific results. Patterns of functional dependency play also a role. The researchers are mainly interested in motifs of 3 to 5 nodes, where triads are particularly important.

Next, we describe how CGV can be used to explore the aforementioned data. Currently the data contains about 3000 nodes and roughly the same number of edges, but these numbers will increase in the future as more medical results are entered in the database. Figure 6.12 (a) shows an initial representation of the data. The user has folded away most views because he is mostly interested in the data's hierarchical structure. The symmetry of the left part and the right part can be seen from the hierarchy view. The colors are pre-defined and have a specific meaning to the researcher. He can use the search box to find a node whose label contains the term `Unclassified`. After typing three letters and performing a single click, the hierarchy view updates itself automatically to focus on the node with the label `Unclassified_cell_group_L` (Figure 6.12 (b)). Since this might identify a part of the brain that needs further research, the user decides to fold the hierarchy view and switch to the graph view to see the node's relations to other parts of the brain. The graph view has already performed layout computations and set the focus on the corresponding cluster automatically. By looking at the edges, the user can see that the node is related to other parts of the same cluster and to nodes in other clusters, which however are off-screen (Figure 6.12 (c)). In Figure 6.12 (d), he then uses the composite lens to create a local over-

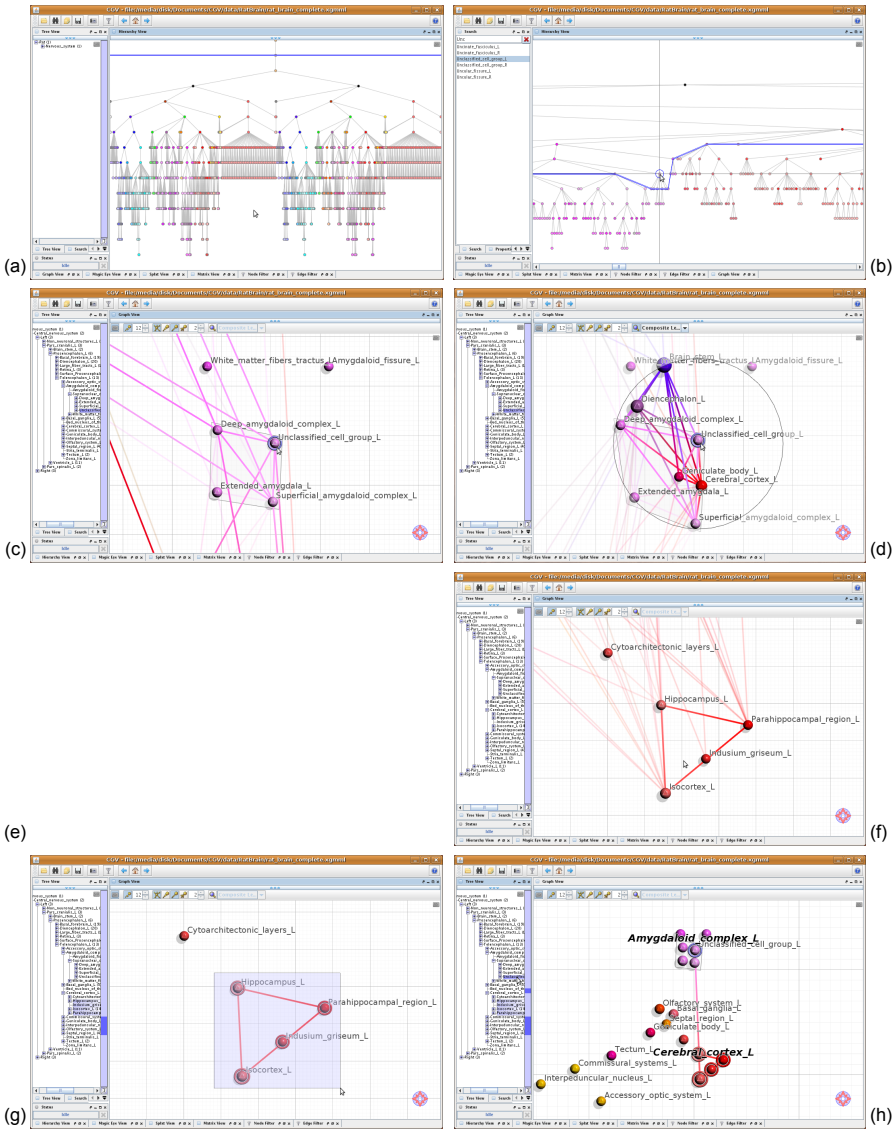


Figure 6.12: Exploration of the topological structure of rat brain.

view of the nodes connectivity. From this local overview he decides that the node labeled `Cerebral_cortex_L` is worth further investigation. In order to navigate to that node, the user releases the lens and uses edge-traveling. The navigation to the selected node is augmented with an animation, which also slightly zooms out to create a better overview (Figure 6.12 (e)). During zooming, the infinite grid serves as a spatial reference to help the user judge the distance traveled. After the node has been reached, the researcher expands it to see details (Figure 6.12 (f)). However, at this point, the connection to the "Unclassified" node is no longer apparent. To really focus on that connection, nodes can be selected as shown in Figure 6.12 (g). To shortcut the navigation back to the "Unclassified" node, the researcher performs the selection in the textual tree view (left part of Figure 6.12 (h)). By zooming out, a full screen overview of the connectivity of the selected

nodes has been created. The researcher can now decide whether it is worth to experimentally investigate this connectivity in more detail or can continue his exploration.

6.10 CASE STUDY: EXPLORATION OF "WORDNET" DATA

In a second usage example, we explore the well-known "WordNet" data [120]. This data has about 100k nodes and 150k edges. CGV automatically computes a hierarchy tree from the big graph. The user can then use the hierarchy for interactive information drill down. We exemplify this procedure in the following.

Figure 6.13 (a) shows an initial macro-view of the underlying "WordNet" data. The textual tree view represents the hierarchical structure

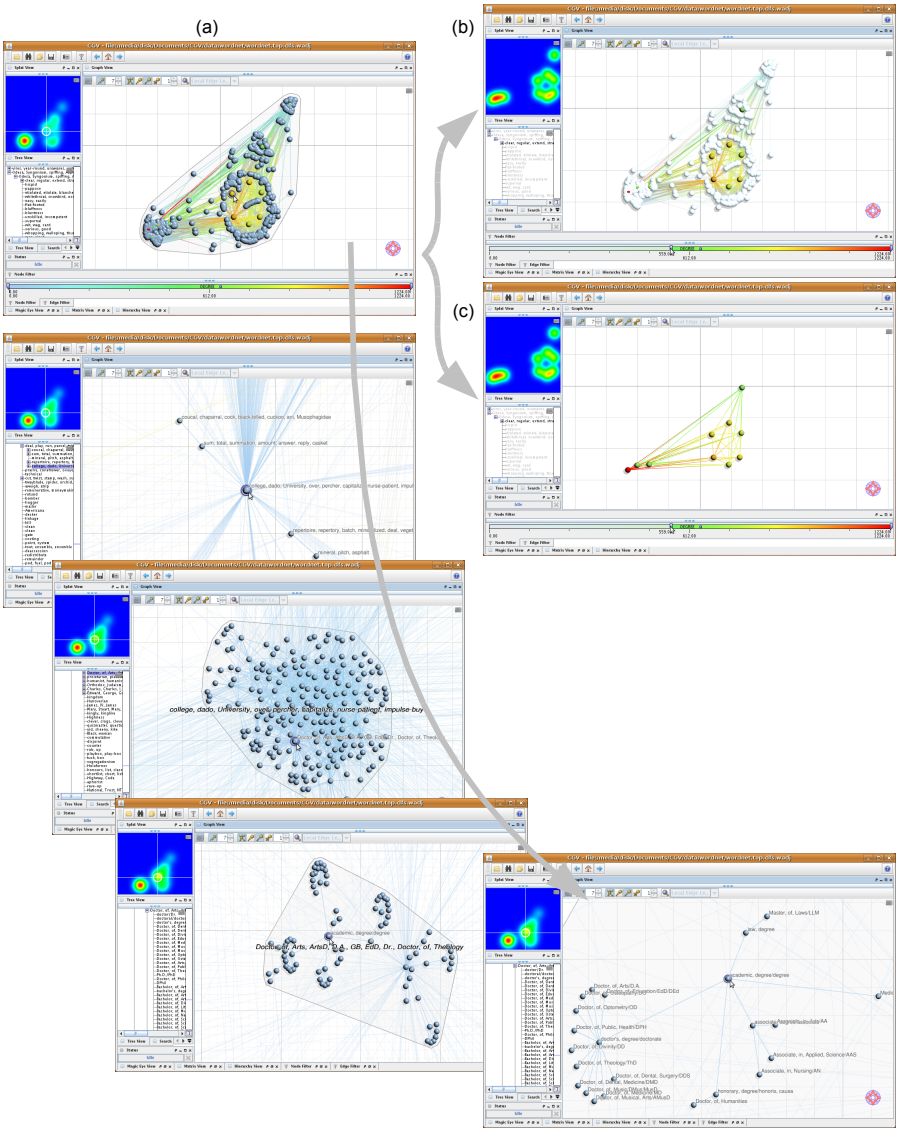


Figure 6.13: Exploration of "WordNet" data.

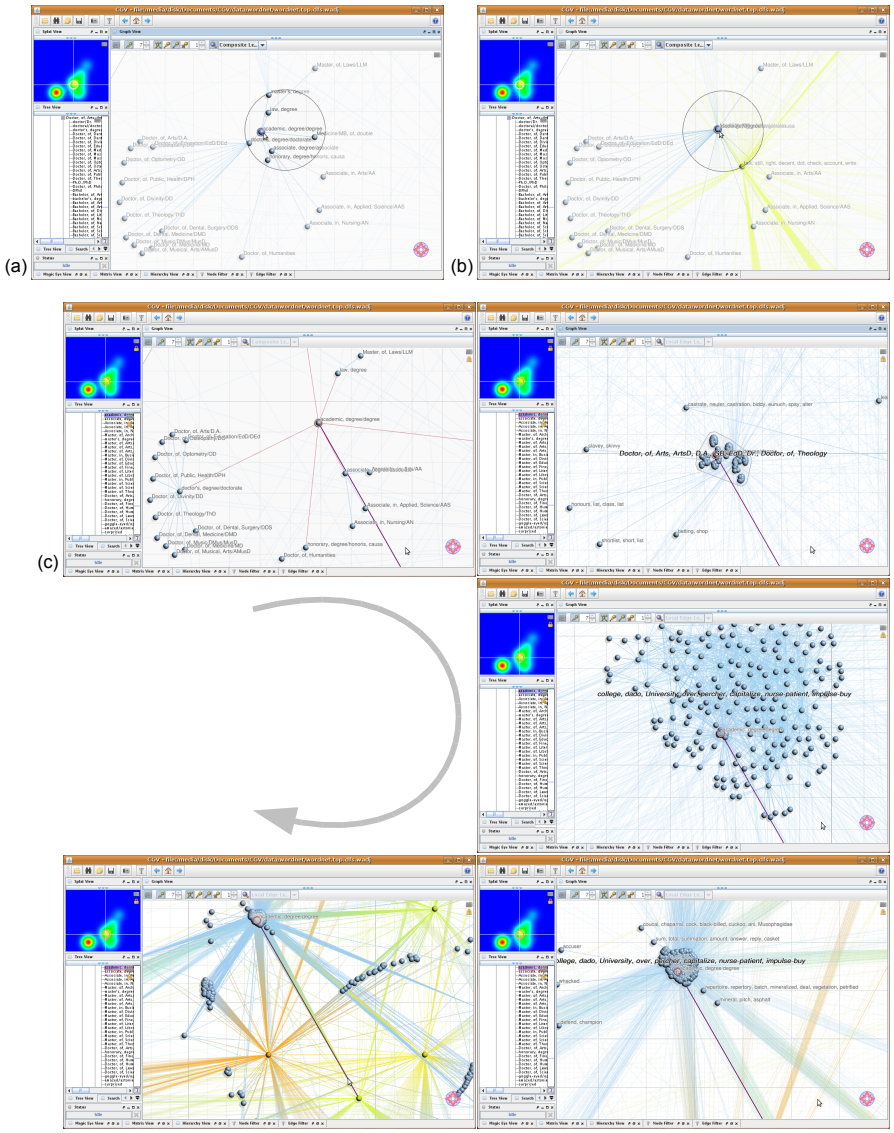


Figure 6.14: Exploration of "WordNet" data continued.

and the splat view gives a coarse overview. Since the main graph view is cluttered with many nodes and edges, the user performs a filter operation with the slider to reduce the number of nodes. Figures 6.13 (b) and (c) show the effects of dimming filtered nodes and of omitting them, respectively. Based on the filtered view the user decides to start his exploration with the node selected in Figure 6.13 (a). He disposes off the filter to have more screen space for the main graph view. After a series of expand operations, the user reaches a node labeled `academic, degree/degree`, which draws his attention. There are several related nodes in the proximity. To get a better overview, the user applies the layout lens to attract the related nodes as shown in Figure 6.14 (a). As he moves the lens he recognizes that the neighbors accumulate in the lens center and a yellow node appears at the lens boundary (Figure 6.14 (b)). Because the nodes accumulate even

when applying the composite lens, which should spread them, the yellow node must be very far away in the layout. The user decides to explore this distant node in more detail. In order to get an impression of where the node is located, the preview mode of edge-based traveling is used. As shown in the sequence in Figure 6.14 (c), several zoom levels are passed before the preview is reached and the traveled edge is centered on the screen. At this point, the user can decided to click the edge again, to actually move to the yellow node.

7

STACKING-BASED VISUALIZATION OF TRAJECTORY ATTRIBUTE DATA

CONTRIBUTION This chapter contributes a novel interactive visualization approach for the exploration of trajectory attribute data. The integrated interaction techniques are specifically designed to match the requirements imposed by the characteristics of the data and the associated tasks. The novel 2D/3D navigation techniques and the introduced time lens promote fluidity at the level of intermediate interaction and seamless transitions between different data facets.

ABSTRACT Visualizing trajectory attribute data is challenging because it involves showing the trajectories in their spatio-temporal context as well as the attribute values associated with the individual points of trajectories. Previous work on trajectory visualization addresses selected aspects of this problem, but not all of them. We present a novel approach to visualizing trajectory attribute data. Our solution covers space, time, and attribute values. Based on an analysis of relevant visualization tasks, we designed the visualization solution around the principle of stacking trajectory bands. The core of our approach is a hybrid 2D/3D display. A 2D map serves as a reference for the spatial context, and the trajectories are visualized as stacked 3D trajectory bands along which attribute values are encoded by color. Time is integrated through appropriate ordering of bands and through a dynamic query mechanism that feeds temporally aggregated information to a circular time display. An additional 2D time graph shows temporal information in full detail by stacking 2D trajectory bands. Our solution is equipped with analytical and interactive mechanisms for selecting and ordering of trajectories, and adjusting the color mapping, as well as coordinated highlighting and dedicated 3D navigation. We demonstrate the usefulness of our novel visualization by three examples related to radiation surveillance, traffic analysis, and maritime navigation. User feedback obtained in a small experiment indicates that our hybrid 2D/3D solution can be operated quite well.

ORIGINAL PUBLICATION [337] — C. Tominski, H. Schumann, G. Andrienko, and N. Andrienko. Stacking-Based Visualization of Trajectory Attribute Data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2565–2574, 2012.

7.1 INTRODUCTION

Exploring trajectories of moving objects is relevant to people in a number of application domains. Examples are traffic planners who need to find bottlenecks in traffic networks, physicists who seek to understand particle movements, or sociologists who analyze the behavior of human individuals. For these scientists, trajectories are valuable sources of information because they encompass spatial and temporal aspects of the movement of objects and additional quantitative and qualitative attributes about the movement and the environment or context in which the movement took place.

The three components *space*, *time*, and *attributes* lead to an information richness that makes analyzing trajectories a profitable task. But understanding spatio-temporal trajectory attributes is also difficult, because it involves a variety of aspects. One needs to assess spatial and temporal dependencies, which need to be set into relation to gain insight into the spatio-temporal dynamics of attributes. Trajectory data might contain interesting facts not only at the level of individual trajectories, but also at the level of sets of trajectories (e.g., trajectories that cross specific regions in space and/or that cover particular spans in time). For larger data sets it is usually unclear where interesting facts can be found and which trajectories needs to be looked at in detail.

In consequence of the complex interplay of different data aspects and analysis tasks, providing appropriate support for interactive exploration of trajectory attribute data is challenging. Hence, existing visualization methods usually focus on one or two particular aspects, but not all of them. According to our research, none of the existing methods provides sufficient support for investigating individual trajectories *and* sets of trajectories with regard to space *and* time *and* attributes.

With this work, we develop a novel solution that covers all facets involved in the analysis of trajectory attribute data. According to the nature of the data and based on a study of relevant analysis tasks, we suggest the following general visualization design: Attribute data of individual trajectories are visualized as *color-coded bands* and sets of trajectories are visualized by *stacking* the bands.

Because showing all data aspects in full detail at all times regardless of the analysis task is infeasible, we provide complementary visual representations. The hybrid 2D/3D *trajectory wall* visualizes trajectory attribute data by stacking 3D color-coded bands on a 2D map. The association to time is established through temporal ordering and through the *time lens*, a circular time display that is connected to a dynamic query mechanism. Additionally, the 2D *time graph* shows trajectories as horizontal bands along which the time-dependency of an attribute is encoded by color. All representations are coordinated,

enabling analysts to link temporal and spatial aspects in order to make spatio-temporal discoveries.

This basic visualization solution is further equipped with supplemental components, including construction of meaningful subsets of trajectories based on interactive selection and analytical calculations, interactive adjustment of the color-coding based on statistical properties of the data, and dedicated navigation mechanisms.

In summary, our contribution is a novel approach that (1) integrates space, time, and attributes, (2) considers relevant analysis tasks, and (3) combines visual, analytical, and interactive components to facilitate trajectory attribute exploration.

We derive our novel approach and describe its individual components in detail in Section 7.3. To demonstrate the usefulness of the proposed solution, we apply it in Section 7.4 to visualize several interesting data sets, including the radiation around Fukushima Daiichi nuclear power plant, the taxi traffic at San Francisco airport, and vessel movement in the harbor of Brest. The usability of our solution has been evaluated in a small experiment. The generally positive feedback of the participants and their constructive suggestions for improvements are briefly discussed in Section 7.5. This article ends in Section 7.6 with a conclusion and ideas for future work.

7.2 DATA, TASKS, AND RELATED WORK

Next, we start with introductory comments on the data we are concerned with, study the questions that analysts might ask about such data, and take a look at related work.

7.2.1 *Data*

The general goal of our work is to explore dynamic attributes along trajectories in space and time – for individual trajectories as well as across sets of trajectories. Achieving this goal is becoming increasingly relevant, because new sensors and data collection infrastructures support the acquisition of contextual attributes of movement better and better.

For example, GPS devices used by runners annotate position records with attributes representing physical conditions such as heart rate or body temperature. Web sites like `movebank.org` provide the infrastructure for enriching trajectories with environmental attributes reflecting weather, land cover, and other phenomena.

Moreover, attributes can be derived directly from raw trajectory data. Examples are speed, direction, acceleration, turn, sinuosity, and distance to selected places or trajectories. A classification of potentially interesting attributes is provided in [34].

Trajectory data D that are associated with attributes can be formally defined as follows. A trajectory $\mathbf{d} \in D$ is an ordered set of data points $\mathbf{d} = \langle d_1, \ldots, d_{l_\mathbf{d}} \rangle$. Each data point $d_k : 1 \leqslant k \leqslant l_\mathbf{d}$ is of the form $d_k \in (S^n \times T \times A_1 \times \cdots \times A_m)$, where S^n defines the spatial coordinates of the point (e.g., geographical latitude and longitude if $n = 2$, plus elevation if $n = 3$), T defines time, and $A_i : 1 \leqslant i \leqslant m$ are the value ranges of quantitative or qualitative attributes. This definition shows the complexity of the problem we face: The data encompass spatial and temporal aspects as well as numerical and/or categorical data.

Here we consider trajectories in 2D space with a moderate number of attributes. The number of trajectories and the number of points per trajectory varies between a few dozens and several thousands, resulting in data sets with about a million individual measurements. Furthermore, we consider domains with hard-constrained (e.g., road traffic or indoor movement of people) or soft-constrained (e.g., seasonal migration of animals or traffic lanes in sea or sky) trajectories. An essential aspect of such constrained trajectory data is that there are large subsets of trajectories with similar geometry. This similar geometry is crucial for our approach.

7.2.2 *Tasks*

In exploratory trajectory analysis the analyst aims at understanding the interrelations between the data components, in particular, between the spatial (S), temporal (T), and attributive (A) components in trajectories of moving objects. Based on distinguishing between *independent* dimensions and *dependent* attributes, exploratory data analysis can be viewed as analogous to the investigation of the *behavior* of a mathematical function, i.e., the way in which the values of the dependent variable(s) vary with respect to the independent variable(s) [36]. For trajectory data, the main goal is to understand the functional dependency $S \times T \to A$, i.e., the *behavior* of the attributes with respect to space and time.

Depending on the focus of the investigation, the analyst may pursue the following behavior-related objectives:

BEHAVIOR CHARACTERIZATION Observe the value distribution of A over the whole S and T or selected parts of S and T and characterize, mentally or explicitly, the behavior of A. It can be characterized as constant or piecewise constant in regions in space or periods in time or as having gradual or abrupt changes, temporal or spatial trends, repetitions in space and time, periodicities in time, local or global outliers, and so forth. An example is to characterize the behavior of the vehicle speed along a highway over a day.

BEHAVIOR SEARCH Detect occurrences of a particular behavior of interest and locate them in S and T. An example is to find out in which parts of the highway and during which times of the day traffic congestions occurred, i.e., low speeds of multiple cars.

BEHAVIOR COMPARISON Compare the behaviors of A in different regions of S or in different intervals of T or in different subsets of the trajectory data D. Examples are to compare the behaviors of the vehicle speeds on different highway segments, or on different days (e.g., work days vs. weekend), or in the subsets of trajectories going in opposite directions.

Since the investigation of the overall behavior $S \times T \to A$ is a complex task, the analyst may decompose it into simpler subtasks. One type of subtask is to focus on selected places $s \in S$ and consider the corresponding behavior of A over T: $T \to A$ for $s = const$. An example is to consider the temporal variation of the speed over the day at a selected crossing. This kind of behavior can be called *local* with respect to space.

Another subtask type is to focus on selected times $t \in T$ and consider the corresponding behavior of A over S: $S \to A$ for $t = const$. An example is to consider the variation of the speed along the highway at around 8AM. This kind of behavior may be called *local* with respect to time. In both cases, one of the dimensions T or S is handled at an *elementary level*.

After exploring the local behaviors in different places and times, the analysis is lifted to a *synoptic level*, where the goal is to understand the overall behavior $S \times T \to A$. The term *synoptic level* combines Bertin's [56] overall and intermediate reading levels (as opposed to the elementary reading level).

Hence, visualization tools for exploring trajectory attribute data should provide appropriate support for the characterization, search, and comparison of local behaviors $T \to A$ and $S \to A$ and overall behaviors $S \times T \to A$ for the whole S, T, and D and subsets thereof.

7.2.3 *Related Work*

A recent review [31] indicates that the analysis of movement data in general is still one of the most important topics in many fields of research, including data mining, GIScience, and visual analytics (see for example [136, 32, 145]). The majority of the existing works concentrate on (1) analyzing spatial and temporal aspects of trajectory shapes (in 2D for space and 3D for space-time), (2) detecting stops, interactions between trajectories, and other types of events, or (3) aggregating trajectories in space and time. However, only little has been done so far on analyzing trajectory attributes.

Among the first attempts to visualize attributes of trajectory data are Charles Minard's maps (see [285] for a review of Minard's work). A classic example is his famous map of Napoleon's Russian Campaign. The map depicts the size of the French army by the width of a band on the map, and air temperature by a visually connected time graph.

At present, trajectories are often represented in a space-time cube, which combines time and space in a single display [213, 198]. In principle, it is possible to show also attributes in this display, but this approach is quite limited in respect to the number of trajectories.

Contemporary works on visualizing trajectory attributes confirm that plotting attributes in geographic space (2D or 3D) is beneficial for their analysis. For instance, Ware et al. [368] developed the GeoZui4D system to display multiple attributes along 3D trajectories of underwater movement of whales using color, texture, and glyphs.

Kraak and Huisman [214] use a combination of time graphs for two attributes (speed and heart rate), a map, and a space-time cube (representing a selected attribute by coloring trajectory segments) for identifying interesting events. However, this approach considers only single trajectories and not sets of them.

Spretke et al. [315] apply color-coding to the segments of multiple trajectories on a 2D map for showing different classes of segments based on multiple attributes. This facilitates separating day flights of migratory birds from night flights and from stops, as well as showing footprints of different classes on the map. However, overplotting hinders detecting spatial behavior along individual trajectories.

Crnovrsanin et al. [89] use a time graph together with a trajectory map for displaying the dynamics of distances to selected places (e.g., forest roads or exits from a building). To compensate for overplotting on the map, the authors transformed the geography using so-called proximity PCA. This approach works for multiple trajectories, but overplotting on maps and in time graphs remains a critical issue.

[33] also use a combination of a time graph display with a map for multiple trajectories. To resolve overplotting in the time graph, they use a so-called time band display (similar to [206, 247]) with coloring based on class intervals.

The reviewed approaches work well for basic tasks. Depending on the focus of the visualization design, $S \rightarrow A$ and $T \rightarrow A$ tasks can be accomplished. Arguably, some approaches (e.g., those based on the space-time cube) can be useful to support $S \times T \rightarrow A$ tasks, but as described earlier such tasks remain complicated anyway. The idea of splitting this complex analysis task into more focused and easier to accomplish subtasks is not explicitly supported by existing solutions, neither at an elementary level nor at a synoptic level.

As a result of the review of the related work, we can summarize that the distribution and the dynamics of attribute values in space

and time remain difficult to analyze, especially, when analysts are interested not only in individual trajectories, but also in collections of trajectories.

7.3 TRAJECTORY ATTRIBUTE VISUALIZATION

In order to support analysts in exploring trajectory attribute data, we have to address the complex data characteristics and the tasks as described in Sections 7.2.1 and 7.2.2. We propose a novel approach that deals with these challenges by combining visual, analytical, and interactive means.

7.3.1 *Solution Overview*

Inspired by Dang et al.'s [92] stacking of graphic elements, our solution visualizes attribute values along stacked trajectory bands. Individual color-coded bands support elementary $* \rightarrow A$ tasks and the stack of the bands as a whole supports synoptic $* \rightarrow A$ tasks. The utility of this approach depends on appropriate color-coding, and appropriate grouping, selection, and stacking of trajectories, which will be discussed in Section 7.3.2.

Because time and space differ in their intrinsic properties and also in terms of how humans perceive them and reason about them, the general visualization design needs to be adapted to space and time. The flexibility of trajectory bands and the generality of the stacking concept enables such an adaptation. As a result, we provide two complementary displays:

- The hybrid 2D/3D *trajectory wall* focuses on the spatial behavior $S \rightarrow A$ by embedding 3D bands into a virtual map space. The temporal information cannot be displayed in full detail in this view. A part of the temporal information, namely, temporal ordering, can be conveyed through ordering of the bands. Additionally, an integrated dynamic query tool, called the *time lens*, allows the analyst to access temporally aggregated information.
- The 2D *time graph* focuses on the temporal behavior $T \rightarrow A$ by showing attribute values along horizontal 2D bands.

Understanding the spatio-temporal behavior $S \times T \rightarrow A$ for individual trajectories and across sets of trajectories requires switching frequently between space and time, and between elementary and synoptic tasks. The full potential of our approach lies in applying the provided visual representations, interactive mechanisms, and analytical tools in a linked and coordinated way.

7.3.2 *General Visualization Issues*

As indicated before, our approach requires an appropriate color-coding of attribute values and an appropriate selection and ordering of the trajectories to be visualized as stacked trajectory bands.

COLOR-CODING OF ATTRIBUTE VALUES Because we use the spatial coordinates on the screen for showing the dimensions of time and space, attribute values must be encoded using another visual variable. We rely on color because it is a widely accepted approach, it fits well with the trajectory band design, and it is both selective and associative [56] and therefore can support very well elementary and synoptic tasks.

To make $* \rightarrow A$ behavior of attribute values easily detectable and interpretable, an appropriate mapping of the values to colors is required. There is a long standing discourse in the visualization community about whether to use isomorphic vs. segmented color scales [55]. In cartography, this topic corresponds to the discussion about unclassified vs. classified thematic maps. According to cartographers, classified thematic maps (i.e., maps using segmented color scales) can represent behavior better [242]. However, this requires not only an appropriate color scale, but also an appropriate definition of class intervals used for the mapping of data values to colors.

In terms of the color scale, we rely on the ColorBrewer [152], which provides evaluated color scales for different numbers of classes. We give the user the freedom to choose one of the ColorBrewer scales according to his/her preferences or domain-specific conventions.

The definition of appropriate class intervals is intricate because there is no perfect method that produces a single "best" partition of an attribute's value range into classes [242]. The partitioning can be based on different criteria [307]. The cartographic literature recommends choosing class breaks according to the statistical distribution of the values so that similar data values are placed in the same class and dissimilar in different classes [307].

Our tools support this recommended strategy in two ways. First, the division can be done fully automatically using the algorithm for statistically optimal classification [191, 307]. Second, the user can interactively set class breaks according to "natural breaks" in the data, which can be detected visually by means of a dot plot or a cumulative curve display [36]. The classification tools also support the divisions into equal intervals and by quantiles, which also have certain advantages [307]. Furthermore, the user can arbitrarily set the breaks according to his/her understanding of the data or according to domain-specific standards or conventions.

Irrespectively of the criteria and strategy used for defining class intervals, it is reasonable to test the sensitivity of the observed patterns

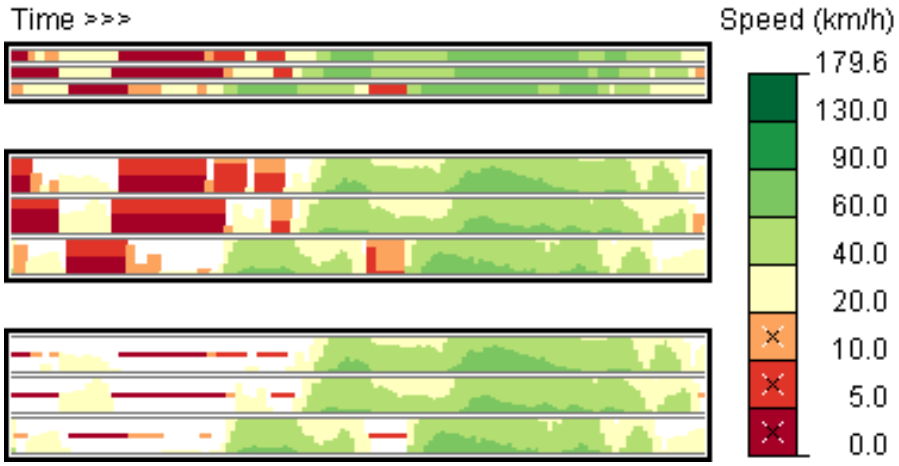


Figure 7.1: Alternatives for color-coding attribute values along trajectory bands. Top: plain color-coding requires less space; middle: two-tone pseudo-coloring [290] increases precision; bottom: color filtering reduces visual load.

to the class break setting. To do such a testing, the user can move the class breaks interactively by means of sliders, which results in immediate updating of all displays where these classes are represented.

Data values can be discerned more precisely when applying two-tone pseudo-coloring [290] (see middle bands in Figure 7.1). This technique (a.k.a. Horizon Graphs) requires sufficient height of the bands to achieve higher precision [163]. As we have found empirically, the two-tone coloring is visible when the height is at least seven pixels, whereas just two or three pixels are enough for plain class-based coloring (see top bands in Figure 7.1). This may have implications when large numbers of trajectories need to be visualized.

To facilitate focusing on value ranges of interest, which may be particularly useful for behavior search and behavior comparison, we allow the user to decrease the visual prominence of selected value intervals (see bottom bands in Figure 7.1). This *color filtering* is activated by clicking on the corresponding rectangles in the color legend. The operation affects the bands in the trajectory wall and the time graph.

GROUPING AND SELECTING TRAJECTORIES Grouping is useful for dividing a large set of trajectories into manageable portions, which can be analyzed one by one, or for focusing on interesting subsets of trajectories (with respect to the analysis goals) and disregarding uninteresting ones.

For analyzing trajectory attributes in respect to space $S \rightarrow A$, we start with identifying groups of trajectories that have similar geometries. The analyst can do this by means of spatial queries (e.g., trajectories passing a series of user-selected regions), or by means of clustering trajectories by similar origins, destinations, or route similarity [284]. When analyzing the temporal behavior $T \rightarrow A$, it makes

sense to construct groups based on temporal queries, e.g., selecting evening or weekend trajectories. The results of the spatial and/or temporal grouping can be further refined by attribute queries (e.g., selecting trajectories with median speed higher than 70 km/h).

STACKING TRAJECTORIES In order to enable analysts to carry out tasks at the synoptic level, an appropriate stacking of trajectory bands is needed. As mentioned earlier, chronological ordering of the trajectory bands brings a part of temporal information into the trajectory wall display. The ordering can be done according to the absolute times of the starts or ends of the trajectories (i.e., using the linear time model) or according to their positions within one of the temporal cycles, such as daily, weekly, or seasonal (i.e., using the cyclic time model).

The temporal ordering of trajectories is important for supporting synoptic $S \times T \rightarrow A$ tasks. With a temporally ordered stack of trajectories, vertical neighborhood of band segments corresponds to *temporal* neighborhood of the trajectory points. Similarly, horizontal neighborhood of band segments corresponds to *spatial* neighborhood of the trajectory points.

Due to the associative property of color, neighboring band segments with the same or close colors (representing similar attribute values) are perceptually united into larger spots. Hence, relatively homogeneously colored spots (perceived by human vision as unities) correspond to spatio-temporal regions of constant behavior.

Furthermore, like gradual changes of the color along the horizontal dimension signify a spatial trend, gradual changes along the vertical dimension signify a temporal trend, and changes in a diagonal direction correspond to a spatio-temporal trend. Hence, owing to the temporal ordering of the trajectory bands, the user can perceive local behaviors $S \rightarrow A$ and $T \rightarrow A$ and overall behavior $S \times T \rightarrow A$.

Additionally, it can be useful to stack trajectory bands according to other criteria. For example, ordering by the average speed supports comparison of faster and slower trajectories and detecting places of major speed differences between them, which may signify re-occurring traffic problems. We allow the user to arrange trajectories based on any attribute or sequence of attributes referring to the whole trajectories, while the temporal ordering is used as default. The generated trajectory sequences are then fetched to the visualization components described next.

7.3.3 *Design of the Visualization Components*

To facilitate the decomposition of the analysis into local $S \rightarrow A$ and $T \rightarrow A$ subtasks, we need to define complementary views that focus on each subtask's specific character. One view focuses on the spa-

tial aspect $S \rightarrow A$ and another one on the temporal aspect $T \rightarrow A$. Moreover, each view must be equipped with facilities to establish a connection to the dimension it is not focusing on.

Because the concept of trajectory bands and their stacking is simple yet flexible by design, it can be easily adapted to the aforementioned requirements. We propose two instantiations: the *trajectory wall* with focus on space (see next section) and the *time graph* with focus on time (see Section 7.3.3.2).

7.3.3.1 *Visualizing Spatial Attribute Behavior*

In order to support $S \rightarrow A$ tasks, it is necessary to show trajectory attribute values in their spatial context. As we consider 2D trajectories, it appears as if a 2D solution would be fully sufficient. However, this is not true. A 2D solution would quickly suffer from severe overplotting because we deal with trajectories with very similar geometry. It would be difficult to separate individual trajectories and discern attribute values along trajectory paths. A 2D solution would also fall short in terms of maintaining an order of trajectories, which is required for detecting behavior at the synoptic level. Utilizing the third dimension for a 3D stacking of trajectory bands will help us resolve these concerns.

On the other hand, stacking trajectories along the third dimension means detaching them from their 2D reference space. This can make it more difficult for the analyst to understand the trajectory paths through space. Therefore, we need to integrate mechanisms that retain the basic 2D character of trajectories.

According to this thinking, we designed the trajectory wall as a hybrid 2D/3D approach. The spatial context is visualized by a 2D map that resides in a virtual 3D viewing space. The map is constructed in a straight-forward way by tiling the bounding box of the trajectories with appropriately scaled bitmaps from the OpenStreetMap project [7] (or any other tile server).

The virtual 3D viewing space serves as the spatial reference continuum into which the visualization of trajectories and attribute values needs to be embedded. For this purpose, we have to construct trajectory paths whose individual segments can be colored using the mechanisms described in Section 7.3.2.

CONSTRUCTING TRAJECTORY PATHS Given a trajectory $\mathbf{d}$ one can connect its individual points $\langle d_1, \ldots, d_{l_d} \rangle$ to form a path through space. Then each path segment can be color-coded according to the associated attribute value. There is a subtle detail that needs to be be taken care of: A trajectory $\mathbf{d}$ has l_d points, but there are only $l_d - 1$ path segments to be colored. Unfortunately, we cannot simply append an artificial segment to the path, because it would inappropriately alter the spatial *characteristics* of $\mathbf{d}$. Neither can we rely on color

interpolation along the path, because we are using segmented color scales.

Our solution is to insert split points midway between any two consecutive original trajectory points. Instead of connecting the original points directly, we create segments between the split points and the original points. The color of a segment is chosen so as to correspond to the attribute value of the original trajectory point incident to the segment. Figure 7.2 illustrates the mapping for 2D paths. Note that our solution requires rendering $2l_d$ segments, whereas simply appending an artificial line segment results in paths of length l_d. So, for trajectories with very many points one might need to trade off correctness of the mapping against performance of the rendering.

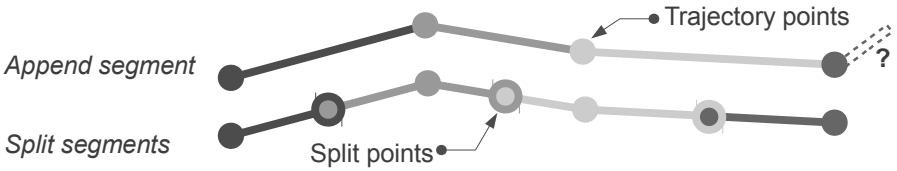


Figure 7.2: Mapping trajectory points to form a colored path.

VISUALIZING TRAJECTORIES AND ATTRIBUTE VALUES The described trajectory paths to are used to construct color-coded 3D trajectory bands as illustrated in Figure 7.3. The bands are stacked along the z-axis perpendicular to the map. This 3D approach provides each trajectory with an exclusive layer on the z-axis. Going into 3D also enables us to indicate the directions of trajectories by means of arrows embedded into the bands.

Although our solution allows for ordering by any attribute, ordering by time is the most appropriate for the trajectory wall, because it brings a part of the temporal information (relative order) to the display. In this way the vertical dimension also represents time, but relative time rather than absolute time.

To better preserve the trajectories relation to the 2D map, they are additionally visualized as color-coded 2D paths. The 3D bands and 2D paths are used in a smoothly blended fashion. When the display is rotated to a bird's eye view, the bands fade out, whereas the paths fade in. When approaching a horizontal perspective, paths fade back out and the bands reappear. The angles at which the fading occurs can be adjusted, including the possibility to allow for an overlap where paths and bands are visible at the same time.

To further facilitate understanding the trajectory bands in connection to the map, we add a highlighting plane for the focused trajectory and project the focused trajectory point onto the map. Optionally, the highlighting plane can show a locally confined duplicate of the map, which is useful when the base map is outside of the current view.

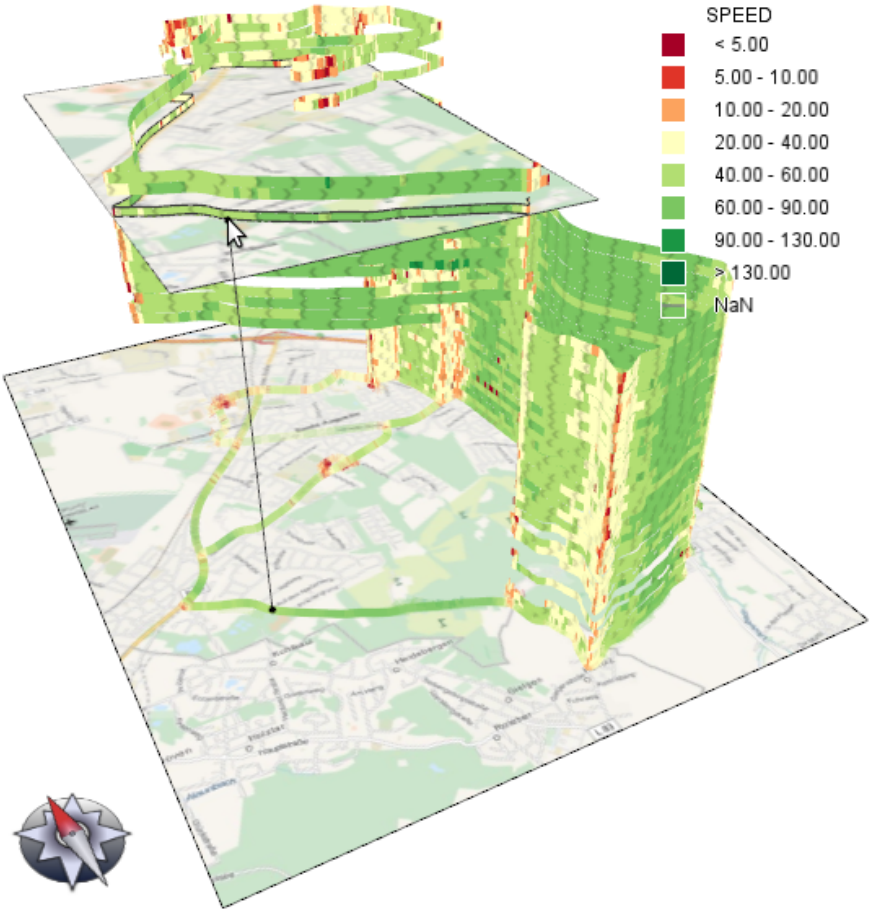


Figure 7.3: Visualization of trajectories as colored 3D bands and 2D paths.

Our hybrid 2D/3D design allows analysts to switch seamlessly between a 2D overview of all trajectory paths and the detailed investigation of attribute behavior in the 3D bands. Figure 7.3 illustrates that the 2D paths are useful to assess the spatial characteristics of trajectories and the overall spatial distribution of attribute values, but as already mentioned, overplotting hinders detailed analysis of individual attribute values along trajectories. The 3D bands make attribute values of individual trajectories and of groups of trajectories easier to explore thus facilitating elementary and synoptic tasks.

DEALING WITH THE IMPLICATIONS OF 3D We carefully analyzed the implications of our 3D environment. In particular, we need to address the problems of 3D navigation and occlusion [306, 207, 108].

Our goal is to make zoom, pan, and rotation functions in 3D simple and convenient. One way to achieve this is to consider the fact that the analyst is usually focused on something in particular when carrying out these operations. By adapting the 3D navigation to this focus point, we can reduce the complexity of the interaction. The zoom follows the point under the mouse cursor (e.g., zoom toward a specific

segment of a trajectory), and the pan grabs exactly that point allowing the analyst to drag it closer. Rotation in 3D is realized as orbiting around the focus, where a virtual compass is provided to maintain user orientation. A specific requirement of our design is to enable the navigation along the stacking order of the trajectories. Therefore, we give analysts the opportunity to use the so-called *elevator*, which corresponds to a smooth navigation along the z-axis. User feedback obtained in a small study (see Section 7.5) indicates that the *focused* zoom, pan, and orbit and the *elevator* are practical in our scenario.

Occlusion can be addressed by two means. One option is to make the wall transparent allowing the user to look through it. Apparently, this option should be used only on demand due to the adverse effects of unintended color blending. An alternative is to use the *color filtering* (see Section 7.3.2) to narrow the trajectory bands in places where irrelevant data is shown. The analyst could for example focus on extreme values and narrow the bands for mean values. The filtering has two benefits: occlusion is reduced where bands are narrow and relevant data stand out where bands are of regular width.

To further reduce possible interference of the map display and the trajectory visualization, the user can temporarily switch off the map or dim it to retain the spatial context.

The ensemble of visual and interactive components described so far facilitates the exploration of the spatial behavior of attributes $S \rightarrow A$. In order to better support $S \times T \rightarrow A$ tasks, a more direct link to time has to be established, in addition to the temporal ordering of trajectories.

MAINTAINING THE CONNECTION TO TIME Integrating time in full detail is hardly possible, because the trajectory wall is already visually rich with two dimensions showing the spatial frame of reference and the third dimension being utilized for the stacking. In order to limit the time-related information to a displayable amount, we developed the *time lens* (see Figure 7.4), which shows temporally aggregated information for an interactively defined spatial query area. The time lens is a circular display (similar to the trip view in [236]) that consists of two basic components: (1) the lens interior for showing spatial aspects and (2) the lens ring for visualizing temporal aspects.

The interior of the lens shows those trajectory points that match a circular spatial query area. The spatial query is interactively specified directly within the trajectory wall display by means of a *query circle* (see Section 7.7 for details). Moving the query circle allows the user to determine which trajectory points are displayed in the time lens, and resizing the query circle controls how many of them. Optionally, the user can extend the 2D query circle to a 3D query cylinder to further refine the query to specific trajectories from the stack of trajectories. Trajectory points that match the query are represented as dots whose

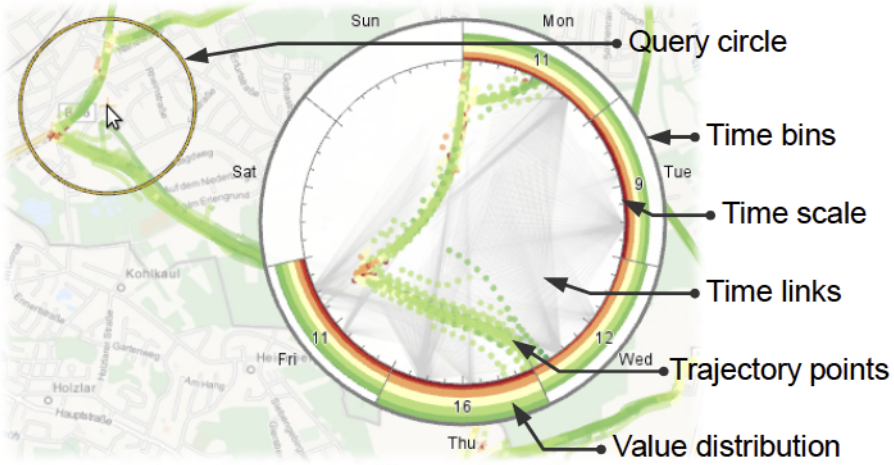


Figure 7.4: The time lens visualizes temporally aggregated information.

color correspond to the points' attribute values. The dots are embedded into the time lens according to their spatial layout (as shown in Figure 7.4).

The ring of the time lens is segmented into *time bins* based on the data's time model (see Aigner et al. [21]). Cyclic time is modeled as a set of recurring time primitives as for instance the 4 quarters of the year, 12 months of the year, 7 days of the week, or 12/24 hours of the day. If only linear time is semantically meaningful for the analyzed the data (e.g., eye-movement data), the time bins correspond to a suitable partition of the linear time domain.

The fill levels of the time bins visualize temporally aggregated information about the trajectories that intersect with the query. We provide three alternative aggregates: (1) *count* calculates how many trajectories intersect with the query area, (2) *total duration* accumulates the time spent by all trajectories in the query area, and (3) *average duration* averages the time spent by individual trajectories in the query area. Additionally, the time bins visualize the distribution of attribute values per time primitive. The time lens in Figure 7.4 shows clearly that no trajectories pass on weekends and that the distributions of values are similar for all workdays.

In addition to displaying aggregated information, we can establish a direct connection to time on demand. This is done by means of so-called *time links*, which connect each trajectory point to the time lens' inner *time scale*. The inner time scale is chosen so as to represent a sensible subdivision of the time bins (e.g., hours if bins are days).

When the query area is sufficiently small (i.e., only few trajectory points are shown), the time links are useful to directly connect space and time. Even with larger numbers of time links it is possible to reveal temporal patterns. For example, the time links in Figure 7.4 accumulate mostly at specific points at the time scale. These accumulations indicate that trajectories pass the query area only during

specific hours of the day. In order to make such pattern discernible, the overplotting in the lens interior can be resolved interactively by alpha-blending the time links and rotating the outer ring of the lens.

The tight integration of the time lens into the trajectory wall facilitates $S \times T \rightarrow A$ tasks. Figure 7.6 on page 139 illustrates the time lens for the count of trajectories and the temporal resolution of months applied to trajectories of mobile sensors that measured radiation along the Tokio-Fukushima highway. In the particular example from Figure 7.6 we can see that the proportion of the medium radiation values (yellowish color in the July and August bins) significantly decreased in September where lower radiation values (greenish colors in the September bin) became more frequent. Given the query area highlighted in the center of the figure, we can conclude that the situation improved in this part of the road. Further discoveries that can be made in the radiation data set will be described in Section 7.4.

Because the time lens depends on restricting the displayed temporal information via dynamic query mechanisms and aggregation, an additional display is needed that focuses entirely on the time aspect.

7.3.3.2 *Focusing on the Temporal Behavior*

To explore the temporal dimension in full detail ($T \rightarrow A$ tasks), we propose the complementary time graph display as illustrated in Figure 7.5. Following our principal visualization design, this display shows individual trajectories as stacked horizontal bands. Designed in this way, the time graph provides a synoptic view in respect to time, as overall temporal behavior can be characterized and be searched for. By sorting the trajectories in the vertical dimension according to different criteria, analysts can conveniently compare the temporal behaviors in individual trajectories and in groups of trajectories.

The number of trajectory bands that can be simultaneously seen in the time graph is limited by the available screen height. Larger sets of trajectories can be explored by means of scrolling or by grouping trajectories as discussed in Section 7.3.2. To enable analysis of long time

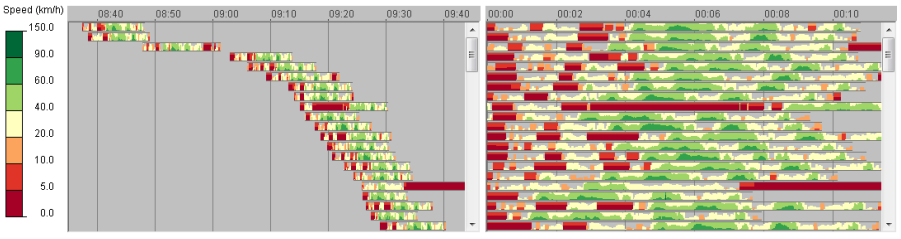


Figure 7.5: The time graph shows the dynamics of attribute values within trajectories by two-tone pseudo-coloring. The alignment of the trajectories has a significant impact on what can be seen in the display (left: alignment by time of the day, right: alignment by start time).

intervals, the display allows temporal zooming and panning, which can be done either fully interactively or through automated animation. To facilitate comparisons between trajectories that are far apart in time, the display supports two kinds of time transformation [30]: (1) in relation to the individual life times of the trajectories (start and/or end times) and (2) in relation to temporal cycles: daily, weekly, monthly, seasonal, or annual. The latter kind of transformation also supports the exploration of how the trajectories and attribute values are distributed with respect to the chosen time cycle, i.e., synoptic $T \rightarrow A$ tasks where T stands for a time cycle.

Figure 7.5 shows the impact of time transformation by aligning the start times of the trajectories. This transformation allows us to see that all trajectories have similar patterns of speed dynamics over time: very low - low - very low values in the beginning, followed by rather high speed later, and ending with rather low speed. The display also supports the *color filtering* introduced in Section 7.3.2, which enables the user to focus the behavior analysis on particular value ranges (see bottom bands in Figure 7.1).

To facilitate $S \times T \rightarrow A$ tasks, the time graph is dynamically linked to the map in the trajectory wall. This means that highlighting a trajectory segment in the time graph will highlight the same segment, and hence its spatial position, in the trajectory wall. The linking is bi-directional and enables elementary support in respect to S.

Note that a direct integration of spatial aspects into the time graph (similar to the lens tool of the trajectory wall) is hardly possible for the following reasons. First, space has no natural ordering which could be reflected by ordering of the trajectory bands. The bands may be ordered by the average or minimal distance to a selected point or object in space, but this is only a very limited part of the spatial context. Second, adequate representation of positions in the geographical space requires providing a recognizable cartographic background for reference, and this needs sufficient space. Hence, there is no good way to aggregate the space into a small display element directly integrated in the time graph display. Nonetheless, the coordination mechanisms between the spatial and temporal displays are fully sufficient for establishing connections between space and time.

7.3.4 *Approach Summary*

With the trajectory wall and the time graph, we have described two complementary visualizations that focus on $S \rightarrow A$ and $T \rightarrow A$ tasks, respectively. In order to support the analyst in making spatio-temporal findings with regard to $S \times T \rightarrow A$ at the elementary and synoptic level, time and space must be linked appropriately.

Therefore, both visualizations are integrated into an infrastructure that provides additional user controls and bi-directional coordination

between spatial and temporal displays. Our infrastructure enables analysts to select trajectories (by clusters or by spatial, temporal, and/or attributive queries) and to order them according to space, time, and attributes, and the visualizations react to these operations dynamically. The class intervals and colors are coordinated as well to keep them consistent throughout all displays.

To further facilitate understanding, we provide direct access to attribute values and additional textual information by mouse over. An interactive legend highlights the class of the currently focused trajectory point, which makes it easier to associate colors with class intervals. By clicking the legend the analyst can filter out classes that are less relevant to the task at hand.

So far we have not yet discussed the aspect of multi-attribute analysis. To deal with multiple attributes, two approaches are possible. One approach is to visualize and explore each attribute separately. Our infrastructure allows for multiple instances of the visual displays, each with a distinct spatio-temporal attribute. This way, a few attributes can be investigated simultaneously, for example to compare them.

The other approach to explore the spatio-temporal behavior of multi-attribute value combinations is to cluster these combinations by similarity and represent the cluster membership of the trajectory points again by color-coding. Our infrastructure provides a variety of clustering methods for this purpose, and includes tools to establish the correspondence between the clusters and the attribute values (e.g., with color-coded parallel coordinates plots, frequency histograms, maps, or space-time cubes).

Overall, our solution provides a comprehensive set of tools to facilitate the exploration and analysis of attribute data of several hundred trajectories. The usefulness of our approach will be demonstrated by several examples in the next section.

7.4 EXAMPLES

In this section we demonstrate how the stacking based visualization approach can be used to gain insight into trajectory attribute data related to radiation surveillance, traffic analysis, and maritime navigation.

7.4.1 *Data Set 1: Radiation Measurements in Japan*

The Fukushima Daiichi nuclear disaster is a series of equipment failures, nuclear meltdowns, and releases of radioactive materials at the Fukushima-I Nuclear Power Plant, following the earthquake and tsunami on 11 March 2011. A community of volunteers created a sensor network for collecting and sharing radiation measurements to empower people with data about their environment (see [8]).

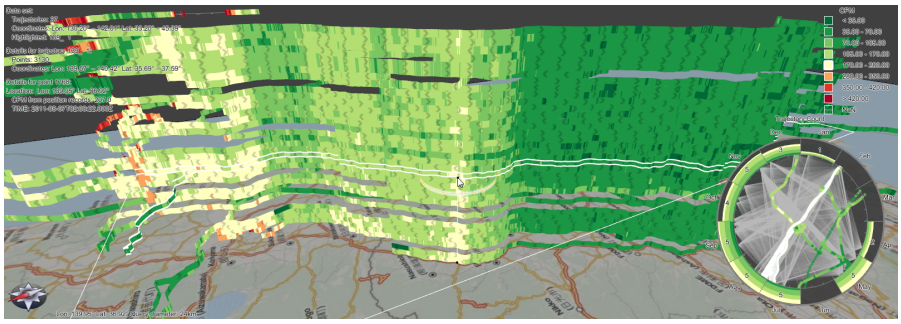


Figure 7.6: Visualization of radiation (CPM values) along the Tokio-Fukushima highway.

While the situation in major cities and around the station is carefully controlled by the authorities, it is also important to understand how radiation is developing on the neighboring roads. Data for the roads are collected by mobile sensors installed in the cars of the volunteers. The result is trajectories with radiation values attached to the position records. A publicly available data set consists of 1,014 trajectories including 1,936,261 measurements of so-called counts per minute (CPM).

The available data allow, in particular, the characterization of the radiation behavior along the major highway connecting Tokyo and Fukushima. After selecting trajectories along the highway using a spatial query, the CPM values have been shown in the trajectory wall (see Figure 7.6) using the class intervals and color scale suggested by experts. The bands are chronologically ordered from bottom to top. The time lens is organized by months, from April 2011 till January 2012. The view has been rotated for maximum visibility of the trajectory wall.

The following behaviors can be observed. There is a spatial trend of the values increasing as the distance to the station decreases ($S \rightarrow A$), which is observed at all times. The values in different places at medium distances from the station (from 25 to 75km) tend to decrease over time ($T \rightarrow A$). Closer to the station this temporal trend is less prominent ($T \rightarrow A$). In the places that are farther from the station the values are constantly low ($T \rightarrow A$). Hence, the overall spatio-temporal behavior ($S \times T \rightarrow A$) is that the radiation increases with approaching the station and decreases over time at medium distances from the station while being constantly low at farther distances and constantly high closely to the station.

7.4.2 *Data Set 2: Traffic Jams in San Francisco*

This example demonstrates behavior search. The goal is to detect traffic jams on a highway connecting San Francisco downtown with the international airport and locate them in space and time. We use a

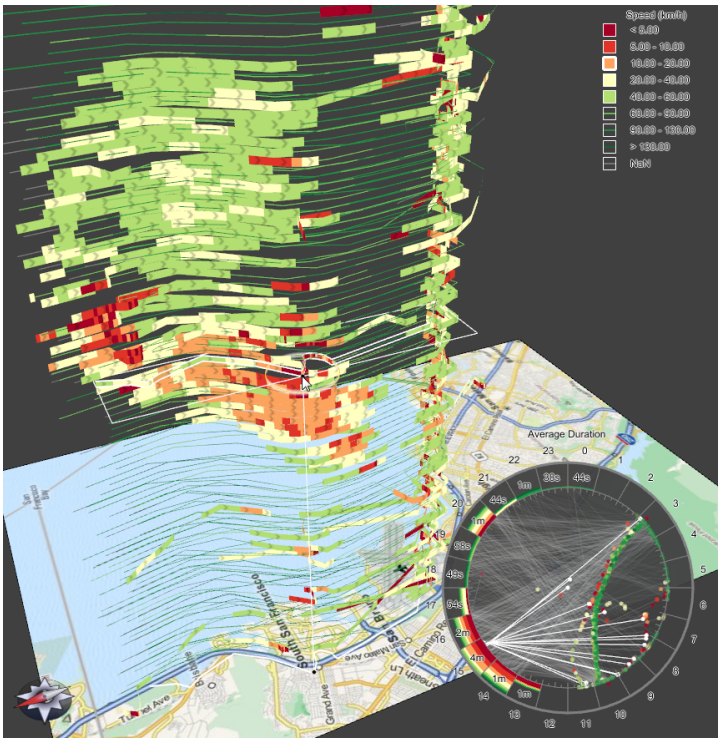


Figure 7.7: The development of a traffic jam in San Francisco.

publicly available data set [270] consisting of tracks of 535 taxi cabs during 4 weeks from 2008-05-17 till 2008-06-10 with 10,564,877 position records in total.

Using the controls provided by our infrastructure, the analyst has filtered the trajectories by visited regions (entrance to the highway and airport area) thus extracting 14,646 trajectories. The speed along these trajectories has been displayed in time graph and trajectory wall displays with class breaks at 5, 10, 20, 40, 60, 90, and 130km/h.

Next, the analyst searches for a particular behavior of interest ($S \times T \rightarrow A$) corresponding to a traffic jam: the speed decreasing to very low values in some part of the highway in multiple temporally close trajectories but not over the whole time. To facilitate the detection of this behavior, the analyst filters out the high speeds (more than 60km/h). Using temporal focusing, the analyst sequentially checks weekly portions of the data. Low speeds are observed in the area close to the airport but they occur all the time; hence, this is not the behavior of interest. The analyst also detects temporally limited "spots" of low speeds in other parts of the road, which signify the traffic jam behavior. Such spots occur once or twice per working day. The most prominent traffic jam happened around 16:00 on Friday, 2008-06-06. Figure 7.7 shows 92 trajectories that passed the highway from 2PM till 5PM. The traffic jam began between South San Francisco and San Bruno. Later the situation has improved in the southern part, but the traffic jam has extended northward towards Brisbane.

The time lens shows that the cars in the selected query area (see circle around the mouse pointer in the center of the figure) used on average 5 minutes to cross the area of about 2km diameter. In a similar way, the other occurrences of the traffic jam behavior can be detected, located in space and time, and characterized in more detail.

Further investigation supported by navigation in the trajectory wall reveals how some taxi drivers tried to bypass traffic jams by taking alternative routes. Most of these attempts had no success. However, one of the drivers (highlighted under the mouse pointer) successfully used an alternative road in South San Francisco.

7.4.3 *Data Set 3: Vessel Traffic in the Harbor of Brest*

This example is meant to demonstrate behavior comparison. The data set with vessel trajectories in the harbor of Brest (France) has been collected and kindly provided by Dr. C. Ray, Naval Academy Research Institute. There are 4,137 trajectories (see Figure 7.8) consisting of 782,404 position records for the period from 2009-02-11 till 2009-12-20. The analyst is interested in the spatio-temporal behavior of the movement tortuosity. The tortuosity values are computed from the position data using a sliding window of 1 minute. High tortuosity means frequent change of the vessel heading. Such situations are unpleasant for the passengers and may indicate dangerous situations.

By observing the trajectory wall for all trajectories, the analyst notices that the tortuosity is usually low on the lanes between the ports and high values mostly occur near the ports in small segments of trajectories ($S \rightarrow A$). However, there are outliers, i.e., trajectories with long segments of high tortuosity. One such trajectory is highlighted on the map (in black in Figure 7.8 right). The zigzagged character of the movement is well visible. Possibly, the vessel had some technical problems.

The analyst then focuses separately on each port to investigate the tortuosity in its vicinity and, more specifically, to compare the tor-

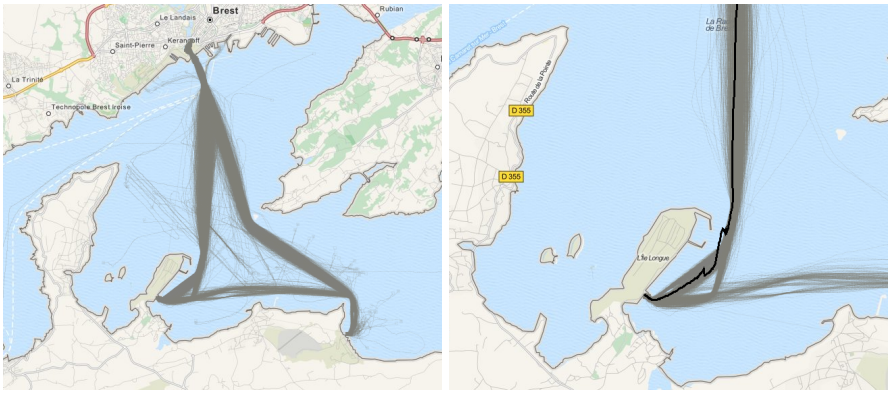


Figure 7.8: Traffic in the Brest harbor (left) and in its south-west part (right).

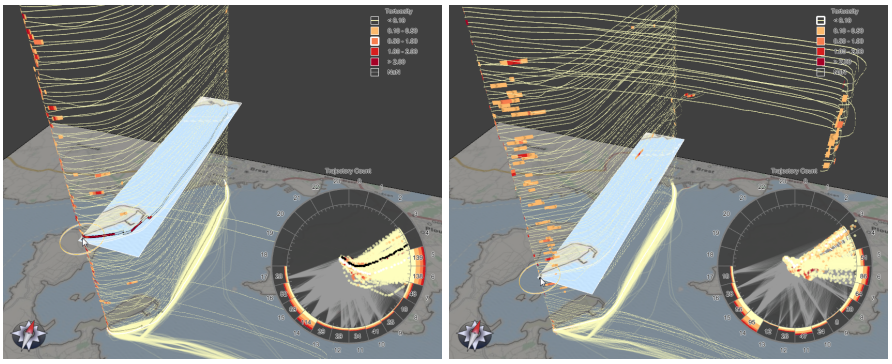


Figure 7.9: Comparison of the tortuosity of incoming (left) and outgoing (right) vessels at the Ile Longue peninsula in the Brest harbor.

tuosity behavior in the subsets of incoming and outgoing trajectories. We show the analysis by example of the port on the Ile Longue peninsula (southwest). By filtering the trajectories using a spatial directional query, the analyst selects two subsets: 1,433 incoming and 1,413 outgoing trajectories. These subsets are visualized in two different instances of the trajectory wall display in Figure 7.9.

Since high tortuosity values are of primary interest, the analyst decreases the visual prominence of the segments with low values by means of color filtering in the legend. The displays clearly show how the spatial behaviors of the tortuosity ($S \rightarrow A$) differ for the incoming and outgoing vessels. For the incoming vessels (see Figure 7.9 left), high values usually occur closely to the destination point. For the outgoing vessels (see Figure 7.9 right), high values occur at about 1km distance from the port. The segmented time lenses show us that both patterns are stable in time. However, at some time periods (6AM-8AM, 2PM-4PM) the traffic is more intensive and, respectively, the high tortuosity values are more frequent, especially in the afternoon. We can also see that the connecting lines accumulate at specific minutes at the time scale, which indicates that vessels follow a defined time table.

The examples demonstrate that our visualization design is useful for gaining insight into spatio-temporal behaviors of various kinds of attributes associated with trajectory positions. The interaction capabilities support the analyst in exploring the data with respect to different subsets of space, time, and attributes.

Unfortunately, in the available data sets we could not find interesting behaviors of combinations of multiple attributes. However, this does not cancel the principal possibility of doing multi-attribute analysis using multiple views or clustering.

7.5 USER FEEDBACK

In order to evaluate the usability of our solution, we conducted a small study. Our main intention was to investigate how easily and conveniently users can explore the hybrid 2D/3D trajectory wall given the implication of 3D as discussed in Section 7.3.3.1. Because the other parts of our solution are based on accepted designs we did not integrate them in the study as well.

For the study, we recruited 15 participants (2 female, ages 32 and 35; 13 male, ages 27–47) from the computer science department. 5 participants considered themselves experienced in visualization, the others had little or no exposure to this field. All participants were confident with mouse and keyboard interaction, 11 participants were familiar with 3D software (mostly from 3D games). None of the participants had used our techniques before.

Each session of the study started with an explanation of the trajectory wall and a demonstration of our focused zoom, pan, and orbit interaction as well as the elevator interaction, which were also summarized on a reference handout. Then the participants were asked to apply the interaction techniques to freely navigate around a trajectory wall showing "speed" of 250 clustered GPS-tracks of daily commuting. During the study, one experimenter encouraged the participants to "think-aloud" about what they were doing, why they were doing it, and how they would like to do it. The experimenter took notes of the participants feedback and provided assistance when it was needed. The participants were explicitly asked to include any negative comments they might have. At the end of each session, the participants had to answer additional Likert-type questions regarding the ease of use, helpfulness, smoothness, and learnability of the interaction. Individual sessions took between 15 and 20 minutes.

The feedback of the participants was mostly positive. 13 out of the 15 participants agreed that our focused zoom, pan, and orbit interactions made it easy to explore the trajectory wall. The idea of using a focus point for the interaction was immediately clear to the participants. Several participants highlighted the particular importance of the elevator navigation along the stacked trajectories. Although we had many participants who were used to classic 3D fly-through navigation, none of them had difficulties using our solution.

13 participants agreed that the switch between 3D trajectory bands and 2D paths on the map is easy to understand. All acknowledged the usefulness of the highlighting plane with the projection of the focused trajectory point onto the map. When being presented with the option of showing a map duplicate (as shown in Figure 7.3), some participants saw this as a useful approach, others were concerned about the occlusion of the wall caused by the additional map.

All participants answered that the interaction was easy to learn; some participant spontaneously said that the interaction "is intuitive and works very much as expected". The implementation was positively rated by all participants as being smooth and fluid.

The participants also made constructive suggestions for improvements. 4 participants commented that the orbit was too sensitive to mouse movement, among them were the two participants who rated the interaction as not being easy. This problem can be corrected by providing user-adjustable interaction speeds. One participant mentioned that the 3D stacking elevator would be even more useful if it had a design similar to scroll bars. This would make navigation across larger distances easier. As a solution, we could enhance the elevator with an in-situ scrolling widget.

Four participants noted the absence of the possibility to temporarily lock the focus of the highlighting. A particularly interesting suggestion was made by one participant who felt it difficult to use the mouse to follow a trajectory precisely. His idea was to lock the trajectory highlighted in the wall, and to project all further mouse movement to this locked trajectory. This way he could traverse a path without accidentally switching to another trajectory.

Overall we can conclude from the user feedback that the 2D/3D approach can be operated quite well using the provided interaction techniques. Several participants made impromptu comments about the visualization as well, including "I can easily compare the trajectories in the wall." or "This looks like stop-and-go traffic, maybe there is a traffic light or a construction site.". We take these comments as an additional indication that our solution is useful when analyzing trajectory attribute data, as already demonstrated in the previous section.

7.6 CONCLUSION AND FUTURE WORK

We presented a novel visualization approach that facilitates gaining insight into trajectory attribute data. By integrating spatial and temporal displays, we support exploratory spatio-temporal analysis. Our visualization design is based on color-coded trajectory bands, by which we address elementary tasks, and on stacking the bands, by which we address synoptic tasks. The usefulness of our solution has been demonstrated by several examples and usability has been tested in a small experiment. We can conclude that the novel approach is indeed useful and usable.

Yet there is potential for improvement and future research. Our current solution works well with hard- and soft-constrained trajectories with similar geometry. As soon as the tracked movements become less and less constrained or even chaotic (e.g., for particle movements at the cellular level), clustering methods and so our visualization ap-

proach will produce worse results. More research is needed to find ways to cope with such data.

An interesting aspect in terms of the visual representation is to investigate further the combination of time and space. So far we use spatially separated views for full-detail time and space. We plan to integrate both views into a single display and use temporal separation instead (i.e., show one after the other). To this end, we could use dynamic transitions to deform bands smoothly between a spatial layout and a temporal one.

Additional interaction mechanism can be considered to better handle larger data sets. We imagine tools that allow analysts to interactively collapse and expand groups of trajectories (user-specified or computed), similar to collapsing and expanding nodes in a regular tree view. This idea requires enhancing the data model from a linear stack to an ordered hierarchy of groups, and enhancing the design of bands, because a single band must then be capable of showing the paths of multiple grouped trajectories.

Furthermore, the exploration process deserves more research. So far our solution leaves the decision which parts of the data to analyze with which visualization entirely to the analyst. To ease the decision making it is worth investigating guiding principles based on a more detailed study of analysis tasks and characteristics of the trajectory data. In combination with the previously mentioned ideas, we could then compute and present "grand tours" through space and time.

Finally, we encourage geo-science researchers and usability experts to use and evaluate our approach to identify and quantify its strengths and weak spots. An interactive prototype is readily available [338].

7.7 SOME DETAILS ON THE DYNAMIC SPATIAL QUERY

The dynamic query area is defined by two mandatory parameters related to the spatial domain S and two optional parameters related to the stacking order. The mandatory parameters *position* and *radius* define a query circle $S' \subset S$. The optional parameters *z-index* and *height* define a query cylinder to select only specific trajectories from the stack.

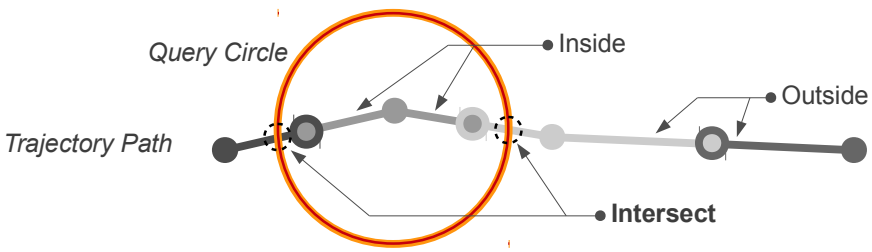


Figure 7.10: Cases of path-segment-circle intersections.

Testing a trajectory $\mathbf{d}$ against S' is based computing line-circle-intersections for the $2l_{\mathbf{d}}$ path segments of the trajectory. Fig. 7.10 illustrates the three different cases that need to be handled. For the case *outside*, nothing needs to be done. For the case *inside*, the corresponding trajectory segment is passed to the temporal aggregation. The *intersect* case requires cutting off the part of the path segment that is outside, where the precise intersection point (incl. coordinates, time, and attribute values) is computed by interpolation. Then the situation is a regular *inside* case. Optionally, the number of required line-circle-intersection calculations can be reduced by first checking if the index of $\mathbf{d}$ in the stacking order falls in a range $z\text{-}index \pm height/2$, as defined by the query cylinder.

8

INTERACTION SUPPORT FOR VISUAL COMPARISON INSPIRED BY NATURAL BEHAVIOR

CONTRIBUTION The contribution of this chapter is a general interaction design to support higher-level visual comparison tasks. Three methods, side-by-side, shine-through, and folding are combined in an comprehensive interaction approach that is broadly applicable for various types of data. The individual interaction techniques are virtual replications of natural comparison behavior. This makes them easy and intuitive, as documented by the results of a user study.

ABSTRACT Visual comparison is an intrinsic part of interactive data exploration and analysis. The literature provides a large body of existing solutions that help users accomplish comparison tasks. These solutions are mostly of visual nature and custom-made for specific data. We ask the question if a more general support is possible by focusing on the interaction aspect of comparison tasks. As an answer to this question, we propose a novel interaction concept that is inspired by real-world behavior of people comparing information printed on paper. In line with real-world interaction, our approach supports users (1) in interactively specifying pieces of graphical information to be compared, (2) in flexibly arranging these pieces on the screen, and (3) in performing the actual comparison of side-by-side and overlapping arrangements of the graphical information. Complementary visual cues and add-ons further assist users in carrying out comparison tasks. Our concept and the integrated interaction techniques are generally applicable and can be coupled with different visualization techniques. We implemented an interactive prototype and conducted a qualitative user study to assess the concept's usefulness in the context of three different visualization techniques. The obtained feedback indicates that our interaction techniques mimic the natural behavior quite well, can be learned quickly, and are easy to apply to visual comparison tasks.

ORIGINAL PUBLICATION [336] — C. Tominski, C. Forsell, and J. Johansson. Interaction Support for Visual Comparison Inspired by Natural Behavior. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2719–2728, 2012.

8.1 INTRODUCTION

Visual comparison tasks take a central role in visual data exploration and analysis. By comparing and relating different parts of the data in detail, users may formulate, confirm, fine-tune, or reject initial hypotheses, and thus can gain a better understanding of the data.

Gleicher et al. [140] argue that appropriate support is needed to facilitate visual comparison in information visualization. Existing solutions support comparison tasks mainly by visual means, including special visual encodings (e.g., visualization of differences) and special visual layouts (e.g., side-by-side views). Because visual exploration is a dynamic process where users repeatedly identify parts of the data to be compared, it is also necessary to provide dedicated interaction support.

With this work, we contribute a novel interaction concept and supplementary visual cues and add-ons to support visual comparison. Our concept has been designed so as to facilitate three phases common to most comparison tasks: (1) *selection* of pieces of information to be compared, (2) *arrangement* of the pieces to suit the comparison, and (3) carrying out the actual *comparison*.

The approach we present here is inspired by real world interaction as people may perform it when comparing information printed on paper [181]. In an initial step, people usually pick from a pool of papers a few sheets to be studied in detail. Common ways of comparing the selected sheets are illustrated in Figure 8.1. One approach is to lay out the sheets *side by side* and look at them alternately. Additionally, when comparing graphical depictions such as line drawings or simple charts, people may stack a few sheets of paper and hold them against the light to let the information hidden on the back sheets *shine through* and merge with the information on the front page. Alternatively, when comparing such overlapping, but in other respects well-arranged prints, people often *fold* pages back and forth in quick succession without otherwise disturbing the arrangement to reveal and relate the information shown on the different sheets. These behaviors can be observed in a multitude of scenarios, for example

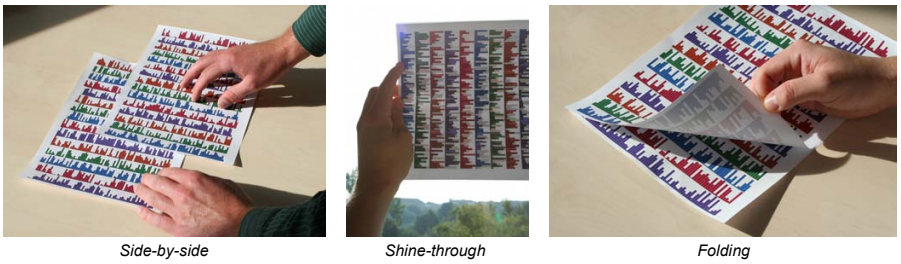


Figure 8.1: Natural behaviors observed when people compare information printed on paper: side-by-side arrangement, shine-through, and folding.

when friends compare photographs, when decision makers contrast figures and tables in reports, or when artists draw the frames of cartoon animations. Our goal was to design interaction techniques for visual comparison that resemble these natural behaviors as closely as possible.

The proposed solution enables users to dynamically specify pieces of graphical representations as the visual objects to be compared. We call these visual objects *views*. Depending on the data and the task, users can arrange views freely on the screen, for example to put them *side by side* or to let one view lie on top of another, if this better suits the comparison. To support users in comparing overlapping views we provide the *shine-through* and the *folding* interaction techniques. Supplementary tools help users manage the views and indicate computed differences to assist in the data analysis where possible. Our approach is generally applicable and can be combined with existing visualization solutions. Design requirements and goals, and the corresponding solutions integrated in our approach will be described in detail in Section 8.3.

We expected that our novel interaction techniques are intuitive thanks to their real-world origin. The anticipated benefit for the users is that comparison tasks feel natural and hence are easy to accomplish. In Section 8.4, we present the results of a user study indicating that the developed solution meets our expectations. In the first part of the study, the participants were able to quickly learn how to use the new interaction techniques. In the second part, we studied the developed techniques for selected visualization scenarios and found that they indeed facilitate the addressed tasks.

In the next section, we will first introduce the general background of our work and briefly review related approaches from the literature.

8.2 PROBLEM DESCRIPTION AND RELATED WORK

Let us begin with a motivating example that dates back to a collaboration with scientists from the bio-informatics department. The scientists visualized clustered multivariate data in a table lens, which enabled them to spot local trends and patterns [193]. For comparing the local phenomena, the scientists extensively used manual scrolling and had to rely on their short-term memory. First they scrolled to a pattern and memorized it and then they navigated to another pattern and compared that to the stored mental image of the first one. The scientists experienced this procedure as inefficient because it requires to scroll over and over again, and as error-prone because the actual comparison is carried out based on a mental image.

So, our goal was to come up with a concept that would allow the scientists to conduct the comparison more efficiently. This specific goal of supporting the comparison of local phenomena in a table lens

can be translated into the more general goal of providing interaction support for visual comparison for a wider range of visualization techniques.

8.2.1 *Problem Description*

What precisely do we mean when speaking of visual comparison? Andrienko and Andrienko [36] provide formal definitions of several variants of this visualization task. In a most general interpretation, *comparison* can be understood as the task of formulating a relationship that holds for particular subsets of the data. On the other hand, *relation seeking* is the task of searching for subsets in the data that match with a predefined relationship. According to this thinking, the only difference between comparison and relation seeking lies in what is given and what is sought (relationship or data subsets). Hence, many practical solutions do not make this distinction between comparison and relation seeking, but subsume both tasks under the common term *visual comparison*. So do we in this work.

For exploratory data analysis, the distinction between comparison and relation seeking is further blurred, because there are usually no a-priory assumptions neither about the data subsets to be compared nor about the specific relations to be tested. In such a setting, users may at all times identify interesting data subsets to be studied in detail (e.g., individual data values or clusters of values) or potential relations to be tested (e.g., comparison of outliers, comparison of the slopes of trends, or comparison of the shapes of clusters). The actual comparison can mean that users formulate a relation based on the data (e.g., a set of values exceed a threshold) or that they confirm or reject a supposed relation (e.g., the cluster shapes are indeed identical).

This multitude of aspects makes developing general support for visual comparison a challenging endeavor. Therefore, we simplify the problem by abstracting from the specific details of the data to be compared, be it tuples in a table, nodes in a graph, sequences of genomes, or whatsoever. Our assumption is that suitable visualization techniques are available to generate appropriate visual representations of the data. Under this assumption we can resort to *pieces of graphical information* as the visual objects to be compared. Delegating the task of dealing with the specific semantics of the data to the visualization allows us to focus on the design of the interaction.

8.2.2 *Related Work*

As previously described, we aim to develop interaction techniques that support users in visually comparing pieces of graphical information. Work that is related to this goal will be briefly surveyed next.

INTERACTION IN VISUALIZATION Spence [310] describes visualization as a tool to support people in forming mental models of otherwise difficult-to-grasp complex data. The fact that people *form* mental models implies that interacting with the visual output and with the data is a vital aspect. The importance of interaction has been advocated by several researchers, including Thomas and Cook [325], Pike et al. [269], and Fekete [118]. We understand our work as a contribution to the interaction side of visualization.

There are a variety of reasons why users want to and need to interact. Yi et al. [385] categorize seven key user intents for interaction. These intents are supported by a large number of available interaction techniques, where direct manipulation [303] appears to be the preferred underlying design. Among the available techniques are classic brushing and linking to mark interesting parts in the data (e.g., [52, 155]), navigation techniques to assist users in exploring larger information spaces (e.g., [109, 256]), techniques to manipulate the layout and adapt the encoding of visual representations (e.g., [252]), or interactive lenses to provide locally adapted information where it is needed (e.g., [177, 111]).

SUPPORT FOR COMPARISON TASKS Due to its central role, visual comparison is addressed by a number of approaches. Here we can list only a few examples. Munzner et al. [258] focus on guaranteed visibility to support the comparison of hierarchically organized data. Holten and van Wijk [172] use bundled edges to explicitly link similar regions in the data. Tominski et al. [331] support the comparison of multivariate data in general by specifically designed color-scales. Jiang et al. [192] support comparison across application boundaries by integrating multiple views in a single shared display.

Far more examples are given by Gleicher et al. [140], who list over 110 references in a survey on visual comparison for information visualization. To establish a meta view of this immense solution space, Gleicher et al. propose a general taxonomy of visual designs for comparison (which also abstracts from the concrete data being compared). The three fundamental categories in this taxonomy are *juxtaposition*, *superposition*, and *explicit encoding*, which can further be combined to form hybrid approaches.

Although Gleicher et al. focus on *visual* techniques, they also mention *interaction* as invaluable tool to assist in the comparison. Two general examples of commonly provided functionality are given: (1) techniques to make connections between related objects (e.g., interactive highlighting) and (2) techniques to reorder and rearrange objects. These two examples correspond to the *connect* and the *reconfigure* categories of Yi et al.'s [385] taxonomy of interaction intents.

Interestingly, Yi et al.'s work does not contain a separate category for interaction techniques for visual comparison. The authors argue

that to compare something can mean so many different things, making it difficult to uniquely distinguish comparison tasks from other interaction intents. We could not single out interaction techniques that are explicitly designed to support visual comparison either. Often the interaction techniques are tightly integrated as part of the visualization. Our approach is different in that we aim to abstract from the particular type of a visualization and focus on a general interaction concept, thus filling a gap in the literature.

Yi et al.'s argumentation is also an indicator that a single technique alone will not suffice to support visual comparison. Instead, multiple interaction techniques need to work in concert to support all the aspects of visual comparison. Our solution accounts for this requirement by combining interaction techniques inspired by natural behavior and supplementary visual cues and add-ons.

NATURALLY INSPIRED INTERACTION One can find different themes in the literature that address naturalness as a key to efficient interaction between the human and the computer. A classic and still relevant theme is *direct manipulation* [303], that is, the direct interaction with the graphical representation on the screen. *Tanglible interaction* [184] is based on more natural interaction with tangible objects in the real world. *Instrumental interaction* [50] conceptualizes the idea of naturally using interaction instruments to manipulate domain objects.

Reality-based interaction [187] and *natural interaction* [348] are the next steps in a line of recent developments that include aspects of greater awareness of the user and the environment in which the interaction takes place. Typically such interactions are applied in scenarios with large or multiple displays, where awareness is achieved through detection and tracking approaches (e.g., detection of hand gestures [359] or tracking of position and viewing direction of the user [230]).

Also notable is the recent progress in utilizing interactive surfaces to facilitate visualization [182]. The advantage of multi-touch surfaces is that the interaction is intuitive because it is much like working with objects on a table. One particularly related instance is Isenberg and Carpendale's [179] work on collaborative comparison of trees.

Although the inspiration for our solution comes from natural behavior, we should clearly distinguish our work from other areas that strive for naturalness in interaction. The bulk of visualization applications still resides in the realm of desktop computers, where expensive tracking systems and multi-touch devices are not yet commonplace.

Therefore, we aim to develop a general interaction concept. We think that being independent of the underlying technology is beneficial, as it allows for later implementation of the concept in different contexts using the technology that is most suitable, be it natural inter-

action, multi-touch interaction, or classic mouse and keyboard interaction. Nonetheless, we hope that the novel approach presented next contributes to more naturalness of interaction in visualization.

8.3 INTERACTION SUPPORT FOR VISUAL COMPARISON

In the following, we elaborate on our concept of interaction support for visual comparison tasks. Starting with a set of design goals, we develop the basic environment into which to embed our new techniques. Then each individual technique will be explained in more detail.

8.3.1 *Design Goals*

As mentioned before, visual comparison tasks are usually carried out in three phases: (1) selection of pieces of information, (2) arrangement of the pieces, and (3) actual comparison. Theses phases are to be supported by interaction techniques that borrow ideas from natural comparison of information printed on paper. According to this thinking, our approach has to fulfill three specific requirements R1-R3:

R1 INTERACTIVE SPECIFICATION OF COMPARISON OBJECTS: During exploratory analysis of unknown data there are usually no a priori restrictions on what may constitute interesting candidates for the comparison. Therefore, users must be enabled to flexibly specify what they want to compare. While we can assume that a good visualization already helps the user in spotting interesting candidates for the comparison, it makes sense to provide additional support to aid in the search (e.g., by showing aggregated differences).

R2 INTERACTIVE RELOCATION TO SUIT COMPARISON: It may very well be that the candidates selected for comparison are located in different parts of the display or are even not visible at the same time (e.g., when comparing the first and the last rows of a scrollable table visualization). This can severely impede the comparison because the eyes have to travel larger distances and because the user has to temporally memorize parts of a possibly complex visual representation (see Lind et al. [234] and Alvarez and Cavanagh [23]). Therefore, the selected parts of the visualization must be relocatable dynamically to make the comparison easy to carry out.

R3 INTERACTIVE RESOLVING OF OCCLUSION TO FACILITATE COMPARISON: Even with a side-by-side layout the comparison might be demanding because the eyes still have to move from one part to the other part. To minimize the eye movement, the objects being compared ideally would have to be superimposed one

on the other. But in this case, the occlusion of possibly relevant graphical information can be disadvantageous. Therefore, appropriate interaction techniques are needed to dynamically unhide occluded parts. Fulfilling this requirement will enable the user to balance the comparison process in between the extremes: much eye movement vs. no eye movement and no occlusion vs. full occlusion.

While the requirements R1-R3 are specific to our work, there are also some general interaction goals G1-G4 we should strive to achieve:

G1 MIMIC NATURAL BEHAVIOR: Our major goal is to mimic real world behavior observed when people compare information printed on paper. We expect that building upon a natural origin is beneficial for interaction support for visual comparison tasks. Achieving this goal requires developing a realistic look & feel. The major challenge will be to design the approach so that it bridges the gap between the natural interaction and the interaction modalities available on computer machinery.

G2 FOSTER FLUID INTERACTION: A second goal is to design the approach so that it is easy and enjoyable to use, where "easy" and "enjoyable" can be associated with usability and user experience, respectively. A related aspect is what Elmqvist et al. [112] recently coined "fluid interaction for information visualization", which comprises aspects of promoting flow, direct manipulation, and smooth interaction-feedback cycles.

G3 PROVIDE COMPUTED ASSISTANCE: Third, we strive to go beyond what is possible in the real world and provide "computed" assistance. Our solution is to be complemented with appropriate tools that help users in accomplishing comparison tasks. However, the augmentation should be balanced and must not interfere with the aims for naturalness and fluidity, which forbids integrating computationally complex calculations.

G4 PROMOTE GENERALITY: Finally, we aim to develop an approach that is general in terms of applicability. We do not seek a solution for a particular visualization technique, but one that can be combined with many of the exiting visualization methods. Generality should also be achieved with regard to the technical realization. That is, our approach should be implementable for different interaction modalities such as mouse and keyboard interaction or multi-touch interaction.

With the specific requirements R1-R3 and the general goals G1-G4 in mind we designed the basic setup for our solution.

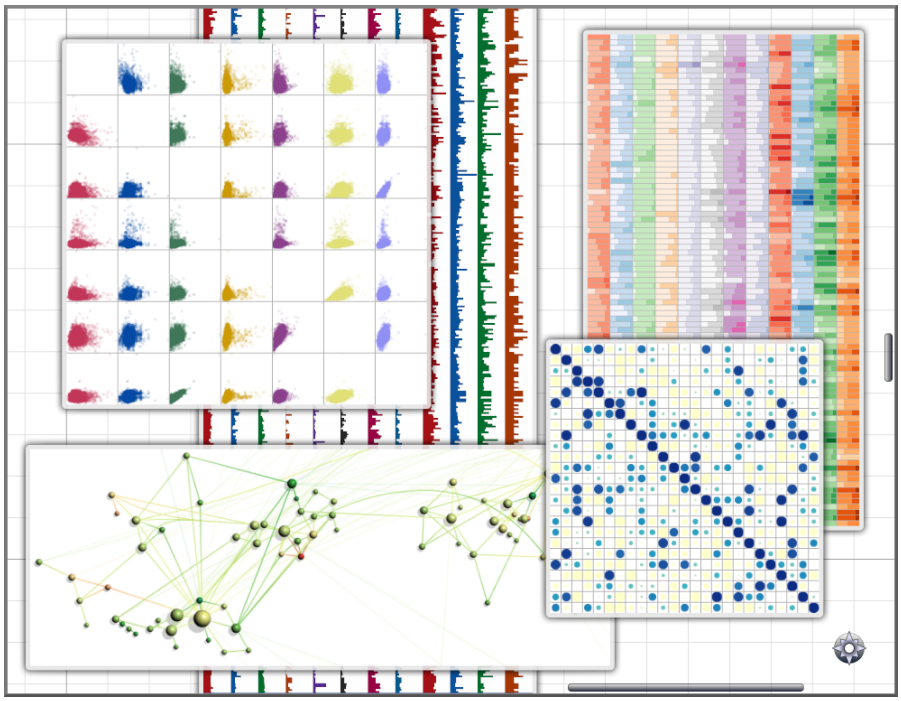


Figure 8.2: Illustration of the zoomable visualization space in which views show graphical representations of data.

8.3.2 *Basic Setup*

As our inspiration originates from natural work with sheets of paper, we need a corresponding virtual paper representative on the computer. For the sake of simplicity we define such representatives as 2D *views* that show a graphical representation of data, where we do not impose any particular restrictions on the type of graphical representation.

Much like in the real world, we need a work space in which views can be compared or analyzed for relations. In accordance with our design goals, we decided to build upon the notion of zoomable user interfaces (ZUIs). In addition to being engaging, visually rich, and simple, ZUIs facilitate overview and detail exploration of the information displayed in the views (see Bederson [53]). In this work, we define the *visualization space* as a virtual zoomable 2D space. This space can hold an arbitrary number of views. In a sense, we combine the advantages of ZUIs and multiple views (see Wang Baldonado et al. [363]) in a way similar to what was suggested by Plumlee and Ware [275]. Figure 8.2 shows an example of the visualization space with five views. For the purpose of illustration, the figure shows different and not necessarily related visual representations.

The classic way of navigating in such a visualization space is to apply zoom and pan operations. However, when multiple views are scattered across the space at different locations and zoom levels, man-

ual navigation can be time consuming and cumbersome [53]. Therefore, we integrate automatic navigation methods that enable the user to quickly travel between views without having to approach them manually. Our solution uses the smooth and efficient zoom and pan animation by van Wijk and Nuij [355] and the infinite grid by Tominski et al. [332] to help maintain user orientation when larger distances need to be covered.

This setup of views residing in a zoomable and animated visualization space is the basis upon which we develop the interaction techniques to facilitate visual comparison.

8.3.3 *Flexible Specification*

According to requirement R1, users must be enabled to specify which pieces of information they want to compare in detail. In the real world, people simply pick sheets of papers, notes, or photos depending on the task at hand [181]. In a regular visualization, the user has to identify and shape regions of interest mentally and keep the extracted information in mind throughout the course of the comparison, which can be quite difficult [23].

Our solution relieves the user from keeping too many things in mind: At any time during the exploration of the data, if the user spots something interesting it can be marked and the system creates a new view (a *sub-view* so to say) corresponding to the selection. Once created, a view is detached from its base view and shown as a full-fledged view in the visualization space. All views are collected in a *view hierarchy*, which stores the parent-child relationships of the views. This hierarchy enables us to keep track of created views, to maintain data integrity when views are removed, and to keep views visible on top of their parents.

The actual specification of a new view is based on drawing a selection shape (e.g., an elastic selection rectangle for our case of rectangular views) on top of an existing view. The selection shape is used to determine what to display in the new view. The selection can take effect in *image space* or in *data space*. For image-based view creation, the new view simply shows a copy of the graphical content within the selection shape. For data-based view creation, the selection shape is projected into the data space, where it is used to filter out all data that do not correspond to the selection. Only the selected data are then set as the input to be visualized in the new view.

Figure 8.3 on the next page shows rectangular views extracted from a node-link diagram. Because the selection was made in image space, we still can see edges that connect to nodes outside of the view. If we had used a data-based selection instead, the induced subgraph corresponding to the selection would not contain these edges.

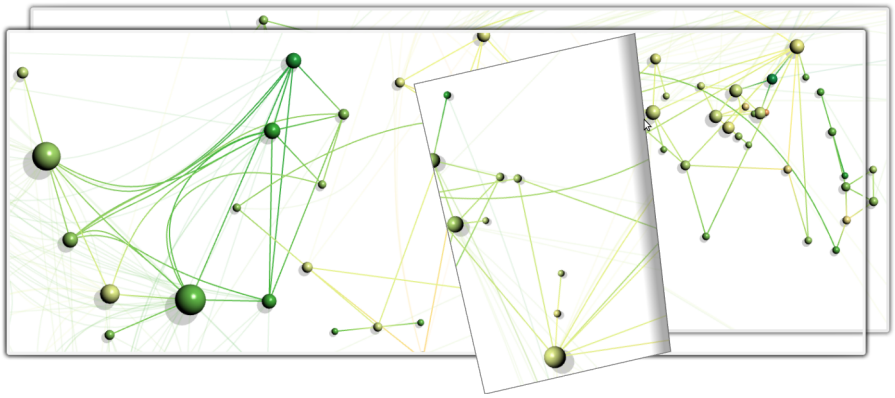


Figure 8.3: Folding interaction to reveal and relate information shown in overlapping node-link diagrams.

Our flexible specification mechanisms contribute to fulfill requirement R1. They have a clear advantage over natural paper comparison, where extracting and duplicating information is usually more complicated (e.g., create a hard copy and cut out pieces of interest). Next we provide the details on three interaction techniques that address the requirements R2 and R3.

8.3.4 *Interactive Arrangement*

Traditionally, visual comparison is supported by showing the information to be compared in a *fixed* layout (i.e., juxtaposition or superposition in [140]). Our solution is different in that we allow users to *flexibly* create layouts that best suit the comparison at hand. By simple drag and drop interaction users can arrange views in the visualization space as if they were shifting paper on a table.

In addition to translation, which is mandatory for side-by-side comparison, rotation and scaling can be useful, depending on the application scenario. For example, collaborative comparison on interactive surfaces requires rotation [179]. If comparison on different scales of the data is semantically meaningful, it can be useful to scale views independent of the already available zooming of the visualization space.

While fully flexible arrangement can be advantageous, it can also be tedious to control. In the real world, people use edges of papers or patterns on the table surface to guide the arrangement. In the virtual world, so-called *snapping* assists in the alignment of views to be compared. Depending on the data and the applied visualization technique, a number of alignments make sense, including grid-based, axis-based, object-based, and image-based alignment.

Grid-based alignment constrains the horizontal and vertical translation to multiples of a grid width and a grid height. This is useful for visualizations that construct regular arrangements of the data. Ex-

amples are matrix representations of graph data or table-based representations of multivariate data, where the matrix or table cells define the grid.

Axis-based alignment is useful for the many visualization techniques that display data along axes, where fixed (e.g., Kiviat graph [210]) or flexible layouts (e.g., FLINA [81]) are possible.

Object alignment is useful when comparing visual representations that contain visually distinguishable objects, such as glyphs, or clusters of nodes in a node-link diagram. If sufficient information (e.g., coordinates, object geometry, or bounding shapes) is available, one can employ the techniques by Bier [57] to compute a variety of guides to drive the snapping.

Where this is not the case, and in all other situations where plain image data is the only source for the snapping, image-based methods can be applied. Line detection, feature point detection, or difference images are simple examples. More complex mechanisms such as the gradient-based snapping by Gleicher [139] can provide better alignment assistance.

By providing the mechanisms for interactive arrangement we fulfill requirement R2. Users can place views side-by-side if it suits the data and tasks at hand. As with natural comparison of paper, it is also possible to let views overlap partially or to position views exactly on top of each other. However, the resulting occlusion renders any comparison impossible. Therefore, requirements R3 demands that users are provided with tools to look "through" or "behind" views.

8.3.5 *Shine-Through Interaction*

In the real world, it is quite common to purposely stack papers on top of each other and to resolve the occlusion by holding the papers against the light. The degree to which information shines through is controlled by altering the viewing direction with respect to the light source or by controlling the luminance of the light.

A common approach to momentarily reveal otherwise occluded information on a computer display is to apply see-through techniques (see Bier et al. [59]). We allow the user to temporally fade out views by alpha-blending with a variable level of transparency.

There are two ways to steer this process: the user can control the transparency manually or the fading animates automatically between fully visible and invisible. The former solution offers more control, but is more expensive in terms of interaction costs [221] as it requires entering a concrete alpha value. The animation is less expensive as it only requires triggering the animation (e.g., press button to fade out and release button to fade back in), but this also means less control. The time costs involved when rendering the animation can be restricted by choosing different animation speeds.

Designed in this way, the shine-through interaction fulfills requirement R3 and it corresponds to natural comparison behavior G1. In line with our experience from the real world, we found the shine-through quite useful, especially when comparing visualizations that employ the visual variables size, length, position, and orientation to encode data.

But there are also some issues related to alpha-blending. The first thing that comes to mind is the unfavorable blending of colors when comparing color-coded visual representations. An alternative solution would be to use weaving as suggested by Hagh-Shenas et al. [148] or Luboschik et al. [241]. However, with both blending and weaving another problem persists: it is difficult for the user to figure out which of the views being compared contributes to a particular feature in the blended or weaved image. This is only natural, because blending and weaving favor a merged view on the data at the cost of loosing separability of individual data items. The folding interaction described next addresses this aspect.

8.3.6 *Folding Interaction*

Folding back and forth to reveal information shown on different pages is a natural behavior when comparing overlapping pieces of information. This inspired us to develop an alternative interaction technique for comparing subjects that are superimposed one on the other. To uncover occluded information we allow the user to temporarily fold away or peel off views as if they were virtual paper.

Folding has been applied successfully for other purposes related to coping with occlusion (requirement R3). Beaudouin-Lafon [51] suggests using folding to assist in window management. Dragicevic [102] and Chapuis and Roussel [75] extend the basic folding to support drag and drop as well as copy and paste operations between overlapping windows. Furthermore, many online brochures allow users to flip pages, mainly for the purpose of being an engaging and fun-to-use website, which is related to our goal G2.

According to our research, the application of folding to support visual comparison in the realm of visualization is novel. Therefore, we describe it in a bit more detail in the next paragraphs.

The advantage of using folding for comparison is that the occluding view can be folded away temporarily, while otherwise keeping the views in place to preserve the arrangement initially created for the comparison. We designed the folding so that it occurs where the *focus* of the user is, usually determined by a primary pointer (e.g., mouse cursor or touch by index finger). To resolve the occlusion at that location (i.e., the folding point P), the user simply presses a trigger to initiate the folding interaction.

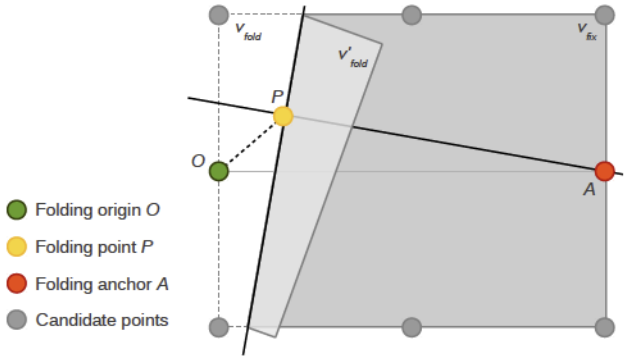


Figure 8.4: Folding based on the folding point P, and on a folding origin O and a folding anchor A chosen from a set of candidate points.

This is in contrast to common folding implementations, where the user has to move the pointer to grab an edge or corner in order to fold. We consider this harmful because such a movement disrupts the user's focus. Moreover, it could very well be, that the edges or corners of a view residing in our zoomable visualization space are not visible at all, making a corner-grab-based folding infeasible.

COMPUTING THE FOLDING GEOMETRY As illustrated in Figure 8.4, our solution folds directly at the folding point P, taking into account a folding origin O and a folding anchor A. In the real world, the folding origin O corresponds to the spot where we grab a page for folding it. The anchor A is the virtual counterpart of the fixture around which the paper is folded (e.g., a staple or the binding).

Because we do not want to force the user to move the pointer to either of the borders of the view to initiate the folding, we developed a simple heuristic to define O and A based on P as follows. First, we define a set of candidate points on the border of the folded view, where each such candidate has a corresponding point on the opposite side of the view (e.g., the corners and the centers of the edges). The second step is to compute the distance of P to each of the candidate points. We choose the closest candidate point as the folding origin O and the point that is opposite to it as the folding anchor A. Finally, the folding axis is constructed as a line originating at P and being perpendicular to the line PA.

If a secondary pointer is available (e.g., in multi-touch scenarios), there is no need to apply the heuristic. Instead, the primary and secondary pointers can be used to define the line PA directly, and with it the folding axis.

Using the folding axis, we can compute polygonal shapes $v_{\mathtt{fold}}$ and $v_{\mathtt{fix}}$ that correspond to the folded part of the view and to the part that remains fix. Further, $v_{\mathtt{fold}}$ is reflected in the folding axis to create the shape $v'_{\mathtt{fold}}$ representing the folded backside of the view. These shapes are needed to render the folding effect.

Figure 8.5: Different folding styles enable users to balance information-richness, naturalness, and the degree of occlusion.

RENDERING THE FOLDING EFFECT The goal of the folding is to unhide the information that would otherwise lie underneath $v_{\mathtt{fold}}$. To this end, when rendering a folded view, the visualization is cropped at the edges of $v_{\mathtt{fix}}$, effectively leaving $v_{\mathtt{fold}}$ blank. To generate appropriate visual feedback for the folding interaction, it is necessary to display the folded backside $v'_{\mathtt{fold}}$ instead. The visual effect of folded virtual paper can be achieved by rendering a white backside and attaching a shading gradient to the folding axis, which gives the fold a more realistic appearance.

However, this basic approach, although being quite natural (goal G1), is not very efficient, because it leads to self-occlusion (and possibly to occlusion beyond the folded view) without utilizing the screen space occupied for the backside for any additional gains. Therefore, we designed alternative rendering styles that aim to either use space more efficiently or reduce the occupied space.

The first alternative style enhances the visualization by utilizing $v'_{\mathtt{fold}}$ to display additional information (e.g., a reflected copy of the front side, an alternative visual encoding of the data, or an explicit representation of differences). The second style, shows $v'_{\mathtt{fold}}$ as semi-transparent shape to mitigate occlusion, while still indicating the folding. The third style shows only the shading gradient mentioned before and thus requires minimal screen space.

Figure 8.5 illustrates the different rendering styles for the folding effect. We can see that the styles vary in their naturalness, information-richness, and in the degree to which they account for occlusion concerns. By offering the different styles, we allow users choosing the one that best suits their preferences and balances the advantages and difficulties of the folding (e.g., some users might favor naturalness over information-richness, others might prefer sacrificing naturalness in favor of a minimalistic style).

ANIMATING THE FOLDING INTERACTION Using the computed folding geometry and the chosen style, we display the folding effect. But applying the folding in an instant once the user triggers the interaction would be unnatural. Depending on the size of $v_{\mathtt{fold}}$, the folding could affect a rather large part of the display, which is contrary to smoothness of interaction and might confuse the user.

Using a force-based animation will give the folding a realistic appearance and make the interaction easy to grasp and fluid to apply (goals G1 and G2). We model a simple spring-mass system, based on a point S, which describes the point where the spring is fixed, and a second point M, which defines a displaced mass that is attached to the spring. For each frame of the animation, we evaluate the forces affecting M and derive its new position, which we use in turn to construct the folding axis.

When the user triggers the folding, we set $S = P$ and $M = O$ which results in a smooth folding from the folding origin O to the folding point P. To smoothly unfold the view when the folding operation has ended, we do the inverse and set $S = O$ and $M = P$. During the folding, when the user moves the folding point, S is simply updated to $S = P$ to account for the movement.

The user can choose from three predefined animation speeds, which have been obtained by varying the spring constant, the damping factor, and the mass of M used for the spring-mass system. Note that the duration of the animation cannot be set directly, because the time required to reach equilibrium depends among other factors on the displacement. Therefore, bigger foldings take longer, but the animation is fluid and smooth at all times as demanded by goal G2.

The folding as described before is a helpful alternative to resolve occlusion (requirement R3) in situations where the shine-through interaction leads to unintended blending of information. In contrast to the shine-through, which affects a view globally, the folding interaction is focused and of localized nature. As our concept aims to be generally applicable (goal G4), both shine-through and folding will have their merits depending on the concrete application scenario.

8.3.7 *Visual Cues and Add-Ons*

With the tools introduced so far, we provide the essential interaction techniques required to support comparison tasks in information visualization. To increase the utility of our approach, we integrate supplementary visual cues and add-ons. The enhancements help users maintain overview and context, reduce interaction costs, and visualize computed differences.

VIEW HIERARCHY OVERVIEW To relieve users of keeping track of the created views and the information displayed in them, it makes sense to show an overview of the view hierarchy in a separate window (e.g., as a regular tree view as depicted in Figure 8.6). This display serves as an overview in which views can be annotated with captions to characterize them. According to Plumlee and Ware [275] (p. 183), making annotations is related to offloading cognitive costs from the visual memory to the verbal memory.

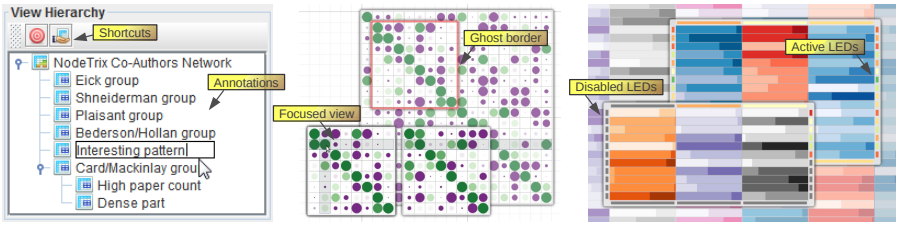


Figure 8.6: Additional tools provide further support to maintain overview and context, to simplify costly interactions, and to display computed differences.

The overview is also the place where the user can pick a view and trigger interaction shortcuts for otherwise costly interaction tasks. We provide two useful shortcuts. The *go-to* shortcut applies the smooth and efficient zoom and pan animation to center the visualization space's camera on the picked view. The *bring-in* shortcut does the inverse: it smoothly moves the picked view towards the center of the display. By using the shortcuts users can avoid repeated zoom and pan and drag and drop operations when exploring larger datasets.

GHOST OF ORIGIN Defining and arranging views freely (requirements R1 and R2) implicate that data are separated from their immediate context. In the real world, this is related to removing the documents to be compared from their trays. Unless documents and trays are appropriately labeled, it can be difficult to maintain their connection.

To better preserve the data context in our visualization space, we support interactive highlighting of the location where a view has originally been detached from its parent. To this end, we embed ghost borders into the display to mark a view's origin in a dimmed fashion (see Figure 8.6). To avoid excessive overplotting, we show only the ghost borders for the currently focused view. This visual cue helps users reestablish the connection between the data shown in the focused view and the data's original context.

DIFFERENCE LEDS To assist users in spotting interesting candidates for detailed comparison, additional support is desirable (requirement R1). Where feasible we provide such support by visualizing computed differences on demand.

Computing differences between views is possible, for instance, for visualizations that show data in regular cell arrangements (e.g., tabular visualization of multivariate data or matrix visualization of graph data). Where two such visualizations overlap, each cell in one view has a corresponding cell in the other view. For each such pair of overlapping cells, we compute the individual difference of their data values. These individual differences are then averaged along columns

Tasks	Provided solutions
Navigate	• Smoothly animated zoom, pan, and scroll • *Go-to* shortcut
Select	• Dynamic specification of views • Annotatable view hierarchy display • Indication of view origins via ghost borders • Assistance via difference LEDs
Arrange	• Flexible view arrangement • Alignment via snapping methods • *Bring-in* shortcut
Compare	• View juxtaposition or superposition • Interpolated and fine-tunable shine-through • Plausible view folding (via simple heuristic) • Balanced set of folding styles • Force-based folding animation

Table 8.1: Summary of interaction tasks and provided solutions.

and rows to obtain a single aggregated difference value per column and row.

To visualize the column-wise and row-wise differences in an unobtrusive fashion, we embed so-called *difference LEDs* into the borders of views. As illustrated in Figure 8.6, the differences are color-coded using a suitable color scheme. Gray color indicates that no differences were computed due to the particular overlap configuration.

When users move a view, the differences are dynamically updated depending on the current overlap situation. During data exploration, the difference LEDs provide aggregated information about the similarity of the overlapping views. Based on this information, users can decide if it makes sense to compare the underlying data in more detail using the shine-through or folding interaction; otherwise they can continue the exploration.

8.3.8 *Approach Summary*

In this section, we developed an ensemble of interaction techniques and supplementary tools to support visual comparison. Table 8.1 provides an overview of the interaction tasks that we address and the corresponding concepts and methods integrated in our approach. Figure 8.7 provides a summarizing overview of the key interactions side-by-side arrangement, shine-through, and folding.

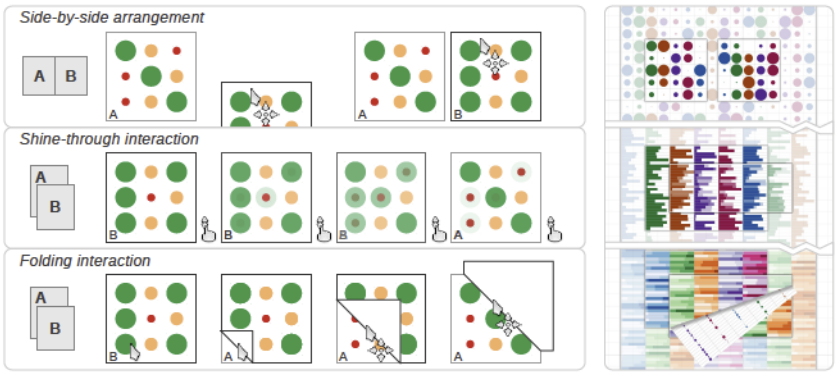


Figure 8.7: Three key techniques facilitate visual comparison: interactive side-by-side arrangement, shine-through interaction, and folding interaction.

To test our solution, we implemented an interactive prototype for visual comparison on regular desktop computers using mouse and keyboard interaction. It integrates all solutions listed in Table 8.1. Our prototype directly supports table and matrix visualizations (e.g., bar chart table or scatter plot matrix) with different visual encodings, including color, two-tone color, size, and length. By supporting the comparison of images, we indirectly allow any kind of visualization to be integrated and tested with our concept. We used our software to compare screen captures of node-link diagrams, parallel coordinates, stacked bar graphs, as well as photos and spot-the-difference images.

The interested reader can try out the developed techniques with a tabular visualization of a small dataset [328]. Note that the support for snapping is currently limited to the grid-based approach.

8.4 USER STUDY

For the evaluation of our techniques, the aim was to gather as much information and feedback as possible regarding the potential usefulness of the interaction techniques and on the fulfillment of the requirements and goals described in Section 8.3.1.

We opted for a qualitative study based on the "think aloud" method in combination with observation and a flexible interview format. Our study pursued two objectives. The first objective was to assess whether the techniques mimicked natural behaviors and if they could be applied easily. The second aimed at assessing the usefulness when using different visualization techniques.

8.4.1 *Participants*

18 persons participated in the study. They were all employees or students at a university. Among them were 6 participants (all male, ages

24–39) who judged themselves as experienced (if not experts) in visualization and 12 participants (four female, ages 19–35, eight male, ages 19–35) with no or little experience in this field. All participants were familiar with (if not experts) in using interactive software, but none had used our interaction techniques prior to the study. Except for refreshments, the participants received no compensation for taking part.

8.4.2 *Stimuli, Tasks, and Devices*

For the first part of the study, we embedded photographs and the pages of a research article into the zoomable visualization space. The task for the participants was to familiarize with the interface and its navigation methods, and to apply the developed interaction techniques to compare the photographs and browse the article much like in the real world.

In the second part, two common and one advanced visualization techniques were used: a table visualization, a matrix visualization, and the NodeTrix technique [165].

The table visualization displayed bars whose lengths were proportional to the underlying data values. Each table column was shown in a saturated color that was different from the color in adjacent columns. We used random data with 12 columns and 200 rows into which we embedded 6 artificial patterns: two ideal linear trends, two noisy trends, and two correlation patterns. The single column trends spanned between 10 and 15 rows and the correlation patterns covered 4 columns and 10 rows. The participants were pointed to these patterns and the task was to compare the lengths and the slopes of the trends, and the strengths of the correlation.

For the matrix visualization, we constructed two random 100×100 matrices. We further used three random 15×15 pattern matrices, each of which contained either a square (7×7), a plus (8×8), or a slash (10×10) as artificial pattern. The three pattern matrices were embedded into each of the random matrices at varying positions to generate two test matrices. The participants were presented with the test matrices, where one matrix used a color-coding and the other a different size+color encoding. The task was to pick a pair of patterns (e.g., the plus in both matrices) and to compare the vicinity of the patterns.

For the third test, we used a more complex NodeTrix representation of a co-author network. A NodeTrix is a hybrid design that shows communities in the network as matrices and embeds them into a node-link diagram to preserve the overall network structure. In the study, the participants were presented with Figure 7 from the original NodeTrix article [165], and were asked to freely compare anything they find interesting in the visualized co-authors network.

While the introductory part and the NodeTrix part were more of exploratory nature, the parts with the table visualization and the matrix visualization were designed to assess our techniques' usefulness when comparing length-coded and color-coded visual representations.

The study was performed using a regular desktop computer with a 24 inch monitor and mouse+keyboard interaction. A large reference sheet with the interaction techniques was pinned to the wall so that the participants could refer to them when they needed to recall a particular mouse or key mapping.

8.4.3 *Procedure*

Each individual session started with obtaining background information, including age, visualization expertise, and interaction experience (classic mouse and keyboard and modern multi-touch). Then an introduction and a demonstration was given, providing an overview of the program and a step-by-step walkthrough.

Thereafter the actual study began. First, the participants were asked to explore the program freely trying out the interaction techniques working on the photographs and the pages of the research article described earlier. Second, they were asked to carry out comparison tasks using the table, matrix, and NodeTrix visualizations. Before each task, the participants were given oral instructions about the particular visualization. In order to stimulate as much feedback as possible, the participants were encouraged to use all interaction techniques. The next visualization was presented when the participants had agreed of being finished with the current task. The order of presentation of the three visualizations was not assumed to be a factor that could have any negative impact on the outcome of this evaluation.

While the participants were carrying out the tasks, the experimenter instructed them to "think aloud" meaning that they should describe what they were doing, why they were doing it, and also what they would like to do. Notes were taken to document each participant's session. During the study, a prepared interview guide was used which included a set of predefined questions that covered various aspects of the design goals and requirements from Section 8.3.1. The experimenter engaged in the conversation and made sure that all questions were covered. Most questions were discussed while the participants were working with the program, although some were reviewed afterwards. At all times, the participants could ask questions and ask for assistance. Total participation time lasted approximately 60 minutes, about 15 minutes each for the introductory part and the three visualizations.

8.4.4 *Results*

The feedback from the participants was mostly positive. There were also useful comments on the negative side, which have already helped us to improve our solution. The evaluation of our approach according to the goals G1-G4 led to the following results.

GOAL G1 All 18 participants said that the provided interaction techniques resemble natural comparison well. 6 participants commented that the interaction is *"better than natural comparison"*, because the techniques provide more degrees of freedom and because things are easier than in the real world (e.g., cutting out and duplicating pieces of information). 12 participants commented that the folding is too flexible. They wished for constrained folding along the horizontal or vertical axes of a view. On the other hand, they recognized that constraining the folding should be optional depending on the underlying visual representation. One participant realized only then that he could create an exactly diagonal fold to compare the symmetric matrices in the NodeTrix much easier.

GOAL G2 All participants experienced the interaction techniques to be fluid and smooth, and enjoyed using them. Several participants expressed their excitement in comments like *"Oh, this looks great."*, *"The folding feels realistic because it follows the pointer smoothly."* or *"The software is nicely implemented, everything is very harmonic."*. The participants' feedback also confirms that using a zoomable interface as the basis for engaging and simple interaction was an appropriate decision.

GOAL G3 Although the complementary visual cues and add-ons were not in the focus of the user study, most participants considered them useful. In particular, the highlighting of a view's origin and the shortcuts were appreciated. The similarity LEDs were only rarely applied due to the nature of the tasks: The clearly described patterns to be compared made a search for comparison candidates unnecessary. 4 participants commented that an integrated overview would ease activating the short cuts.

GOAL G4 We also asked the participants if they could imagine applying the interaction techniques with different interaction modalities. From the 18 participants of the study, 6 considered themselves as intermediates or experts in multi-touch interaction. All 6 of them said that the interaction would work on a touch device as well; 3 thought that it would work even better on such devices due to the natural *paper-on-table* metaphor. One of the participants was experienced in alternative interaction modalities. She commented that applying the

techniques using Wii-controllers or depth cameras in front of a large high-resolution display could work as well, but the utility of the techniques might depend on the distance of the user to the display.

Given the feedback from the users, we can conclude that our approach fulfills all four goals G1-G4. The user study also yielded insight with regard to the requirements R1-R3.

REQUIREMENTS R1-R3 When the participants carried out the comparison tasks for the three different visualization techniques, we observed that the assumed general procedure of (1) selecting, (2) arranging, and (3) comparing pieces of information was indeed followed by the participants. Depending on the task, all participants created juxtapositions or superpositions of the views and compared them afterwards. For superposition, the participants applied the shine-through and folding interactions to fine-tune the arrangement. When being asked if anything was essentially missing the participants answered in the negative. All participants answered that the interaction tools and visual cues supported them in accomplishing the comparison tasks easily, quickly, and with confidence (although there was no ground truth to be found). Hence, we can infer that the requirements R1-R3 are valid for interaction support for visual comparison and that our approach fulfills them.

OBSERVATIONS The way how participants applied the techniques suggests that side-by-side arrangement are useful when comparing smaller views (e.g., the linear trend patterns) and that superposition is beneficial for larger views (e.g., the pattern matrices). We can further infer that using shine-through is good for getting an overview of the compared views, while the folding facilitates making direct comparison of more focused regions (e.g., individual rows) in the display.

When working with the different visualization techniques, some participants noticed that the color blending of the shine-through can be a disadvantage. They compensated this by using the folding interaction. With regard to the occlusion caused by the folding, we could argue that it is not a severe problem, because we can assume that the user's attention rests where the folding uncovers hidden information and not where information is possibly about to be occluded. Yet, this argument needs to be proven experimentally.

Until then it is good to have the different styles (see Figure 8.5). We asked the participants which style they found useful: the information-rich backside, the transparent backside, and the minimal backside received 15, 13, and 14 votes respectively. The blank backside received only 3 votes, although one participants explicitly acknowledged this style's naturalness.

ADDITIONAL IMPROVEMENTS The participants made a number of suggestions for additional improvements. 3 participants commented that the mouse wheel zooming works opposite to what they expected. Detailed inquiry confirmed that these participants had experience with other zoomable tools to which they were adapted. As described by Grew [143], there are competing models behind the wheel interaction, so it makes sense to allow users to choose the wheel zoom direction.

The grid-based snapping of our table visualization was critiqued as well. 9 participants said that they could not compensate for a certain offset in the value ranges when comparing the slope of trends in detail. Therefore, the suggestion was to make the snapping an optional feature, active only when a particular key is held down. Interestingly, when comparing the NodeTrix, for which we do not provide snapping, some participants commented that snapping would be needed, whereas others said that the shine-through and manual arrangement work so well that snapping is not a must.

One participant suggested that it should be possible to fix a view in place so that it can not be moved unintentionally, another recommended that dragging a view beyond the window borders should automatically activate scrolling. We consider these suggestions to be mostly details of the implementation, which can be easily fixed.

SUMMARY Overall, we can conclude that most participants are generally satisfied with the interaction techniques and acknowledged their usefulness in comparing visual representations.

8.5 CONCLUSION AND FUTURE WORK

Inspired by real world behavior, we developed a general interaction concept and an implementation thereof to support comparison tasks in visualization. Our approach covers all categories of Gleicher et al.'s taxonomy [140]: the interaction techniques facilitate *juxtaposition* and *superposition*, while *explicit encoding* is realized by visual cues. In this sense we can argue that our solution is a first step to complete the interaction support for visual comparison.

A qualitative study with 18 participants confirmed that the provided interaction techniques resemble natural behavior and that they are easy to apply and understand. The study indicates that the interaction works well for different visualization techniques.

Yet, we think that users should be better supported (beyond visual indication) in finding interesting candidates for detailed comparison. One idea for future work is to design viscosity-based interaction so that moving a view across other views is dependent on the similarity of the overlapping data. A view can be moved fluidly where the data are dissimilar anyway, but the movement is more viscose where it

would be worth taking a detailed look due to the stronger similarity of the underlying data.

We also plan to implement our concept on a multi-touch surface and for interaction in front of a large display wall. The new implementations should consider the specifics of these environments as well as the suggestions of the participants of our user study.

A limitation of our work is that it is not entirely clear which interaction to apply under which circumstances. In the real world, shine-through and folding are usually applied with line graphics or simple shapes (e.g., the flipping technique used by artists when creating animated comics), but not with dense graphical contents such as photos. We may conjecture that applying shine-through and folding for visual comparison in information visualization is more suitable for abstract graphical depictions than for very dense displays. However, this has to be confirmed by additional experimental studies. Therefore, we encourage controlled studies to investigate the usefulness of the interaction techniques for different classes of visualization techniques and visualization tasks beyond comparison. To this end, we will make our software available to other researchers and interested users.

9

SEMI-AUTOMATIC EDITING OF CUSTOMIZED GRAPH LAYOUTS

CONTRIBUTION This chapter contributes a novel approach to editing graphs with customized layouts. The proposed *EditLens* combines interactive and computing methods to support insert, update, and delete operations. The key idea to ease these typically complex tasks is to relax point-wise interaction to region-wise interaction, shifting the problem of accuracy from the user to the computer. Modern touch technology is employed to make the editing operations easy and intuitive to carry out.

ABSTRACT Usually visualization is applied to gain insight into data. Yet consuming the data in form of visual representation is not always enough. Instead, users need to edit the data, preferably through the same means used to visualize them. In this work, we present a semi-automatic approach to visual editing of graphs. The key idea is to use an interactive *EditLens* that defines where an edit operation affects an already customized and established graph layout. Locally optimal node positions within the lens and edge routes to connected nodes are calculated according to different criteria. This spares the user much manual work, but still provides sufficient freedom to accommodate application-dependent layout constraints. Our approach utilizes the advantages of multi-touch gestures, and is also compatible with classic mouse and keyboard interaction. Preliminary user tests have been conducted with researchers from bio-informatics who need to manually maintain a slowly, but constantly growing molecular network. As the user feedback indicates, our solution significantly improves the editing procedure applied so far.

ORIGINAL PUBLICATION [138] — S. Gladisch, H. Schumann, M. Ernst, G. Füllen, and C. Tominski. Semi-Automatic Editing of Graphs with Customized Layouts. *Computer Graphics Forum*, 33(3):381–390, 2014.

9.1 INTRODUCTION

Visualization has become a widely accepted means to facilitate data intensive work. Many excellent approaches exist to support exploratory or confirmatory analysis as well as the visual communication of analysis results. In these scenarios, the data on the machine are transferred via a visual representation to insight or understanding on the side of the user [353].

Often there is also the need to go the opposite direction, meaning that user knowledge needs to be fed back to the data by manipulating or editing them. This can be necessary, for instance, to correct obviously erroneous data entries, to add missing values, or to incorporate new knowledge into an existing database. Hence, there is a need to develop solutions, where methods to visualize data work hand in hand with methods to edit data [48].

As we will see in Section 9.2, the literature is rich of interaction techniques allowing users to control the visualization. Computer science in general provides powerful means to manipulate data. Yet there are two serious problems. First, manual data manipulation is often time-consuming business and lacks user support [197]. Second, and more importantly, visualization and editing are usually carried out separately, which hinders a smooth data analysis and data manipulation cycle. Therefore, our work advocates an integrated data visualization and editing process.

We aim to support the visual editing of graphs with established customized layouts. Users working with such graphs over a longer period of time develop a mental map of the data. Care has to be taken to maintain the integrity of the mental map on incremental updates to the graph. Moreover, customized graph layouts often obey application-dependent constraints (e.g, certain nodes may appear only in specific regions). Such constraints are usually incompatible with existing automatic graph layout methods. Hence, users have to resort to diagram editors to carry out manipulations manually, which is cumbersome especially when working with an already complex layout of nodes and edges.

Our approach to editing graphs with customized layouts is *semi-automatic*, combining the strengths of interaction and computation. We address the editing of the graph's structure, more specifically, the insertion of new nodes and edges as well as the update of existing elements to improve the layout. The key idea is to use an interactive lens, the *EditLens*, to define a local region where edit operations are to take effect. Node positions and edge routes of the entities being edited are automatically computed based on different optimization criteria. As we do so within the confines of the *EditLens*, the computational costs are kept at a manageable level and the mental map is maintained. By moving the lens or switching between optimization criteria, the user

can on-the-fly explore different solutions and customize the suggestions made by the lens before an edit operation is finally committed. In Section 9.3, we introduce our approach in more detail.

An interactive tool for visualizing and editing graphs has been implemented, combining established visualization concepts with the advantages of the EditLens and standard manual editing facilities. As described further in Section 9.4, we take advantage of novel multitouch gestures, but still support classic mouse and keyboard interaction. In Section 9.5, we apply our solution to a concrete problem from bio-informatics where researchers collect and maintain a network of molecular interactions in mouse, the so-called *PluriNetWork* [308] with several hundred nodes and edges. With our tool, the maintainers of the PluriNetWork can now conduct the necessary steps more easily as can be seen from the user feedback described in Section 9.6. Discussion and conclusion will be given in Sections 9.7 and 9.8.

9.2 RELATED WORK

We structure the review of related work into two parts: aspects of interaction and algorithmic aspects.

9.2.1 *Aspects of Interaction*

Interactive manipulation is a key theme of visualization. While interaction is mostly applied to alter the visualization, there are also approaches to edit the underlying data. Next we take a brief look at both aspects.

Several techniques have been proposed for interacting with graphs in particular. Examples for adjusting graph layouts are interactive sticks and arcs [278], magnet-based attraction [316], or radial menus and hotboxes [252]. There are also interactive lenses to reduce node and edge clutter and to create local overviews of subgraphs [330]. Interactive adjustment of edge curvature via bundling, fanning, magnets, and legends is addressed in [166]. Individually, the aforementioned approaches provide excellent means for interaction with graphs, but editing of graphs, in particular of graphs with customized layouts, has received only little attention.

On the other hand, Baudel [48] argues convincingly for manipulation of the data through means of visual representation and direct interaction. Off-the-shelf graph editors provide the functionality to interactively manipulate graph data represented as node-link diagrams. Yet, much work has to be done manually by the user, with only little or no assistance from the computer. Sketching and multitouch approaches strive to ease graph editing by employing more natural drawing gestures. Examples are works on sketching of UML diagrams [77] and on beautifying sketches of node-link diagrams on-

the-fly [38]. Combining overview+detail with constraint-based layout has been demonstrated to be beneficial [103]. Multi-touch and pen interaction can help to support the editing of graphs [127].

Usefulness and ease-of-use of existing graph editing approaches have been demonstrated for moderately-sized graphs being constructed from scratch. However, the existing approaches fall short of addressing larger graphs with hundreds of nodes and edges forming a customized and established layout. In such scenarios, accuracy and efficient use of screen space become important aspects. Addressing them by manual positioning of nodes and edges is no longer sufficient. Our semi-automatic editing approach tackles these problems by integrating interactive and algorithmic methods. Related work on algorithmic aspects of graph drawing are reviewed next.

9.2.2 *Algorithmic Aspects*

In our work, we consider evolving graphs. Although changes do not occur automatically, but are the result of editing operations by the user, there is still a certain analogy to time-varying graphs. Therefore, we shall indicate some related algorithms for these kinds of graphs.

In general, visualization of time-varying graphs follows two approaches: either time steps are visualized individually in sequence or a super graph including all time steps is constructed. Visualizing time steps individually usually means showing small multiples or an animation of the evolving graph (e.g., [71] or [322]). Approaches based on a super graph can optimize the layout according to a holistic view of the graph (e.g., [342] or [121]). None of these approaches can be used in our scenario, because they require a finite set of time steps, which is not given for an open-ended editing procedure.

Such scenarios are dealt with by online graph drawing algorithms. Examples include online drawing of directed acyclic graphs [264], approaches based on Baysian networks [64], solutions for clustered graphs and corresponding quality metrics [128], or approaches based on simulated annealing [233]. These algorithms aim to achieve stability and coherence between subsequent drawings, which is also needed for editing. However, these algorithms are all fully automatic and do not provide the facilities to adjust the layout according to application-dependent customizations.

In summary, according to our review of related work, editing customized graph layouts remains an under-researched problem to whose solution we contribute.

9.3 SEMI-AUTOMATIC GRAPH EDITING

We propose to combine interactive and computing methods to create a semi-automatic approach to graph editing. Before going into detail, we take a look at the problem to be solved.

9.3.1 *Problem Analysis*

In general, data manipulation is one of the following operations: insert, update, or delete. In our specific case of editing the structure of graphs, nodes and edges are the entities to be manipulated. Editing data attributes or data values associated with nodes and edges is beyond the scope of this paper and a concern for future investigation.

Editing graphs with an established customized layout requires maintaining the mental map that users have already developed of the data. This is especially true when the layout obeys certain semantic properties, for example, when specific nodes may only appear in certain regions of the layout. Therefore, edit operations must not result in a global change of the graph layout, but the effects must remain local. Otherwise, we would risk that the graph can no longer be recognized as the one that it had been before.

When editing graphs with customized layouts, users should be allowed to input their ideas or constraints of the layout into the system to control the final outcome. Placing single nodes or routing individual edges by hand results in local changes and clearly allows for full customization of the graph layout. However, such manual editing is infeasible when editing larger graphs. Imagine yourself using a diagram editor for inserting a node with say a dozen of edges into an existing graph with hundreds of nodes. Positioning this node and routing each connected edge by hand is tedious work and would take a lot of time, especially when certain quality criteria (e.g., shortest overall length of all connected edges) have to be evaluated mentally.

On the other hand, we could let automatic graph layout algorithms do the math and compute high quality layouts with respect to general quality criteria. But these algorithms are usually incompatible with customized layouts. Furthermore, most graph layout algorithms work globally, which contradicts our need for only local changes to preserve the mental map. A third difficulty is that computational costs rise significantly when considering multiple layout criteria for larger graphs, which hinders interactivity of edit operations.

In summary, fully manual editing techniques are not sufficient for larger graphs with customized layouts due to the immense effort required from the user and fully automatic layout algorithms dismiss custom application-dependent layout constraints and lack mechanisms for controlling the desired editing outcome. Our idea is to

combine the power of interaction (yet without intensive labor) with the power of computation (yet with local effect and customizable).

9.3.2 *General Approach*

In the first place, we need to think about what tasks are to be interactive and what can be done automatically, and how visual feedback ties interaction and computation together.

USER INTERACTION Interaction gives the user control of what to edit (e.g., insert node) and of what the editing outcome will be, including the possibility for customization. However, we do not want to burden the user with placing nodes at exact positions or defining exact routes of edges. Therefore, we relax *point-wise* editing to *region-oriented* editing. Instead of specifying precise points, the user interactively defines a local region where an edit operation is to take effect. This local region acts as a *coarse solution*, and finding a *precise solution* will be the task of automatic computations, effectively reducing the users workload.

AUTOMATIC COMPUTATION Algorithmic calculations enhance the interactive editing by providing suggestions for exact node positions and edge routes. The underlying algorithms consider a set of layout criteria to compute "good" results with respect to the local region as defined by the user. Different heuristic layout strategies leave space for customization via prioritizing specific layout criteria. The computational part relieves the user from evaluating the layout quality mentally and finding optimized exact solutions by hand.

VISUAL FEEDBACK The interplay of interaction and computation also demands suitable visual feedback. Of primary concern is to accentuate elements being considered for an edit operation and to dim those that are not affected. Moreover, the local region acting as a coarse solution plays a special role, while the rest of the graph layout fades into the background. To ensure a smooth editing cycle while interactively exploring the space of "good" solutions, visual feedback about the suggestions of the automatic computations is continuously provided to the user.

To support the aforementioned functionalities, we developed the novel concept of an interactive lens for edit operations, which we describe in detail next.

9.3.3 *The EditLens*

Magic lenses as introduced by Bier et al. [58] naturally lend themselves to our purposes. Magic lenses affect a local region, which is

defined by the lens' shape, size, orientation, and position. The actual effect is defined by a lens function. The lens functions known so far mostly alter the visual content within the lens, for instance, to magnify focus regions or to de-clutter overcrowded parts of the display.

We propose the EditLens whose function is to edit the underlying data and the visual representation. According to our research, this is a novel type of lens function, not yet considered in the literature. With the EditLens, we incorporate (1) interactive control by adjusting the lens, (2) locally confined automatic computation of the lens function, and (3) the presentation of corresponding visual feedback in a single editing tool.

The EditLens has a rectangular shape, which is suitable for most edit operations on customized graph layouts. Alternative shapes might be useful for special applications (which we are still unaware of). The rectangular lens can be resized and repositioned interactively to define the local region where the edit operation is to take effect. By using a larger lens, the user widens the space of possible node positions and edge routes. Using a rather small lens can be useful when the user has a strong idea of a suitable solution in mind or to adhere to application-dependent constraints. As such, the EditLens utilizes the coarse-positioning skills of humans (as any magic lens does [59]).

Precise positioning is left to exact calculations by the computer. These calculations suggest "good" node positions and edge routes within the lens. The calculations are kept local in order to maintain the integrity of the already existing layout and to keep computational costs low. Naturally, computations with a local scope rarely lead to a global optimum. Therefore, the user is free to explore alternative solutions by adjusting the EditLens or prioritizing a different layout criterion. Eventually, the user may accept a suggested solution and commit the edit operation. Finally, there is still the option to manually refine a suggested solution via standard means of editing altogether.

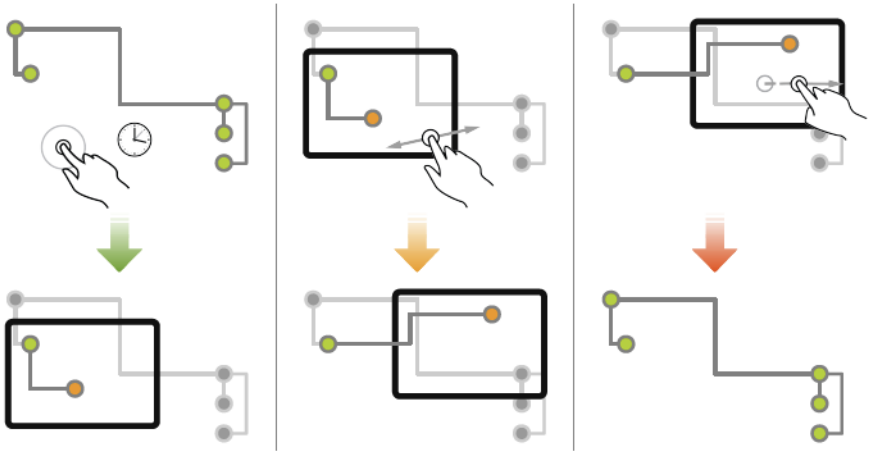
The EditLens has been designed to support the following edit operations as depicted in Figure 9.1:

- Insertion of nodes and edges into a graph
- Update of node positions and edge routes of the layout
- Deletion of nodes and edges from a graph

We start with describing the insertion of a node, as it best illustrates how the EditLens works in general. Additional details for the remaining operations are given later on.

9.3.3.1 *Inserting a node*

Once node insertion has been triggered for the local region defined by the EditLens, two computational steps need to be carried out. First,



(a) Insert (b) Update (c) Delete

Figure 9.1: Schematic depiction of insert, update, and deletion with the EditLens. The node being edited is highlighted in orange, regular nodes are shown in green. Nodes that are not considered for the editing operation are dimmed.

the node position must be computed and based on that, as the second step, the edges connected to the node must be routed. For this, aesthetics and quality criteria have to be considered [47]. General goals are to avoid clutter and ambiguities, which usually means utilizing the available screen space efficiently.

Since it is impossible to handle all possible criteria simultaneously [277], we resort to a plausible set of four criteria. In the first place, nodes should be easily discernible, which can be achieved by leaving sufficient space to other nodes. Second, neighbors should be easily detectable, which demands edges to be short. Third, edges should circumvent existing nodes, to prevent misinterpretation of connectivity. A well-accepted way to achieve this is to use orthogonal edge routing. Finally, edges should be easily traceable, which is the case for edges with as few bends as possible. This set of criteria is certainly not exhaustive, but we assume that alternatives can be taken into account on demand.

For finding a "good" position p_v for the node v being inserted we follow a three step procedure:

1. Determine a set $\mathfrak{R}$ of areas that are potential candidates for node placement. The areas must be within the EditLens and must be free of existing nodes or edges.

2. Select a suitable area $R \in \mathfrak{R}$ within which p_v is to be computed. Necessary preconditions are that R be large enough to contain the node's shape and that application-dependent semantic regions be obeyed.

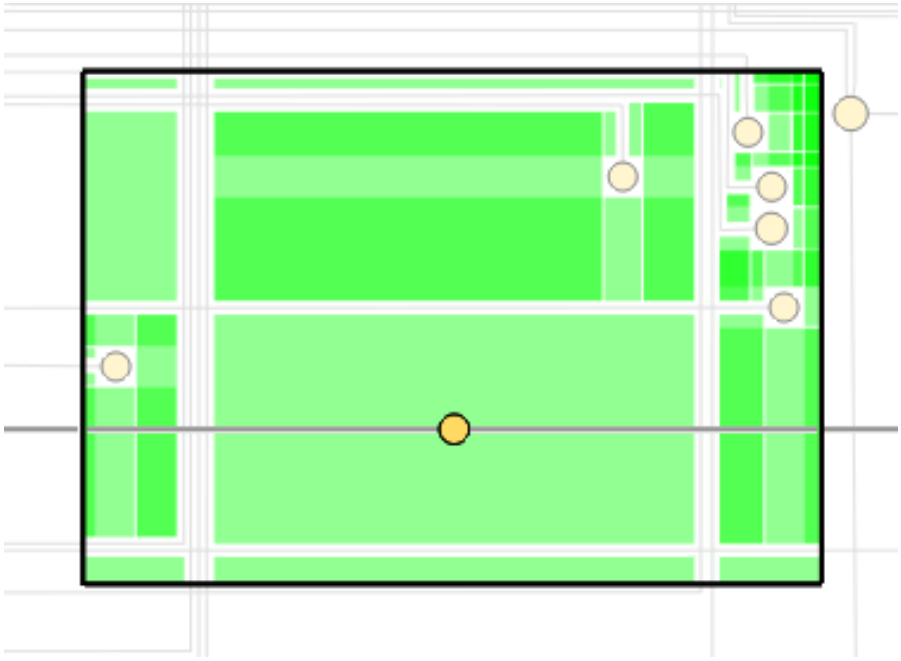
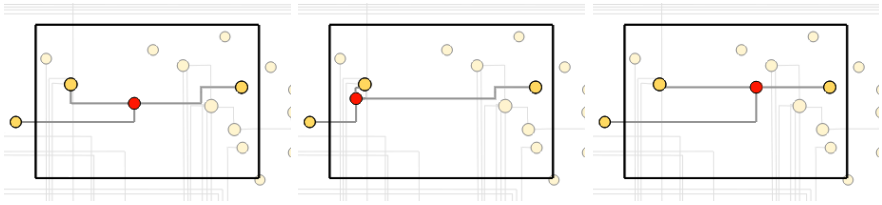


Figure 9.2: Empty space rectangles (green) under the EditLens (dark frame) with the node being edited (yellow). Different shades of green indicate where rectangles overlap.

3. Compute a "good" position p_v within R taking into account the layout criteria indicated before.

A sufficiently fast method to determine $\mathfrak{R}$ is the *dynamic space management* by Bell and Feiner [54]. Their method is based on *full* and *empty space rectangles*, which describe occupied and empty space, respectively. The dynamic space management is initialized by creating full space rectangles for the bounding boxes of all existing nodes and edge segments. Note that the bounding boxes are enlarged by a certain δ to preserve a minimal gap between existing and newly placed graph elements. The dynamic space management can then be queried for *empty space rectangles* inside the EditLens to compile the set $\mathfrak{R}$ as illustrated in Figure 9.2.

The next steps are to select an $R \in \mathfrak{R}$ and to compute the exact position p_v in R. For these steps, we have to consider the necessary conditions formulated in step (2) and the sufficient conditions as imposed by the addressed layout criteria. However, it is hard or even not possible to obtain node positions and edge routes that fulfill all criteria optimally. For example, orthogonal edge routing depends on the node position. But the node position for the optimal orthogonal edge route could be too close or even overlap with an existing node. To support the search for good positions, the EditLens can prioritize certain criteria. For this purpose, we have developed three placement strategies:



(a) Node space first (b) Edge length first (c) Number of edge bends first

Figure 9.3: Three placement strategies determine a node's position based on prioritizing different layout criteria, including favoring much free space around nodes, shortening edges to neighbors, and reducing the number of edge bends. The inserted node is shown in red, its neighbors are yellow, unaffected elements are dimmed.

NODE SPACE FIRST A node is easily discernible when there is enough free space around it. Following this thinking, we select R as the largest empty space rectangle available that falls into the semantic region associated with the node being inserted. Then we compute p_v as the center of R. Figure 9.2 illustrates the available empty space rectangles, with the largest one selected and the node positioned in its center. After placing the inserted node this way, the edge routing is invoked to establish links to the node's neighbors.

EDGE LENGTH FIRST To easily identify neighbors in a graph, it makes sense to prioritize short edge lengths. According to this thinking, the sum of the lengths of the edges connected to the node being inserted should be minimized. As this sum depends on the node's position, we have to find the position within the EditLens where the sum of edge lengths is minimal. However, the computation costs for edge routing prevent testing every possible position within the lens. Therefore, we developed the following heuristic.

First, select R as the empty space rectangle that falls into the semantic region associated with the node being inserted and whose center c_R has the minimal sum of Manhattan-distances between c_R and every neighbor of v. Then we define a linear optimization problem. Let (x, y) be a position in R, and $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ be the positions of the n neighbors of v. Further, (l, t) denotes the top-left corner

of R and (r, b) is R's bottom-right corner, and d_v is the diameter of the shape of v. Our goal is to optimize:

$$f(x,y) = \sum_{i=1}^{n} |x - x_i| + |y - y_i| \rightarrow \min$$

such that

$$\begin{aligned} x &> l + d_v/2 \\ x &< r - d_v/2 \\ y &> t + d_v/2 \\ y &< b - d_v/2 \end{aligned}$$

By solving this linear optimization problem with the simplex algorithm, we derive the optimal position $p_v = (x, y)$ for v in R. This heuristic, however, does not account for obstacles along the shortest routes (according to Manhattan distance) to neighbors of v. The exact routes around obstacles are finally computed by an orthogonal edge router.

NUMBER OF EDGE BENDS FIRST When tracing edges, bends along the way can be disruptive. Therefore, it makes sense to keep the number of edge bends to a minimum. Again the position at which edge routes have a minimal number of bends could be determined by invoking the edge routing for all possible positions and selecting the best one. As this is prohibitively expensive, we came up with a simple heuristic to prioritize the reduction of edge bends.

A key observation is that if the node v being inserted is aligned horizontally or vertically with one of its neighbors u, the edge (v, u) does not bend. Further, if two neighbors u_1 and u_2 are already on a horizontal or vertical line, inserting v on that line in between u_1 and u_2 generates edges (v, u_1) and (v, u_2) that do not bend either. Therefore, our heuristic searches exactly for such situations. In a first step, we determine existing straight horizontal and vertical lines between all neighbors of v. These lines are intersected with all empty space rectangles in $\mathfrak{R}$. Then we select an $R \in \mathfrak{R}$ as follows.

If an empty space rectangle is intersected by both a horizontal line and a vertical line, select that rectangle as R, and the intersection of both lines will be the position p_v of the node v. If no such rectangle exists, we look for a rectangle R that is intersected by at least one horizontal or one vertical line, and place p_v on that line and centered in R. If we encounter multiple possible solutions during our search, we can either favor larger rectangles or shorter intersection lines to drive the final decision for R and p_v.

Figure 9.3 compares the results of the three placement strategies. These strategies together resizing and relocating the EditLens offer sufficient room for controlling and customizing the outcome of the

node insertion. At the same time, the automatic calculations reduce the user's mental load and manual work significantly.

If no suitable node position can be found by the automatic means due to insufficient space or due to unsatisfiable constraints (e.g., EditLens placed in wrong semantic region), the color of the lens border changes to a signaling color such as red. The user can then either adjust the lens parameters or place the node manually, overriding any quality criteria.

9.3.3.2 *Inserting an edge*

When inserting an edge (u, v) the idea is to use the EditLens as a constraint for the edge routing, more precisely as a region through which the edge has to be routed automatically. Consider two nodes u and v of the same type (i.e., both nodes are placed in the same dedicated semantic region) that have to be connected by an edge. It makes sense that the route of this edge be within the confines of the dedicated region. However, when using a routing algorithm that is unaware of the dedicated region and simply calculates a route from the source u to the target node v, one cannot guarantee that the result satisfies this specific constraint.

With the EditLens, the user first defines the local area through which the edge has to be routed. Second, the EditLens automatically calculates a single point p in the local region through which the edge has to be routed precisely. For determining p, we again use the aforementioned placement strategies. Third, the automatic edge routing algorithm calculates a route from the source node u to p and from p to the target node v. This guarantees that the route will connect u and v passing through p. As edge routing algorithms usually aim to minimize edge length, it is also almost certain that the edge overall remains within the bounding box of u, v, and p.

9.3.3.3 *Deleting and updating nodes and edges*

Delete operations are straight-forward. When a node is deleted, all connected edges are deleted as well. If the deleted node was an obstacle for other edges, the edge routes of these edges can now be improved by update operations. When deleting an edge, nodes that have been placed near the route of that edge might need further layout improvement to better utilize the screen space made available.

Updating a node or an edge can be reduced to virtual delete and insert operations in the following way. When updating an existing node, delete this node first and then re-insert it to a new position. When updating an edge in the diagram, delete this edge first and the re-insert it with a new edge route.

9.3.4 *Approach Summary*

The EditLens supports insertion, update, delete operations on graphs with customized layouts. By interacting with the lens, users can explore different placement or routing suggestions for different regions and different layout strategies. We see this as a key advantage of the EditLens. Such an exploration of solutions is not possible with either manual editing techniques or automatic layout algorithms alone.

Yet, as the EditLens computes heuristic-driven suggestions only, minor refinements might be necessary to fully satisfy the user. Such refinements of node positions or edge routes can be done using classic means of editing (e.g., dragging a node or the bends of an edge to slightly adjusted positions). As the coarse solution is already "good", the effort for manual refinement will remain at a reasonable level.

9.4 IMPLEMENTATION AND INTERACTION

We implemented a prototype that combines classic means for exploring graphs with our novel techniques for editing graphs. The graph is shown as a node-link representation. Nodes are represented as circles whose size and color are set according to data attributes. Textual labels for nodes are placed via a dedicated labeling algorithm [240]. Edges are represented by orthogonal polylines as computed by the edge routing algorithm. The node-link diagram is embedded into a zoomable space [53].

While the visualization part follows well-known ideas, the editing part brings in our new ideas. We implemented a dual-interaction prototype to take advantage of the efficiency of familiar mouse+keyboard interaction and the potential of novel multi-touch gestures. This gives users the chance to choose the modality they deem best for the task at hand.

Standard GUI elements such as buttons, scroll bars, and menus are operated identically with mouse+keyboard and touch interaction. Interaction differs, however, when editing in the zoomable node-link diagram. Our design has been inspired by previous work on touch interaction for graphs [127, 295]. Figure 9.4 summarizes the gestures we employ. Next we indicate how users actually carry out navigation and selection as well as manipulation tasks.

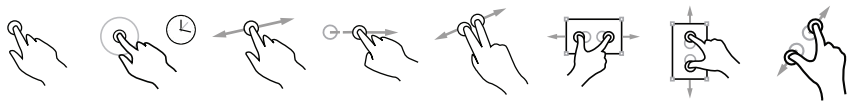


Figure 9.4: Touch gestures used in the EditLens prototype.

Navigation and Selection Pan-navigation works via mouse drag in empty space or by two finger touch-slide in empty space. Zoom-

navigation is activated via the mouse wheel or by using a pinch gesture, which is very common for this task. Individual nodes or edges can be either selected by single mouse clicks or by a single tap with a finger. Selected elements are highlighted and detailed textual information about them is shown. For selecting multiple graph elements, the user can also use a lasso-selection gesture.

Manipulation The lens is our main interaction element. It can be spawned for node insertion with a long tap or press in empty space or for node editing with a long tap or press on an existing node. The lens can be enlarged in x-direction, y-direction, or both combined by using pinch gestures. Resizing is also possible via the keyboard or the mouse. Translation of the lens is done via single finger slide or regular mouse drag. To move the lens over large distances, the user can use a double tap or click as a shortcut, which triggers a smooth transition of the lens to the desired location. Adjusting the lens initiates an automatic re-computation of the suggested position and routes of the elements being edited. Users can accept the placement suggestion by a single finger tap or mouse click. Elements can be deleted using a flick gesture or by pressing the DEL key.

As fall-back solutions, we also support fully manual editing of node positions and edge bends. This is done through classic drag and drop gestures.

With these interaction techniques and the underlying algorithmic solutions for node positioning and edge routing, our prototype is capable of dealing with graphs with several hundreds of nodes and edges. Next we describe an application of our solution to a real-world problem from bio-informatics.

9.5 APPLICATION EXAMPLE: EDITING THE PLURINETWORK

A use case for our approach is the manual maintenance of networks discovered in complex systems. An example is the *PluriNetWork* [308], a literature-curated network of 365 nodes and 631 edges, where nodes represent genes and edges describe molecular interactions that are important for the cellular state of pluripotency in mouse. Bio-informatics scientists have developed the network from scratch using the editor of the Cytoscape platform [302]. The established layout follows a circuit-like design motivated by the analogy of the flow of gene regulation and the flow of electricity. As illustrated in Figure 9.5, the layout is divided into three distinct regions of semantically related nodes: signaling on the top (blue), epigenetics on the left (green), and transcriptional gene regulation in the center (orange). Once new verified knowledge becomes available in the literature, the network is edited manually. This manual editing is still carried out using the Cytoscape editor. However, with the increased size and complexity of the PluriNetWork, preserving the already existing semantic map

is a major difficulty. A description of the original editing procedure makes clear why this is so.

ORIGINAL EDITING PROCEDURE Given a newly reported interaction between two genes $(gene_1, gene_2)$ as input, the editing workflow is as follows. First, it has to be determined if $gene_1$ and $gene_2$ are already present in the graph. If either of the genes does not exist, the curator has to find a good spot for placing a node for the missing gene according to its associated semantic region. If one gene already exists, the new node should be as close as possible to the existing node, but node overlap has to be avoided.

Then the curator heuristically determines a path from $gene_1$ to $gene_2$ that (1) is short, (2) has a low number of bends, and (3) is as far away as possible from existing edges. Finding such a path could turn into a major challenge, as the optimal solution satisfying the three listed criteria is very often not at all obvious or even does not exist. Often, a suboptimal solution was attempted, yielding a displeasing layout and triggering further follow-up adjustments of nodes and edges that were not affected in the first place.

If both genes exist already, and only their interaction is new, the genes are usually already close in the network layout, because the layout reflects biological relatedness, and new interactions tend to connect related genes. In some cases, however, the nodes of $gene_1$ and/or $gene_2$ need to be moved to accommodate a visually more pleasing layout according to the mentioned criteria. Manually finding a path between $gene_1$ and $gene_2$ would then pose the same difficulties as described before. Further complications arise if a node in question has so many edges that there is insufficient space to start a new one. In such cases, existing edges are rearranged, and if needed, the node is enlarged to be able to attach a new edge to it.

The bio-informatics researchers transferred the graph (connectivity as well as layout) to the *WikiPathways* platform [203] to enable community contributions. However, the built-in editor of WikiPathways turned out to be very cumbersome to use, effectively inhibiting any public editing of the PluriNetWork.

APPLYING THE EDITLENS To help the curators of the PluriNetWork in editing their data, we integrated the EditLens into the original editing workflow. In the first part, nothing changes. The first step is to classify a newly reported interaction of two genes $(gene_1, gene_2)$ according to whether none, one, or both of the genes already exist in the layout. If none of the genes exist we first apply the EditLens to insert $gene_1$ (operation: insert node). Then $gene_2$ with the connection to $gene_1$ is inserted (operation: insert node and edge). The latter operation also applies if only one gene is already present in the layout.

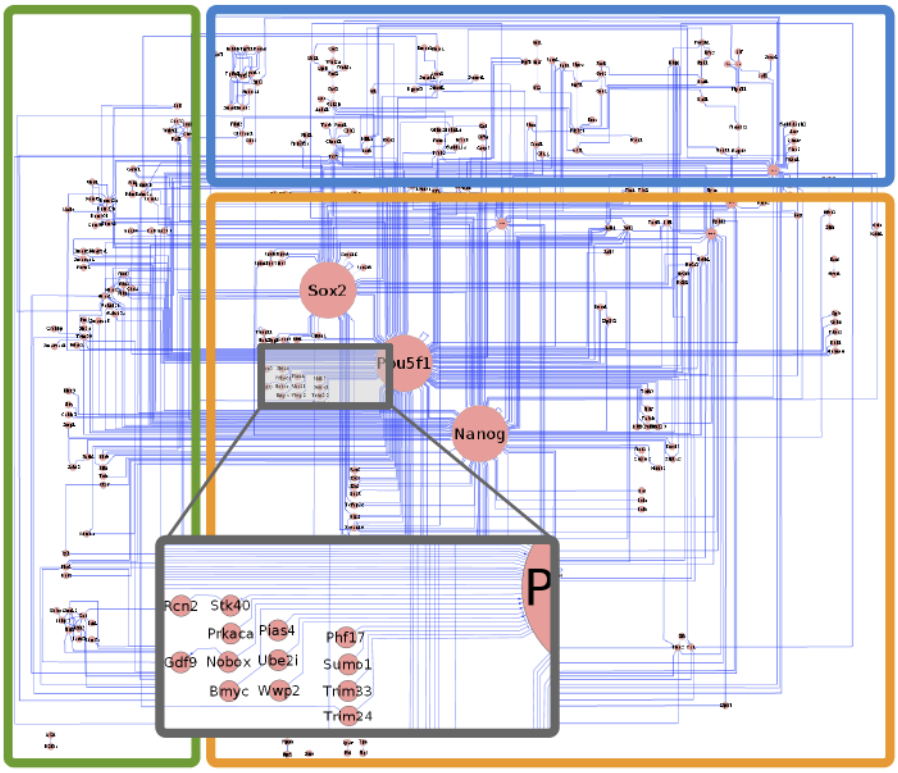


Figure 9.5: The PluriNetWork. Marked are the three semantic regions (blue, green, and orange). The detail view (gray) indicates how cumbersome manual editing must be.

If both genes already exist, we use the EditLens to insert a new edge between $gene_1$ and $gene_2$ (operation: insert edge).

When genes need to be repositioned or gene interactions need to be rearranged for layout improvement in general, the EditLens' update functionality is employed. Deletion of genes or interactions among them is not applicable in the PluriNetWork as all existing elements are backed by scientific literature. In the unlikely event that a scientific discovery should be refuted, the EditLens would support carrying out the necessary correction in the PluriNetWork.

We expected that integrating the EditLens into the PluriNetWork's editing workflow will significantly help the maintainers of the network in carrying out their editing tasks. A small user evaluation has been set up to verify this.

9.6 PRELIMINARY USER FEEDBACK

For the evaluation, the aim was to gather feedback regarding the potential usefulness of the EditLens. Prior to the user evaluation, a visualization specialist was recruited for pilot testing. Then we performed a small qualitative study using a flexible interview format.

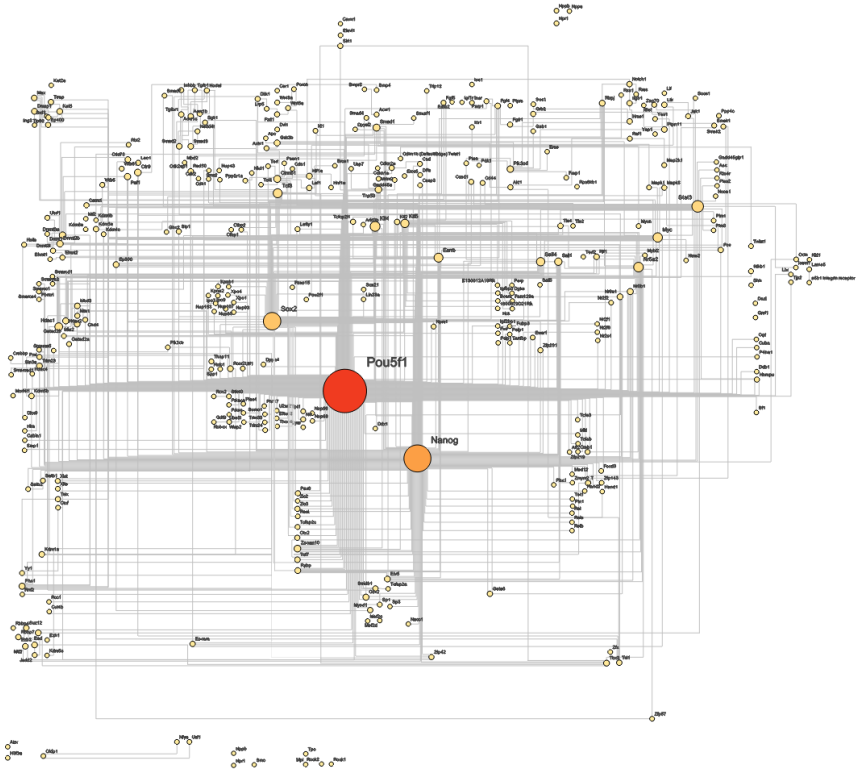


Figure 9.6: The improved PluriNetWork as visualized by our prototype. Node size and color visualize node degree. Labels were placed using a dedicated labeling algorithm.

PARTICIPANTS Our primary interest was to receive feedback from domain experts who edit networks on a regular basis. Therefore, we invited two of the researchers maintaining the PluriNetWork. In order to contrast expert feedback against feedback from more casual users, we recruited two additional non-experts, who had never worked with the PluriNetWork before. All four participants were male and employees or students at a university. They all were familiar with using interactive software, but none had used our EditLens prior to the study. The participants judged themselves as experienced in using touch gestures.

TASKS AND DEVICES For the first part of the study, the participants' task was to familiarize themselves with our prototype, especially with navigation techniques, manual editing techniques, and with the EditLens. The experimenter first demonstrated the available functionality and then the participants were prompted to try them out on their own. The network presented throughout the study was an updated and improved PluriNetWork as depicted in Figure 9.6. The introduction lasted approximately 15 minutes.

In the second part, the participants had to solve three tasks on the PluriNetWork. The first task was to insert a new node along with

two edges to existing nodes. For the second task, the participants were asked to insert an edge between two already existing nodes. Finally, the third task was to improve the network layout by updating an existing node. All three tasks were motivated by real-world tasks from the context of the PluriNetWork.

The participants had to carry out all tasks using both the original manual editing procedure and our novel EditLens. Manual editing was done with mouse+keyboard interaction, whereas the EditLens was operable via touch-gestures. Two participants (one expert and one non-expert) used the EditLens first, the other two participants used the manual editing procedure first. On every task, the participants were instructed to "think aloud" meaning that they should give any feedback they have in mind. The second part of the study took between 30 and 40 minutes.

The study was performed using a regular desktop computer with a 23" touch-enabled monitor, which was arranged in a horizontal position. A reference sheet with the available multi-touch and mouse gestures as well as key mappings was presented on another monitor so that the participants could look them up quickly when needed.

RESULTS All participants were able to finish all tasks using the original editing procedure and the EditLens. Questions regarding the usefulness of the EditLens were answered positively by all participants. The different layout strategies were also positively received by all participants. Ease-of-use of the EditLens was generally acknowledged. When asked which technique they would prefer, both experts and both non-experts answered in favor of the EditLens, as we expected. One expert said that "the EditLens is very useful and can clearly reduce the editing effort". A non-expert user said that "the automatic suggestion of node positions and edge routes is obviously beneficial".

Additional comments suggest that there is space for improving our approach. One participant of the study suggested routing inserted edges along existing edges to create bundles of edges. Another user would have liked to experiment with alternative lens shapes.

Note that our study did not formally control all influencing factors (e.g., mouse+keyboard vs. multi-touch, horizontal vs. vertical display, simple vs. complex editing tasks). Therefore, the results obtained are to be understood as qualitative indicators to be confirmed in further formal studies.

9.7 DISCUSSION

This section is to discuss the EditLens as one piece of the larger editing puzzle and to indicate limitations and open questions yet to be addressed.

An important point for discussion is the scalability of the EditLens, in terms of both computation and interaction. Our solution works well with the hundreds of nodes and edges of the PluriNetWork. As the computational costs are usually bounded by the size of the EditLens and the limited number of entities affected by an edit operation, we expect that even larger graphs are possible to work with. Critical are update operations that affect high-degree nodes with many edges. Elaborate connector routing algorithms (e.g., [254] or [380]) are required to cope with such cases.

In terms of interaction, we have focused on edit operations for single graph elements. Scaling to multi-element edit operations is a sensible next step. Such operations could work on groups of elements, where certainly additional constraints would need to be introduced, for example, to maintain intra-group ordering and alignment.

An interesting question to be investigated is the long-term effect of locally optimal edit operations. It is yet to be observed, if the global layout quality deteriorates after many local edit operations. The PluriNetWork application would be an ideal candidate to further study this behavior.

The EditLens has been designed with orthogonal graph layouts in mind. To overcome this limitation and to be applicable more broadly, alternative aesthetic criteria (e.g., uniform edge-length, edges in bundles) and corresponding algorithmic solutions need to be considered. One can also imagine lenses that automatically apply different constraints to different regions of a graph layout according to available meta information or data characteristics. Investigating non-rectangular or even adaptive lens shapes is another interesting direction for future work.

Balancing all influencing factors will be a formidable challenge, which we think requires collaboration of interaction, visualization, and algorithm experts.

9.8 CONCLUSION

In this work, we presented a semi-automatic approach to editing graphs with customized layouts. Our novel EditLens combines interactive means with automatic computation. In contrast to existing point-wise editing, the EditLens follows a region-oriented paradigm. This significantly eases editing operations, because the user only needs to define a coarse region interactively, rather than a precise position. The algorithmic part of the EditLens computes suggestions for precise solutions, which the user can customize on-the-fly by adjusting the lens or selecting different heuristic placement strategies. As far as we know, no such lens has been considered in the literature.

A prototype implementation of our approach has been applied to a real-world problem in the context of manually curating a larger

biomedical network. In a small user evaluation we tested the utility and usefulness of the EditLens. Overall our approach received quite positive feedback from domain experts as well as casual users. We are convinced that the EditLens is a valuable addition to the existing tools for editing data through interactive visual means.

10

TANGIBLE VIEWS FOR INFORMATION VISUALIZATION

CONTRIBUTION This chapter contributes the concept of *tangible views*, by which interaction and display blend into a single interactive visual tool. Driven by modern interaction and display technology, tangible views offer a novel way of interacting with visual representations of data. Possible applications are manifold and include exploring of multivariate data, browsing graphs at different levels of abstraction, comparing graph matrices, and analyzing spatio-temporal data in a space-time cube.

ABSTRACT In information visualization, interaction is commonly carried out by using traditional input devices, and visual feedback is usually given on desktop displays. By contrast, recent advances in interactive surface technology suggest combining interaction and display functionality in a single device for a more direct interaction. With our work, we contribute to the seamless integration of interaction and display devices and introduce new ways of visualizing and directly interacting with information. Rather than restricting the interaction to the display surface alone, we explicitly use the physical three-dimensional space above it for natural interaction with multiple displays. For this purpose, we introduce *tangible views* as spatially aware lightweight displays that can be interacted with by moving them through the physical space on or above a tabletop display's surface. Tracking the 3D movement of tangible views allows us to control various parameters of a visualization with more degrees of freedom. Tangible views also facilitate making multiple – previously virtual – views physically "graspable". In this paper, we introduce a number of interaction and visualization patterns for tangible views that constitute the vocabulary for performing a variety of common visualization tasks. Several implemented case studies demonstrate the usefulness of tangible views for widely used information visualization approaches and suggest the high potential of this novel approach to support interaction with complex visualizations.

ORIGINAL PUBLICATION [313] — M. Spindler, C. Tominski, H. Schumann, and R. Dachselt. Tangible Views for Information Visualization. In *Proceedings of the International Conference on Interactive Tabletops and Surfaces (ITS)*, pages 157–166. ACM Press, 2010.

10.1 INTRODUCTION

In visualization science, it is commonly known that encoding all information in a single image is hardly possible once a data set exceeds certain size or complexity, or when multiple users have to look at the data from different perspectives. This problem can be resolved spatially by providing multiple views on the data [363] or by embedding additional local views in the visualization [58]. It can also be resolved temporally by changing representations over time. Except for a few automatic methods, in most cases changing a visualization is a result of user interaction [385].

Mouse and keyboard are the predominant interaction devices to adjust the representation according to the data and the task at hand. Compared to the richness of available visualization methods, the number of dedicated interaction techniques for information visualization is moderate. Reasons might be that complex interactivity must be squeezed through the degrees of freedom offered by mouse and keyboard and that display and interaction device are physically separated. Recent research on tabletop displays demonstrates that the integration of display and interaction device is beneficial for interactive visualization [179, 180]. In particular multi-touch gestures strive for naturalness. However, interaction is still mainly based on 2D positional input generated by pointing or moving fingers on the display's surface.

On the other hand, visualizations printed on paper are limited in terms of interactively altering the graphics. However, it is quite intuitive to grab a piece of paper, to move it towards the eyes to see more detail, and to put it back for an overview. Similarly, it is quite easy to fold pages in a report or to arrange multiple printouts on a desk to compare figures side-by-side. Doing so in multiple view environments on a computer display might involve several steps of reconfiguration of the visualization, which may turn out to be cumbersome when using mouse and keyboard alone. In a sense, an advantage of hardcopy visualizations is that they serve as a device for direct interaction and as a display at the same time.

In our research, we aim to narrow the gap between common interaction performed on the display and the natural manipulation we perform with paper. To that end, we developed what we call *tangible views*. A tangible view is a physical surface, for example a piece of cardboard, that users can hold in their hands. As long as it is handy, there is no restriction on a tangible view's shape and size. A tangible view serves two purposes – it is used as a local display in conjunction with a tabletop display, and it is used as an input device. Display functionality is realized by projecting specific graphical information onto tangible views. The three-dimensional manipulation of a tangible view is tracked in space to make more degrees of freedom avail-

able for interacting with the visualization and the data. While a single tangible view can already be a promising alternative to classic interaction, the true potential of our approach lies in the possibility to use multiple tangible views at the same time. In the latter case, tangible views do not only cater for natural interaction, but they also supersede virtual multiple views by physical ones, which can be freely moved in space. In summary, tangible views:

1. *Integrate display and interaction device.* By holding a display in your hand, one can interact *with it* in several gestural ways to change the displayed view and visualization parameters. The support of touch and pen interaction directly *on* the handheld display allows for additional interactivity.

2. *Enhance common 2D interaction with additional 3D interaction.* The usage of a graspable display that can be moved freely in 3D space implies a very natural way of interaction based on the metaphor of looking at pictures or documents.

3. *Replace virtual views by physical, tangible views.* Tangible views provide additional physical display space that can be utilized to support multiple coordinated views, overview & detail as well as for focus + context techniques.

The main contribution of this paper is a conceptual investigation of tangible views in the context of information visualization. We start by an analysis of related work, followed by a description of properties and degrees of freedom of tangible views as a tool of both representation and interaction. Subsequently, the applicability of tangible views to a variety of information visualization solutions is illustrated with several cases studies. Thereby, we demonstrate that tangible views are an interesting alternative to classic interaction and that they can be used for novel kinds of interaction that are more natural and intuitive compared to traditional input and output devices. We continue with a discussion of early user feedback and possible limitations. Later, technical aspects of the system are briefly described. Finally, we close with a reflection of our approach and indicate directions for future work and potential applications of tangible views.

10.2 RELATED WORK

10.2.1 *Conventional Interactive Visualization*

Conventional information visualization solutions address desktop computers with a single virtual desktop (possibly one that spans multiple stationary displays) and standard input devices (e.g., mouse, trackball, keyboard). One or multiple virtual views are shown that provide different visualizations of the data under investigation. Common use

cases for multiple views are to provide overview and detail or to compare alternative visual encodings or different parts of the data [363].

To accomplish exploration tasks, the interactive adaptation of the visualization to the task and data at hand is crucial. Yi et al. [385] identified several high-level user intents for interaction. Users want to mark something as interesting, e.g. specific data items by brushing [52]. For exploratory analyses, users also need to alter views. This can be achieved by navigating the view space [83, 355] or the data space [333], or by using common user interface controls to adjust the visual encoding and to rearrange views on the virtual desktop. Particularly for larger data sets it is necessary to filter the data interactively [19] and to switch between different levels of abstraction [107]. For higher order visualization tasks users often need support for relating and comparing data items [110, 330].

Technically, any interaction can be modeled as adjustments of visualization parameters [189]. With direct manipulation [303], users interact directly with the visual representation. Physical movement of pointing devices is translated into specific interaction semantics, for instance, to select data items of interest (see [155, 170]) or to transform the view on the data (see [164, 109]). Indirect manipulation uses control elements, such as sliders, to adjust numeric visualization parameters or to filter out data items that are irrelevant.

A special class of techniques are virtual lenses [58]. Lenses combine different visualization and interaction concepts in one interactive tool. Lenses exist that magnify interesting items or regions [293], that filter represented information [106], that rearrange visualized data items [330], or that adjust the visual encoding [324]. The diversity of lens techniques indicates that they are a universal tool to support most of the user intents identified by Yi et al. [385]. Generally, a lens is defined as a spatially confined sub-region of the display and a lens-specific visual mapping. By means of direct manipulation, users can move the lens to specify the part of the visual representation that is to be affected by the lens mapping.

10.2.2 *Towards More Direct Interaction*

Direct manipulation in information visualization can be accomplished with *indirect* pointing devices, such as the prevalent mouse, where the input space does not coincide with the display space. *Direct input*, by contrast, unites the interaction and display space and is often performed using a digital pen or finger on touchscreens. An enhancement is multi-touch technology that allows users to execute commands by performing complex gestures with multiple fingers on the display surface simultaneously. Even though natural direct manipulation concepts lends themselves to the field of information visualization, the mouse still dominates the field.

Approaches that investigate direct or tangible interaction in information visualization are scarce. Isenberg and Carpendale explicitly make use of interactive tabletop displays for the purpose of performing comparison tasks [179]. Via direct interaction on the tabletop, users can compare aspects of tree representations. Isenberg and Fisher apply multi-touch technology to support collaborative visualization tasks [180]. The iPodLoupe introduced by Voida et al. [360] goes one step further and adds a physical local display to the visualization environment. While a large interactive tabletop display shows the visualization context, a small focus display (iPod) is used to show details. Yet, the interaction remains on the tabletop display; users cannot interact by manipulating the focus display in space.

The traditional visualization methods reviewed above are mostly using indirect input and are based on virtual views, i.e., windows on a physical display or local views embedded into the visualization. Spatially aware displays, which know precisely about their position and orientation in 3D space, are a promising approach to make virtual views physically touchable and to accomplish direct and natural interaction.

A pioneer work in making virtual views physically tangible is the metaDESK system by Ullmer and Ishii [346]. The system consists of a tabletop display and an LCD panel that is attached to the tabletop via a mechanical arm. By moving around the LCD panel users can navigate through polygonal 3D models. Yee's Peephole Displays [384] support the interaction with a digital desktop environment (calendar, web browser, street map) that is virtually wrapped around the user. A prominent example of a paper-based passive display are Paper Windows by Holman et al. [171], which support various ways of interacting with a graphical user interface. Sanneblad and Holmquist used spatially aware monitors to magnify details of a street map that is displayed on a large stationary vertical display [291]. In [255], Molyneaux et al. present a technical architecture for bi-directional interaction with tangible objects (input/output), similar as proposed in our work. However, their discussion is mostly on technical aspects and focuses only briefly on modalities of interaction. To allow for simultaneous back-projection of different contents onto a tabletop surface and a tangible lens, respectively, Kakehi and Naemura use a special projection foil that changes its translucency depending on the projection angle [196]. The SecondLight system by Izadi et al. [186] supports dual projections by using electronically switchable diffusers. The PaperLens [312] is a technically less complex combination of a tabletop context display and a spatially aware lightweight focus display. The system allows users to explore spatial information spaces simply by moving the lightweight display through the physical 3D space above a tabletop surface.

10.2.3 *Closing the Gap*

In summary, we see a twofold gap. On the one hand, information visualization strives for natural direct manipulation with the visual representation and the data, but only few approaches utilize the available technologies to this end. On the other hand, various approaches have been developed to support direct interaction with lightweight physical displays, but none of them addresses the specific representational and interaction aspects of information visualization.

Our aim is to narrow this gap by means of tangible views. The work we present here builds upon the previous PaperLens system, where a tabletop display provides the contextual background for the exploration of spatial information spaces with a spatially aware tangible magic lens [312]. Four different classes of information spaces were identified and are supported by the system: *layered*, *zoomable*, *temporal*, and *volumetric* information spaces. While horizontal translation (x-y position on or above the tabletop) is reserved for panning operations, lifting or lowering the magic lens enables users to choose from a set of two-dimensional information layers, to perform zooming of a high-resolution image, to go forward or backward in time in a video, and to explore the third dimension of a volumetric data set. Thanks to this explicit mapping of the magic lens' z-position (height above the tabletop) to the defining characteristics of each data class, users experienced the exploration of these information spaces as intuitive and natural. This motivated us to use PaperLens as the basis for our work.

In this paper, we technically and conceptually extend this approach in the following key points: (1) generalization of the interaction vocabulary including novel gestures and support for multiple tangible views, and (2) mapping of the vocabulary to semantics specific to information visualization.

10.3 TANGIBLE VIEWS

In this section, we will systematically investigate tangible views as a class of devices that serves two purposes at the same time: as a tool of representation and as a tool of interaction. We begin our discussion by focusing on the general characteristics and illustrate what is syntactically possible when using tangible views. In the next section, we will add semantics to these possibilities by mapping them to tasks that are common in the field of information visualization.

10.3.1 *Tool of Representation*

In its simplest form, a tangible view is a spatially aware lightweight display or projection surface onto which arbitrary information can be

projected. Tangible views usually do not exist on their own, but instead are integrated into an environment of one or more stationary displays of arbitrary size, shape and orientation. By displaying graphical information, these stationary displays or surfaces both define and provide the contextual background of a virtual information world in which a tangible view exists. A basic display configuration will be used throughout this paper: a horizontal tabletop whose purpose is to serve as the main context view and tangible views as local views into the information space. This thinking relates to the focus + context concept.

One important advantage of tangible views is that they can be used with other tangible views simultaneously. Thus, they can be understood as a multiple view environment with each tangible view representing a unique physical view into a virtual information world. This characteristic makes them an ideal tool for collaboration or comparison tasks and for supporting the overview and detail approach. Besides that, tangible views usually appear in different shapes and sizes. Most commonly a tangible view will be of rectangular or circular shape, but other more sophisticated shapes, like hexagonal or metaphorical shapes (e.g., "magnifying glass"), are possible and may play a special role during interaction.

10.3.2 *Tool of Interaction*

Throughout our investigations of the various aspects of tangible views, we aimed at as-easy-to-learn and as-natural-as-possible usage that is inspired by everyday life interaction principles. Interacting with tangible views is basically as simple as grabbing a lightweight physical object (the tangible view) with one or both hands and then moving it around in the real-world space, while the tangible view constantly provides appropriate visual feedback. The actual interaction takes places within the physical space that is defined by the stationary display that serves as the contextual background. In our case, the space above the horizontal tabletop's surface is used as the three dimensional reference system that we refer to as the *interaction space*. Despite previous research on interacting with non-rigid tangible screens, such as foldable [229] or bendable [299] approaches, we restricted our investigations on rigid tangible displays only.

As with all rigid objects in a 3D space, there are six degrees of freedom (6DOF) available for interaction. More precisely, the basic degrees of freedom are the position (x, y, and z) with respect to the interaction space and the local orientation of the tangible view (α, β, and γ). Corresponding interactions are translation and rotation, respectively. Both are very easy to learn and simple to execute. Additionally, interaction can be enhanced by introducing higher level interaction gestures (on the basis of basic degrees of freedom). Such

gestures enrich the interaction vocabulary of users and thus can make it easier for them to solve particular sets of problems.

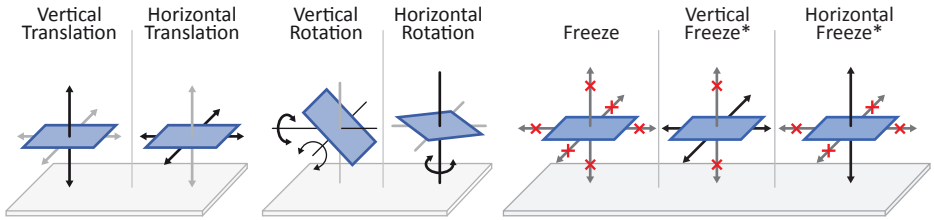
It is important to note that the ways of interaction that we discuss here are similar to those in the field of tangible interaction, where "graspable" objects represent specialized tools that can be used to physically interact with a display surface, in particular tabletops. However, there are three major differences between *tangible* and *tangible view* interaction. First, traditional tangible interaction is limited to the tabletop surface itself, whereas the usage of the space above it is rarely seen with the Multi Layer Interaction for Digital Tables by Subramanian et al. [318] being a minor exception. By contrast, with tangible views we propose a technique that utilizes the space above a tabletop explicitly for the purpose of interaction. Second, tangibles usually are characterized by specialized form factors or well-defined shapes that make them fit perfect for a particular task or set of tasks, e.g. for adjusting parameters such as in SLAP Widgets by Weiss et al. [373]. On the contrary and although tangible views can come in various shapes too, they provide a much more generic and multipurpose way of interaction. This is probably due to the third important difference: tangible views provide a constant visual feedback and thus their appearance is customizable. This is a feature that traditional tangibles lack or at least provide very seldom or only in a limited way.

10.3.3 *Interaction Vocabulary*

The design space for tangible views is more complex and richer than it looks at a first glance. Therefore, some fundamental principles need to be found and understood that help both users and system designers. In this respect, many interaction techniques, such as gestures, have been described and used previously. Our intention was to organize, combine, and extend these ideas in a meaningful way and with focus on tailoring them towards the domain of information visualization. This was one goal of our research and as a result we identified the following eight basic usage patterns for tangible views: *translation*, *rotation*, *freezing*, *gestures*, *direct pointing*, the *toolbox metaphor* as well as *multiple views*, and *visual feedback*. The first six patterns are mainly motivated by the available degrees of freedom and additional interaction modalities, and thus are features of the "tool of interaction". In contrast, the last two patterns (visual feedback, multiple views) are motivated by properties of the "tool of representation". In the following, we will discuss these eight patterns for tangible views in more detail.

10.3.3.1 *Translation*

One way of interacting with a tangible view is to interpret its current 3D position and thus to utilize shifts of movement as a form of inter-



(a) Translation (b) Rotation (c) Freezing

Figure 10.1: Overview (1 of 3) of the interaction vocabulary of tangible views (asterisks denote novel techniques).

action [312]. The resulting three degrees of freedom (3DOF) can then be interpreted either by utilizing all 3DOF at the same time or by restricting them to one or two axes: *horizontal translation* as movement in the x-y-plane and *vertical translation* as movement along the z-axis (see Figure 10.1a).

10.3.3.2 *Rotation*

Another way of interacting with a tangible view is to use its local orientation, i.e., changes of α, β, and γ (3DOF). Without the claim of completeness, we distinguish between two types of rotation: *horizontal rotation* [227] around z and *vertical rotation* [229] as rotations around x and/or y. This is illustrated in Figure 10.1b.

10.3.3.3 *Freezing*

In certain situations, it is necessary to move a tangible view without the intention of interacting with the system. This happens, for example, when users want to study a specific view in more detail or when they want to keep it for later examination by physically placing the view on or beside the table surface. For this purpose, we introduce the possibility of freezing a tangible view (see Figure 10.1c). In terms of degrees of freedoms used for interaction, this means that the system ignores shifts of movement for all or some principle axes. We distinguish between three different freezing modes: normal *freeze* [311] where x-y-z are locked, *vertical freeze* where only z is locked, and *horizontal freeze* where only x-y are locked. Hereby, the latter two techniques are new to the field.

10.3.3.4 *Gestures*

So far, we used the available 6DOF in a very direct manner. But there is room for more complex types of interactions by using the concept of gestures. In order to enrich the interaction with tangible views, we propose the use of following (non-exhaustive) set of simple gestures: *flipping* [171], *shaking* [374], and *tilting* [91] (see Figure 10.2a). The principle idea of flipping is to attach different meanings to the front

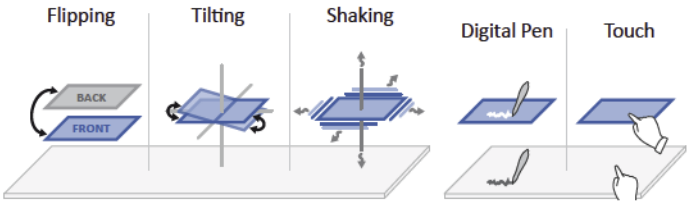


Figure 10.2: Overview (2 of 3) of the interaction vocabulary of tangible views (asterisks denote novel techniques).

and the back side of a tangible view and thus to let users interact with the system by turning a tangible view around. As the name implies, shaking lets users interact with the system by arbitrarily moving a tangible view to and fro. In contrast, *sideways* and *frontways tilting* is like slanting the tangible view slightly to the left/right (around the y-axis) and to the front/back (around the x-axis), respectively.

10.3.3.5 *Direct Pointing*

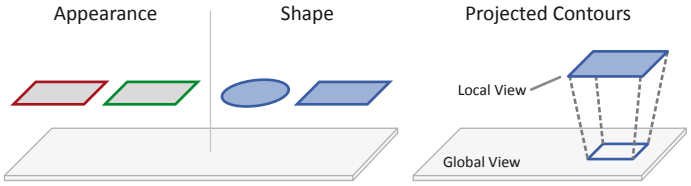
Direct pointing borrows its ideas from the fact that in addition to interacting *with* tangible views, it is also possible to perform interaction *on* them by providing further methods of input. Without loss of generality, we distinguish between *multi-touch* and *digital pen* for interacting on both the tangible views and the context display (see Figure 10.2b). These technologies allow users to interact with a visual representation by means of *direct pointing*. Thumb movements or touch, for instance, can be recognized to control context-sensitive user interface elements on tangible views. Digital pens are utilized for more precise input such as for writing or exact pointing [311].

10.3.3.6 *Toolbox Metaphor*

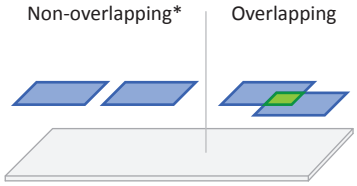
The main idea of the toolbox metaphor is to assign specialized tasks to the physical properties [123, 347] of tangible views. In particular the *shape* (e.g., circle or rectangle) and the *visual appearance* (e.g., color or material) of a tangible view are relevant. As hinted in Figure 10.3a, these properties can be used as a basis to decode certain tasks or tools by the physical look of a tangible view. Following this concept, a set of pre-manufactured tools (tangible views) is presented in close proximity to the tabletop. Depending on their aim of interaction, users can then easily select the appropriate tool for a particular problem by simply grabbing another tangible view.

10.3.3.7 *Visual Feedback*

Visual feedback is a fundamental requirement for the interaction with a visual system such as tangible views. When interacting with tan-



(a) Toolbox Metaphor (b) Visual Feedback



(c) Multiple Views

Figure 10.3: Overview (3 of 3) of the interaction vocabulary of tangible views (asterisks denote novel techniques).

gible views, users expect instant visual feedback in correspondence with the interaction. Such feedback is provided directly on a tangible view or on the stationary tabletop display. Visual feedback also serves to illustrate the interplay of views by projecting the tangible view's contour lines onto the tabletop surface [312] (see Figure 10.3b).

10.3.3.8 *Multiple Views*

As depicted in Figure 10.3c, tangible views support the concept of multiple *local views* within the reference system of a *global view*. We distinguish between *non-overlapping* and *overlapping* local views. We define the term overlapping as whether two or more tangible views consume the same or partly the same horizontal (x-y) space above the tabletop (by ignoring the z-axis). In our understanding, overlapping tangible views can influence each other, i.e., the visual output of one tangible view may depend on the output of another one. In contrast, non-overlapping tangible views are independent from each other. In combination with *freezing*, *multiple views* provide the foundation for several two-handed comparison techniques as described in the next section. To the best of our knowledge, such tangible comparison techniques have never been presented before.

10.4 CASE STUDIES

From the previous section we see that tangible views provide a rich vocabulary that comprises interaction aspects *(tangible)* and representation aspects *(view)*. This section will address the question of how this vocabulary can be applied to information visualization. Our discussion begins with some general considerations. Then, we explain

how tangible views can support users in accomplishing common interaction tasks. For this purpose, we have implemented five visualization approaches that demonstrate the versatility of tangible views.

10.4.1 *General Considerations*

Traditional visualization techniques address a two-dimensional *presentation space* that is defined by the axes of the display. In contrast, with tangible views we extend the presentation space by a third axis – the z-axis that emanates perpendicularly from the tabletop surface. The motivation for the extension to the third dimension lies in the data-cube-model with the space above the tabletop display being the physical equivalent of an otherwise virtual data-cube. This allows us to project data not only onto the tabletop's surface, but also into the space above it. As we will see in the following case studies, there are various options how to utilize the additional dimension for interaction and visualization purposes.

Here, two fundamental aspects are *multiple view visualizations* (that provide different visual representations simultaneously [363]) and *lens techniques* (local views with a specific visual encoding embedded into a visualization context). As any tangible view functions as a physical window into virtuality, *multiple views* and *lenses* can easily be made tangible. Beyond that, direct manipulation is naturally supported by tangible views as well: users can move a tangible view around to specify the area or the data items to be affected by the lens.

10.4.2 *Case Study: Graph Visualization*

Node-link-diagrams and hierarchical abstraction are classic means to enable users to interactively explore large graphs. Starting at the abstraction's root, users expand or collapse nodes in a series of discrete interactions until information density suits the task at hand. A contin-

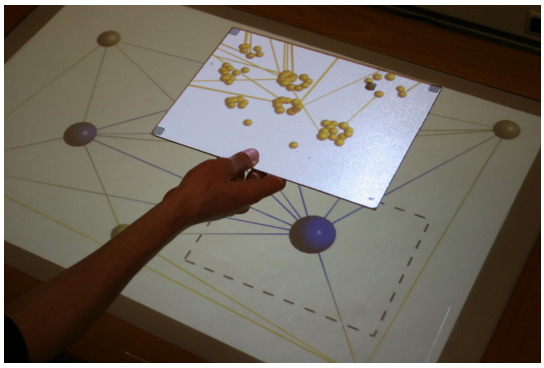
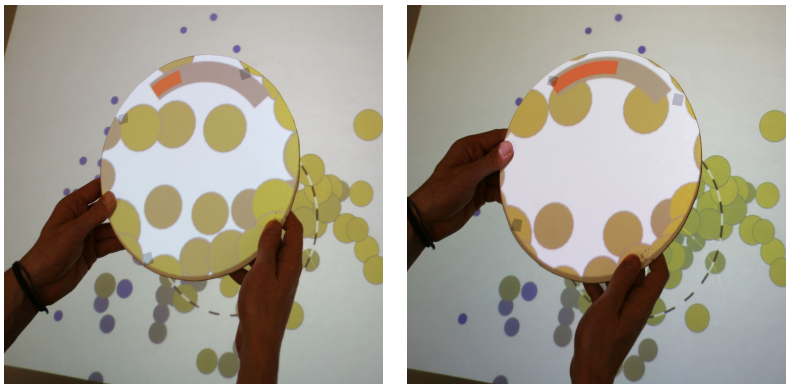


Figure 10.4: A tangible view is used for smoothly exploring a graph at different levels of abstraction.

uous navigation through the different levels of abstraction has been introduced by van Ham & van Wijk [351]. We implemented a tangible variant of such an abstraction lens and applied it to explore relations in the ACM classification. As demonstrated in Figure 10.4, a rectangular tangible view serves as a local abstraction view for the graph that is being shown on the tabletop display. Users can naturally pan the view by using *horizontal translation* and freely change the degree of detail by *vertical translation*. This way it is possible to quickly explore different parts of the graph and compare relations at different scales. At all times, the tabletop display provides *visual feedback* about the current positions of the local view within the global view.

10.4.3 *Case Study: Scatter Plot*

Scatter plots visualize correlation in a multivariate data set by mapping two variables to x-y position of graphical primitives, where color, size, and shape of these primitives can be used to encode further variables. However, graphical primitives could become very tiny and could overlap or occlude each other, which impedes recognition of color and shape. To make size, color, and shape of the data items discernible, our scatter plot implementation is equipped with a graphical zoom lens and a simple fisheye lens, which temporarily sacrifices the positional encoding to disentangle dense parts of a scatter plot. According to the *toolbox metaphor*, a rectangular tangible view and a circular tangible view represent the zoom lens and the fisheye lens, respectively. The tabletop display serves as the visual context showing two data variables (out of four of the well-known IRIS data set) mapped onto the tabletop's x and y-axis, respectively. Users can easily alternate the variables to be visualized by using *frontways* (x-axis)



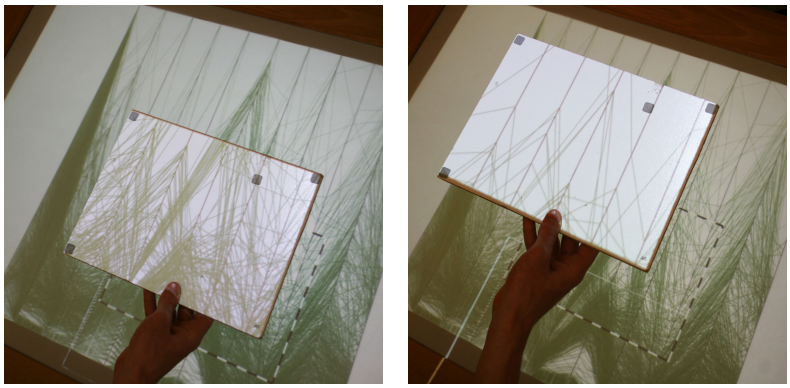
(a) Low degree of displacement. (b) Higher degree of displacement.

Figure 10.5: Scatter plot: A circular fisheye lens allows users to control the parameters lens location and degree of magnification by using *horizontal* and *vertical translation*, respectively. The fisheye-lens' degree of displacement is adjusted by *horizontal rotation*.

and *sideways tilting* (y-axis). *Horizontal translation* is again applied to set the lens location and *vertical translation* controls the geometric zooming factor for the zoom lens. The degree of displacement for the fisheye lens is manipulated with *horizontal rotation*. During this latter interaction, a curved slider on the view's surface provides *visual feedback* of the current parameter value (see Figure 10.5).

10.4.4 *Case Study: Parallel Coordinates Plot*

Classic parallel coordinates encode multiple quantitative variables as parallel axes and data items as polylines between these axes. This encoding is useful when users need to identify correlated variables or clusters in the data. However, as the number of data items increases, parallel coordinates suffer from cluttering. Ellis and Dix suggest using a sampling lens to mitigate the problem [106]. As sampling is often random, it is not clear in advance, what a good sampling factor is. We implemented a tangible sampling lens (see Figure 10.6) that supports users in interactively finding a suitable sampling factor. While the background visualization shows the whole dataset (11 variables and 1100 records of a health-related dataset), the tangible lens shows only every i-th data item. Analogous to the previous case studies, the lens location is set by *horizontal translation*. By *vertical translation*, users can traverse through possible values for i (degree of sampling). For the purpose of demonstration, our basic prototype simply uses $i \in (1, 2, 4, 8, 16 \ldots)$. Beyond that, attribute axes of the parallel coordinates plot can be reordered with *direct pointing* by using digital pens (Anoto). Axes rearrangements can be performed on both tangible view and tabletop.



(a) Every 32nd polyline of the original dataset is displayed.

(b) Every 256th polyline of the original dataset is displayed.

Figure 10.6: A tangible sampling lens supports users in finding an appropriate sampling factor by using *vertical translation*. Projected outlines on the tabletop helps users mentally linking local and global views.

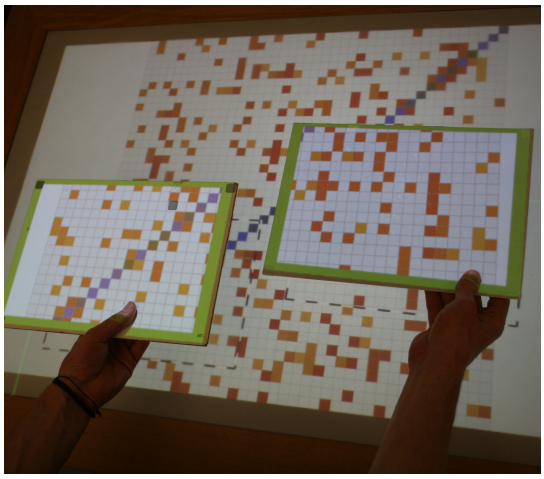


Figure 10.7: Using multiple tangible views simultaneously facilitates visual comparison tasks.

10.4.5 *Case Study: Matrix Visualization*

Yi et al. [385] list visual comparison as an important interaction intent that involves various steps, as for instance, selecting subjects for the comparison, filtering the data to compare specific subjects only, or encoding of additional information to support the comparison. Performing visual comparison with traditional means is usually difficult due to the numerous heterogeneous interactions participating. On the other hand, direct interaction on a tabletop can facilitate comparison [179]. How tangible views can be applied for visual comparison will be illustrated next. For the sake of simplicity, we use rectangular tangible views and a matrix visualization of a synthetic graph (42 nodes and 172 edges) that is displayed on the tabletop display as shown in Figure 10.7. In the first phase of comparison, tangible views are used to select data subsets. By *horizontal* and *vertical translation* users can determine position and size of a subregion of the matrix and then *freeze* the selection. Once frozen, the user can put the tangible view aside and take another one to select a second data subset. The two frozen tangible views can now physically be brought together either by holding each one in a separate hand or by rearranging them on the tabletop. As additional visual cues, smooth green and red halos around compared data regions indicate similarity and dissimilarity, respectively. If a selection is no longer needed it can be deleted by the *shaking* gesture.

10.4.6 *Case Study: Space-Time-Cube Visualization*

Space-time-cubes are an approach to integrate spatial and temporal aspects of spatio-temporal data in a single visual representation [213]. The analogy between a space-time-cube and the three-dimensional

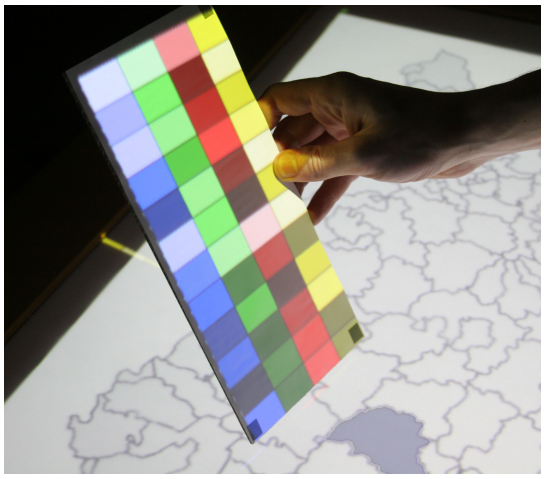
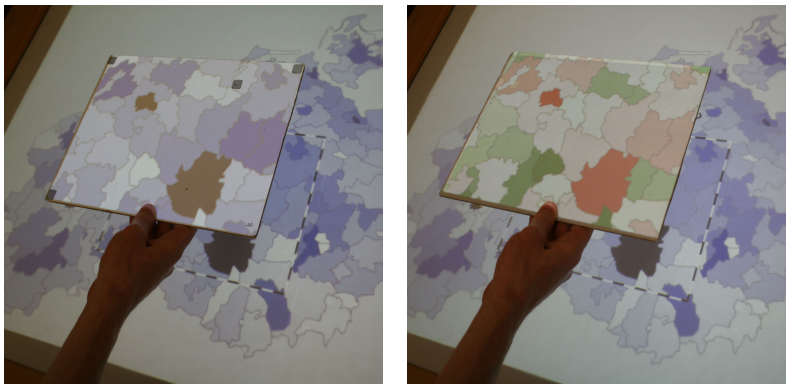


Figure 10.8: Tangible views can be used to augment a map display with additional visual representations.

presentation space used for tangible views motivated this case study: the tabletop display's x and y-axis show the spatial context as a geographic map, and the dimension of time (12 months) is mapped along the z-axis.

Tangible views are used as physical viewports into the space-time-cube. Interactive exploration is driven by *horizontal* and *vertical translation* to navigate the map and the time axis, respectively. When held in a horizontal orientation, a tangible view shows the data for a selected month, i.e., a horizontal slice through the space-time-cube. To get an overview for all months (i.e., a vertical slice), users can *rotate* a tangible view into upright orientation. Then the visual representation changes to a simple color-coded table that visualizes multiple variables for all 12 months for the area above which the tangible view is hovering (see Figure 10.8). Depending on whether the user's task is to identify data values or to locate data values, different color schemes are used to encode data values (see Figure 10.9) [331]. Simply *flipping* the tangible view switches between both tasks.

Exploring spatio-temporal data usually involves comparing different areas, different time steps, or both in combination. *Freezing* a tangible view helps users to accomplish these goals more easily. With *vertical freeze*, a tangible view can be locked to a certain month, effectively unlinking *vertically translation* and navigation in time. When frozen, the tangible view can even be put down to relocate the entire interaction to the tabletop surface itself. This can be quite useful for handling *multiple views* simultaneously in order to compare attributes between different areas, or for marking a certain detail for later examination by simply leaving the tangible view on a particular area. *Horizontal freeze* lets users lock a tangible view to a certain area. This is useful for comparing different months of the same location. To this end, the user simply locks two tangible views onto the same area. It is



(a) Before flipping: visualization supporting the task of identification.

(b) After flipping: visualization supporting the task of localization.

Figure 10.9: By *flipping* a tangible view, users can choose between visualizations that support different tasks.

then possible to lift or lower the two views independently to compare two months, while the horizontal freeze guarantees that the focused area does not change unintentionally (two-handed comparison, see Figure 10.10).

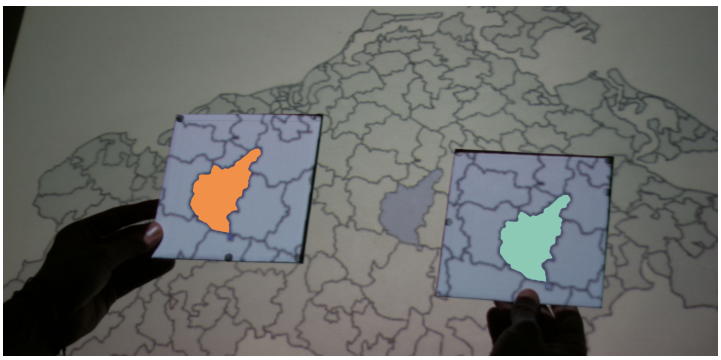


Figure 10.10: After locking the focus of two tangible views to the same location by *horizontally freezing*, users can visually compare between the two views by lifting or lowering them simultaneously.

10.5 DISCUSSION

From designing the case studies and initial user feedback, we crystallized a set of observations that may be useful guidelines for further more complex applications. We also discuss potential limitations and critical comments of users.

10.5.1 *Observations*

Based on the case studies, we derived following observations:

I. Providing direct *visual feedback*, such as cast shadows of tangible views on the tabletop, helps users mentally linking local and global views.

II. *Translation* should be reserved for navigation in presentation space.

III. *Freezing* is essential to temporarily decouple a tangible view from one or multiple axes of the interaction space. This is necessary to support tasks that require rearrangement of views, most prominently, comparison tasks, but also helps switching to traditional interaction, such as multi-touch.

IV. *Direct pointing* is essential for interacting *within* local or global views (tangible or tabletop). It is a requirement for precise selection tasks.

V. By favoring orthogonal interaction (e.g., *shape* for choosing a tool, *translation* for navigating the presentation space, *horizontal rotation* for navigating the parameter space, and *tilting* for navigating the data space), users can implicitly communicate their intent to the system without the need of explicitly changing any global interaction states.

10.5.2 *Limitations*

We have shown the case studies to a variety of users and generally received positive feedback. Even domain experts, at first reluctant, were quickly convinced of the techniques after seeing a live demo. Interestingly, before testing the demo and by only knowing the theoretical concept, some of them suspected that it would be: "too tiring to hold and move the tangible views through the air" if compared to: "use a stylus on a tabletop, where users can rest their elbow on the surface". Although this is true for extensive use, users commented that the mix of tangible interaction and the use of more traditional pen or touch input, e.g., after *freezing* a tangible view and laying it down on the table, reduced this problem considerably. In general, users did not have problems with lifting up the tangible views too high, because we restricted the physical interaction volume to 40 cm above the table. Thus, users were able to find boundaries of the interaction volume quite easily (no visual feedback above a certain height). In some cases, we also provided additional navigational aids, such as height indicators inspired by Spindler et al. [312]. Users felt that

this was helpful for finding certain layers of information more efficiently. In addition, the system allows to tilt lenses slightly in order to prevented viewing angle becoming too glancing. Sometimes, users complained about problems with precise interaction and hand tremor when moving or rotating tangible views in order to adjust an accurate position or angle. Here, convincing solutions need to be found and evaluated, which is beyond the scope of this paper. Also, one user suggested to provide better support for letting users know which actions are mapped to what. Similar to traditional GUI widgets, labels and tooltips could reveal what the widget does or even show that there is an affordance. The same user remarked that: "each tangible view has a fixed size and shape, unlike standard windows in a GUI". This could be tackled by having a collection of different sized tangible views or even by future hardware that allows unfolding of displays, similar to [229]. Despite of these issues, we are very confident that tangible views are a very powerful and generic technique that paves the way for an exciting field of research with many challenging questions.

10.6 TECHNICAL SETUP

For the technical setup of tangible views we extended the PaperLens approach by Spindler et al. [312], particularly in terms of tracking technology, gesture recognition, and direct input techniques (digital pens and touch) for both tangible views and the tabletop. The setup consists of a tabletop display as well as several infrared (IR) cameras and a top projector that are attached to the ceiling. This setup is enriched with various tangible cardboard displays (tangible views) that can be freely moved through the space on or above the tabletop.

In order to bring such a system alive, several problems need to be solved: tracking of tangible views to make them spatially-aware, displaying image content, recognizing gestures, support for direct pointing, as well as providing application functionality. Many of these tasks can be tackled independently from each other and thus have been split up (on a technical basis) between different computers. Hereby, for the purpose of inter-computer communication, we use public protocols for streaming device states (VRPN) and remote procedure calls (XML-RPC).

TRACKING The problem of determining position and orientation of tangible views is solved by tracking. Various tracking approaches have been used in the past, such as mechanical (arm-mounted) [346], ultrasonic [291], and optical solutions with visible markers [228]. However, a major design goal of PaperLens [312] is to conceal the technical aspects from users as much as possible (no cables, no disturbing markers, etc.). This has been accomplished by sticking small, unob-

trusive IR-reflective markers (4mm) to the corners/borders of tangible views. These markers can then be tracked by Optitrack FLEX: V100R2 cameras. As opposed to the original PaperLens implementation that only uses one camera and a simple home-made tracking, we extended the system by using six cameras and a commercially available tracking solution (Tracking Tools 2.0) that is more accurate and allows for defining arbitrary marker combinations. These are used to encode lens IDs (for the *toolbox*) as well as front and back sides of lenses.

DISPLAYING IMAGE CONTENT Tangible views are implemented as a passive display solution, i.e., image content is projected from a ceiling-mounted projector onto an inexpensive lightweight projection medium (cardboard or acrylic glass). This allows for a low-cost production of tangible views in various sizes and shapes (rectangles, circles, etc.) and also includes control of the visual appearance (color, material) as well as using the tangible views' back sides as displays. Once position and orientation of a tangible view is known, this information is fed to a computer. Thus, the connected top projector can project arbitrary image content onto the tangible view. In order to maintain a perspectively correct depiction of the image content, OpenGL is used to emulate the physical space above the tabletop including all tangible views that reside there. The OpenGL camera is precisely located at the virtual position of the top projector and the physical properties of lenses are represented by textured polygons (shape, position, and orientation). Image content is then rendered independently into FrameBufferObjects (FBO) that are attached to these textures. In this way, application code is separated from the more generic projection code. This will allow us to easily exchange the top-projected passive displays with lightweight active display handhelds in the near future.

RECOGNIZING GESTURES In order to support *flipping*, *shaking*, and *tilting*, a simple gesture recognizer has been implemented. *Flipping* is recognized with the help of unique markers that identify front and back side of a tangible view. For other gestures, we identified characteristic movement patterns that can be detected by the system, i.e., for *shaking* a rapid irregular movement with small extent and for *tilting* a back-and-forth rotation along an axis in a range of about 20°.

DIRECT POINTING In terms of interacting *on* tangible views, the system was augmented with support for direct pointing, in particular *touch* and *digital pens*. Digital pen technology can easily be incorporated by gluing Anoto-paper [1] onto a tangible view's surface. The Anoto paper shows a unique dot pattern that is scanned by special pens with a built-in camera for determining their position on the

lens's surface. This 2D position is then transmitted to the application via Bluetooth in real-time. The system was further enhanced with basic support for *touch* input. For this purpose, additional IR-reflective markers have been affixed to the surface of tangible views. By hiding these "marker buttons" with their thumbs, users can activate certain states, such as the *freeze* mode.

10.7 CONCLUSION

Conventional desktop display solutions and indirect interaction by means of traditional input devices are notable limitations for information visualization. To overcome these limitations, we introduced *tangible views*, which integrate display and interaction device. Tangible views provide additional display space and allow for a more natural and direct interaction. They serve as a viewport into a 3D presentation space and utilize the additional axis for various interaction and visualization purposes. To the best of our knowledge, this is the first time that spatially-aware displays were employed in the field of information visualization.

In this paper, we composed a versatile set of orthogonal interaction patterns serving as a basic vocabulary for carrying out a number of important visualization tasks. Users can perform a variety of gestures directly *with* a tangible view, or use touch and pen input *on* a tangible view. Tangible views provide haptic affordances combined with clear proprioception by means of body movements. At the same time, we are employing the well-known metaphors of moving sheets of paper on a desk as well as lifting photos and other documents to look at them in detail. As previous studies suggest [312], interaction with tangible views is perceived as very natural.

We see the true potential of our approach in the possibilitiy to provide interesting alternatives to classic techniques and to supersede *virtual* views by *physically tangible* ones. With that, fairly direct mappings can be achieved for multiple coordinated views, overview & detail techniques, and focus + context techniques, in particular lens techniques. In addition, bimanual interaction allows for the natural control of various visualization parameters in parallel, which cannot be accomplished with traditional desktop interfaces. Here, we contributed two-handed comparison techniques. By means of the *toolbox metaphor*, we can utilize tangible views to facilitate task-oriented visualization, which resembles the usage of physical workshop or kitchen tools.

FUTURE WORK The very positive early user feedback we received suggests that the application of tangible views for information visualization tasks is a very promising approach. However, further thorough studies of particular combinations of tangible views and visual-

ization techniques are required. For that, we need to refine the interaction techniques, especially with regard to touch input and parameter controls. In addition to the already investigated and mentioned applications of tangible views, there are further visualization challenges that may benefit from our approach, among them interactive visual assistance for data fusion with tangible view and collaborative problem solving supported by tangible views. With *tangible views*, we hope to have made a contribution especially to the interaction side of information visualization and to stimulate a discussion on more natural ways of looking at and interacting with data.

11

PHYSICAL NAVIGATION TO SUPPORT GRAPH EXPLORATION ON A LARGE HIGH-RESOLUTION DISPLAY

CONTRIBUTION The contribution of this chapter is a study on utilizing new interaction modalities to support the visualization on modern display devices. Physical navigation and head movement are employed in novel ways as low-level means of interaction to facilitate the visual exploration of large graphs on large high-resolution displays. Based on the user's position and viewing direction, detailed information is visualized in the users field of view, while aggregated information is shown in the periphery.

ABSTRACT Large high-resolution displays are potentially useful to pre-sent complex data at a higher level of detail embedded in the context of surrounding information. This requires an appropriate visualization and also suitable interaction techniques.

In this paper, we describe an approach to visualize a graph hierarchy on a large high-resolution display and to interact with the visualization by physical navigation. The visualization is based on a node-link diagram with dynamically computed labels. We utilize head tracking to allow users to explore the graph hierarchy at different levels of abstraction. Detailed information is displayed when the user is closer to the display and aggregate views of higher levels of abstraction are obtained by stepping back. The head tracking information is also utilized for steering the dynamic labeling depending on the user's position and orientation.

ORIGINAL PUBLICATION [230] — A. Lehmann, H. Schumann, O. Staadt, and C. Tominski. Physical Navigation to Support Graph Exploration on a Large High-Resolution Display. In G. Bebis et al., editors, *Advances in Visual Computing*, pages 496–507. Springer, 2011.

11.1 INTRODUCTION

Visualization is challenged by the fact that the available display space usually cannot keep up with the amount of information to be displayed. Commonly this problem is addressed by a kind of overview+detail technique [84], where users interactively switch between an abstract overview of the entire data and detailed views of smaller parts of the data. Large high-resolution displays (LHRDs) are an emerging technology and a promising alternative to the classic overview+detail approaches. LHRDs combine a large physical display area with high pixel density for data visualization (see Ni et al. [259]). A unique feature of LHRDs is that they allow users to perceive the global context of complex information presented on the display by stepping back, while enabling them to explore finer detailed data by stepping closer. This makes LHRDs suitable for visualization applications, such as the visual exploration of graphs.

However, because classic mouse and keyboard interaction is infeasible in LHRD environments, we require new approaches to interact with the visualization. Ball et al. [45] found that physical navigation is quite useful in this regard. We utilize this fact for a basic exploration task. In particular, the user's physical position in front of a LHRD determines the level of abstraction of the visual representation of hierarchical graphs. Graph nodes are expanded dynamically when the user moves closer to the display (i.e., more detail). When the user steps back, nodes are collapsed, by which a higher level of abstraction (i.e., less detail) is obtained. This provides a natural way to get an overview or detailed information similar to virtual zoom interaction via mouse's scroll wheel. A benefit of our approach is that the users do not need their hands for the interaction, which opens up possibilities to use them for other tasks.

Switching between different levels of abstraction also requires adaptation of the visual representation. While the operations expand and collapse correspond to addition and removal of nodes in the graph layout, respectively, the question remains how to appropriately label the differently abstracted objects shown on the LHRD. We utilize an existing labeling algorithm (i) to ensure readability of labels depending on the level of abstraction (determined by the user display distance) and (ii) to avoid readability problems caused by LCD-panel-based LHRDs.

In the next section, we will briefly describe basics of graph exploration and related work on interactive visualization on LHRDs. In Section 11.3, we will describe details of our approach to support graph exploration on LHRDs. Section 11.4 will outline the design of our prototype system and preliminary user feedback. We will conclude with a summary and an outlook on future extensions and applications of our approach in Section 11.5.

11.2 RELATED WORK

Our literature review is structured as follows. We briefly describe graph exploration in general and then consider visualization on large displays and shed some light on what modern input concepts beyond mouse and keyboard input can offer for interactive visualization.

11.2.1 *Graph Exploration*

Lee et al. [223] identified several tasks that users seek to accomplish with the help of graph visualization. Lee et al. structure low-level tasks by means of a taxonomy with three high-level categories: topology-based tasks, attribute-based tasks, and browsing-tasks. In each of these categories, we find tasks of exploratory nature, such as "find adjacent nodes", "find edge with largest weight", "follow a given path", or "return to previously visited node".

In general, interactive graph exploration is supported by appropriate visualization techniques (see Herman et al. [168]) and well-known overview+detail and focus+context interaction (see Cockburn et al. [84]). Such techniques enable users to zoom in order to view details or an overview, and to pan in order to visit different parts of the data.

A common approach to support the exploration of larger graphs is to structure (e.g., to cluster or aggregate) them hierarchically (see Herman et al. [168]). Elmqvist and Fekete [107] describe the various advantages of this approach, including the fact that it allows us to visualize different abstractions of the underlying data. In order to access the level of abstraction that is required for the task at hand, users interactively expand or collapse individual nodes of the hierarchy or switch between entire levels of the hierarchy.

Additionally, special lens techniques have been proposed for interacting in locally restricted areas of the visualization. For instance, van Ham and van Wijk [351] use a virtual lens to automatically adjust the level of abstraction (i.e., expand/collapse nodes) depending on the position of the lens. Tominski et al. [332] and Moscovich et al. [256] describe lenses that create local overviews of the neighborhood of selected nodes and provide direct navigation along paths through a graph.

All of these examples demonstrate the usefulness of dedicated interaction techniques for graph exploration. However, most of the existing techniques have been designed for classic desktop environments with regular displays and mouse interaction. In this work, we adapt existing concepts to make them applicable in LHRD environments. The exploration in terms of different levels of abstraction is of particular interest to us, because of the inherent overview+detail capabilities of LHRDs.

11.2.2 *Interactive Visualization on Large Displays*

The advantages of LHRDs (i.e., many pixels and natural overview+detail) make them an interesting alternative to desktop-based visualization scenarios. Next, we review existing approaches that utilize LHRDs for visualization.

As demonstrated by Keim et al. [202], pixel-based visualization approaches are a good example of visualization techniques that benefit from the larger number of pixels available on LHRDs. Another example is the visualization of complex spatiotemporal data. Booker et al. [61] explain that the exploration of spatiotemporal data can significantly benefit from LHRDs. They also conjecture major benefits for other information exploration scenarios. While these examples focus on the output capabilities of LHRDs, other researchers have addressed questions of interaction.

While mostly hand-held devices (e.g., wireless mouse, tracked button device) are used to pan or zoom in a desktop-based visualization application, such devices, if at all, are cumbersome to use in LHRD environments. A commonly accepted means to drive interaction in LHRD is tracking the user's physical movements. Vogel and Balakrishnan [358] use head tracking to switch between the display of ambient, public, or personal information depending on the user's distance in front of a large display. Ashdown et al. [39] switch the mouse pointer between monitors by head tracking to increase mouse movements in multi-monitor environments. Ball et al. [43, 45] identify that in LHRDs the performance of simple navigation tasks for finely detailed data can be increased by using physical navigation. This knowledge is realized by Peck et al. [267] who adjust the mechanisms for interactive selection and navigation based on the user's distance to the display. A survey of interaction techniques for LHRDs is presented by Khan [205].

Although physical interaction has a number of advantages, there are some limitations. For example, Ball et al. [44] use head rotation as input for panning navigation in a geospatial visualization application. However, virtual navigation and head motion were tightly coupled, which made it difficult for users to pan the map while also scanning it. Consequently, head tracking is better suited for simple interaction mechanisms than for fine motor control.

Overall, prior work shows that using physical navigation in LHRD environments can improve both perception of the visualized data [43, 45] and interaction with the visualization [267]. Considering these advantages, we devised an interactive visualization approach to support the exploration of graphs on LHRDs at different levels of abstraction.

11.3 EXPLORING GRAPHS ON A LARGE HIGH-RESOLUTION DISPLAY

In recent years, graphs have gained importance in many application backgrounds such as social networks, power networks, climate networks, biological networks and others. We present an approach that is suitable to support graph exploration by exploiting the advantages of LHRDs and physical navigation.

First we will explain the data that we address, which are basically graph hierarchies. Then we describe the node-link-based visualization employed in this work. Finally, we introduce novel interaction techniques for adjusting the level of abstraction of the displayed data based on head tracking.

11.3.1 *Data*

We use a *graph hierarchy* H as the main data structure to drive the exploration [168, 107]. H is a rooted tree whose leaves represent information on the finest level of granularity. Non-leaves of H are abstractions of their corresponding child nodes. Any "full cut" through H defines a view of the data with a specific level of abstraction.

There are two basic means to enable users to choose a suitable level of abstraction: (i) one can globally switch from one level of the hierarchy to another or (ii) one can expand and collapse nodes in order to adjust the level of abstraction locally. Switching the level globally means replacing all nodes of the current level with the nodes of another level. On the other hand, expansion of a non-leave node locally replaces that node with its children, which results in more information and less abstraction. Collapsing a set of nodes replaces these nodes with their parent node, which results in less information and more abstraction.

For the purpose of demonstration, we visualize the hierarchy of the ACM computing classification system, where nodes correspond to text labels of the categories and edges between nodes indicate related categories. With its 1473 nodes and 464 edges this data set is not too large and the explicitly given labels on the different levels of the hierarchy are expressive and easy to understand. Both facts make this data set quite useful for first experiments.

Note that edges have been added by hand, which is the reason why an edge that exists on a higher level of abstraction may have no corresponding edges on lower levels. In particular, no edges exist on the finest granularity of the ACM data set.

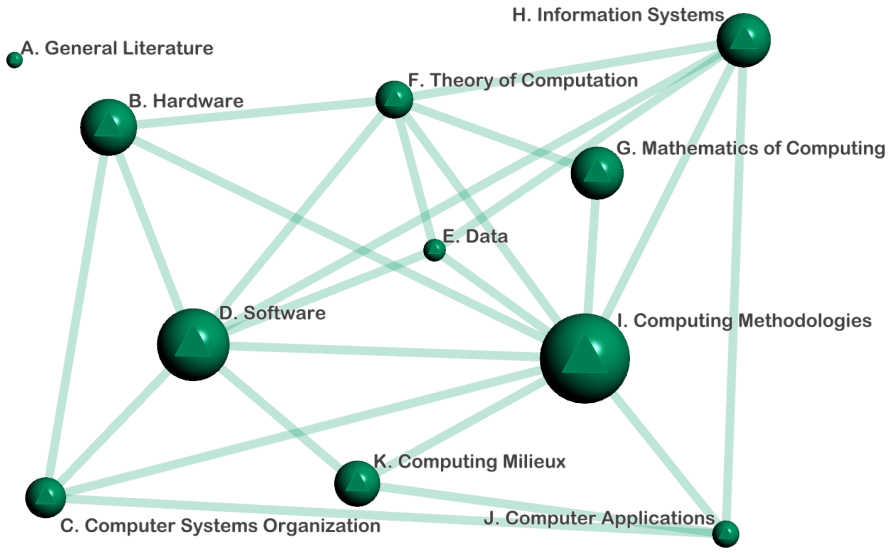


Figure 11.1: Node-link view of the top level of the ACM computing classification system.

11.3.2 *Visualization*

The visualization relies on a basic node-link representation (see Figure 11.1). The required graph layout has been pre-computed using a recursive hybrid algorithm: A force-directed mechanism determined node positions for connected parts of the graph [47] and a variant of the squarified treemap layout handled unconnected parts [65]. In a second phase, node positions were adjusted manually to better echo the semantics of the data (e.g., the ordering of categories). Third, an automatic local adjustment of the layout is computed to avoid that nodes are placed behind bezels of our LCD-panel-based LHRD.

In our node-link visualization, nodes are visualized as spheres. To separate the various hierarchy levels, the spheres are colored depending on the depth of the node in the graph hierarchy using a sequential color scheme from ColorBrewer [152]. The links between spheres are shown as colored lines. Along a line, we interpolate the color of its two incident spheres. This way, edges between nodes of the same level in the hierarchy (i.e., no interpolation, because both spheres have the same color) are clearly distinguishable from edges between nodes of different levels (i.e. color interpolation, because sphere colors are different). Although the layout already communicates the hierarchical structure of the data quite well, we use convex hulls as visual envelopes for nodes that share the same parent.

Because we work with a graph whose nodes represent category captions, we have to attend to label placement. This is also relevant when it comes to showing text labels with associated node properties (e.g., node degree, node depth, etc.). To ensure label readability, we have to account for two aspects. First, we have to deal with bezels

of LCD-panel-based LHRDs. Usually, bezels are handled as part of the display space and an empty virtual space is placed behind them in order to make virtual objects (e.g., spheres) drawn across monitors appear more natural and undistorted. What might be good for virtual objects is unfavorable for labeling, because we risk loosing important textual information in the empty virtual space behind the bezels. In order to avoid this, labels must not overlap the bezels. Secondly, we have to account for the larger range of distances from which labels must be readable. This requires adjustment of label sizes depending on the viewing distance.

Due these requirements we need a dynamic calculation of the labeling at real-time. We employ the labeling algorithm by Luboschik et al. [240], which satisfies our needs. The algorithm allows the placement of so-called conflict particles in those parts of the labeling space where labels must not appear. We insert such particles exactly where the bezels are located. Hence, we can guarantee that labels never overlap with bezels. However, labels that are larger than the size of a single LCD-panel cannot be placed anymore. To mitigate this problem, we split labels where possible. Once the label positions have been calculated, we use scalable vector fonts for the text rendering.

Given the number of pixels of LHRDs, we are theoretically able to visualize the entire data. However, in that case, it might be difficult for viewers to recognize the multitude of graph elements. Moreover, as in our example, interesting findings might be revealed on different levels of a graph hierarchy. Therefore, it is important to enable the user to explore the data interactively.

11.3.3 *Interaction*

When visually exploring a graph hierarchy, the adjustment of the level of abstraction is an essential interaction. We realize this interaction based on head tracking. By using head tracking we obtain information about the user's head position and orientation (6 degrees of freedom) in front of the display wall. As mentioned in Section 11.2.2, care must be taken that the interaction be robust against small head motions. We implemented two alternative methods to utilize the head tracking information: (1) the zone technique and (2) the lens technique.

Both techniques allow the user to change the level of abstraction of the displayed graph hierarchy by moving in front of the LHRD. Transitions from one level to another are animated to retain the user's mental map. The zone technique corresponds to a level-wise global adjustment, whereas the lens technique realizes a local adjustment in those parts of the visualization that the user is currently looking at. This input offers a hands-free interaction method, where the hands

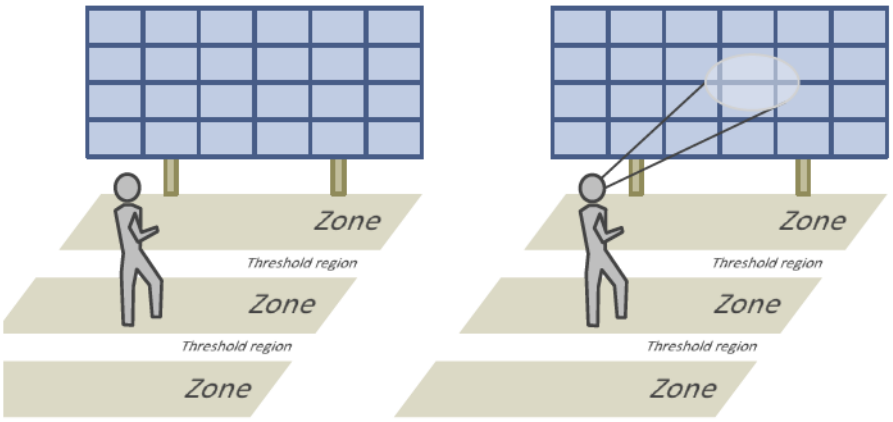


Figure 11.2: Illustration of zone technique (left) and lens technique (right).

can be used for finer motor control tasks (e.g., interactive manipulation).

THE ZONE TECHNIQUE supports intuitive level switching based on head tracking, where only the user's distance is considered. For the zone technique, the interaction space in front of the wall is divided into parallel zones, one zone for each level. The interaction zone for the root of the graph hierarchy is farthest away from the display, the zone for the deepest leaves of the graph hierarchy is closest to the display. As the user navigates physically (i.e., step forward or backward) we switch the level of the graph hierarchy. The closer the user moves toward the display the more details are displayed. In order to avoid sudden unintentional switches, additional threshold regions are inserted between two adjacent zones (see Figure 11.2, left). Thus the user has to cross the threshold region entirely to release a switching.

The advantage of the zone technique is a natural changing of the level of detail by forward and backward moves. However, the user changes the level of detail globally for the entire graph. This reveals edges within the same level, but edges across different levels remain hidden (e.g., a relation between a node and the children of another node).

THE LENS TECHNIQUE facilitates to keep the overall context, while providing detailed information for a user-selected part of the data. The selection is done by an approximation of the gaze direction. Head tracking delivers information about the head position and orientation. Consequently, we create a viewing cone from the user to the display screen to approximate the user's field of view (see Figure 11.2, right). As a visual feedback, the field of view is shown as a lens ellipse on the LHRD (see Figure 11.3).

Figure 11.3: Exploring a graph with the lens technique.

As with the zone technique, the distance determines the currently displayed level of abstraction. By steering the lens with small head movements, the user is able to scan the graph and get insight into detailed information in different parts of the data. Nodes that enter the lens are dynamically expanded to reveal the next lower level of the graph hierarchy (i.e., more detail). When a node exits the lens it is collapsed to return to the original level of detail.

To keep the amount of displayed information understandable, the lens size is adjusted to the user's distance from the LHRD. When the user steps closer to the display, the lens size decreases. This naturally matches with the smaller field of view that the user covers when standing close to the LHRD.

The lens technique offers an interesting focus+context interaction, where the overall context is preserved while the user accesses details by moving the head. However, unintentional head movements can effect frequent expand/collapse operations, which cause a kind of disturbing flicker. Therefore, we enabled the Kalman filter provided by the tracking software to reduce the natural head tremor.

Both of our interaction techniques require that the labels be adjusted depending on the user distance (and level of abstraction) in order to improve readability. When the user is close to the display we can decrease the font size to free display space, which allows the algorithm to place more labels (see Figure 11.3). In contrast, when the user is far away from the display, small labels are unreadable. In such a setting, we traverse up in the graph hierarchy and pick aggregated labels, which are rendered using a larger font face. We use the common "Arial" font in a range from 12pt to 32pt.

11.4 PROTOTYPE AND PRELIMINARY USER FEEDBACK

We implemented a prototype to study the visual exploration technique on our LHRD. Next we describe the technical details and report on preliminary user feedback.

11.4.1 *Technical Details*

We use a flat tiled LCD wall consisting of 24 DELL 2709W displays, where each tile has a resolution of 1920 × 1200 pixels, with a total resolution of 11520 × 4800 (55 million pixels). The interaction space in front of the display wall (see Figure 11.4) has an area of approximately 3.7m × 2.0m and a height of about 3.5m. The displays are connected to a cluster of six render nodes (slaves) and one additional head node (master). For the rendering, we utilize the graphics framework CGLX [2] and the font rendering library FTGL [3].

Our prototype uses an infrared tracking system (Naturalpoint OptiTrack) with 12 cameras, which are arranged semicircularly around the interaction space. The user wears a baseball cap that has attached to it reflective markers. Tracking these markers enables us to determine the position and the orientation of the user's head in the interaction space.

11.4.2 *User Feedback*

Using the aforementioned prototype, we collected preliminary user feedback. We asked computer science students (one female and seven male students) to test the application with the described interaction techniques about ten minutes. The users explored the graph (see Section 11.3.1) by walking around in the interaction space. They could switch between the zone technique and the lens technique by flipping the baseball cap during the trial. In the interview afterwards the participants were asked about readability of labels from any distance, ease-of-use, and their preferred technique.

All participants indicated that the head tracking was easy to use and that the labels were readable from any distance. The participants reported that the zone technique was easier to use because there were no unintentional level switches during the interaction. This indicates to us that the threshold regions for the zone technique are effective. Although the zone technique was easier to control, the subjects found it less interactive and they felt that there was too much information in their peripheral visual field.

The lens was named as an eye-catcher and as being more interactive than the zone technique. Providing detail information inside the lens while maintaing the context outside the lens helped users to cope with the amount of information. On the other hand, the lens

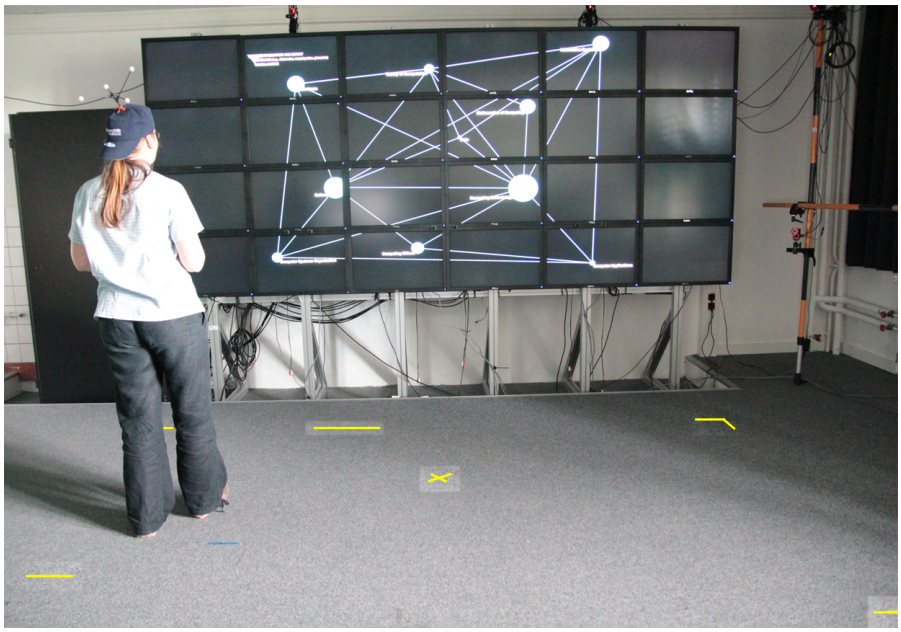


Figure 11.4: Tiled LCD wall and interaction space indicated with floor marks.

technique required a head calibration for every user to enable a reliable interaction behavior. Moreover, even after careful calibration, the users experienced unintended interaction caused by small head motions. This feedback suggests to us that we require improved filter methods to mitigate the effects of natural head tremor.

11.5 SUMMARY AND FUTURE WORK

We have introduced an approach to support the exploration of graphs on large high-resolution displays. The visualization is dynamically adjusted as the user moves in front of the display. Based on head tracking input, the labeling of the visual representation is recomputed to ensure label readability. Moreover, we implemented two interaction techniques based on physical navigation. Both the zone technique and the lens technique utilize the user's position to determine the displayed level of abstraction. Additionally, the lens technique considers an approximation of the viewing direction to provide more details for a selected part of the data while preserving the overall context. Preliminary user feedback indicates that users can easily learn and apply the developed techniques. Our prototype is capable of showing two thousand nodes and labels at interactive frame rates.

We understand our work as an initial step to be followed by further research on visualization on LHRDs. The next step is to integrate a very large data set (e.g., a metabolic pathway from systems biology) into the LHRD environment and to perform a detailed user study.

There are a number of general questions to be answered as well as specific issues to be addressed. In terms of the visualization, we have to improve our algorithmic solutions for avoiding overlap of important visual information with bezels. We already achieved quite good results for the labeling and are confident that similar solutions can be found for clusters of nodes and edges of graphs. Further investigations are necessary to find general solutions for other visualization techniques. This also has to include the enhancement of existing methods to better utilize the available pixels.

The large display space and the interaction space in front of the display are potentially useful for collaborative work. We have experimented with a setting where two persons collaborate. In such a setting, we have to consider enhanced visual feedback of the participating users and conflicts in the determination of the level of detail to be shown – globally as well as locally. However, this work is still in a very early stage. In future projects, we will investigate ways to enable users to explore graphs with physical navigation in a multi-user scenario.

More specifically, with regard to our prototype, we plan to consider further interaction tasks to be accomplished in the LHRD environment. For instance, the user should be able to freeze selected nodes for the purpose of visual comparison. We are also thinking of manipulation techniques for graphs to allow users to edit the data. It is also conceivable that visualization parameters (e.g., font size) are interactively customizable within limits. We conjecture that the so far unused hands of the user are most suitable for these finer interaction tasks.

12

EVENT-BASED CONCEPTS FOR USER-DRIVEN VISUALIZATION

CONTRIBUTION This chapter contributes a novel approach to automatically adjusting visual representations according to user interests. Event-based methods are applied in an event-reaction scheme to automate operations that otherwise would have to be carried out manually. This significantly reduces interaction costs for the benefit of the human analyst. The developed approach has an impact on all levels of interaction and the underlying formal model facilitates engineering better interactive visualization systems.

ABSTRACT Visualization has become an increasingly important tool to support exploration and analysis of the large volumes of data we are facing today. However, interests and needs of users are still not being considered sufficiently. The goal of this work is to shift the user into the focus. To that end, we apply the concept of event-based visualization that combines event-based methodology and visualization technology. Previous approaches that make use of events are mostly specific to a particular application case, and hence, can not be applied otherwise.

We introduce a novel general model of event-based visualization that comprises three fundamental stages. (1) Users are enabled to specify what their interests are. (2) During visualization, matches of these interests are sought in the data. (3) It is then possible to automatically adjust visual representations according to the detected matches. This way, it is possible to generate visual representations that better reflect what users need for their task at hand.

The model's generality allows its application in many visualization contexts. We substantiate the general model with specific data-driven events that focus on relational data so prevalent in today's visualization scenarios. We show how the developed methods and concepts can be implemented in an interactive event-based visualization framework, which includes event-enhanced visualizations for temporal and spatio-temporal data.

ORIGINAL PUBLICATION [327] — C. Tominski. Event-Based Concepts for User-Driven Visualization. *Information Visualization*, 10(1):65–81, 2011.

12.1 INTRODUCTION

Visual representations are useful tools to help people understand larger volumes of data. The visualization community offers a wealth of techniques for visually exploring and analyzing various kinds of data as, for instance, multivariate data, geo-referenced data, temporal data, or network data. Making sense of such data without the help of visualization would be a tedious task.

Although information visualization is brought to use in many contexts, there are still problems to be solved. Visualizing larger volumes of data without generating overcrowded and visually cluttered displays is one problem. Even though it has been and still is being addressed in visualization research, most of today's visualization techniques reach their limits if a certain volume of data is exceeded. A second problem concerns the applicability of visualization techniques. North et al. [263] state that management and storage of data via relational database technology is flexible. In contrast, today's visualization techniques are often tailored to a single visualization task that is to be solved for one particular data set. As such, most visualizations are rather inflexible. Lack of relevance in visual representations is a third critical problem. Amar and Stasko [25] refer to this problem as the *Worldview Gap*. Among other gaps, they describe the Worldview Gap as: "[...] the gap between what is being shown in a visual representation and what actually needs to be shown to intuitively draw representational conclusions."

These problems aggravate when considering different users who are all interested in different aspects of a data set. There is no single visual representation that suits all possible user interests and all imaginable data aspects. Therefore, visual representations have to be flexible so that they can be adjusted either interactively or automatically (by appropriate methods) to meet the users' specific needs.

The goal of this work is to alleviate the aforementioned problems by involving the users (in fact, their interests) more tightly in the visualization process. To that end, an event-based approach to visualization is pursued. Event-based approaches have proved useful in various application fields like software development, databases, or modeling and simulation. Event-related concepts have also been used in the context of visualization (see Section 12.2). However, most of them are tailored to a specific type of data and a particular visualization problem, and hence, cannot be applied in a general manner.

We address this issue by combining event methodology and visualization technology in a *general model of event-based visualization*. We present the idea of the model and its basic components in Section 12.3. The three main aspects to be investigated in detail are: (1) *event specification* (see Section 12.3.2), (2) *event detection* (see Section 12.3.3), and (3) *event representation* (see Section 12.3.4).

To substantiate our concept, visualization examples for multivariate time-series and for spatio-temporal data are discussed in Section 12.4. Appropriate interaction facilities taken for granted, the presented techniques can also react automatically to events of interest to create visual representations that are adapted to the user's task at hand.

Since the term *event* is often strongly related to time, we explicitly underline that the approach developed here is not limited to the visualization of time-dependent data. In fact, we use events in a more general manner. In anticipation of the definition of events in Section 12.3, one can think of an event as a match of an entity in a data set with an (abstract) data-centric description of user interests.

12.2 RELATED WORK

An event is a ubiquitous element of life. The term *event* is stressed in virtually every field of science with a diversity of meanings. It is beyond the scope of this work to provide an exhaustive review of different conceptions of events. We rather try to identify the core of events in general and review specific applications of events in visualization.

12.2.1 *Basic Considerations*

The different terms commonly related to events can be categorized into those that describe static characteristics (facts, state, condition) and those for dynamic aspects (event, action, process) of some world or system. Events are usually embedded into an event-action-schema – in fact, events are the basis for reactive behavior in computer systems. However, it is not always clear what the cause is in this schema and what the effect. On the one hand, events are described as occurrences identified by some specific action taken (e.g., a button was pressed) and on the other hand, events are the cause for specific actions to be taken (e.g., cancelation of a database transaction).

It is generally the case that event types and event instances are distinguished. A differentiation between these terms is also important for the research presented here. The role of both terms can be briefly described as follows:

EVENT TYPE An event type is an abstract description of conditions or circumstances under which an entity is considered to be of interest. In other words, event types describe why something is of interest. For instance, the event type "airplane take-off" can be described by the condition "altitude > 0".

EVENT INSTANCE Event instances (or short events) are actual occurrences of interest, usually described by a pair consisting of a specific event type and an entity of some observation space. Depending on the observation space, different event instances are possible. For the mentioned event type "airplane take-off", one can image an event instance in time ("airplane take-off", 6:37pm) or an event instance that considers flights ("airplane take-off", "Flight CO96").

The precise meaning of event types and event instance in the context of event-based visualization will be discussed in detail in Section 12.3.

12.2.2 *Events in Visualization*

Many different visualization techniques have been developed in recent years. The focus, however, has mostly been on visualizing data, rather than on representing events that are somehow related to the data. We will now briefly review some of the visualization approaches that are connected to or make use of event-related terminology. It can be said in advance that events are again used in manifold ways with very different meanings.

FEATURE VISUALIZATION AND EVENTS Reinders et al. [282] follow a feature-based approach combined with events to visualize large time-dependent flow data. Their approach is divided into four basic steps: feature extraction, feature tracking, event detection, and visualization. While the feature extraction and tracking steps determine the evolution of features, the event detection step aims at detecting specific events in this evolution. Reinders et al. describe events as *...any development in the evolution of a feature that is significant*. This implies that events are dependent on the application domain, i.e., significant events have to be figured out for each addressed application. For features in time-dependent flow data, several basic events are suggested (e.g., birth and death or split and merge of features). Interested readers are referred to [281] for details. Once events have been detected in the data, two visualization techniques are used – a 3D iconic representation for features and a graph visualization for events.

The event detection process in Reinders et al.'s approach lifts the visualization of flow data to a higher level of abstraction. By focusing on events, the volume of data to be visualized is reduced while significant changes in the evolution of flow data are still clearly conveyed. The concentration on significant events allows for a simple, yet expressive representation as a graph that reveals the very characteristics in the analyzed flow. However, Reinders' events are strongly tied to the features derived from time-dependent flow data. An integration of the user into the specification of events has not been considered.

EVENTS IN PROCESS VISUALIZATION Matković et al. [246] describe a system that combines information visualization with classic monitoring techniques. Multiple process variables can be monitored via enhanced virtual instruments, which are arranged on the display using a focus+context approach. Variables of interest are represented at higher levels of detail (focus) to convey more information. Other instruments show lower levels of detail (context). Besides allowing users to set the focus interactively, there is also the possibility of automatic adjustment. This is where events come into the picture. Matković et al. are concerned with undesired system states (predefined). Upon occurrence of the event that the system switches to one of these undesired states, instruments representing critical variables are automatically put into the focus to make them visually more present and to enrich them with additional information. Since this attracts the user's attention, necessary actions can be initiated more quickly.

The event specification, however, is rather limited. Neither are events described that relate to more than one variable nor are combinations of events considered. It remains unclear if and how users can actually specify their interests as events. Nonetheless, the approach presented in [246] is a salient example of how events can be successfully applied to improve flexibility of visualizations.

EVENTS IN CLINICAL VISUALIZATION Chittaro et al. [80] describe a system that is dedicated to supporting physicians in the analysis of clinical data. Well-known 3D bar charts are used as the basis for the visualization. To display multiple time series, a corresponding number of 3D bar charts are created and arranged in a rectangular fashion. A variety of interaction techniques is provided by the system to support the visual exploration of the data and to alleviate the difficulties with 3D representations. What is special about the proposed system is that not only clinical data, but also clinical events are considered. Such events can be interventions by clinic personnel or unexpected actions taken by patients. Blue signs are displayed on top of those 3D bars whose time step and treatment session are connected with an event. This way they are highlighted among the rest of the visualization and the physicians analyzing the data can easily recognize and identify unexpected situations.

Again, however, events are used in a very basic manner. Events are not differentiated by some type and are only loosely coupled with the clinical data by an indication of their occurrence time. Physicians do not have the option to specify dedicated event types for some special situations.

NETWORK VISUALIZATION WITH EVENTS In Erbacher et al.'s [114] idiom, events are network-related messages residing in log files (e.g.,

successful or rejected login attempts, different types of connections to a system, or port scans). The visualization is based on a central glyph that represents a monitored server. Events of different type are shown as lines of different appearance emanating from the server glyph to smaller glyphs representing remote hosts. Whereas all drawings are usually done with a shade of gray, unexpected or suspicious activity (i.e., particular messages in log files) results in a change of color – red for critical activity, yellow for suspicious activity. Connections that have been identified (by an external system) as intrusions are represented by lines and glyphs of even brighter red. The authors argue that presenting colored (red or yellow) lines among gray lines attracts the attention of administrators to suspicious activity and actions to counter network attacks can be taken more quickly.

However, the considered events are restricted to a mixture of events defined by the authors and external events as generated by an intrusion detection system. Although users can visually recognize suspicious behavior, they cannot specify their insights as new events for later automatic highlighting.

Xiao et al. [382] go one step further and allow users to interactively specify patterns of interest. The basic idea is to apply brushing in a scatter plot diagram and to derive predicate logic formulas from the brushed data items. To that end, pre-defined network-related predicates are combined using logic operators. Xiao et al. augment the data space with knowledge captured as patterns (i.e., each data item is annotated with its matching patterns) and make use of this knowledge by changing colors in scatter plots, by adapting the level of detail, and by supporting enhanced filtering. This leads to more effective visualization of network traffic.

Even though the predicates used are specific to the network domain and despite the fact that the visual representations are quite simple, Xiao et al. see potential that their approach is generalizable to other visualization contexts.

FURTHER APPROACHES More visualization approaches exist that make use of events. To give an indication of their variety, we briefly present a non-exhaustive set of examples in the following.

Events often relate to a spatio-temporal context. In the domain of geo-spatial visualization, [134] utilize a technique known as space-time cube to represent spatio-temporal event data. Text and document visualization are also related to events. Approaches exists that cluster and visualize events that are given in some text [379]. Other approaches concentrate on the detection of events from streaming texts as a preprocess prior to visualization [391]. Software visualization and algorithm animation is known to make use of events. A comparison of data-driven and event-driven software visualization approaches can be found in [94]. Again, the "[...] precise definition

of what constitutes an 'interesting event' varies between authors." [287]. Debugging of distributed software systems benefits from visual approaches that make use of events [208]. In visual debugging, events denote "[...] actions without duration that take place at specific points in time and change the state of a process." [216]. In general, the analysis of dynamic systems can gain from event-based methods. A coupling of active databases and VRML visualization is suggested in [231] to achieve automatic updates of represented data.

DISCUSSION Several event-related visualization approaches are known in the literature. The diversity of application scenarios suggests that a combination of event-based concepts and visualization technology is useful. However, only few approaches (e.g., Reinders et al. [282]) use events as an integral component to improve visualization. Others (e.g., Matković et al. [246]) understand events more or less as add-ons. In some cases, it even seems that the potential of integrating events into the visualization is underestimated.

What caught our attention was the fact that users are only rarely involved in the event specification. Usually, visualization designers determine what should be of interest and users are restricted to only viewing the visual representations. This limits not only the expressiveness of a visualization approach, but also its applicability to different or even similar visualization tasks.

As a matter of fact, users know their interests best. Therefore, they should participate more closely in the creation of visual representations. User-definable events are one possibility to address this shortcoming (e.g., Xiao et al. [382]). Indeed, users need to be supported in specifying their interests as events and surely user-defined events will not be as complex as those specified by visualization designers, but nonetheless, a higher degree of user involvement bears great potential.

Additionally, event-based visualization can help in coping with larger volumes of data. Usually, visual representations of large data sets are cluttered and overcrowded, and thus, difficult to understand. Besides using approaches like overview+detail and focus+context, large data sets can be dealt with by visualizing only relevant parts of the data. Extracting interesting events is one possibility to concentrate on relevant things. Among the reviewed event-related approaches, Reinders et al. [282] and Xiao et al. [382] actually reduce the amount of information to be represented. However, their descriptions of events are specific to features in flow data and to network traffic data, respectively, and hence are not generally applicable.

This potential of getting closer towards user-adapted visualization and the chance to better deal with larger data sets motivated us to extract a generic theory and to develop a general model of event-based visualization. The model subsumes previous work in the field,

but abstracts from domain-specific details. Hence, our model is not restricted to a specific application (as it is the case for all of the reviewed approaches), but can be applied and adapted in different visualization scenarios. A particular focus will be on the event definition by the user.

12.3 EVENT-BASED VISUALIZATION

We will now introduce the general model of event-based visualization. First, an overview of the model will be given. The main aspects – event specification, event detection, and event representation – will be discussed in detail in separate sections.

12.3.1 *Overview and Formal Model*

The basic idea of event-based visualization comprises three steps. (1) Let users specify their interests as event types, (2) determine if and where these interests match with the data (i.e., detect event instances), and (3) consider detected event instances when generating the visual representation.

Let's clarify this general procedure by an example. We assume a physician who visualizes data that contain the daily numbers of cases of different diseases. Increases and decreases in the data can be discerned from a suitable visualization. However, there is a special condition that is of interest to the physician: If the number of new cases of influenza increases for three successive days and the absolute number of cases of influenza exceeds 300, then this might be an indication of a possible wave of influenza. In an event-based visualization session, the physician would do the following: He specifies the condition of interest as an event type. The system tries to find actual instances of that event type. If no events are detected, the visualization is presented as usual. But if an event is detected, the visual representation is adjusted automatically to highlight the special situation. Charts are enhanced with markers or color scales are altered to emphasize the data record for which the event has been detected. The physician can quickly and easily recognize the importance of the current health situation, which enables him to take steps to prevent a possible epidemic of influenza.

Figure 12.1 contrasts the classic visualization approach with the event-based visualization approach. When using the classic approach, all users are provided with the same visual representation, although they are interested in different aspects of the data (indicated by different colors). By following the event-based approach, it is possible to generate individually adjusted visualizations. The integration of the user into the process of creating visual representations helps increase the relevance in the resulting images.

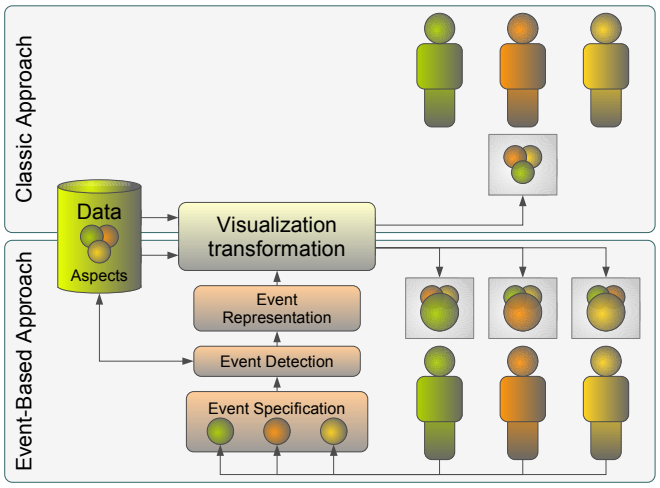


Figure 12.1: Comparison of classic and event-based visualization – The figure depicts users who are interested in different aspects of the data (denoted by different colors). Using the classic approach results in a standard representation. The event-based approach integrates users more tightly into the visualization process. As a consequence, the relevance of the generated representation increases.

To make our model truly general and flexibly applicable we put it on a formal basis. First, it is necessary to define event types and event instances as well as their frame of reference on an abstract level. Then, a visualization transformation can be defined that considers event instances so as to achieve event-based visualization.

To describe the frame of reference in which events occur, the notion of event domains is introduced. An *event domain*

$$\mathrm{ED}$$

contains entities with respect to which event types can be specified. Such entities can be, for instance, tuples or attributes of a relational data set, or simply time points. We define that event instances occur with regard to entities of an event domain.

A distinction between event types and event instances is generally required in event-based systems. In our case, an *event type*

$$et \in \mathrm{ET}$$

is used to express a concrete interest with regard to entities of an event domain. Here, ET denotes the set of all possible event types. To differentiate between event types that address different event domains, the notion of abstract event types is introduced. The *abstract event type* for an event domain ED is defined as

$$\widehat{\mathrm{ED}} \subseteq \mathrm{ET}$$

where $\widehat{\mathrm{ED}}$ contains only those event types of ET that can be evaluated with respect to the elements of ED. In other words, an abstract

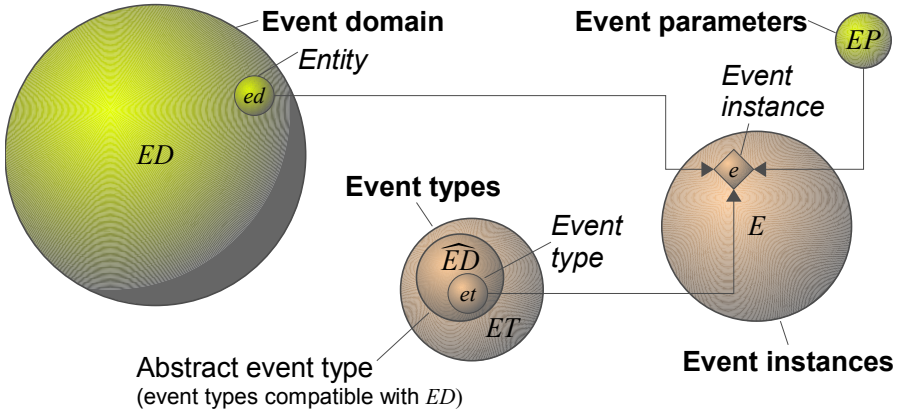


Figure 12.2: Illustration of the notions event domain, event type, and event instance.

event type is a notation to associate an event domain with a set of compatible event types.

Actual event instances can be defined as follows. Assumed $et \in \widehat{ED}$ is an event type in some abstract event type $\widehat{ED}$ and an entity $ed \in ED$ conforms to the interest expressed as et, then the triple

$$e = (ed, et, EP) \in E$$

denotes an *event instance* (or short *event*) of type et. The set of all possible event instances is denoted by E. The set EP denotes *event parameters* that can be assigned to an event instance. Event instances establish a connection between interests (i.e., the event type) and concrete entities of some event domain. The introduced terms and their relation are illustrated in Figure 12.2.

Event-based visualization can now be defined formally as follows. The detection of event instances of some event types in a data set can be modeled as a mapping from $\mathfrak{P}(ET)$ and D (the set of data sets) to $\mathfrak{P}(E)$ ($\mathfrak{P}$ denotes powersets)

$$\texttt{detect} : \mathfrak{P}(ET) \times D \to \mathfrak{P}(E).$$

Visualization in general can be modeled as a mapping from D (the set of data sets) and P (the set of visualization parameters) to V (the set of visual representations)

$$\texttt{vis} : D \times P \to V.$$

For event-based visualization, the mapping `vis` is extended so as to take the event instances detected in the data into account

$$\texttt{evis} : D \times P \times \mathfrak{P}(E) \to V.$$

This formal description is the basis for our general model of event-based visualization, which consists of the three steps *event specification*, *event detection*, and *event representation*. In order to combine these

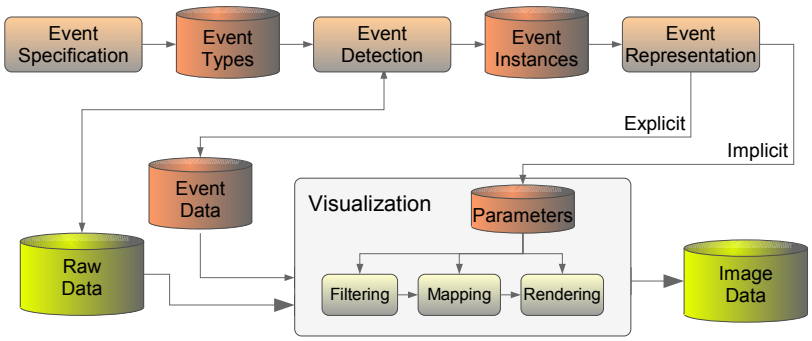


Figure 12.3: General model of event-based visualization – The major components of event-based visualization – event specification, event detection, and event representation – are attached to the classic visualization pipeline.

event-based concepts with visualization technology, we attached them to the classic visualization pipeline with the stages data analysis, filtering, mapping, and rendering. Since most visualization approaches follow this pipeline, virtually any visualization technique can be easily enhanced by the advantages of events. Figure 12.3 illustrates the extended visualization pipeline. Next, we will discuss the steps of our model in detail.

12.3.2 *Event Specification*

The task of the event specification is to compile event types that are or might be of interest to visualization users. To translate informal user interests to the digital language of computers, a formalism is required that provides suitable expressiveness while still allowing users to declare their interests as easily as possible. Furthermore, intuitive tools must be developed to allow users with different experience to specify their interests as event types. We suggest adapting formulas known from predicate logic and building upon them a model for user-centered event specification.

In our approach, event types are defined via so-called event formulas. These formulas make use of elements of predicate logic(PL), including variables, predicates, functions, aggregate functions, logical operators, and quantifiers to encapsulate the particular conditions that users are interested in. Different variants of event formulas can be used to address different event domains. In our work, we concentrate on relational data sets, and therefore, support the following abstract event types: tuple event types $\langle\rangle_T$ for expressing interest with respect to tuples of a data set (e.g., values within a tuple exceed certain thresholds), attribute event types $\langle\rangle_A$ for a data set's attributes (e.g., the attribute with the highest value), and sequence event types $\langle\rangle_S$ for sequences of tuples (e.g., sequence of days with rising stocks).

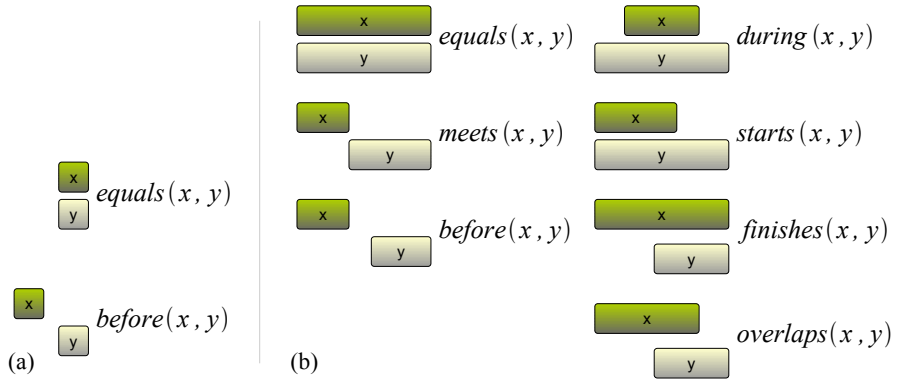


Figure 12.4: Temporal predicates – (a) Predicates for time points; (b) Predicates for time intervals (inverse predicates are possible).

For sequence event types, event formulas are extended by sequence-related notations (inspired by Sadri [288]). A combination of event types to composite event types is also possible. They are realized via set operators.

Predicates are crucial for event specification. Apparently, the expressiveness of the available predicates determines the overall expressiveness of event types. Though basic predicates (e.g., check for equality or comparison of values) are sufficient for defining a variety of simple event types, it makes sense to provide more sophisticated predicates that address the specific semantics of different kinds of data. Next, we illustrate dedicated temporal and spatial predicates, which can be used to ease event definition.

Temporal predicates describe relations between time points or between time intervals [149]. The temporal predicates `equals` and `before`, as well as their inverse predicates `not-equal` and `after` are used for time points. Allen [22] identifies the following predicates for time intervals: `equals`, `before`, `meets`, `overlaps`, `during`, `starts`, and `finishes`. Corresponding inverse predicates do also exist. An overview of temporal predicates is depicted in Figure 12.4.

A comprehensive collection of spatial predicates is presented in [279]. These predicates allow for various statements with regard to spatial objects (e.g., Are two regions externally connected, are they internally connected, or are they disconnected?) Spatial predicates are presented in Figure 12.5. Detailed axiomatizations and further references can be found in [279] and [115].

Temporal and spatial predicates enable users to specify manifold event types with respect to time-oriented and geo-referenced data. These and further advanced predicates can encapsulate complex semantics, and hence, can reduce the efforts for the event specification.

To illustrate the introduced event specification formalism, we present some examples of event types with regard to relational climate data that contain among others the attributes date, region, average temperature (avgtemp), and precipitation (prec). A first, very

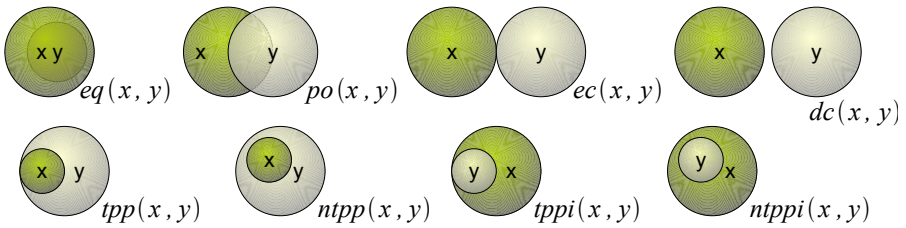


Figure 12.5: Spatial predicates – The figure shows a pairwise disjoint set of base predicates for spatial regions [279]. The predicates are `eq` (equals), `po` (partially overlaps), `ec` (is externally connected with), `dc` (is disconnected from), `tpp` (is tangential proper part of), `ntpp` (is non-tangential proper part of), `tppi` (inverse of `tpp`), and `ntppi` (inverse of `ntpp`).

simple event type detects a tuple with a specific value for an attribute. In particular, the event type

$$\langle x \mid x.\mathtt{prec} = 0 \rangle_T$$

can be used to detect tuples that describe a day with no precipitation, where x is a tuple variable and is also called the *target* of the event formula. Event types that describe the exceeding of a threshold are commonly required. An example of an event that will be detected whenever the average temperature drops below 0 °C could be

$$\langle x \mid x.\mathtt{avgtemp} \leqslant 0 \rangle_T.$$

Tuple event types that relate the target tuple variable to other tuple variables can be used to state more complex conditions. For instance, the hottest day in the data set can be described by the event type

$$\langle x \mid \forall_T y : x.\mathtt{avgtemp} \geqslant y.\mathtt{avgtemp} \rangle_T,$$

relating the target variable (x) to all other tuples ($\forall_T y$) in the data set. Attribute event types are useful when aggregate functions over attributes are considered. In this example, we use the aggregate function `dev`, which can be used to calculate the standard deviation of an attribute. Then, the attribute with the largest fluctuation can be detected using the following event type

$$\langle a \mid \forall_A b : \mathtt{dev}(a) \geqslant \mathtt{dev}(b) \rangle_A.$$

The temporal predicate **equals** (see Figure 12.4) and the spatial predicate **ec** (see Figure 12.5) can be used to detect regions for which all neighbors show lower average temperatures.

$$\begin{aligned} \langle x \mid \forall_T y : (&y.\mathtt{date}\ \textbf{equals}\ x.\mathtt{date} \wedge \\ &y.\mathtt{region}\ \textbf{ec}\ x.\mathtt{region}) \rightarrow \\ &y.\mathtt{avgtemp} < x.\mathtt{avgtemp} \rangle_T \end{aligned}$$

In the next example (inspired by [289]), we define a sequence event type by using the $*$ notation and the keyword previous. Here, arbitrary repetition of a variable y is denoted by $*y$ and access to the sequence element that precedes a variable y is denoted by y.previous. Our goal is to declare our interest in five successive days with increasing temperature and an overall increase in temperature of more than 50%.

$$\langle (x, *y, z)^{region}_{date} \mid$$
$$y.avgtemp > y.\text{previous}.avgtemp \wedge$$
$$1,5 \cdot x.avgtemp < z.\text{previous}.avgtemp \wedge$$
$$x.date + 5 \textbf{ equals } z.\text{previous}.date \rangle_S$$

These examples show that different interests can be expressed as event types. However, the reader can also easily see that relying solely on textual event specification is likely to stress users. We have to offer dedicated support, because otherwise the whole approach becomes questionable. Therefore, we developed a user-centered model of event specification and a visual event editor. Our goal was to assist users in specifying event types, in altering already specified event types, and in selecting event types currently relevant for a particular visualization session. To that end, it is necessary to take the expertise of users into account: Novice visualization users must be provided with tools different from those provided for expert data analysts. Our three-level event specification model aims to fulfill these requirements. It consists of the three stages (1) direct specification, (2) specification by parameterization, and (3) specification by selection (see Figure 12.6).

This model uses the proposed event formulas as its basis. The complete functionality of event formulas is made available to expert users

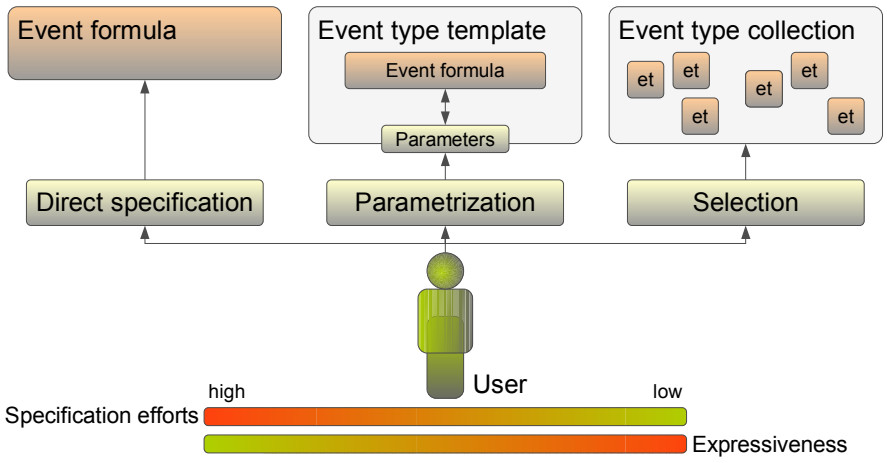


Figure 12.6: Model for user-centered event specification – The figure illustrates that users can specify their interests by direct definition of event formulas, by parameterization of event type templates, or by selection from an event type collection.

at the first level of the model only. By allowing for direct specification of event formulas experts can specify any event type they are interested in. It must be mentioned that direct specification does not mean that event types can only be entered by typing them on the keyboard. Quite the opposite, it is important to support even expert users in the direct specification of events. This support can be achieved by visual approaches as we will see later in this section.

We suggest providing predefined event type templates that can be parameterized effortlessly by non-expert users. Event type templates use event formulas internally, but allow access to them only via easy-to-set parameters. Since the complexity of event formulas is hidden from the user, the event specification is significantly simplified by templates. An example for an event type template is a threshold template with parameters for the considered attribute and the particular threshold. Users need no deep knowledge of the event type formalisms, but instead can simply select an attribute and set a concrete threshold to specify their interest.

The third level of event specification is based on simple selection. The idea is to provide a collection of predefined event types from which users can select the most interesting ones for their task at hand. Evidently, the collection of event types to choose from has to be compiled by domain experts or visualization designers. To facilitate easy selection of relevant event types, it is necessary to provide expressive identifiers and an expressive verbal descriptions of the conditions intended to be detected by an event type. This third level of event specification addresses not only novice users, but also users who are in the position of a decision maker, or any other user who seeks quick access to relevant information.

Figure 12.6 illustrates the three levels of the user-centered event specification model. It can be seen that the effort required is high for direct specification and low for specification by selection; the effort required for parameterizing event type templates is in between, depending on the parameters that need to be set. Obviously, the model implies a tradeoff between expressiveness and the level of user expertise needed for event specification. According to Tang et al. [320], this is generally the case whenever formal means must be handled by human operators.

As already indicated, relying on textual input of event formulas alone will not suffice to achieve user acceptance. Further support on the level of direct specification is needed. We investigated how the specification of event types can be supported via visual methods. The goal is to use visual abstractions not only to show data, but also to aid in the interactive specification of events. Basically, three visual approaches are suitable for visual event specification: *brushing*, *dynamic queries*, and *visual editors*.

Brushing is usually seen as the process of interactively selecting data items and highlighting them in all views of the visualization [76]. Brushing is intuitive as it is performed directly on the visualization. In addition to common brushing operations like elastic rectangles and lassos, several dedicated brushing techniques are known in the literature. Doleisch et al. [99] describe how formal feature descriptions can be specified interactively by brushing. Another example are Hochheiser's [169] Timeboxes, which can be used to query time series or other linear sequences. Timeboxes have later been revised by Buono et al. [68]. Xiao et al. [382] apply brushing for interactive pattern specification. Especially the last example gives evidence that brushing can support interactive event type specification. Depending on the particular brushing semantics, a variety of conditions can be captured easily.

Dynamic queries are similar to brushing in that they are used to select data items of interest [304]. In contrast to brushing, however, dynamic querying is usually performed via dedicated user interface elements, and selected data items are not highlighted, but unselected items are filtered out. Nonetheless, the net effect is the same: data of interest stand out. The user interface is critical with regard to expressiveness and effectiveness of the querying process. An example of a query interface that supports boolean combinations of query conditions is presented in [332]. The similarity between the formalism used there and the introduced event types suggests that dynamic query interfaces can be beneficial for event specification.

However, relying on brushing or dynamic queries alone is not sufficient because they are limited to the range of values already contained in the data. By using either method, it is impossible to specify conditions with respect to values that are not contained in the data, but might appear in later instances of a data set. As this limits the expressiveness of event types significantly, further visual methods must be offered.

Visual editors (or visual languages) can help in this regard. Such editors aim to exploit the capabilities of the human visual system for the specification of complex, formally defined structures [63]. Many applications benefit from visual editors (e.g., Lee et al. [225], Balkir et al. [42], or Chittaro and Combi [79]). The advantage of using visual editors for event specification is that any event type can be specified regardless of whether the particular condition is already present in the data (as it is required when using brushing) or not. To make use of this advantage, we implemented an extensible editor framework for event types. Currently, the framework supports event types that address tuples of relational data sets. The visual editor provides a graphical abstraction of the underlying formalism and makes it easier for users to describe their interests as event types. To illustrate the editor's use, three examples of tuple event types are depicted

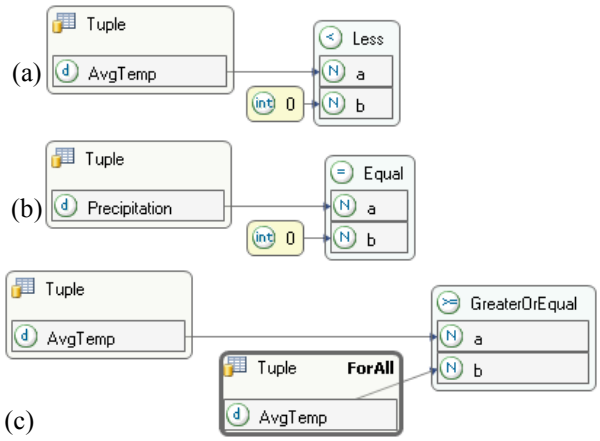


Figure 12.7: Examples of visually specified event types – (a) $\langle x \mid x.prec = 0\rangle_T$; (b) $\langle x \mid x.avgtemp \leqslant 0\rangle_T$; (c) $\langle x \mid \forall_T y : x.avgtemp \geqslant y.avgtemp\rangle_T$ (maximum average temperature).

in Figure 12.7. Figure (a) and (b) show how predicates ("Less" and "Equal") are used to connect tuple variables and constants, while (c) illustrates the connection between a free tuple variable and a tuple variable bound to a "ForAll" quantifier.

12.3.3 *Event Detection*

Once users have specified their interests as event types, the next step is to determine if and where these interests match with the data. While the event specification has been designed so as to be as general as possible to allow for a great variety of event types, we certainly require dedicated methods for the event detection.

The basic task is to evaluate the conditions encapsulated as event types. To that end, the variables used in event formulas are substituted with concrete entities of the data (tuples, attributes, or sequences of tuples). In a second step, predicates, functions, and logical connections are evaluated, so that the event formula as a whole can be evaluated to either true or false. Conducting the event detection results in a set of event instances $E' \subseteq E$, which mark those entities of an event domain that comply with user interest.

Since the detection procedure is very costly in terms of computation time, we make use of sophisticated methods to find event instances. Tuple events can be detected efficiently with the help of relational database management systems (RDBMS). Such systems employ optimization methods when evaluating SQL queries to the data. Since the SQL and our event formulas share a common formal basis – predicate logic – it is possible to map tuple event types to SQL query statements. If event types make use of special predicates, unavailability of such predicates in RDBMS can be a limiting factor. With the help of stored procedures (i.e., small custom-made pieces of software that re-

side within the RDBMS), this limitation can be alleviated. The following example illustrates the mapping of a simple event type to an SQL statement. Assume $\langle x \mid x.avgtemp \leqslant 0 \rangle_T$ is a tuple event type that detects frosty days in a climate-related data table `climate`. Executing the SQL statement `SELECT * FROM climate WHERE avgtemp < 0` will return all the tuples that comply with the event type. By delegating such SQL statements to an RDBMS, event detection efficiency can be increased significantly. Attribute event types cannot be mapped to SQL statements in this one-to-one fashion. But still, the evaluation of aggregate functions in attribute event types can be accelerated. Sequence event types can not yet be optimized by database management systems. Nonetheless, one can make use of the OPS algorithm, which has proved to be efficient for querying sequenced data [288]. Finding instances of event types that are less general than tuple, attribute, or sequence event types will certainly require development of dedicated detection methods.

If dynamic data (i.e., data that change over time) are considered, efficiency becomes even more crucial because events must be detected on every change of the data. To further improve efficiency, we suggest investigating incremental detection in future work. Such methods operate on a differential data set, rather than on the whole data. However, incremental methods are still a current topic in database research and as such are not yet readily available. Moreover, relying on incremental detection might impose restrictions on what event types are possible.

The event instances found during event detection can now be handed over to the event representation step.

12.3.4 *Event Representation*

The last important step of event-based visualization is the event representation. The goal of this step is to incorporate detected event instances (which reflect the interests of users) into visual representations. The event representation has major impact on the extend to which event-based visualization can bridge the Worldview Gap. For an expressive event representation we (1) have to *communicate the fact* that something interesting has been found, (2) must *emphasize events* among the rest of the data, and (3) should *convey what* makes the events interesting. The most important requirement is that the visual representation must show that something interesting has been found in the data. To meet this requirement, easy to perceive visual cues (e.g., a red frame around the visual representation, exclamation marks, or annotations) are used. Alpha blending can be applied to fade out past events. The second requirement aims at emphasizing those parts of the visual representation that reflect the interests of users. Additionally, the visualization should communicate

what makes the highlighted parts interesting (i.e., what is the particular event type). However, facing arbitrarily definable event formulas, this last requirement is difficult to accomplish.

We suggest two different concepts for representing event instances – *implicit event representation* and *explicit event representation*. Implicit event representation aims at automatically adjusting given visualization techniques. The major challenge in representing events implicitly is to adequately alter visualization parameters so as to achieve the previously stated goals. On the other hand, explicit event representations focus on visualizing events, rather than the underlying data. Since detaching events from data implies a high degree of abstraction, explicit event representation is a promising alternative for gaining insight into large data sets.

An interesting question we had to face was who should be in charge of the event representation. While the human is clearly in charge of specifying event types and the machine is obviously responsible for detecting event instances, the event representation cannot be assigned so easily. Quite contrary, it seems to be the best solution to make the event representation a joint effort of both human and machine.

IMPLICIT EVENT REPRESENTATION By implicit event representation, the large number of available visualization approaches can be enhanced with event-based concepts without the need to develop entirely new visualization techniques. A requirement for incorporating events successfully into known visualization techniques is to find suitable parameters of the visualization transformation. Apparently, the available parameters and their degrees of freedom determine the expressiveness of implicit event representation.

If a visualization technique provides adequate visualization parameters, adjustments can be implemented either as *instantaneous* or *gradual* changes. Instantaneous changes of visualization parameters have immediate effect on the visualization. A simple example is setting a color parameter from red to blue. Gradual changes, on the other hand, consume time during the adjustment. An example is the movement of the focal point in a focus+context visualization toward an interesting data portion that is associated with an event. The distinction between instantaneous and gradual parameter changes raises the question which possibility to use in particular situations. In other words, does it make sense to switch a parameter immediately to achieve an emphasis on events, or is it better to apply a gradual change? A definite answer to this question cannot be given due to the complexity of human perception and cognition and because the answer depends on size and complexity of the considered data. Usually a trade-off must be found between instantaneous changes, which are more suited for

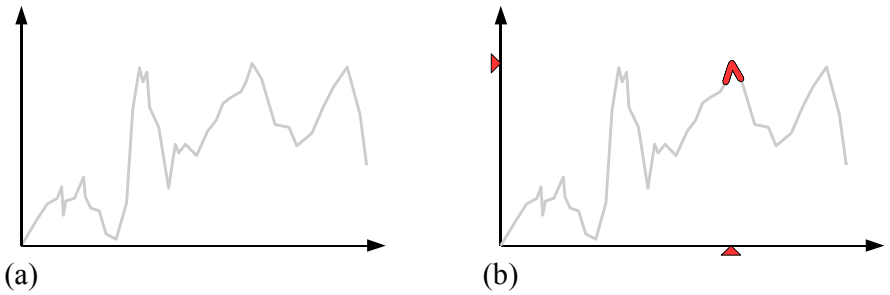


Figure 12.8: Local changes in a diagram plot – (a) The original diagram; (b) Markers and adaptation of color and line thickness introduce local changes to the diagram.

achieving preattentive effects, and gradual changes, which are more suited for maintaining the users' mental map.

Irrespective of whether instantaneous or gradual, adjustments of visualization parameters result in changes in the visual representation. Such visual changes have either a *local* or a *global* scope. An example for a change with local scope is changing the color and width of those spots in a diagram plot that represent event instances (see Figure 12.8). When speaking of global changes, we mean that the visual representation is affected as a whole, that is, it is perceived in an entirely different way. An example is to move the focal point of a globally effective fisheye distortion (see Figure 12.9). Again, the distinction between changes with local and global scope makes it necessary to discuss the question when to use which. Local changes, on the one hand, are limited in their spatial extent on the screen, and hence, enable the representation of multiple event instances in a single view. Furthermore, the locality of the changes leaves space for conveying the type of event instances. On the other hand, local

(a) (b)

Figure 12.9: Global change of a Magic Eye View visualization – (a) The original Magic Eye View; (b) Adjustment of the focal point results in a global change.

changes, especially many of them, might be hard to grasp, because local changes compete with each other for the users' attention. One also has to be careful with regard to the question whether a local change is due to an event or due to variance in the visualized data. Global changes, on the other hand, do not compete for the users' attention. Since global changes affect the whole display, rather than just portions of the screen, they are useful for conveying events that abstract from concrete data items (e.g., composite events). However, the global scope of the changes also implies that only a few events can be incorporated into the visualization simultaneously.

From the previous discussion it becomes clear that finding expressive implicit event representations is a challenging task. Particularly the diversity of application scenarios, which require different ways of considering events in the visualization, makes this task demanding. As indicated, perceptual issues play an important role. For more information and references on the cognitive aspects touched in the previous discussion, the interested reader is referred to [341, 156, 389, 46, 251, 309, 344].

EXPLICIT EVENT REPRESENTATION In the previous section we explained how events can be visualized implicitly by merging their representation into known visualization techniques. In contrast, explicit event representation aims for visualizing the event instances detected in the data, rather than the data. That is, the detected events are the only input to the visualization transformation. According to Reinders et al. [282], focusing on relevant events raises the level of abstraction of the data analysis. Although visualizing only events is contradictory to Tufte's [343] claim "Above all else show the data!", providing a more abstract event representation (certainly in addition to a fine-grained data visualization) is useful for achieving more complex visualization tasks like finding correlations between different interesting aspects of the data (described as event types). It can be stated that the major goal of representing events explicitly is to facilitate higher order visual analysis. Secondly, it is definitely important to mention that commonly the number of detected event instances is much smaller than the number of data items $|E'| \ll |D'|$, where $E' \subseteq E$ and $D' \subseteq D$. This fact can be exploited to alleviate the visualization of very large data sets.

For explicit event representation, we suggest transforming the event instances E' detected in some data set D' into a new data set $D_{E'}$ that in turn is subject to visual analysis. Since the event type is a crucial characteristic, any implementation of such a transformation should consider the event type as an attribute of the new data set $D_{E'}$. Further event parameters like event importance as well as selected characteristics of the entity to which an event is bound should be mapped to attributes in $D_{E'}$. The advantage of this procedure is

that a broad range of available visualization techniques can be used to represent the event instances E' (now mapped to the new data set $D_{E'}$) explicitly. Since the event type is a categorical attribute, it is a major requirement that the visualization techniques be capable of efficiently representing categorical data. Since it might be necessary to visualize more than one event parameter, potential visualization techniques should also be able to handle multivariate data.

Depending on the application scenario, different ways of representing events explicitly are possible. Abstract representation of events as graph (like in [282]) are imaginable. To concentrate on temporal aspects of events many dedicated visualization techniques for time-oriented data are available [20]. In the next section we will also see how event instances can be represented in time and space.

12.4 VISUALIZATION EXAMPLES

We implemented a framework called *eVis*, which serves as a research testbed for event-enhanced visual representations. It is implemented in C# using .NET and DirectX. The framework's architecture is depicted in Figure 12.10. The core of the framework implements the general model of event-based visualization. On the data side, *eVis* provides interfaces to relational data sets either stored in files or in RDBMS. Event types enter the system in form of structured XML files, which can be generated either by hand or via a visual editor. For the event detection, we rely on RDBMS functionality as long as the data

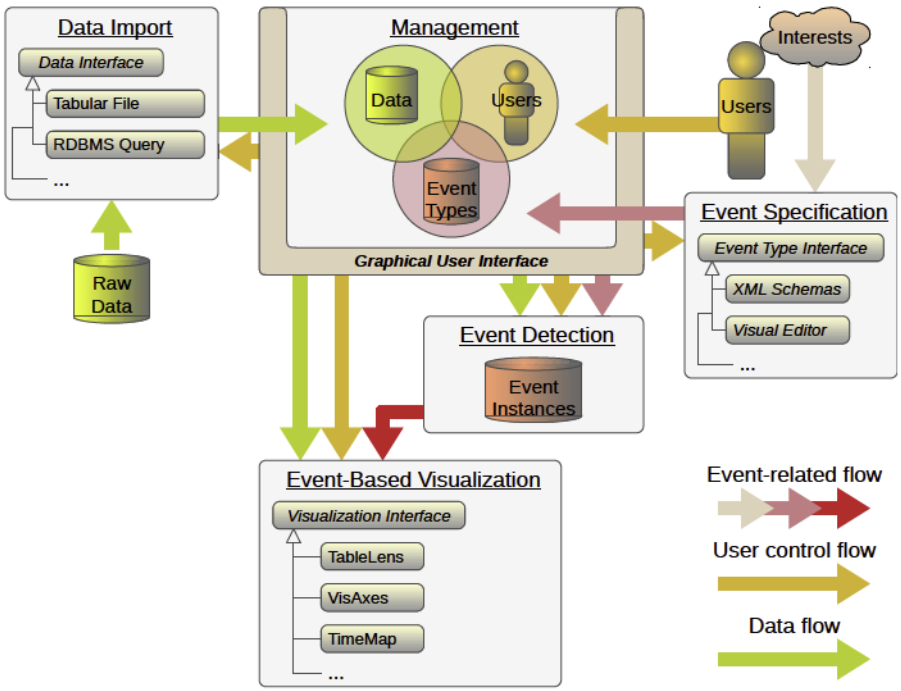
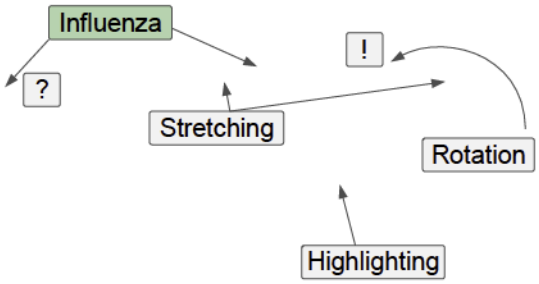


Figure 12.10: Architecture of the framework *eVis*.

is stored in a database. Data residing in files is scanned for events using a plain non-optimized implementation of the event detection formalism. The framework integrates several visualization techniques that follow the model of a parameterized visualization pipeline. On the one hand, this allows for interactive visual data exploration, and on the other hand, all techniques can be subject to automatic adjustments through implicit event representation. Techniques like scatter plots, parallel coordinates, or the Table Lens can be used for abstract multivariate data. Temporal data can be explored using dedicated techniques like stacked charts or the TimeWheel [329]. In addition to that, we implemented an explicit event representation technique that visualizes events in time and space.

We will now illustrate four visual examples generated with *eVis*. Three examples use implicit event representation for the analysis of a multivariate time-oriented data set, the fourth example will demonstrate the explicit representation of event detected in geo-referenced data.

We assume a user who has to search time-dependent human health data (1100 data rows and 11 attributes) for uncommonly high numbers of cases of influenza. The task at hand is to detect where in time these situations have occurred. A possible way to accomplish this task is to use the TimeWheel technique [329], which is a multi-axes representation for visualizing multivariate data over time. The TimeWheel shows a time axis in the center of the display. Multiple axes that encode time-dependent attributes are arranged around the central time axis. For each time point in the data, lines descend from the time axis to the corresponding points on each of the attribute axes. The TimeWheel can be rotated to bring different attributes into the focus. Furthermore, each axis can be equipped with a slider to zoom into value ranges of interest. Without event integration the user in our example will be provided with a TimeWheel that uses a standard parameterization (see Figure 12.11 (a)). The standard view shows influenza on the upper right axis (light green), time is represented on the central axis. Alpha-blending has been applied by default to reduce visual clutter. From the TimeWheel in Figure 12.11 (a) one can only guess from the labels of the axis showing influenza that higher values are present; the alpha-blending made the particular lines almost invisible (see question mark). Several interaction steps are necessary to re-parameterize the TimeWheel to accomplish the task at hand. In *eVis*, the user can specify the interest "Find days with a high number of cases of influenza." as the event type $\langle x \mid x.flu \geqslant 300 \rangle_T$ to be considered for the current analysis task. The event type can be stored and may be reused in later visualization sessions or by other users. The event detection will now determine whether or not the data conform to the interest expressed by the user and will create event instances for those data entities that do so. To provide the analyst with an individ-



(a) (b)

Figure 12.11: Implicit event representation for the TimeWheel – (a) TimeWheel representing time-dependent health data; the interests of the user are not considered in this standard parameterization, which aims at showing main trends. (b) TimeWheel representing the same data; the user's interests were recognized in the data and have been emphasized via highlighted lines and automatic rotation; the presentation is better targeted for the user's task at hand.

ually adjusted TimeWheel that reflects his interests, the parameters of the visual representation have to be altered. Parameter changes can be implemented either as instantaneous actions or gradual processes (e.g., smooth animation). In this particular example, we use an action that switches color and transparency of those line segments that represent event instances. Days with high numbers of influenza cases are excluded from alpha-blending and are drawn in white color. Additionally, the TimeWheel is rotated such that the axis representing influenza is moved gradually to an exposed position. A focus+context stretching of the axes is applied to provide the influenza axis (and the axis opposite to it) with more screen space than the other axes. The application of a gradual process is important in this case to help users maintain their mental map of the visual representation. The result of applying parameter changes as response to event instances is depicted in Figure 12.11 (b). This figure illustrates that event-based visualization is capable of focussing on user interests by adapting the visual representation to the current visualization task. In the automatically adjusted TimeWheel, identification of days with higher numbers of influenza infections is easier than in the default view.

In order to see how the high numbers of influenza cases are correlated with other diagnoses, the user can use a scatter plot visualization as presented in Figure 12.12. Again, the *eVis* framework allows us to highlight those parts of the data where event instances occur. This makes it easier for the user to pick up relevant information.

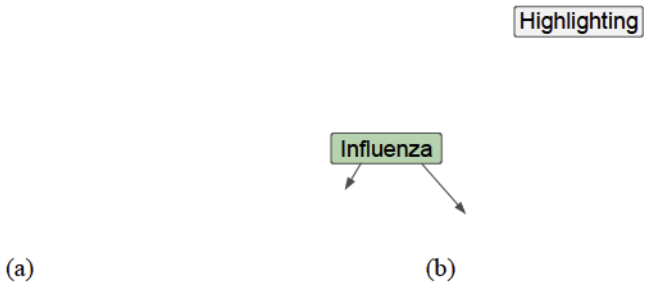


Figure 12.12: Implicit event representation for scatter plots – (a) Scatter plot representing selected diagnoses of the health data; (b) Relevant data portions have been automatically highlighted.

As alternative view to assess correlations, *eVis* provides a Table Lens [280]. The focus point of the Table Lens is a prominent parameter to be adjusted to accomplish implicit event representation. In our case, we implemented an adjustment of the focus parameter according to the most recently detected tuple event in the table. Despite the simplicity of this adjustment, it achieves an automatic focusing on relevant information. Figure 12.13 illustrates the difference between considering and not considering events in a Table Lens representation.

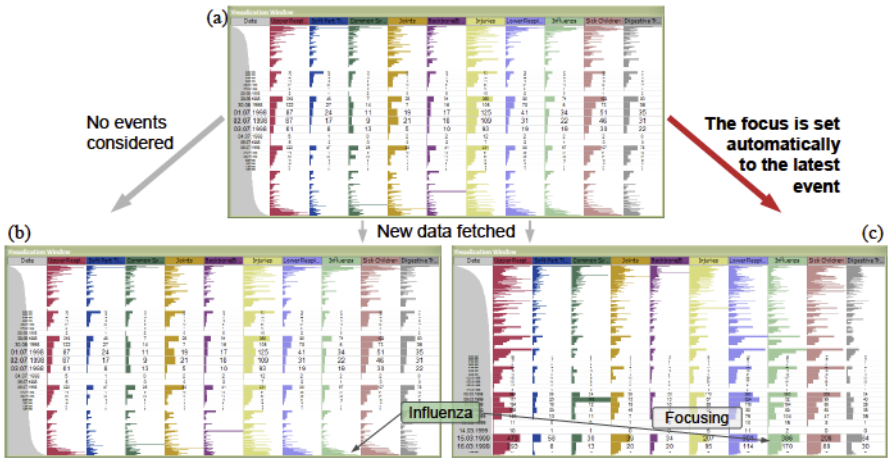


Figure 12.13: An event-enhanced Table Lens – (a) A Table Lens that visualizes multivariate temporal health data; (b) Several new data tuples have been fetched to the Table Lens. However, it can hardly be seen that an interesting data tuple is present now; the user remains unaware of the special situation; (c) The event "number of cases of influenza is greater than 300" has been detected in the newly added data and the Table Lens automatically focuses on that event (in the last row of the table). The user can identify relevant information easily.

In addition to the automatic adjustments illustrated before, users always have the chance to adjust the visual representation via interaction. This is possible since all our visual techniques implement visualization parameters as first-class objects. Where applicable we display visualization parameters in a unified user interface, which is automatically constructed from parameter descriptions, including id, caption, data type, and value range. However, this is not yet possible for all parameters. Particularly the parameters of complex visual mappings are hard to transfer to an efficiently usable user interface. This is one aspect for future work.

For the example of explicit event representation, we consider health data that were collected in a spatio-temporal frame of reference (ca. 750.000 data rows and 12 attributes). These data represent the numbers of cases of different diagnoses for different regions of a map on different temporal granularities. For this example, the assumed user interest is extended from a purely temporal context to a spatio-temporal one: The user wants to know where and when maxima occurred and how these evolve in time and space. The event type $\langle x \mid \forall_T y : (x.date \textbf{ equals } y.date) \rightarrow (x.flu \geqslant y.flu)\rangle_T$ detects per time step where in space (i.e., for which region) the maximum number of cases of influenza occurred. By utilizing efficient RDBMS technology, we can detect events in less than ten seconds. For the visualization, we use a map to represent the spatial frame of reference and time axes that emanate perpendicularly from each region of the map to depict the temporal dimension of the data (see Figure 12.14). Instances of the maximum event type are shown as small red spheres attached to the time axes. In contrast to the red spheres, blue spheres are used to represent instances of the logical negation of the considered event type (i.e., a maximum did not occur). Red and blue spheres allow users to see when and where maxima occurred. To enable users to track maxima, successive event occurrences are explicitly connected using a space-time-path [72], which is represented as white line segments. To help users concentrate on relevant information, the "not" events (blue spheres) of those regions that do not show at least one "positive" event are faded out. The result is a visualization that explicitly represents event instances, visualizes the trajectory of the event instances through time and space, and that is clean of irrelevant information. The number of relevant events in the visual representation is much smaller than the raw data. With respect to the example, it is now possible to find a potential trace of how influenza has spread in time and space. Surely, we have to admit that scientists from the field of epidemiology might have used a different and most likely a more complex event type for this example. But given the generality of our approach, they can easily do so.

All examples indicate that integrating events-based concepts into the visualization process is a promising approach to create visualiza-

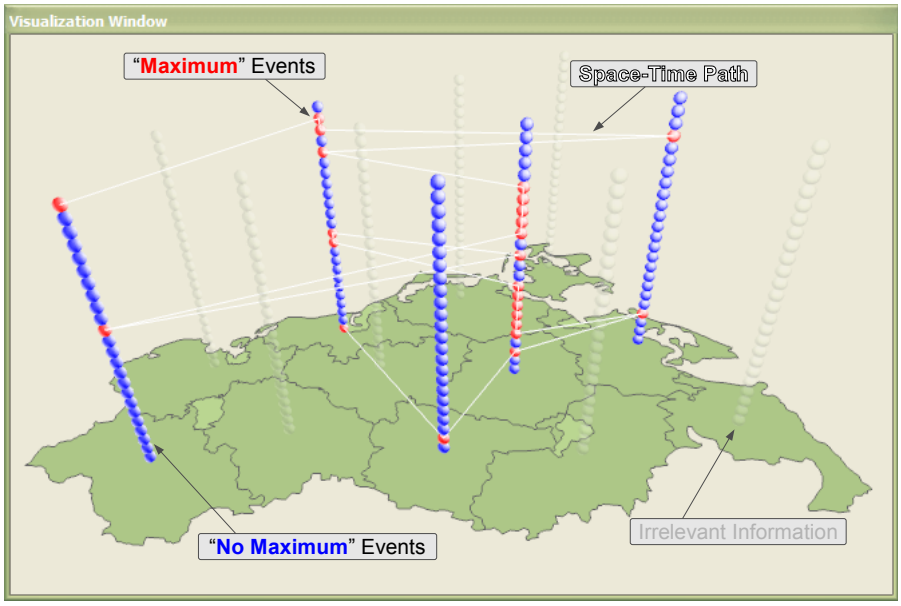


Figure 12.14: Explicit event representation as space-time-path – The figure shows when in time and where in space the maximum number of cases of influenza have occurred. Irrelevant information is faded out so that the user can fully concentrate on relevant events.

tion that are adapted to user interests. Yet more research is required to achieve the ambitious goal of fully automated generation of user-centered visual representations with high relevance. We will discuss this in the next section.

12.5 DISCUSSION AND CONCLUSION

In this paper, we presented an event-based approach to user-driven visualization. We developed a general theory and model that can be applied irrespective of the specific visualization scenario. We described in detail how users can specify their interests as event types, how these interests can be detected in the data, and how one can integrate detected event instances into visual representations.

The generality and theoretical foundation of the introduced concepts are major benefits. Thanks to this generality, the advantages of event-based visualization, that is, the focus on user interests and the reduction of the data to be visualized, can be transferred more easily to other visualization contexts. The framework *eVis*, which implements the model's core functionality, enabled us to illustrate the visual advantages of event-based visualization for multivariate temporal data and for spatio-temporal data.

With regard to the specification of user interests as event types, we have concentrated on tuple, attribute, and sequence events in relational data sets. As most data come in or can be transferred into

relational form, these major event types cover a wide range of application scenarios. Since the formalism relies on predicate logic and set theory, the expressiveness of the introduced event types is limited to that of these formalisms. The model's generality, however, allows for experimentation with different event types. Data nuggets as described in [383] or the knowledge-based approach discussed in [382] are interesting candidates. Another interesting aspect, especially in the context of graph analysis, are event types that are capable of capturing structural properties (e.g., motifs).

The visual specification of event types via a visual editor, as we pursue it here, has the advantage that arbitrary event types can be specified. On the other hand, brushing and dynamic queries allow for a more direct specification of interests, but are restricted to facts actually present in the data. Bringing both concepts closer together, that is, combining the flexibility of editor-based specification with the intuitiveness of brushing and dynamic queries poses an interesting research question for future work.

On the event detection stage, we suggested exploiting database technology to find events in relational data. As an alternative, the field of data mining offers a wealth of automatic event detection methods for different kinds of data. The question that arises is how well do the semantics of such automatically detected events resemble the interests of users. Or in other words, how can we make sure that the event detected by some automatic algorithm is really that what the user intended? An avenue to follow is to break the black box of automatic methods and to integrate them more tightly with interactively steerable visual analysis tools [325]. This implies that the implementation of event specification and event detection get closer together.

Since event detection is time-critical, especially for larger data sets, we highlighted the need for incremental detection methods. As soon as such methods become available in standard databases, the event detection should utilize them to increase detection efficiency. However, further research is required to investigate possible benefits (increased performance) and limitations (restricted expressiveness of event types) of incremental detection.

In this article, we discussed basic procedures of integrating events into visual representations. Implicit event representations try to improve relevance in visual representations by automatically adjusting visualization parameters. The key to success is to perform the right adjustments. However, it is not yet clear what the right adjustments are. For the TimeWheel example, we as the visualization designers implemented parameter changes so as to achieve expressive event representation. But as interests vary from user to user, so does what is perceived as expressive. Even though our approach gives users the opportunity to define their interests, it is not yet possible for users to specify how they want their interests to be represented on screen.

We think it is a formidable challenge to make implicit event representation as flexible as the event specification, including sufficient support, as for instance visual editors. This challenge could be tackled by breaking a visualization down into its building blocks, both from the perspective of construction (What visual elements make up the visualization?) and the perspective of perception (How are the elements perceived?). A possible way to do so is to use – again – formal descriptions [376]. Similarly, descriptions of basic visual adjustments and their effect on perception would be needed. Only then can one find matches of characteristics of events with characteristics of visual adjustments, which is a requirement for fully automatic adaptation. The generic building blocks can then be used to assemble domain-specific visualization solutions, which in turn can be subject to user evaluation.

For explicit event representation the situation is similar. The key question here is how to model event instances as a new data set to be fed into a dedicated visualization technique. Apparently, this question can only be answered taking the concrete application background into account. This fact shifts the problem in the general direction of visualization design, which is an actively investigated research topic.

Generating user-centered visual representations is a challenging task that requires joint effort of visualization, perception, human computer interaction, and data mining scientists. It would be false to claim that our model of event-based visualization provides a solution to this challenge. We rather see it as a step toward a solution – an initial answer to the question asked at the beginning "[...] what actually needs to be shown to intuitively draw representational conclusions." [25].

We approached user-driven visualization more from a technical point of view. Our model provides a technical platform that assists in focussing on user interests. However, further in-depth investigation is required to address questions of perception, user acceptance, and semantics of user tasks and interests. The model we present here can provide a basis for such investigations.

13

NAVIGATION RECOMMENDATIONS FOR EXPLORING HIERARCHICAL GRAPHS

CONTRIBUTION This chapter contributes a novel approach for assisting the human user in interacting with visual representations of data. The key concern is to narrow the gulf of evaluation by providing recommendations for intermediate-level navigation in large hierarchical graphs. Potentially interesting navigation destinations are automatically derived based on the concept of degree of interest (DOI). Visual hints point to these interesting destinations, which reduces the cost for evaluating the current exploration situation and makes triggering the next navigation steps easier.

ABSTRACT Navigation is a key interaction when analyzing graphs by means of interactive visualization. Particularly for unknown graphs, the user often faces situations where it is not entirely clear where to go next. For hierarchical graphs, the user may also ponder whether it is useful to look at the data at a higher or lower level of abstraction.

In this paper, we present a novel approach for recommending places in a hierarchical graph that are worth visiting next. A flexible definition of interestingness based on the notion of a degree of interest (DOI) allows us to recommend horizontal navigation in terms of the graph layout and also vertical navigation in terms of the level of abstraction. The actual recommendation is communicated to the user through unobtrusive visual cues that are embedded into the visual representation of the graph. A proof-of-concept implementation has been integrated into an existing graph visualization system.

ORIGINAL PUBLICATION [137] — S. Gladisch, H. Schumann, and C. Tominski. Navigation Recommendations for Exploring Hierarchical Graphs. In G. Bebis et al., editors, *Advances in Visual Computing*, pages 36–47. Springer, 2013.

13.1 INTRODUCTION

When exploring unknown graphs, users need to switch between overview and detail representations and they need to navigate to different parts of the graph. These tasks are typically supported by a zoomable representation of the graph, where the graph is hierarchically structured to provide different levels of abstraction [168]. The user can zoom & pan to visit different parts of the graph, and can expand or collapse nodes to adjust the level of abstraction. There are several existing systems that implement this strategy [41], [245], [332]. A big plus of these systems is that users can freely choose the part of the data they are interested in and the level of abstraction that suits their needs.

However, a problem is that users may be overwhelmed with the seemingly infinite number of possibilities for navigation. According to Spence [310], a key question for the user is: *Where should I go now?* Figure 13.1 illustrates this problem. Considering a current position arrived at during the exploration, the user does not know where interesting data could be located.

In this sense, navigating in an unknown graph to find interesting data is often a tedious trial-and-error procedure. This prompted us to investigate some kind of navigational guidance to interesting data. The aim of such a guidance is to facilitate the user's navigation decisions (i.e., recommend navigation to interesting targets) and to mitigate the trial-and-error character of navigation (i.e., minimize unconscious navigation through regions with uninteresting data).

In the following, we present a novel data-driven approach for navigation recommendations to support the exploration of hierarchical graphs. In Section 13.2 we describe in more detail the problem we are dealing with and briefly review existing state-of-the-art solutions. Section 13.3 introduces our novel approach, including means to define what the user is interested in, to compute navigation recommendations, and to communicate recommendations visually to the user.

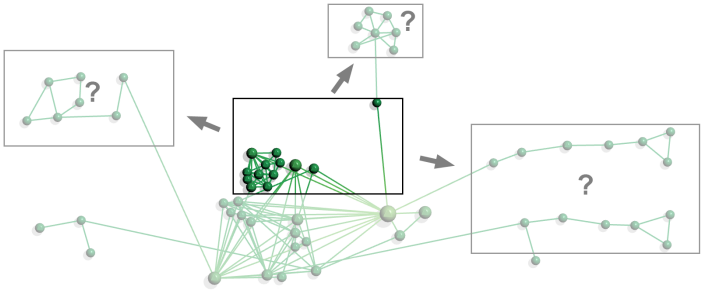


Figure 13.1: The problem of navigation. Given a partial view on a graph layout (center rectangle) the user does not know where to navigate in order to find "interesting" data (rectangles with question mark).

A demonstration of the proof-of-concept implementation is given in Section 13.4. Section 13.5 concludes our work and indicates directions for future work.

13.2 PROBLEM DESCRIPTION AND RELATED WORK

Next we describe the problem addressed by our research, review existing work that is related to this problem, and identify gaps to be filled with our approach.

13.2.1 *Problem Description*

We consider hierarchical graphs as input data. A hierarchical graph is defined as a rooted tree whose leaves correspond to a graph at the finest level of granularity. Nodes and edges of the graph may be associated with data attributes. Inner nodes of the tree correspond to aggregations or abstractions of their associated child nodes [168]. We assume that a suitable layout of the graph can be computed with existing methods [15].

To allow users to explore the graph, its layout is visualized as a node-link diagram that is embedded in a zoomable space. The zoomable space enables what we call *horizontal* navigation: The user can pan to any rectangular partial view of the graph layout. A hierarchical graph allows for additional navigation on its hierarchical structure: The user can expand or collapse nodes in order to get to a lower or higher level of abstraction [107]. We call this *vertical* navigation. Figure 13.2 illustrates both types of navigation.

One can easily imagine that the number of possible navigation steps is quite large. Should I pan this direction or the other to find some high-degree node? Which node should I expand to uncover a clique? Should I collapse these nodes to catch sight of the maximum-value node? In fact, the user can derive some more or less vague answers from the visual representation itself (e.g., navigate to where many edges connect). But we argue that a dedicated support to assist the user during navigation would be a promising addition to the user's analytical toolbox. With this thinking we are not alone, as documented in the next paragraphs.

13.2.2 *Related Work*

Looking at the literature one can find two categories of approaches to assist users in navigating graphs. On the one hand, there are approaches that focus on providing *orientation help* to keep users oriented. On the other hand, *navigation recommendation* approaches aim to actually suggest navigation steps to the user. In the following, we briefly review a few important examples.

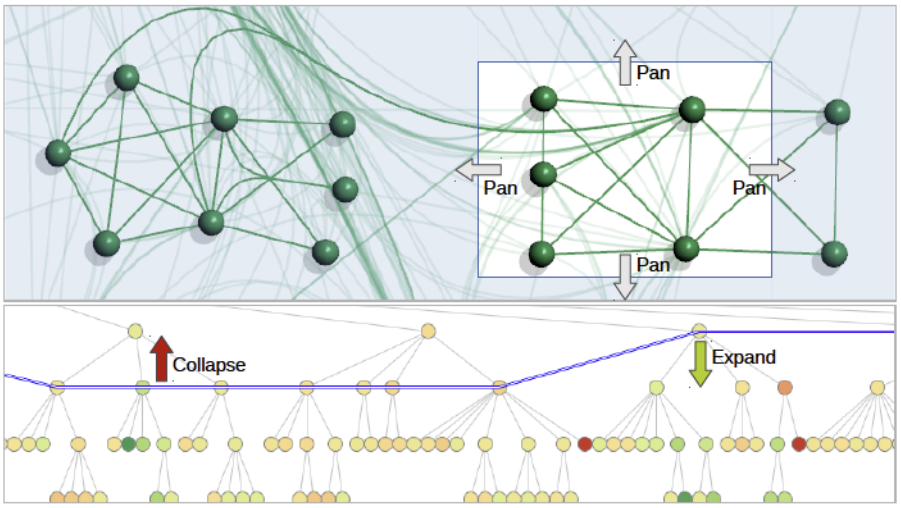


Figure 13.2: Navigation in a hierarchical graph. Horizontal navigation means altering the partial view on the graph layout (e.g., by panning the view). Vertical navigation means adjusting the level of abstraction (blue line) along the graph hierarchy by expanding or collapsing individual nodes. (Colors visualize data attributes associated with nodes.)

ORIENTATION HELP This kind of assistance helps users to orient themselves while navigating through the data. In the context of graph exploration, May et al. [249] present a technique that computes landmarks in the vicinity (context) of the current partial view (focus). The visualization is enhanced with labeled signposts that show directions to the determined landmarks. Jusufi et al. [195] investigate orientation guidance in graphs for which a complete partition is given. The approach is based on special glyphs that provide overviews of the subgraphs connected to a focus node. Plaisant et al. [274] also use special glyphs for user orientation. They enhance a tree visualization with preview icons that summarize the topology of subtrees. From a more general perspective, we can also consider off-screen visualization techniques (e.g., [49], [146], [126]) to be orientation help.

NAVIGATION RECOMMENDATIONS The main idea of navigation recommendations is to suggest navigation steps (e.g., a specific target or a direction). Van Ham and Perer [350] describe an exploration model for graphs that includes navigation recommendations. Based on an initial focus, the approach computes and shows the most interesting contexts. Visual hints help users to decide which nodes in the context to expand in order to navigate to the additional information. Crnovrsanin et al. [90] present a technique that recommends interesting nodes based on a set of selected nodes. Interestingness of nodes depends on data attributes, graph topology, and sequences of previous user interaction. Perer and van Ham [268] introduce *querying and browsing* as a new paradigm for graph exploration. They propose a

general model that determines an initial focus and its context on the graph based on a textual query. Special icons within a node-link diagram recommend to the user where to browse the context in order to find interesting information. Additionally, the approach computes and visualizes the shortest path from a focus node to a recommended node in the context.

OPEN RESEARCH QUESTIONS The reviewed examples from the literature demonstrate quite nicely how useful user assistance can be. A detailed look into the mechanisms behind the existing solutions reveals that most of them define a notion of a current *focus* that is associated with a *context*, where focus and context are defined exclusively on the graph structure. However, this implies that navigation recommendation can be given only for entities being connected to the focus in terms of the graph's topology. Interesting but disconnected nodes (e.g., in graphs with disconnected components) cannot be recommended, even if they are located close to the focus in the graph's layout (which is what users see on the display). Our novel solution addresses this limitation by utilizing a broader and more general notion of focus and associated context.

Another aspect common to the reviewed solutions is that they address only horizontal navigation in plain graphs. Hierarchical graphs have not been considered in connection with navigation recommendations so far. Our approach closes this gap by including vertical navigation along the axis of the level of abstraction. In other words, we address both horizontal navigation *and* vertical navigation. The next section will introduce our approach for navigation recommendations for hierarchical graphs.

13.3 NAVIGATION RECOMMENDATIONS FOR HIERARCHICAL GRAPHS

As described earlier, the scenario is that users explore an unknown hierarchical graph by means of a zoomable visualization that shows a layout of the graph and an encoding of associated data attributes. Our goal is to support the user in deciding which navigation step to take next to arrive at interesting data. To this end, we need to address the following key issues:

DETERMINING RECOMMENDATION CANDIDATES Given a hierarchical graph and the current state of the visual exploration process we need to derive a set of recommendation *candidates*.

SELECTING INTERESTING RECOMMENDATIONS In order to compile a set of navigation *recommendations* we need to rank the candidates according to their *interestingness* and select those that are worth visiting next.

COMMUNICATING RECOMMENDATIONS VISUALLY The selected navigation recommendations need to be *communicated* to the user in an unobtrusive fashion with as little distraction from the actual visualization as possible.

Following this line of thinking, we will next describe in more detail how our approach handles these issues. But first of all, we need to define what the targets for navigation recommendations could be. In general, one could recommend navigation to any entity related to a hierarchical graph, for example, nodes, edges, connected components, cliques, or any other semantically meaningful subset of nodes and edges. For the sake of simplicity, we restrict our considerations to nodes as the targets for navigation recommendations.

13.3.1 *Determining Recommendation Candidates*

As commonly accepted, the starting point for determining candidates is the user's current focus. Based on the focus we define a context, which contains the candidates. The context must include a sufficiently large number of candidates to choose from, and it must be sufficiently small to stay focused and to avoid computations on a huge search space. The size of the context and hence the number of candidates is controlled by means of a distance measure. In summary, we use three components: (1) a set of focus nodes to start with, (2) a sufficiently sized set of context nodes – the candidates, and (3) a distance measure to control the size of the context. Figure 13.3 illustrates how these components can be realized.

An intuitive and often used definition of these components is based on the graph structure. A set of focus nodes is selected by the user, and the context is defined by the k-neighborhood of the focus nodes. Here k is the parameter to be adjusted to control the size of the context.

As already indicated, we generalize focus and context to a broader definition. So, as a second facet, we additionally consider the user's current view on the graph layout. That is, all nodes that are currently visible on the display are considered to be the focus. The context is

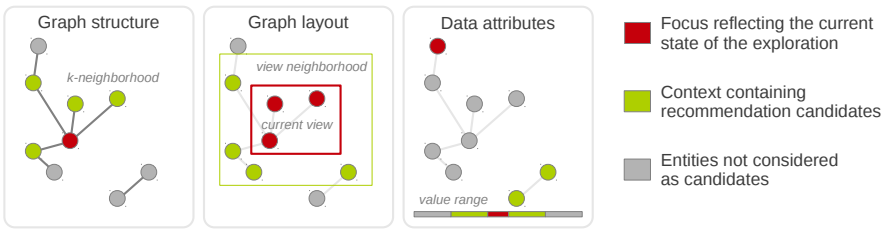


Figure 13.3: Different definition of focus and context in terms of the graph structure, the graph layout, and the data attributes yield different candidates for recommendation.

again defined in terms of a neighborhood, but this time a neighborhood in terms of the view space. The size of the context is again controllable.

So far we have not yet taken into account the data attributes that might be associated with the nodes. Consequently, we allow for a focus in attribute space. This focus can be determined in different ways, for example by dynamic filtering sliders or by fixing the value range of what is currently visible on the display. The context can then be defined as a range of values enclosing the focus, where the range's size can be set as needed.

With this general definition we can better capture the different aspects being relevant when exploring graphs – the graph structure, the graph layout, and the data attributes. The broader definition also allows us to circumvent problems that occur when considering either of the aspects alone. An example are nodes that are close to the focus in the layout, but that are far away in terms of the graph structure. In contrast to existing solutions, our approach is able to recommend navigation to such nodes.

13.3.2 *Selecting Interesting Recommendations*

Given our definition of candidates in the context, the next step is to assign an interestingness to each candidate. As we want to recommend interesting nodes to navigate to, we need a concept that describes how interesting a recommendation candidate is. Given the unpredictability of the visual exploration process, the concept must be capable of handling varying interestingness.

An established and widely-applied concept is the *degree of interest* (*DOI*), a numerical interestingness computed by means of a *DOI* function. Originally, Furnas [130] introduced the *DOI* for trees only. Van Ham and Perer [350] generalized it for graphs. Their weighted *DOI* function considers an a priori interest of the nodes (*API*), a distance to a focus (*DIST*) and a user interest (*UI*):

$$\mathrm{DOI}(x, F, \mathfrak{s}) = \alpha \cdot \mathrm{API}(x) + \beta \cdot \mathrm{UI}(x, \mathfrak{s}) + \gamma \cdot \mathrm{DIST}(x, F)$$

where x is the node to be assigned an interestingness, F is the current focus, $\mathfrak{s}$ are search criteria that describe what the user is currently interested in, and α, β, γ are real-valued weights. With this *DOI* definition, we already have a quite flexible mechanism to incorporate the user's interest into the recommendation computation.

Additionally, it can be important to know which data elements have already been visited (i.e., have been visible or have explicitly been marked as explored) in the course of the exploration. We propose to use an additional weighted *KNOW* component for the *DOI*

function that considers the interestingness of a node according to its exploration state:

$$\mathrm{DOI}(x, F, \mathfrak{s}) = \alpha \cdot \mathrm{API}(x) + \beta \cdot \mathrm{UI}(x, \mathfrak{s}) + \gamma \cdot \mathrm{DIST}(x, F) + \delta \cdot \mathrm{KNOW}(x)$$

By considering the exploration state of a node, we can penalize already explored data or, on the contrary, favor them. Which option to use depends on the user's goal. Visited nodes can be considered less interesting because they do not provide any new information. On the other hand, they could be particularly of interest for comparison tasks.

Given our specification of the *DOI* function, the question that remains to be answered is how to instantiate it and its components. We follow the accepted way of previous *DOI*-related approaches and provide interactive means for the user to specify and adjust the settings. To ease the specification procedure, we use template functions that can be parameterized using classic GUI elements. Which template functions to apply (i.e., what is interesting?) and how to parameterize them depends on the application domain, the use case, and the analyzed graph.

Given an appropriate *DOI* specification, we compute the interestingness of the nodes of a graph. It is worth mentioning that we do so only for the recommendation candidates in the context of the current focus. This spares us computing interestingness values for all nodes of the whole dataset.

For a hierarchical graph, we differentiate between two alternative ways of computing the interestingness. The first is that we compute interestingness for the finest level of granularity and aggregate interestingness along the hierarchy. The second alternative is to compute interestingness explicitly for each candidate irrespective of whether it is a leave node or an inner node. Again, the application scenario and the nature of the data decide on which alternative to apply.

Now that every candidate has a *DOI* value they can be sorted according to their interestingness. The result is a ranking of the recommendation candidates. Since we only want to recommend the *most interesting* nodes, we choose the first m nodes of the ranking as targets for the navigation recommendations, where m should be kept small to avoid overloading the user with too many recommendations. From our experience with test datasets, we suggest recommending $m < 10$ interesting navigation targets.

13.3.3 *Communicating Recommendations Visually*

The last step is to create an adequate visualization for the navigation recommendations. Given a potentially already visually rich graph visualization, how can we enhance it in order to communicate navigation recommendations to the users without interfering too much

with the ongoing visual exploration? Our answer to this question is to embed specifically designed visual navigation cues into the existing node-link visualization. Depending on the type of navigation and on where the target of a recommendation is located, we use different visual cues. The type of navigation can be either *horizontal* or *vertical*.

RECOMMENDATION FOR HORIZONTAL NAVIGATION For horizontal navigation, we distinguish navigation to nodes that are *on-screen* and nodes that are *off-screen*. Recommendations to on-screen nodes are visualized via subtle highlighting rings that encode how interesting a node is according to its *DOI* value.

For recommendations to off-screen nodes, we need a visual encoding that communicates at least the target's direction and better still, the distance to the target as well. For this purpose, we consider known techniques for off-screen visualization, such as arrows, *halos* [49], *wedges* [146], or *proxies* [126]. Arrows are easy to interpret, but communicate navigation direction only. Halos and wedges have the advantage that they encode direction and distance to a recommended navigation target. Further, wedges can be arranged to reduced overlap [146]. Proxies focus not so much on target distance, but more on communicating additional information about the target by means of shape, color, or labels.

Inspired by these approaches, we designed a new solution that combines the advantages of the existing ones. What we call *enriched wedge* is a visual cue that encodes direction and distance, and also additional information about *why* the recommendation was given. This is accomplished by embedding a bar chart into a wedge. The wedge visualizes direction and distance, and each bar visualizes the partial interestingness of a recommended node according to the individual components of the *DOI* function (i.e., *API*, *UI*, *DIST*, *KNOW*). Using enriched wedges can positively influence the user in arriving at a navigation decision (i.e., choosing the "right" navigation target). Figure 13.4 illustrates the enriched wedge in comparison to existing off-screen techniques.

Of course, the idea of enriching the navigation recommendations with a visualization of individual *DOI* values is not restricted to wedges pointing to off-screen targets. The highlighting of on-screen targets can be enhanced in a similar manner to provide information about interestingness at a glance.

RECOMMENDATION FOR VERTICAL NAVIGATION A vertical navigation is necessary when the recommended target is not contained in the currently visualized level of abstraction. As the target is definitely not visible, we need to pick a suitable anchor to attach the navigation recommendation to. We decided to visually highlight the nodes whose expansion (or collapse) would bring the recommended

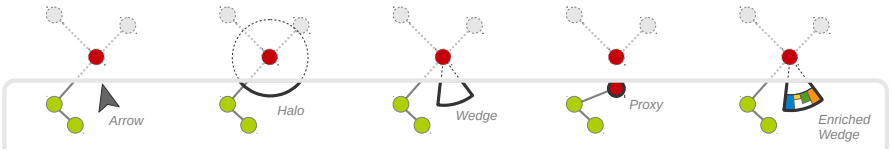


Figure 13.4: Techniques for recommending navigation to off-screen nodes. Green nodes are on-screen, dashed elements are off-screen, and the red node is the recommend target.

target to the display. For example, if a target is below the current level of abstraction, we highlight the target's ancestor that is contained in the current level of abstraction and whose expansion will uncover the target. If the ancestor is off-screen we can again apply one of the off-screen techniques described before.

In order to differentiate the highlighting for vertical navigation from that for horizontal navigation, and further the one for node expansion from that of node collapse, we resort to animated rings around nodes. In accordance with our goal to generate an unobtrusive visual embedding, the highlighting is designed as subtly pulsing animations with a specific direction. The animated rings appear to shrink when collapse navigation is recommended and to grow for recommended expansion. Figure 13.5 shows snapshots of an animation indicating an expand recommendation.

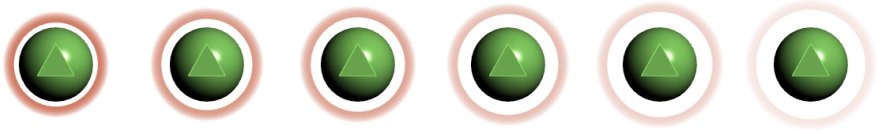


Figure 13.5: Snapshots of the animation that indicates nodes to be expanded to arrive at a recommended navigation target.

13.3.4 *Summary and Additional Concerns*

With the aforementioned mechanisms, we can select interesting nodes and recommend them visually to the user as potentially worthy steps for navigation. A key issue of our approach is balancing it appropriately. Interests vary and also the visual presence of navigation cues will be perceived differently by different users in different stages of the exploration process. Therefore, it is critical to adjust the computation of recommendations and their visualization to the application scenario and to the preferences of the user. Our approach provides the required flexibility to do so. Further, we should recall the on-demand character of our approach. That is, only if users feel that they need assistance they will activate the navigation recommendations.

Two additional concerns need to be addressed in the context of navigational guidance: (1) a good initial view to start with and (2) a

visual encoding of the exploration state. Both are not trivial question and we have not dealt with them in depth. Yet we give some ideas how to address them.

Ideally, a good initial view on the data provides an expressive overview of the data and offers a suitable number of options for further exploration. For determining such an initial view, different criteria can matter. For example, the number of nodes can be considered. Huang et al. [174] state that 20 to 100 nodes are suitable for an overview. Moreover, in specific applications, there may exist data elements being semantically more relevant than others. In such cases, including graph elements of higher relevance (e.g., outliers) can lead to a more appropriate initial view. One could also favor nodes with high degree as they potentially lead to more options for navigation along the graph structure. Despite these initial suggestions, creating a good initial view remains a difficult and largely context-dependent task.

The second concern regards the dependency of interestingness on the exploration state. To make this dependency clear to the user it makes sense to visualize a node's exploration state as well, because it may influence navigation decisions. When exploring hierarchical graphs the user might want to know which subtrees have already been explored. Cramming this additional information into the visualization as well is difficult. Therefore, we experimented with on-demand labeling that classifies nodes into *unexplored*, *partially explored*, and *explored*. Such on-demand labels can help users to decide where to explore further and where no further exploration is necessary.

13.4 PROOF-OF-CONCEPT IMPLEMENTATION

To test our approach, we developed a proof-of-concept implementation. As the underlying zoomable graph visualization, we use the *CGV* system [332]. We implemented a plausible default preset for the interestingness specification (including maximum attribute values and attribute outliers), which enables us to give recommendations at all times, even in cases where the user has not yet made the interests known to the system. The *DOI* function and its components can be altered interactively via a simple graphical user interface. We implemented arrow-based recommendation cues and our *enriched wedge*.

We tested the system with several hierarchical graphs. Here we illustrate its application with a graph that contains search queries as nodes and relations between the queries as edges. The graph is of moderate size with 695 nodes and 4073 edges. Figure 13.6 shows a partial view on the graph as the user may see it during exploration. Note that for the purpose of demonstration we use a visual encoding that might not appear as gentle and subtle as one would use it in a real application. In the figure, we can see recommendations to investigate on-screen targets, indicated by red circles around some

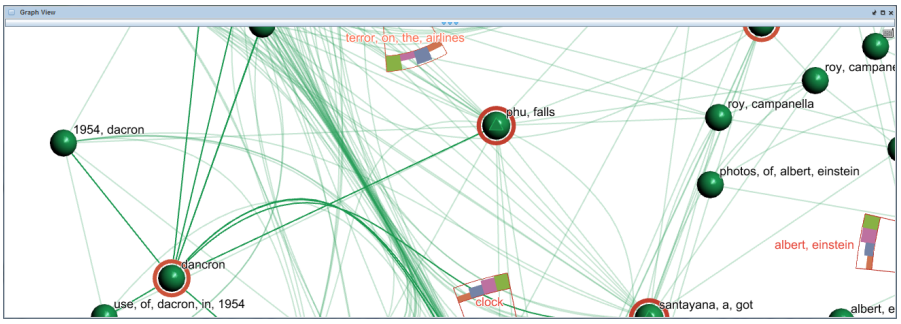


Figure 13.6: Navigation recommendation in the proof-of-concept implementation. Red circles indicate on-screen targets worth investigating next. Enriched wedges indicate off-screen targets that might be of interest to the user as well.

nodes. Enriched wedges at the border of the screen recommend navigation to off-screen targets. Once the user has decided on the next navigation target, it can be visited by following the navigation recommendations manually. For example, the user can pan the view in the direction of an enriched wedge until the target falls into view. The target is then highlighted using the red circle for on-screen targets. As an alternative to manual navigation, we utilize *CGV's* animation facilities to provide automatic animated traveling to the selected target. To this end, the user simply clicks an enriched wedge to trigger the animation.

During the exploration, the recommendations are constantly updated according to the current focus and the specification of the user's interest. The system also keeps track of which nodes have already been visited, where we rely on users explicitly marking a node as explored.

In summary, provided that users are able to express their current interest, our implementation can help in locating interesting nodes in the local context quickly.

13.5 CONCLUSION

In this work, we developed a data-driven approach for navigation recommendations to interesting information. Our solution is based on three basic steps: (1) collection of a set of recommendation candidates based on a compound focus, (2) selection of navigation recommendation based on user interests, and (3) visualization of navigation recommendations via visual cues embedded into an existing graph visualization.

Our solution extends the body of existing work in several aspects. We addressed hierarchical graphs, which require both horizontal and vertical navigation, an issue not studied in previous work. In terms of computing navigation recommendations, we generalized the notion

of focus and context to incorporate the graph structure, the graph layout, and the data attributes. Further, we extended the widely-accepted *DOI* concept by the component *KNOW*, which captures the exploration state of data elements. For communicating navigation recommendations, we suggest several visual encodings, including the novel *enriched wedge*. A proof-of-concept implementation has been developed.

The mechanisms behind our concept work quite well in the proof-of-concept implementation. However, in the future, we need to develop a better interface for specifying the users' interests. Our current approach with classic GUI elements needs to be revised in order to make the overall solution more accessible for users. Further it makes sense to keep a history of what users have already marked as interesting. This would allow us to find better starting points for exploration.

A pressing issue is that the degree to which our solution reduces the trial-and-error character of visual exploration has not yet been quantified. And unfortunately we believe that it will be hard to do so due to the many influencing factors. Therefore, we invite evaluation experts to contact us and we will gladly collaborate and provide our implementation for in depth usability studies.

Part III

THE OUTRO

This final part summarizes the key viewpoints discussed in this work and presents the overall conclusion. The greater picture behind this work will be crystallized and based on that topics for future work on interaction in visualization will be identified.

14 CONCLUSION AND FUTURE WORK

14.1 CONCLUDING REMARKS

This work started out with the goal to strengthen the interaction side of visualization research. We looked at fundamentals of interaction in general and specifically in relation to visualization, and we discussed the challenges involved when designing and engineering interaction techniques in the context of visualization.

To actually achieve the aforementioned goal, this work presented several contributions with a focus on interactivity. The aspect of designing interaction for visualization has been addressed by developing a number of novel low-level, intermediate-level, and high-level techniques aligned with a unified view of interaction and its four cornerstones the data, the tasks, the technology, and the human. We considered the aspect of engineering interactive solutions by introducing software architectures and infrastructures.

Given the cumulative nature of this work it is clear that the proposed solutions address specific problems in the context of visualization. Still there is a greater picture of this work as a whole. This picture is to be crystallized in the following paragraphs.

UNIFIED VIEW AND CORNERSTONES At the core of this work are the cornerstones of interaction in visualization: the data, the tasks, the technology, and the human. While we looked at these cornerstones individually, we also highlighted interrelations among them. For example, while we focused on the task of comparison in Chapter 8, we also considered the specifics of the data (tabular data in our case), the characteristics of the interaction technology (hence the especially designed folding technique), and the needs of the human user (by drawing inspiration from natural behavior). In a similar sense, all other chapters in this work indicate how important the cornerstones are, both individually and as a unified view.

In fact, a key message of this work is that one has to consider all cornerstones of interaction in order to be successful when designing and engineering interaction for visualization. Leaving only one of the cornerstones out of consideration puts us at risk to arrive at inappropriate or unusable solutions.

INTERACTION AND AUTOMATIC MEANS Another major point discussed in this work is that interaction, although powerful and effective, is not a universal answer to all problems. This points us to think

more carefully about where interaction is indeed helpful and in what situations the user is better off with less interaction.

This thinking is reflected several times by approaches that combine interaction with automatic means. Examples were given in Chapter 9, where fully manual data editing is facilitated by automatic computations. In Chapter 12, automatic event-triggered adjustments of visual representation were applied to reduce interaction costs. Analytic DOI-based concepts are the backbone of the navigation recommendations presented in Chapter 13. A less obvious example are the interaction shortcuts from Chapter 8, which take the user or bring a view automatically to a desired position in the visualization space.

The integration of interactive and automatic means has been successful in the mentioned examples. Yet, balancing interaction and automatism is not trivial and, as mentioned before, requires consideration of all cornerstones of interaction in the context of the application domain.

BROAD UTILIZATION OF TECHNOLOGY An important contribution of this work is the development, collection, and illustration of novel interaction approaches utilizing different kinds of modern technologies. Looking at the equipment that we employed to drive the visual interface between the human and the computer, we see a gradual shift from classic desktop settings with mouse and keyboard to modern display technologies and interaction modalities.

Predominant in Chapters 6–8 was the classic mouse and keyboard setup. But still the interactive approaches presented there are novel, a fact that indicates that even in standard settings there is still potential to be exploited. In Chapter 9, we started to go beyond mouse and keyboard by designing interaction for touch-enabled devices. Advancing further, Chapter 10 discussed tangible interaction in the space above a tabletop display. Finally, we also explored physical navigation and head tracking in front of a large display wall in Chapter 11.

As a conclusion, the approaches presented here suggest that a broad utilization of display and interaction technologies bears much potential for visualization applications. In a sense, our work provides some initial results in reply to recent research agendas [224, 183] that call for better utilization of modern interaction technology.

LENSES AS UNIVERSAL TOOLS A recurring pattern throughout this work is the use of interactive lenses to facilitate visualization tasks. Thanks to the lightweight and focused nature of lenses, they are particularly suited for exploratory visualization.

We used lenses to support the exploration of graphs in Chapter 6 and to integrate temporal information into a visual representation that is otherwise focused on spatial aspects in Chapter 7. In Chapter 11, we tracked the user's viewing direction in order to project a fo-

cus+context lens onto a large display wall. While these examples can be considered virtual lenses, which represent a digitized copy of the lens metaphor on the display, the tangible views from Chapter 10 are examples of physical, graspable lenses. Such tangible lenses open up whole new possibilities for visualization purposes as demonstrated in several case studies.

A recent survey confirms that many more interactive lenses exist for various types of data and tasks [339]. From the wide application of lenses in our work and in the work of others, we can conjecture that lenses are universally useful tools in interactive visualization.

THE CHALLENGE OF DEVELOPING INTERACTION So far, we emphasized the results of our work, which come as new interaction approaches and techniques. However, we did not look at the process of generating the results. Unfortunately, due to the multitude and heterogeneity of influencing factors, there is hardly a structured process one could follow. In fact, the process is more of exploratory character. Spence [310] explains this quite illustratively:

> Many ingredients to support representation, presentation and interaction are described [...]: like a good chef or skilled painter, interaction designers must select appropriate ingredients from those available and use established concepts to blend them into a pleasing and effective product. And, as with cooking and painting, good interaction design can be achieved only with practice and the experience of both good and not so good results.
>
> — Spence [310]

Spence's words already indicate that designing and engineering interaction in visualization is hard. During the development of the approaches presented here, many hours if not weeks have been spent on trying out (which essentially means designing and implementing) and throwing away various interaction solutions until eventually the final result surfaced. Although the costs involved were significant, it is worth investing in interaction, because the benefit for the user is a more productive solution and smooth working experience. The user feedback reported here testifies to this.

In summary, we see that interaction indeed deserves special attention in visualization research. By bringing together in one place the aspects covered in the individual research papers underlying this work, we were able to generate a comprehensive picture of interaction in visualization and derive higher-level insight about the topic. In this sense, this work as a whole is greater than the sum of its parts. Hence, we can conclude that our work contributes to strengthening the interaction side of visualization.

14.2 TOPICS FOR FUTURE WORK

Although we made a number of contributions in this work they are but pieces in the larger puzzle of the *science of interaction* in visualization research. This puzzle holds many more unsolved questions that need to be addressed. This also pertains to the novel approaches presented in Chapters 5–13 of Part ii.

Of course it is important to resolve the remaining issues of these approaches in the short run, but it is not the goal of this section to reiterate any details about the involved challenges here. Instead, the next paragraphs provide an agenda of topics for more long-term future research with a focus on interaction in visualization.

SUPPORT THE ENGINEERING OF INTERACTION As already indicated, interaction engineering in the context of visualization is a complex endeavor. A key difficulty is the dependency of the interaction design on the visual design. Because the visual representation serves as the user interface, interaction can only be implemented efficiently after the visual design has been finalized. In fact, the problem with this dependency is that a little change in the visual design can break the interaction design.

Therefore, new conceptual models and strategies have to be developed to support the human engineer in experimenting effortlessly with different visual and interaction designs. To this end, the aforementioned dependency has to be relaxed. The benefit of such a relaxation will be a greater flexibility during the development process and reduced development cost.

In addition to such conceptual considerations, there is room for improvements on the practical side of interaction engineering. Still the standard model is to implement interaction via event handler methods in some programming language. However, this procedure is cumbersome and error-prone. Alternative methods such as modeling interaction via state machines [37] should be investigated for their applicability in the context of visualization. Ideally, in the future, interaction for visualization will be modeled using a graphical editor, rather than coded in a programming language.

GO BEYOND MOUSE AND KEYBOARD Although mouse and keyboard are still the predominant interaction devices and the regular desktop display is still the most frequently used output device, the future will see a shift toward alternative settings. Direct touch on high-resolution surfaces will most likely replace the standard workplace in the near future. Specialized applications will run in large display environments that track their users and integrate the devices the users bring with them.

While the visualization community has begun to recognize this shift (e.g., [335, 224, 183, 190]), still more research is necessary to comprehensively integrate modern display and interaction technologies with visualization approaches. Such research has to tackle several challenges. First, technical issues have to be addressed to create a basis upon which interactive visualization solutions can be built. Second, investigations are needed to determine how and where new technologies can be employed most usefully. Third, studies have to be conducted to evaluate the benefits of the new technologies in comparison to established setups. A concrete example could be a comparative study on graph exploration on the desktop using the CGV system from Chapter 6, graph exploration using tangible views from Chapter 10, and graph exploration using physical navigation as described in Chapter 11. One can easily imagine how complex such a study would be.

PROVIDE GUIDELINES AND GUIDANCE Interaction approaches are as multi-faceted as the problems they aim to solve. With our unified view on interaction, we can somewhat structure the space of useful interaction solutions. Yet, with interaction for different data and different tasks, and maybe even for different technologies, it can become difficult to develop or choose an appropriate interaction technique for a given visualization problem.

Therefore, we have to provide guidelines for visualization engineers and guidance for visualization users. By guidelines we mean a set of established best practices that a visualization engineer can refer to when developing interactive visualization approaches. Such guidelines could, for example, suggest how to map interaction intents to appropriate interaction techniques using the most efficient technology. While guidelines apply in the development phase of visualization, guidance is to support the user while using interactive visualization tools. The navigation recommendations from Chapter 13 are an example for guidance during interactive navigation. But guidance in general is much broader and can be provided with respect to various domains at different degrees [298]. Here we see much potential for future research on guiding the user in making the most of the provided interaction functionality.

OVERCOME THE INTERACTION-VISUALIZATION GAP The gap between interaction and visualization needs to be narrowed further. A still problematic concern is the lack of a consistent model that covers visual aspects as well as aspects of interaction on equal terms. The interaction model by Norman [262] describes how the user interacts with the computer. The various instances of the visualization pipeline [147, 70, 100] detail on the steps that are performed by the computer during the generation of a visual representation. In the former case

the human acts upon the data (in visual form), in the latter case the computer acts upon the data (in symbolic form). This discrepancy has to be overcome.

A new integrated model of a visualization–interaction pipeline would help bridge the gap between interaction and visualization research. One option to build such a model is to compile a conglomerate of the visualization pipeline [147, 70, 100], Norman's [262] interaction model, and the model of visual exploration by Jankun-Kelly et al. [188]. Constructing such an integrated model requires a detailed analysis of where visualization and interaction models differ and what the key influencing factors behind the models are.

ESTABLISH AN INTERACTION VOCABULARY With Bertin's visual variables there is an established vocabulary of basic building blocks for the graphics design of visualization approaches. However, there are no such building blocks for interaction in visualization.

Therefore, an open research question is to define an interaction vocabulary. There have already been first efforts to identify patterns or primitives of interaction in visualization [300, 286]. These studies describe how interaction is employed for different tasks. What we are still missing are constructive building blocks that allow us to flexibly outfit a graphics design with a suitable interaction design depending on the characteristics of the underlying problem. Identifying, collecting, and structuring such building blocks in an interaction vocabulary, will enable us to investigate whole new topics. As one such topic we envision adaptive interaction, where interaction techniques automatically adapt to the data and the task at hand as well as to the available technology and the human user operating it. Maybe it is even possible to extend the interaction vocabulary to a *grammar of interaction* analogous to Wilkinson's [376] *grammar of graphics*.

As mentioned earlier, our ideas for future work represent larger research topics to be studied in the long run. On the one hand, the versatility of the topics suggests that there is still much to do in the context of a *science of interaction*. On the other hand, we also indicated that there is much potential in further strengthening the interaction side of visualization. The positive results obtained with the approaches presented in this work let us conclude that it is worth facing and tackling the challenges ahead of us.

BIBLIOGRAPHY

[1] Anoto Group AB. http://www.anoto.com. Retrieved: 14. Juli 2014. (Cited on page 212.)

[2] CGLX Cross-Platform Cluster Graphics Library. http://vis.ucsd.edu/~cglx/. Retrieved: 14. Juli 2014. (Cited on page 224.)

[3] FTGL Font Rendering Library. http://sourceforge.net/projects/ftgl/. Retrieved: 14. Juli 2014. (Cited on page 224.)

[4] HyperTree Java Library. http://hypertree.sourceforge.net. Retrieved: 14. Juli 2014. (Cited on page 108.)

[5] JGraphT. http://jgrapht.sourceforge.net. Retrieved: 14. Juli 2014. (Cited on page 108.)

[6] JUNG – Java Universal Network/Graph Framework. http://jung.sourceforge.net. Retrieved: 14. Juli 2014. (Cited on page 108.)

[7] OpenStreetMap. http://www.openstreetmap.org. Retrieved: 14. Juli 2014. (Cited on page 131.)

[8] Safecast. http://blog.safecast.org/maps/. Retrieved: 14. Juli 2014. (Cited on page 138.)

[9] Tableau Software. http://www.tableausoftware.com. Retrieved: 14. Juli 2014. (Cited on page 108.)

[10] EU Coordination Action VisMaster. http://www.vismaster.eu. Retrieved: 14. Juli 2014. (Cited on page 67.)

[11] Walrus – Graph Visualization Tool. http://www.caida.org/tools/visualization/walrus. Retrieved: 14. Juli 2014. (Cited on page 108.)

[12] J. Abello. Hierarchical Graph Maps. *Computers & Graphics*, 28(3):345–359, 2004. doi: 10.1016/j.cag.2004.03.012. (Cited on pages 84, 85, and 87.)

[13] J. Abello and J. Korn. MGV: A System for Visualizing Massive Multidigraphs. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):21–38, 2002. doi: 10.1109/2945.981849. (Cited on pages 84, 87, and 90.)

[14] J. Abello, A. L. Buchsbaum, and J. Westbrook. A Functional Approach to External Graph Algorithms. *Algorithmica*, 32(3):437–458, 2002. doi: 10.1007/s00453-001-0088-5. (Cited on page 85.)

[15] J. Abello, F. van Ham, and N. Krishnan. ASK-GraphView: A Large Scale Graph Visualization System. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):669–676, 2006. doi: 10.1109/TVCG.2006.120. (Cited on pages 85, 86, 90, 91, 106, 107, 109, and 259.)

[16] J. Abello, H.-J. Schulz, H. Schumann, and C. Tominski. Interactive Poster: CGV – Coordinated Graph Visualization. Poster at IEEE Information Visualization Conference (InfoVis), 2007. (Cited on pages 85 and 102.)

[17] G. D. Abowd and A. J. Dix. Giving Undo Attention. *Interacting with Computers*, 4(3):317–342, 1992. doi: 10.1016/0953-5438(92)90021-7. (Cited on page 107.)

[18] E. Adar. GUESS: A Language and Interface for Graph Exploration. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 791–800. ACM Press, 2006. doi: 10.1145/1124772.1124889. (Cited on page 109.)

[19] C. Ahlberg, C. Williamson, and B. Shneiderman. Dynamic Queries for Information exploration: An Implementation and Evaluation. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 619–626. ACM Press, 1992. doi: 10.1145/142750.143054. (Cited on pages 23, 84, 98, and 196.)

[20] W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski. Visualizing Time-Oriented Data – A Systematic View. *Computers & Graphics*, 31(3):401–409, 2007. doi: 10.1016/j.cag.2007.01.030. (Cited on page 248.)

[21] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of Time-Oriented Data*. Springer, 2011. doi: 10.1007/978-0-85729-079-3. (Cited on pages 34, 35, and 135.)

[22] J. F. Allen. Time and Time Again: The Many Ways to Represent Time. *International Journal of Intelligent Systems*, 6(4):341–355, 1991. doi: 10.1002/int.4550060403. (Cited on page 238.)

[23] G. Alvarez and P. Cavanagh. The Capacity of Visual Short-Term Memory is Set Both by Visual Information Load and by Number of Objects. *Psychological Science*, 15(2):106–111, 2004. doi: 10.1111/j.0963-7214.2004.01502006.x. (Cited on pages 153 and 156.)

[24] R. Amar, J. Eagan, and J. Stasko. Low-Level Components of Analytic Activity in Information Visualization. In *Proceedings of the IEEE Symposium Information Visualization (InfoVis)*, pages 89–100. IEEE Computer Society, 2005. doi: 10.1109/INFOVIS.2005.24. (Cited on page 22.)

[25] R. A. Amar and J. T. Stasko. Knowledge Precepts for Design and Evaluation of Information Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):432–442, 2005. doi: 10.1109/TVCG.2005.63. (Cited on pages 228 and 255.)

[26] F. Amini, S. Rufiange, Z. Hossain, Q. Ventura, P. Irani, , and M. J. McGuffin. The Impact of Interactivity on Comprehending 2D and 3D Visualizations of Movement Data. *IEEE Transactions on Visualization and Computer Graphics*, 2014. doi: 10.1109/TVCG.2014.2329308. To appear. (Cited on page 15.)

[27] C. Andrews and C. North. The Impact of Physical Navigation on Spatial Organization for Sensemaking. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2207–2216, 2013. doi: 10.1109/TVCG.2013.205. (Cited on page 50.)

[28] C. Andrews, A. Endert, B. Yost, and C. North. Information Visualization on Large, High-Resolution Displays: Issues, Challenges, and Opportunities. *Information Visualization*, 10(4):341–355, 2011. doi: 10.1177/1473871611415997. (Cited on pages 48 and 50.)

[29] K. Andrews. Evaluating Information Visualisations. In *Proceedings of the Workshop on Beyond Time and Errors: Novel Evaluation Methods for Information Visualization (BELIV)*, pages 1–5. ACM Press, 2006. doi: 10.1145/1168149.1168151. (Cited on pages 112 and 113.)

[30] G. Andrienko and N. Andrienko. Poster: Dynamic Time Transformation for Interpreting Clusters of Trajectories with Space-Time Cube. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 213–214. IEEE Computer Society, 2010. doi: 10.1109/VAST.2010.5653580. (Cited on page 137.)

[31] G. Andrienko, N. Andrienko, U. Demsar, D. Dransch, J. Dykes, S. I. Fabrikant, M. Jern, M.-J. Kraak, H. Schumann, and C. Tominski. Space, Time and Visual Analytics. *International Journal of Geographical Information Science*, 24(10):1577–1600, 2010. doi: 10.1080/13658816.2010.508043. (Cited on page 125.)

[32] G. Andrienko, N. Andrienko, P. Bak, D. Keim, S. Kisilevich, and S. Wrobel. A Conceptual Framework and Taxonomy of Techniques for Analyzing Movement. *Journal of Visual Languages & Computing*, 22(3):213–232, 2011. doi: 10.1016/j.jvlc.2011.02.003. (Cited on page 125.)

[33] G. Andrienko, N. Andrienko, and M. Heurich. An Event-Based Conceptual Model for Context-Aware Movement Analysis. *International Journal of Geographical Information Science*, 25

(9):1347–1370, 2011. doi: 10.1080/13658816.2011.556120. (Cited on page 126.)

[34] G. Andrienko, N. Andrienko, C. Hurter, S. Rinzivillo, and S. Wrobel. From Movement Tracks through Events to Places: Extracting and Characterizing Significant Places from Mobility Data. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 161–170. IEEE Computer Society, 2011. doi: 10.1109/VAST.2011.6102454. (Cited on page 123.)

[35] G. Andrienko, N. Andrienko, P. Bak, D. Keim, and S. Wrobel. *Visual Analytics of Movement*. Springer, 2013. doi: 10.1007/978-3-642-37583-5. (Cited on page 36.)

[36] N. Andrienko and G. Andrienko. *Exploratory Analysis of Spatial and Temporal Data*. Springer, 2006. doi: 10.1007/3-540-31190-4. (Cited on pages 22, 38, 124, 128, and 150.)

[37] C. Appert and M. Beaudouin-Lafon. SwingStates: Adding State Machines to Java and the Swing Toolkit. *Software: Practice and Experience*, 38(11):1149–1182, 2008. doi: 10.1002/spe.867. (Cited on pages 17 and 276.)

[38] J. Arvo and K. Novins. Fluid Sketching of Directed Graphs. In *Proceedings of the Australasian User Interface Conference (AUIC)*, pages 81–86. Australian Computer Society, 2006. (Cited on page 176.)

[39] M. Ashdown, K. Oka, and Y. Sato. Combining Head Tracking and Mouse Input for a GUI on Multiple Monitors. In *CHI Extended Abstracts on Human Factors in Computing Systems*, pages 1188–1191. ACM Press, 2005. doi: 10.1145/1056808.1056873. (Cited on page 218.)

[40] D. Auber. Tulip – A Huge Graph Visualisation Framework. In P. Mutzel and M. Jünger, editors, *Graph Drawing Software*, pages 105–126. Springer, 2004. doi: 10.1007/978-3-642-18638-7_5. (Cited on pages 106 and 109.)

[41] D. Auber, D. Archambault, R. Bourqui, A. Lambert, M. Mathiaut, P. Mary, M. Delest, J. Dubois, and G. Melançon. The Tulip 3 Framework: A Scalable Software Library for Information Visualization Applications Based on Relational Data. Research Report RR-7860, INRIA, 2012. URL `http://hal.inria.fr/hal-00659880`. (Cited on page 258.)

[42] N. H. Balkir, G. Ozsoyoglu, and Z. M. Ozsoyoglu. A Graphical Query Language: VISUAL and Its Query Processing. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):955–978, 2002. doi: 10.1109/TKDE.2002.1033767. (Cited on page 242.)

[43] R. Ball and C. North. Effects of Tiled High-Resolution Display on Basic Visualization and Navigation Tasks. In *CHI Extended Abstracts on Human Factors in Computing Systems*, pages 1196–1199. ACM Press, 2005. doi: 10.1145/1056808.1056875. (Cited on page 218.)

[44] R. Ball, M. Dellanoce, T. Ni, F. Quek, and C. North. Applying Embodied Interaction and Usability Engineering to Visualization on Large Displays. In *Proceedings of the British HCI Workshop on Combining Visualisation and Interaction to Facilitate Scientific Exploration and Discovery*, pages 57–65, 2006. (Cited on page 218.)

[45] R. Ball, C. North, and D. A. Bowman. Move to Improve: Promoting Physical Navigation to Increase User Performance with Large Displays. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 191–200. ACM Press, 2007. doi: 10.1145/1240624.1240656. (Cited on pages 216 and 218.)

[46] L. Bartram, C. Ware, and T. Calvert. Moving Icons: Detection And Distraction. In *Proceedings of the TC13 IFIP International Conference on Human-Computer Interaction (INTERACT)*. IOS Press, 2001. (Cited on page 247.)

[47] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999. (Cited on pages 30, 84, 91, 180, and 220.)

[48] T. Baudel. From Information Visualization to Direct Manipulation: Extending a Generic Visualization Framework for the Interactive Editing of Large Datasets. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 67–76. ACM Press, 2006. doi: 10.1145/1166253.1166265. (Cited on pages 23, 41, 174, and 175.)

[49] P. Baudisch and R. Rosenholtz. Halo: A Technique for Visualizing Off-Screen Objects. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 481–488. ACM Press, 2003. doi: 10.1145/642611.642695. (Cited on pages 260 and 265.)

[50] M. Beaudouin-Lafon. Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 446–453. ACM Press, 2000. doi: 10.1145/332040.332473. (Cited on pages 10 and 152.)

[51] M. Beaudouin-Lafon. Novel Interaction Techniques for Overlapping Windows. In *Proceedings of the ACM Symposium on User*

Interface Software and Technology (UIST), pages 153–154. ACM Press, 2001. doi: 10.1145/502348.502371. (Cited on page 159.)

[52] R. A. Becker and W. S. Cleveland. Brushing Scatterplots. *Technometrics*, 29(2):127–142, 1987. doi: 10.2307/1269768. (Cited on pages 12, 30, 151, and 196.)

[53] B. B. Bederson. The Promise of Zoomable User Interfaces. *Behaviour & Information Technology*, 30(6):853–866, 2011. doi: 10.1080/0144929X.2011.586724. (Cited on pages 31, 39, 155, 156, and 185.)

[54] B. A. Bell and S. K. Feiner. Dynamic Space Management for User Interfaces. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 239–248. ACM Press, 2000. doi: 10.1145/354401.354790. (Cited on page 181.)

[55] L. D. Bergman, B. E. Rogowitz, and L. Treinish. A Rule-Based Tool for Assisting Colormap Selection. In *Proceedings of the IEEE Visualization Conference (Vis)*, pages 118–125. IEEE Computer Society, 1995. doi: 10.1109/VISUAL.1995.480803. (Cited on page 128.)

[56] J. Bertin. *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, 1983. (Cited on pages 4, 21, 125, and 128.)

[57] E. A. Bier. *Snap-Dragging: Interactive Geometric Design in Two and Three Dimensions*. PhD thesis, University of California, Berkeley, 1988. URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/1988/5858.html`. (Cited on page 158.)

[58] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose. Toolglass and Magic Lenses: The See-Through Interface. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 73–80. ACM Press, 1993. doi: 10.1145/166117.166126. (Cited on pages 35, 100, 178, 194, and 196.)

[59] E. A. Bier, M. C. Stone, K. Fishkin, W. Buxton, and T. Baudel. A Taxonomy of See-Through Tools. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 358–364. ACM Press, 1994. doi: 10.1145/191666.191786. (Cited on pages 158 and 179.)

[60] R. Blanch and E. Lecolinet. Browsing Zoomable Treemaps: Structure-Aware Multi-Scale Navigation Techniques. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1248–1253, 2007. doi: 10.1109/TVCG.2007.70540. (Cited on page 72.)

[61] J. Booker, T. Buennemeyer, A. J. Sabri, and C. North. High-Resolution Displays Enhancing Geo-Temporal Data Visualizations. In *Proceedings of the ACM Southeast Regional Conference*, pages 443–448. ACM Press, 2007. doi: 10.1145/1233341.1233421. (Cited on page 218.)

[62] D. Borland and R. Taylor. Rainbow Color Map (Still) Considered Harmful. *Computer Graphics and Applications*, 27(2):14–17, 2007. doi: 10.1109/MCG.2007.46. (Cited on page 94.)

[63] P. Bottoni and G. Costagliola. On the Definition of Visual Languages and Their Editors. In *Proceedings of the International Conference on the Theory and Application of Diagrams*, pages 305–319. Springer, 2002. doi: 10.1007/3-540-46037-3_29. (Cited on page 242.)

[64] U. Brandes and D. Wagner. A Bayesian Paradigm for Dynamic Graph Layout. In *Proceedings of the International Symposium on Graph Drawing (GD)*, pages 236–247. Springer, 1997. doi: 10.1007/3-540-63938-1_66. (Cited on page 176.)

[65] M. Bruls, K. Huizing, and J. J. van Wijk. Squarified Treemaps. In *Proceedings of the Joint Eurographics - IEEE TCVG Symposium on Visualization (VisSym)*, pages 33–42. Eurographics Association, 2000. URL `http://diglib.eg.org/EG/DL/WS/VisSym/VisSym00/033-042.pdf`. (Cited on pages 91 and 220.)

[66] C. Buchheim, M. Jünger, and S. Leipert. Improving Walker's Algorithm to Run in Linear Time. In *Proceedings of the International Symposium on Graph Drawing (GD)*, pages 344–353. Springer, 2002. doi: 10.1007/3-540-36151-0_32. (Cited on page 93.)

[67] A. Buja, J. A. McDonald, J. Michalak, and W. Stuetzle. Interactive Data Visualization using Focusing and Linking. In *Proceedings of the IEEE Visualization Conference (Vis)*, pages 156–163, 419. IEEE Computer Society, 1991. doi: 10.1109/VISUAL.1991.175794. (Cited on pages 12 and 30.)

[68] P. Buono, A. Aris, C. Plaisant, A. Khella, and B. Shneiderman. Interactive Pattern Search in Time Series. In *Proceedings of the Conference on Visualization and Data Analysis (VDA)*, pages 175–186. SPIE, 2005. doi: 10.1117/12.587537. (Cited on page 242.)

[69] S. P. Callahan, L. Bavoil, V. Pascucci, and C. T. Silva. Progressive Volume Rendering of Large Unstructured Grids. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1307–1314, 2006. doi: 10.1109/TVCG.2006.171. (Cited on page 66.)

[70] S. Card, J. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, 1999. (Cited on pages 3, 4, 10, 277, and 278.)

[71] S. K. Card, B. Suh, B. A. Pendleton, B. Heer, and J. W. Bodnar. Time Tree: Exploring Time Changing Hierarchies. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 3–10. IEEE Computer Society, 2006. doi: 10.1109/VAST.2006.261450. (Cited on page 176.)

[72] T. Carlstein, D. Parkes, and N. Thrift, editors. *Human Activity and Time Geography*. Edward Arnold Publishing, 1978. (Cited on page 252.)

[73] A. Cedilnik, B. Geveci, K. Moreland, J. Ahrens, and J. Favre. Remote Large Data Visualization in the ParaView Framework. In *Proceedings of the Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)*, pages 163–170. Eurographics Association, 2006. doi: 10.2312/EGPGV/EGPGV06/163-170. (Cited on page 66.)

[74] S.-M. Chan, L. Xiao, J. Gerth, and P. Hanrahan. Maintaining Interactivity While Exploring Massive Time Series. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 59–66. IEEE Computer Society, 2008. doi: 10.1109/VAST.2008.4677357. (Cited on page 64.)

[75] O. Chapuis and N. Roussel. Copy-and-Paste Between Overlapping Windows. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 201–210. ACM Press, 2007. doi: 10.1145/1240624.1240657. (Cited on page 159.)

[76] H. Chen. Compound Brushing Explained. *Information Visualization*, 3(2):96–108, 2004. doi: 10.1057/palgrave.ivs.9500068. (Cited on pages 12, 30, and 242.)

[77] Q. Chen, J. Grundy, and J. Hosking. An E-Whiteboard Application to Support Early Design-Stage Sketching of UML Diagrams. In *Proceedings of the IEEE Symposium on Human Centric Computing Languages and Environments (HCC)*, pages 219–226. IEEE Computer Society, 2003. doi: 10.1109/HCC.2003.1260232. (Cited on page 175.)

[78] E. H. Chi. A Taxonomy of Visualization Techniques Using the Data State Reference Model. In *Proceedings of the IEEE Symposium Information Visualization (InfoVis)*, pages 69–75. IEEE Computer Society, 2000. doi: 10.1109/INFVIS.2000.885092. (Cited on pages 4, 68, and 74.)

[79] L. Chittaro and C. Combi. Visualizing Queries on Databases of Temporal Histories: New Metaphors and their Evaluation. *Data & Knowledge Engineering*, 44(2):239–264, 2003. doi: 10.1016/S0169-023X(02)00137-4. (Cited on page 242.)

[80] L. Chittaro, C. Combi, and G. Trapasso. Data Mining on Temporal Data: A Visual Approach and its Clinical Application to Hemodialysis. *Journal of Visual Languages and Computing*, 14(6):591–620, 2003. doi: 10.1016/j.jvlc.2003.06.003. (Cited on page 231.)

[81] J. H. T. Claessen and J. J. van Wijk. Flexible Linked Axes for Multivariate Data Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2310–2316, 2011. doi: 10.1109/TVCG.2011.201. (Cited on page 158.)

[82] W. S. Cleveland and R. McGill. An Experiment in Graphical Perception. *International Journal of Man-Machine Studies*, 25(5): 491–501, 1986. doi: 10.1016/S0020-7373(86)80019-0. (Cited on page 21.)

[83] A. Cockburn and J. Savage. Comparing Speed-Dependent Automatic Zooming with Traditional Scroll, Pan and Zoom Methods. In *In People and Computers XVII – Designing for Society*, pages 87–102. Springer, 2004. doi: 10.1007/978-1-4471-3754-2_6. (Cited on pages 102 and 196.)

[84] A. Cockburn, A. Karlson, and B. B. Bederson. A Review of Overview+Detail, Zooming, and Focus+Context Interfaces. *ACM Computing Surveys*, 41(1):2:1–2:31, 2008. doi: 10.1145/1456650.1456652. (Cited on pages 216 and 217.)

[85] S. Conversy. Improving Usability of Interactive Graphics Specification and Implementation with Picking Views and Inverse Transformation. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 153–160. IEEE Computer Society, 2011. doi: 10.1109/VLHCC.2011.6070392. (Cited on page 17.)

[86] S. Conversy. Contributions to the Science of Controlled Transformation. Habilitation à Diriger des Recherches de l'Université Paul Sabatier - Toulouse III, 2013. URL `http://tel.archives-ouvertes.fr/tel-00853192`. (Cited on page 27.)

[87] A. Cooper, R. Reimann, and D. Cronin. *About Face 3: The Essentials of Interaction Design*. Wiley, 2007. (Cited on pages 9, 15, and 57.)

[88] J. A. Cottam, A. Lumsdaine, and P. Wang. Abstract Rendering: Out-of-core Rendering for Information Visualization. In *Proceedings of the Conference on Visualization and Data Analysis (VDA)*, pages 90170K–1–90170K–13. SPIE, 2014. doi: 10.1117/12.2041200. (Cited on page 29.)

[89] T. Crnovrsanin, C. Muelder, C. D. Correa, and K.-L. Ma. Proximity-Based Visualization of Movement Trace Data. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 11–18. IEEE Computer Society, 2009. doi: 10.1109/VAST.2009.5332593. (Cited on page 126.)

[90] T. Crnovrsanin, I. Liao, Y. Wu, and K.-L. Ma. Visual Recommendations for Network Navigation. *Computer Graphics Forum*, 30(3):1081–1090, 2011. doi: 10.1111/j.1467-8659.2011.01957.x. (Cited on page 260.)

[91] R. Dachselt and R. Buchholz. Natural Throw and Tilt Interaction between Mobile Phones and Distant Displays. In *CHI Extended Abstracts on Human Factors in Computing Systems*, pages 3253–3258. ACM Press, 2009. doi: 10.1145/1520340.1520467. (Cited on page 201.)

[92] T. N. Dang, L. Wilkinson, and A. Anand. Stacking Graphic Elements to Avoid Over-Plotting. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1044–1052, 2010. doi: 10.1109/TVCG.2010.197. (Cited on page 127.)

[93] W. de Nooy, A. Mrvar, and V. Batagelj. *Exploratory Social Network Analysis with Pajek*. Cambridge University Press, 2005. (Cited on page 109.)

[94] C. Demetrescu, I. Finocchi, and J. T. Stasko. Specifying Algorithm Visualizations: Interesting Events or State Mapping? In S. Diehl, editor, *Software Visualization*, pages 16–30. Springer, 2002. doi: 10.1007/3-540-45875-1_2. (Cited on page 232.)

[95] M. Derthick and S. F. Roth. Enhancing Data Exploration With a Branching History of User Operations. *Knowledge-Based Systems*, 14(1-2):65–74, 2001. doi: 10.1016/S0950-7051(00)00101-5. (Cited on page 108.)

[96] R. Diestel. *Graph Theory*. Springer, 2005. (Cited on page 84.)

[97] A. Dix, J. Finlay, G. D. Abowd, and R. Beale. *Human-Computer Interaction*. Pearson Education, 3rd edition, 2004. (Cited on page 9.)

[98] H. Doleisch. SIMVIS: interactive visual analysis of large and time-dependent 3D simulation data. In *Proceedings of the Winter Simulation Conference*, pages 712–720. WSC, 2007. doi: 10.1145/1351542.1351674. (Cited on pages 29 and 76.)

[99] H. Doleisch, M. Gasser, and H. Hauser. Interactive Feature Specification for Focus+Context Visualization of Complex Simulation Data. In *Proceedings of the Joint Eurographics - IEEE TCVG*

Symposium on Visualization (VisSym), pages 239–248. Eurographics Association, 2003. URL http://diglib.eg.org/EG/DL/WS/VisSym/VisSym03/239-248.pdf. (Cited on pages 63, 80, 81, 98, and 242.)

[100] S. dos Santos and K. Brodlie. Gaining Understanding of Multivariate and Multidimensional Data through Visualization. *Computers & Graphics*, 28:311–325, 2004. doi: 10.1016/j.cag.2004.03.013. (Cited on pages 4, 277, and 278.)

[101] P. R. Doshi, G. E. Rosario, E. A. Rundensteiner, and M. O. Ward. A Strategy Selection Framework for Adaptive Prefetching in Data Visualization. In *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*, pages 107–116. IEEE Computer Society, 2003. doi: 10.1109/SSDM.2003.1214972. (Cited on page 66.)

[102] P. Dragicevic. Combining Crossing-Based and Paper-Based Interaction Paradigms for Dragging and Dropping Between Overlapping Windows. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 193–196. ACM Press, 2004. doi: 10.1145/1029632.1029667. (Cited on page 159.)

[103] T. Dwyer, K. Marriott, F. Schreiber, P. Stuckey, M. Woodward, and M. Wybrow. Exploration of Networks Using Overview+Detail with Constraint-Based Cooperative Layout. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1293–1300, 2008. doi: 10.1109/TVCG.2008.130. (Cited on page 176.)

[104] A. Ebert, G. C. van der Veer, G. Domik, N. D. Gershon, and I. Scheler, editors. *Building Bridges: HCI, Visualization, and Non-formal Modeling*. Springer, 2014. doi: 10.1007/978-3-642-54894-9. (Cited on page 13.)

[105] G. Ellis and A. Dix. An Explorative Analysis of User Evaluation Studies in Information Visualisation. In *Proceedings of the Workshop on Beyond Time and Errors: Novel Evaluation Methods for Information Visualization (BELIV)*, pages 1–7. ACM Press, 2006. doi: 10.1145/1168149.1168152. (Cited on page 112.)

[106] G. Ellis and A. Dix. Enabling Automatic Clutter Reduction in Parallel Coordinate Plots. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):717–724, 2006. doi: 10.1109/TVCG.2006.138. (Cited on pages 64, 196, and 206.)

[107] N. Elmqvist and J.-D. Fekete. Hierarchical Aggregation for Information Visualization: Overview, Techniques, and Design Guidelines. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):439–454, 2010. doi: 10.1109/TVCG.2009.84. (Cited on pages 196, 217, 219, and 259.)

[108] N. Elmqvist and P. Tsigas. A Taxonomy of 3D Occlusion Management for Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1095–1109, 2008. doi: 10.1109/TVCG.2008.59. (Cited on page 133.)

[109] N. Elmqvist, P. Dragicevic, and J.-D. Fekete. Rolling the Dice: Multidimensional Visual Exploration using Scatterplot Matrix Navigation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1539–1148, 2008. doi: 10.1109/TVCG.2008.153. (Cited on pages 151 and 196.)

[110] N. Elmqvist, Y. Riche, N. H. Riche, and J.-D. Fekete. Melange: Space Folding for Visual Exploration. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):468–483, 2010. doi: 10.1109/TVCG.2009.86. (Cited on page 196.)

[111] N. Elmqvist, P. Dragicevic, and J.-D. Fekete. Color Lens: Adaptive Color Scale Optimization for Visual Exploration. *IEEE Transactions on Visualization and Computer Graphics*, 17(6):795–807, 2011. doi: 10.1109/TVCG.2010.94. (Cited on page 151.)

[112] N. Elmqvist, A. V. Moere, H.-C. Jetter, D. Cernea, H. Reiterer, and T. Jankun-Kelly. Fluid Interaction for Information Visualization. *Information Visualization*, 10(4):327–340, 2011. doi: 10.1177/1473871611413180. (Cited on pages 4, 11, 13, 24, and 154.)

[113] A. Endert, L. Bradel, J. Zeitz, C. Andrews, and C. North. Designing Large High-Resolution Display Workspaces. In *Proceedings of the Conference on Advanced Visual Interfaces (AVI)*, pages 58–65. ACM Press, 2012. doi: 10.1145/2254556.2254570. (Cited on page 50.)

[114] R. F. Erbacher, K. L. Walker, and D. A. Frincke. Intrusion and Misuse Detection in Large-Scale Systems. *Computer Graphics and Applications*, 22(1):38–48, 2002. doi: 10.1109/38.974517. (Cited on page 231.)

[115] M. Erwig and M. Schneider. Spatio-Temporal Predicates. *IEEE Transactions on Knowledge and Data Engineering*, 14(4):881–901, 2002. doi: 10.1109/TKDE.2002.1019220. (Cited on page 238.)

[116] G. Faconti and M. Massink. Continuous Interaction with Computers: Issues and Requirements. In C. Stephanidis, editor, *Universal Access In HCI: Towards an Information Society for All*, volume 3, pages 301–305. Lawrence Erlbaum, 2001. (Cited on page 62.)

[117] J.-D. Fekete. The InfoVis Toolkit. In *Proceedings of the IEEE Symposium Information Visualization (InfoVis)*, pages 167–174. IEEE

Computer Society, 2004. doi: 10.1109/INFVIS.2004.64. (Cited on pages 64, 65, 66, and 108.)

[118] J.-D. Fekete. Advanced interaction for Information Visualization. In *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis)*, page xi. IEEE Computer Society, 2010. doi: 10.1109/PACIFICVIS.2010.5429617. (Cited on pages 4 and 151.)

[119] J.-D. Fekete and C. Plaisant. Interactive Information Visualization of a Million Items. In *Proceedings of the IEEE Symposium Information Visualization (InfoVis)*, pages 117–124. IEEE Computer Society, 2002. doi: 10.1109/INFVIS.2002.1173156. (Cited on page 64.)

[120] C. Fellbaum, editor. *WordNet – An Electronic Lexical Database*. MIT Press, 1998. (Cited on page 118.)

[121] K.-C. Feng, C. Wang, H.-W. Shen, and T.-Y. Lee. Coherent Time-Varying Graph Drawing with Multifocus+Context Interaction. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1330–1342, 2012. doi: 10.1109/TVCG.2011.128. (Cited on page 176.)

[122] W. Fikkert, M. D'Ambros, T. Bierz, and T. Jankun-Kelly. Interacting with Visualizations. In A. Kerren, A. Ebert, and J. Meyer, editors, *Human-Centered Visualization Environments*, volume 4417 of *Lecture Notes in Computer Science*, pages 77–162. Springer, 2007. doi: 10.1007/978-3-540-71949-6_3. (Cited on page 14.)

[123] G. W. Fitzmaurice. *Graspable User Interfaces*. PhD thesis, University of Toronto, 1996. (Cited on page 202.)

[124] C. Forlines and R. H. Lilien. Adapting a Single-User, Single-Display Molecular Visualization Application for Use in a Multi-User, Multi-Display Environment. In *Proceedings of the Conference on Advanced Visual Interfaces (AVI)*, pages 367–371. ACM Press, 2008. doi: 10.1145/1385569.1385635. (Cited on page 23.)

[125] M. Frisch. *Interaction and Visualization Techniques for Node-Link Diagram Editing and Exploration*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, 2012. (Cited on page 43.)

[126] M. Frisch and R. Dachselt. Visualizing Offscreen Elements of Node-Link Diagrams. *Information Visualization*, 12(2):133–162, 2013. doi: 10.1177/1473871612473589. (Cited on pages 33, 260, and 265.)

[127] M. Frisch, J. Heydekorn, and R. Dachselt. Investigating Multi-Touch and Pen Gestures for Diagram Editing on Interactive Surfaces. In *Proceedings of the International Conference on Interactive*

Tabletops and Surfaces (ITS), pages 149–156. ACM Press, 2009. doi: 10.1145/1731903.1731933. (Cited on pages 176 and 185.)

[128] Y. Frishman and A. Tal. Dynamic Drawing of Clustered Graphs. In *Proceedings of the IEEE Symposium Information Visualization (InfoVis)*, pages 191–198. IEEE Computer Society, 2004. doi: 10.1109/INFOVIS.2004.18. (Cited on page 176.)

[129] B. Fry. *Visualizing Data: Exploring and Explaining Data with the Processing Environment*. O'Reilly, 2008. (Cited on page 66.)

[130] G. W. Furnas. Generalized Fisheye Views. *ACM SIGCHI Bulletin*, 17(4):16–23, 1986. doi: 10.1145/22339.22342. (Cited on page 263.)

[131] G. W. Furnas. Effective View Navigation. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 367–374. ACM Press, 1997. doi: 10.1145/258549.258800. (Cited on page 54.)

[132] K. Z. Gajos, M. Czerwinski, D. S. Tan, and D. S. Weld. Exploring the Design Space for Adaptive Graphical User Interfaces. In *Proceedings of the Conference on Advanced Visual Interfaces (AVI)*, pages 201–208. ACM Press, 2006. doi: 10.1145/1133265.1133306. (Cited on page 106.)

[133] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Pattern Elements of Reusable Object-Oriented Software*. Addison Wesley, 1994. (Cited on pages 88 and 89.)

[134] P. Gatalsky, N. Andrienko, and G. Andrienko. Interactive Analysis of Event Data Using Space-Time Cube. In *Proceedings of the International Conference Information Visualisation (IV)*, pages 145–152. IEEE Computer Society, 2004. doi: 10.1109/IV.2004.1320137. (Cited on page 232.)

[135] S. Ghani, N. H. Riche, and N. Elmqvist. Dynamic Insets for Context-Aware Graph Navigation. *Computer Graphics Forum*, 30 (3):861–870, 2011. doi: 10.1111/j.1467-8659.2011.01935.x. (Cited on page 33.)

[136] F. Giannotti and D. Pedreschi, editors. *Mobility, Data Mining and Privacy – Geographic Knowledge Discovery*. Springer, 2008. doi: 10.1007/978-3-540-75177-9. (Cited on page 125.)

[137] S. Gladisch, H. Schumann, and C. Tominski. Navigation Recommendations for Exploring Hierarchical Graphs. In G. Bebis, R. Boyle, B. Parvin, D. Koracin, B. Li, F. Porikli, V. Zordan, J. Klosowski, S. Coquillart, X. Luo, M. Chen, and D. Gotz, editors, *Advances in Visual Computing*, pages 36–47. Springer, 2013. doi: 10.1007/978-3-642-41939-3_4. (Cited on page 257.)

[138] S. Gladisch, H. Schumann, M. Ernst, G. Füllen, and C. Tominski. Semi-Automatic Editing of Graphs with Customized Layouts. *Computer Graphics Forum*, 33(3):381–390, 2014. doi: 10.1111/cgf.12394. (Cited on page 173.)

[139] M. Gleicher. Image Snapping. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 183–190. ACM Press, 1995. doi: 10.1145/218380.218441. (Cited on page 158.)

[140] M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts. Visual Comparison for Information Visualization. *Information Visualization*, 10(4):289–309, 2011. doi: 10.1177/1473871611416549. (Cited on pages 38, 148, 151, 157, and 170.)

[141] D. Gotz and Z. Wen. Behavior-driven Visualization Recommendation. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*, pages 315–324. ACM Press, 2009. doi: 10.1145/1502650.1502695. (Cited on page 54.)

[142] J. D. Gould and C. Lewis. Designing for Usability: Key Principles and What Designers Think. *Communications of the ACM*, 28 (3):300–311, 1985. doi: 10.1145/3166.3170. (Cited on page 112.)

[143] P. Grew. Steering Wheel or Driving Wheel: Which Way Is Up? In *Proceedings of the IASTED International Conference Human-Computer Interaction*, pages 164–169. ACTA Press, 2008. (Cited on pages 113 and 170.)

[144] E. Grundy, M. W. Jones, R. S. Laramee, R. P. Wilson, and E. L. C. Shepard. Visualisation of Sensor Data from Animal Movement. *Computer Graphics Forum*, 28(3):815–822, 2009. doi: 10.1111/j.1467-8659.2009.01469.x. (Cited on page 34.)

[145] J. Gudmundsson, P. Laube, and T. Wolle. Computational Movement Analysis. In W. Kresse and D. M. Danko, editors, *Springer Handbook of Geographic Information*, pages 423–438. Springer, 2012. doi: 10.1007/978-3-540-72680-7_22. (Cited on page 125.)

[146] S. Gustafson, P. Baudisch, C. Gutwin, and P. Irani. Wedge: Clutter-Free Visualization of Off-Screen Locations. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 787–796. ACM Press, 2008. doi: 10.1145/1357054.1357179. (Cited on pages 105, 260, and 265.)

[147] R. B. Haber and D. A. McNabb. Visualization Idioms: A Conceptual Model for Scientific Visualization Systems. In G. M. Nielson, B. D. Shriver, and L. J. Rosenblum, editors, *Visualization in Scientific Computing*, pages 74–93. IEEE Computer Society, 1990. (Cited on pages 4, 277, and 278.)

[148] H. Hagh-Shenas, S. Kim, V. Interrante, and C. Healey. Weaving Versus Blending: A Quantitative Assessment of the Information Carrying Capacities of Two Alternative Methods for Conveying Multivariate Data with Color. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1270–1277, 2007. doi: 10.1109/TVCG.2007.70623. (Cited on page 159.)

[149] E. Hajnicz. *Time Structures: Formal Description and Algorithmic Representation*. Springer, 1996. doi: 10.1007/3-540-60941-5. (Cited on page 238.)

[150] T. Hakala, J. Lehikoinen, and A. Aaltonen. Spatial Interactive Visualization on Small Screen. In *Proceedings of the International Conference on Human Computer Interaction with Mobile Devices & Services (MobileCHI)*, pages 137–144. ACM Press, 2005. doi: 10.1145/1085777.1085800. (Cited on page 23.)

[151] M. Harrower and S. Benjamin. Designing Better Map Interfaces: A Framework for Panning and Zooming. *Transactions in GIS*, 9 (2):77–89, 2005. doi: 10.1111/j.1467-9671.2005.00207.x. (Cited on page 101.)

[152] M. A. Harrower and C. A. Brewer. ColorBrewer.org: An Online Tool for Selecting Color Schemes for Maps. *The Cartographic Journal*, 40(1):27–37, 2003. doi: 10.1179/000870403235002042. (Cited on pages 128 and 220.)

[153] M. Hascoët and P. Dragicevic. Interactive Graph Matching and Visual Comparison of Graphs and Clustered Graphs. In *Proceedings of the Conference on Advanced Visual Interfaces (AVI)*, pages 522–529. ACM Press, 2012. doi: 10.1145/2254556.2254654. (Cited on page 33.)

[154] M. Hassenzahl and N. Tractinsky. User Experience – A Research Agenda. *Behaviour Information Technology*, 25(2):91–97, 2006. doi: 10.1080/01449290500330331. (Cited on pages 9 and 24.)

[155] H. Hauser, F. Ledermann, and H. Doleisch. Angular Brushing of Extended Parallel Coordinates. In *Proceedings of the IEEE Symposium Information Visualization (InfoVis)*, pages 127–130. IEEE Computer Society, 2002. doi: 10.1109/INFVIS.2002.1173157. (Cited on pages 12, 30, 151, and 196.)

[156] C. G. Healey, K. S. Booth, and J. T. Enns. High-Speed Visual Estimation Using Preattentive Processing. *ACM Transactions on Computer-Human Interaction*, 3(2):107–135, 1996. doi: 10.1145/230562.230563. (Cited on page 247.)

[157] J. Heer and M. Agrawala. Software Design Patterns for Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):853–860, 2006. doi: 10.1109/TVCG.2006.178. (Cited on pages 66 and 88.)

[158] J. Heer and M. Bostock. Crowdsourcing Graphical Perception: Using Mechanical Turk to Assess Visualization Design. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 203–212. ACM Press, 2010. doi: 10.1145/1753326.1753357. (Cited on page 22.)

[159] J. Heer and D. Boyd. Vizster: Visualizing Online Social Networks. In *Proceedings of the IEEE Symposium Information Visualization (InfoVis)*, pages 32–39. IEEE Computer Society, 2005. doi: 10.1109/INFOVIS.2005.39. (Cited on page 108.)

[160] J. Heer and G. Robertson. Animated Transitions in Statistical Data Graphics. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1240–1247, 2007. doi: 10.1109/TVCG.2007.70539. (Cited on pages 17 and 103.)

[161] J. Heer and B. Shneiderman. Interactive Dynamics for Visual Analysis. *Communications of the ACM*, 55(4):45–54, 2012. doi: 10.1145/2133806.2133821. (Cited on pages 11, 15, and 28.)

[162] J. Heer, S. K. Card, and J. A. Landay. prefuse: A Toolkit for Interactive Information Visualization. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 421–430. ACM Press, 2005. doi: 10.1145/1054972.1055031. (Cited on pages 66 and 108.)

[163] J. Heer, N. Kong, and M. Agrawala. Sizing the Horizon: The Effects of Chart Size and Layering on the Graphical Perception of Time Series Visualizations. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 1303–1312. ACM Press, 2009. doi: 10.1145/1518701.1518897. (Cited on page 129.)

[164] K. Henriksen, J. Sporring, and K. Hornbæk. Virtual Trackballs Revisited. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):206–216, 2004. doi: 10.1109/TVCG.2004.1260772. (Cited on page 196.)

[165] N. Henry, J.-D. Fekete, and M. J. McGuffin. NodeTrix: a Hybrid Visualization of Social Networks. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1302–1309, 2007. doi: 10.1109/TVCG.2007.70582. (Cited on page 166.)

[166] N. Henry Riche, T. Dwyer, B. Lee, and S. Carpendale. Exploring the Design Space of Interactive Link Curvature in Network Diagrams. In *Proceedings of the Conference on Advanced*

Visual Interfaces (AVI), pages 506–513. ACM Press, 2012. doi: 10.1145/2254556.2254652. (Cited on page 175.)

[167] M. Herlihy and N. Shavit. *The Art of Multiprocessor Programming*. Morgan Kaufmann, 2008. (Cited on page 65.)

[168] I. Herman, G. Melançon, and M. S. Marshall. Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000. doi: 10.1109/2945.841119. (Cited on pages 84, 87, 217, 219, 258, and 259.)

[169] H. Hochheiser. *Interactive Graphical Querying of Time Series and Linear Sequence Data Sets*. PhD thesis, University of Maryland, 2003. URL `http://hcil2.cs.umd.edu/trs/2003-20/2003-20.pdf`. (Cited on page 242.)

[170] H. Hochheiser and B. Shneiderman. Dynamic Query Tools for Time Series Data Sets: Timebox Widgets for Interactive Exploration. *Information Visualization*, 3(1):1–18, 2004. doi: 10.1057/palgrave.ivs.9500061. (Cited on pages 22 and 196.)

[171] D. Holman, R. Vertegaal, M. Altosaar, N. F. Troje, and D. Johns. Paper Windows: Interaction Techniques for Digital Paper. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 591–599. ACM Press, 2005. doi: 10.1145/1054972.1055054. (Cited on pages 197 and 201.)

[172] D. Holten and J. J. van Wijk. Visual Comparison of Hierarchically Organized Data. *Computer Graphics Forum*, 27(3):759–766, 2008. doi: 10.1111/j.1467-8659.2008.01205.x. (Cited on page 151.)

[173] C. Holz and S. Feiner. Relaxed Selection Techniques for Querying Time-Series Graphs. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 213–222. ACM Press, 2009. doi: 10.1145/1622176.1622217. (Cited on page 22.)

[174] M. L. Huang, P. Eades, and J. Wang. On-line Animated Visualization of Huge Graphs Using a Modified Spring Algorithm. *Journal of Visual Languages and Computing*, 9(6):623–645, 1998. doi: 10.1006/jvlc.1998.0094. (Cited on page 267.)

[175] W. Huang, editor. *Handbook of Human Centric Visualization*. Springer, 2013. doi: 10.1007/978-1-4614-7485-2. (Cited on page 24.)

[176] C. Hurter, B. Tissoires, and S. Conversy. FromDaDy: Spreading Aircraft Trajectories Across Views to Support Iterative Queries.

IEEE Transactions on Visualization and Computer Graphics, 15 (6):1017–1024, 2009. doi: 10.1109/TVCG.2009.145. (Cited on page 34.)

[177] C. Hurter, A. Telea, and O. Ersoy. MoleView: An Attribute and Structure-Based Semantic Lens for Large Element-Based Plots. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2600–2609, 2011. doi: 10.1109/TVCG.2011.223. (Cited on page 151.)

[178] J.-F. Im, F. G. Villegas, and M. J. McGuffin. VisReduce: Fast and Responsive Incremental Information Visualization of Large Datasets. In *Proceedings of the IEEE International Conference on Big Data*, pages 25–32. IEEE Computer Society, 2013. doi: 10.1109/BigData.2013.6691710. (Cited on page 29.)

[179] P. Isenberg and M. S. T. Carpendale. Interactive Tree Comparison for Co-located Collaborative Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6): 1232–1239, 2007. doi: 10.1109/TVCG.2007.70568. (Cited on pages 152, 157, 194, 197, and 207.)

[180] P. Isenberg and D. Fisher. Collaborative Brushing and Linking for Co-located Visual Analytics of Document Collections. *Computer Graphics Forum*, 28(3):1031–1038, 2009. doi: 10.1111/j.1467-8659.2009.01444.x. (Cited on pages 194 and 197.)

[181] P. Isenberg, A. Tang, and S. Carpendale. An Exploratory Study of Visual Information Analysis. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 1217–1226. ACM Press, 2008. doi: 10.1145/1357054.1357245. (Cited on pages 148 and 156.)

[182] P. Isenberg, S. Carpendale, T. Hesselmann, T. Isenberg, and B. Lee, editors. *Proceedings of the Workshop on Data Exploration for Interactive Surfaces (DEXIS) at the ACM International Conference on Interactive Tabletops and Surfaces (ITS)*, 2011. URL `http://hal.inria.fr/hal-00659469`. (Cited on pages 45 and 152.)

[183] P. Isenberg, T. Isenberg, T. Hesselmann, B. Lee, U. von Zadow, and A. Tang. Data Visualization on Interactive Surfaces: A Research Agenda. *Computer Graphics and Applications*, 33(2):16–24, 2013. doi: 10.1109/MCG.2013.24. (Cited on pages 44, 50, 274, and 277.)

[184] H. Ishii and B. Ullmer. Tangible Bits: Towards Seamless Interfaces Between People, Bits and Atoms. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 234–241. ACM Press, 1997. doi: 10.1145/258549.258715. (Cited on pages 10 and 152.)

[185] ISO. Ergonomics of Human-System Interaction – Usability Methods Supporting Human-Centred Design. ISO/TR 16982:2002, 2002. (Cited on page 9.)

[186] S. Izadi, S. Hodges, S. Taylor, D. Rosenfeld, N. Villar, A. Butler, and J. Westhues. Going Beyond the Display: A Surface Technology with an Electronically Switchable Diffuser. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 269–278. ACM Press, 2008. doi: 10.1145/1449715.1449760. (Cited on page 197.)

[187] R. J. Jacob, A. Girouard, L. M. Hirshfield, M. S. Horn, O. Shaer, E. T. Solovey, and J. Zigelbaum. Reality-Based Interaction: A Framework for Post-WIMP Interfaces. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 201–210. ACM Press, 2008. doi: 10.1145/1357054.1357089. (Cited on pages 10 and 152.)

[188] T. Jankun-Kelly, K.-L. Ma, and M. Gertz. A Model and Framework for Visualization Exploration. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):357–369, 2007. doi: 10.1109/TVCG.2007.28. (Cited on pages 16, 28, 89, 108, and 278.)

[189] T. J. Jankun-Kelly, K.-L. Ma, and M. Gertz. A Model for the Visualization Exploration Process. In *Proceedings of the IEEE Visualization Conference (Vis)*, pages 323–330. IEEE Computer Society, 2002. doi: 10.1109/VISUAL.2002.1183791. (Cited on page 196.)

[190] Y. Jansen and P. Dragicevic. An Interaction Model for Visualizations Beyond The Desktop. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2396–2405, 2013. doi: 10.1109/TVCG.2013.134. (Cited on pages 50 and 277.)

[191] G. F. Jenks and F. C. Caspall. Error on Choroplethic Maps: Definition, Measurement, Reduction. *Annals of the Association of American Geographers*, 61(2):217–244, 1971. URL `http://www.jstor.org/stable/2562442`. (Cited on page 128.)

[192] H. Jiang, D. Wigdor, C. Forlines, M. Borkin, J. Kauffmann, and C. Shen. LivOlay: Interactive Ad-Hoc Registration and Overlapping of Applications for Collaborative Visual Exploration. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 1357–1360. ACM Press, 2008. doi: 10.1145/1357054.1357266. (Cited on page 151.)

[193] M. John, C. Tominski, and H. Schumann. Visual and Analytical Extensions for the Table Lens. In *Proceedings of the Conference on Visualization and Data Analysis (VDA)*, pages 680907–1–680907–12. SPIE, 2008. doi: 10.1117/12.766440. (Cited on page 149.)

[194] C. R. Johnson, S. G. Parker, C. D. Hansen, G. L. Kindlmann, and Y. Livnat. Interactive Simulation and Visualization. *IEEE Computer*, 32(12):59–65, 1999. doi: 10.1109/2.809252. (Cited on page 66.)

[195] I. Jusufi, C. Klukas, A. Kerren, and F. Schreiber. Guiding the Interactive Exploration of Metabolic Pathway Interconnections. *Information Visualization*, 11(2):136–150, 2012. doi: 10.1177/1473871611405677. (Cited on page 260.)

[196] Y. Kakehi and T. Naemura. UlteriorScape: Interactive Optical Superimposition on a View-dependent Tabletop Display. In *Proceedings of the International Workshop on Tabletops and Interactive Surfaces (Tabletop)*, pages 189–192. IEEE Computer Society, 2008. (Cited on page 197.)

[197] S. Kandel, J. Heer, C. Plaisant, J. Kennedy, F. van Ham, N. H. Riche, C. Weaver, B. Lee, D. Brodbeck, and P. Buono. Research Directions in Data Wrangling: Visualizations and Transformations for Usable and Credible Data. *Information Visualization*, 10(4):271–288, 2011. doi: 10.1177/1473871611415994. (Cited on pages 23, 41, and 174.)

[198] T. Kapler and W. Wright. GeoTime Information Visualization. *Information Visualization*, 4(2):136–146, 2005. doi: 10.1057/palgrave.ivs.9500097. (Cited on page 126.)

[199] D. F. Keefe. Integrating Visualization and Interaction Research to Improve Scientific Workflows. *Computer Graphics and Applications*, 30(2):8–13, 2010. doi: 10.1109/MCG.2010.30. (Cited on pages 4 and 14.)

[200] D. F. Keefe and T. Isenberg. Reimagining the Scientific Visualization Interaction Paradigm. *IEEE Computer*, 46(5):51–57, 2013. doi: 10.1109/MC.2013.178. (Cited on pages 15 and 25.)

[201] D. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann, editors. *Mastering the Information Age – Solving Problems with Visual Analytics*. Eurographics Association, 2010. (Cited on page 27.)

[202] D. A. Keim, J. Schneidewind, and M. Sips. Scalable Pixel Based Visual Data Exploration. In P. P. Lévy, B. Le Grand, F. Poulet, M. Soto, L. Darago, L. Toubiana, and J.-F. Vibert, editors, *Pixelization Paradigm*, pages 12–24. Springer, 2007. doi: 10.1007/978-3-540-71027-1_2. (Cited on page 218.)

[203] T. Kelder, M. P. van Iersel, K. Hanspers, M. Kutmon, B. R. Conklin, C. T. A. Evelo, and A. R. Pico. WikiPathways: Building Research Communities on Biological Pathways. *Nucleic Acids Research*, 40(D1):1301–1307, 2012. doi: 10.1093/nar/gkr1074. (Cited on page 187.)

[204] A. Kerren, A. Ebert, and J. Meyer, editors. *Human-Centered Visualization Environments*. Springer, 2007. doi: 10.1007/978-3-540-71949-6. (Cited on page 24.)

[205] T. K. Khan. A Survey of Interaction Techniques and Devices for Large High Resolution Displays. In A. Middel, I. Scheler, and H. Hagen, editors, *Visualization of Large and Unstructured Data Sets - Applications in Geospatial Planning, Modeling and Engineering (IRTG 1131 Workshop)*, pages 27–35. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011. doi: 10.4230/OASIcs.VLUDS.2010.27. (Cited on page 218.)

[206] R. Kincaid and H. Lam. Line Graph Explorer: Scalable Display of Line Graphs Using Focus+Context. In *Proceedings of the Conference on Advanced Visual Interfaces (AVI)*, pages 404–411. ACM Press, 2006. doi: 10.1145/1133265.1133348. (Cited on page 126.)

[207] A. Kjellin, L. W. Pettersson, S. Seipel, and M. Lind. Evaluating 2D and 3D Visualizations of Spatiotemporal Information. *ACM Transactions on Applied Perception*, 7(3):19:1–19:23, 2008. doi: 10.1145/1773965.1773970. (Cited on page 133.)

[208] A. Knüpfer, H. Brunst, and W. E. Nagel. High Performance Event Trace Visualization. In *Proceedings of the Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 258–263. IEEE Computer Society, 2005. doi: 10.1109/EMPDP.2005.24. (Cited on page 233.)

[209] R. Kohavi, R. M. Henne, and D. Sommerfield. Practical Guide to Controlled Experiments on the Web: Listen to Your Customers not to the HiPPO. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 959–967. ACM Press, 2007. doi: 10.1145/1281192.1281295. (Cited on page 112.)

[210] K. W. Kolence and P. J. Kiviat. Software Unit Profiles & Kiviat Figures. *SIGMETRICS Performance Evaluation Review*, 2:2–12, 1973. doi: 10.1145/1041613.1041614. (Cited on page 158.)

[211] R. Kosara. Indirect Multi-Touch Interaction for Brushing in Parallel Coordinates. In *Proceedings of the Conference on Visualization and Data Analysis (VDA)*, pages 786809–1–786809–7. SPIE, 2011. doi: 10.1117/12.872645. (Cited on page 24.)

[212] R. Kosara, H. Hauser, and D. L. Gresh. An Interaction View on Information Visualization. In *Eurographics – State of the Art Reports*, pages 123–137. Eurographics Association, 2003. URL `http://diglib.eg.org/EG/DL/Conf/EG2003/star/star6.pdf`. (Cited on page 84.)

[213] M.-J. Kraak. The Space-Time Cube Revisited from a Geovisualization Perspective. In *Proceedings of the International Cartographic Conference (ICC)*, pages 1988–1995. The International Cartographic Association (ICA), 2003. (Cited on pages 53, 126, and 207.)

[214] M.-J. Kraak and O. Huisman. Beyond Exploratory Visualization of Space-Time Paths. In H. J. Miller and J. Han, editors, *Geographic Data Mining and Knowledge Discovery*, pages 431–443. CRC Press, 2nd edition, 2009. doi: 10.1201/9781420073980.ch17. (Cited on page 126.)

[215] M.-J. Kraak and F. J. Ormeling. *Cartography: Visualization of Spatial Data*. Pearson Education, 2010. (Cited on page 34.)

[216] D. Kranzlmüller, S. Grabner, and J. Volkert. Event Graph Visualization For Debugging Large Applications. In *Proceedings of the SIGMETRICS Symposium on Parallel and Distributed Tools (SPDT)*, pages 108–117. ACM Press, 1996. doi: 10.1145/238020.238054. (Cited on page 233.)

[217] G. E. Krasner and S. T. Pope. A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80. *Journal of Object-Oriented Programming*, 1(3):26–49, 1988. (Cited on page 88.)

[218] M. Kreuseler and H. Schumann. A Flexible Approach for Visual Data Mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):39–51, 2002. doi: 10.1109/2945.981850. (Cited on pages 30 and 93.)

[219] M. Kreuseler, T. Nocke, and H. Schumann. A History Mechanism for Visual Data Mining. In *Proceedings of the IEEE Symposium Information Visualization (InfoVis)*, pages 49–56. IEEE Computer Society, 2004. doi: 10.1109/INFVIS.2004.2. (Cited on page 108.)

[220] R. Krüger, D. Thom, M. Wörner, H. Bosch, and T. Ertl. TrajectoryLenses – A Set-based Filtering and Exploration Technique for Long-term Trajectory Data. *Computer Graphics Forum*, 32(3): 451–460, 2013. doi: 10.1111/cgf.12132. (Cited on page 34.)

[221] H. Lam. A Framework of Interaction Costs in Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1149–1156, 2008. doi: 10.1109/TVCG.2008.109. (Cited on pages 15, 24, 51, and 158.)

[222] C. Law, W. Schroeder, K. Martin, and J. Temkin. A Multi-Threaded Streaming Pipeline Architecture for Large Structured Data Sets. In *Proceedings of the IEEE Visualization Conference (Vis)*,

pages 225–232. IEEE Computer Society, 1999. doi: 10.1109/VISUAL.1999.809891. (Cited on pages 65, 74, and 81.)

[223] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry. Task Taxonomy for Graph Visualization. In *Proceedings of the Workshop on Beyond Time and Errors: Novel Evaluation Methods for Information Visualization (BELIV)*, pages 1–5. ACM Press, 2006. doi: 10.1145/1168149.1168168. (Cited on pages 33, 84, and 217.)

[224] B. Lee, P. Isenberg, N. Riche, and S. Carpendale. Beyond Mouse and Keyboard: Expanding Design Considerations for Information Visualization Interactions. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2689–2698, 2012. doi: 10.1109/TVCG.2012.204. (Cited on pages 13, 24, 44, 50, 274, and 277.)

[225] D. Lee, W. Mao, H. Chiu, and W. W. Chu. TBE: A Graphical Interface for Writing Trigger Rules in Active Databases. In *Proceedings of the Working Conference on Visual Database Systems (VDB)*, pages 367–386. Kluwer Academic Publishers, 2000. (Cited on page 242.)

[226] E. A. Lee. The Problem with Threads. *IEEE Computer*, 39(5): 33–42, 2006. doi: 10.1109/MC.2006.180. (Cited on pages 28, 63, 65, and 90.)

[227] J. Lee, Y. Roh, J.-I. Kim, W. Kim, S. Hong, , and H. Kim. A Steerable Tangible Interface for Multi-Layered Contents Played on a Tabletop Interface. In *Proceedings of the International Conference on Interactive Tabletops and Surfaces (ITS)*. ACM Press, 2009. Poster presentation. (Cited on page 201.)

[228] J. C. Lee, S. E. Hudson, J. Summet, and P. H. Dietz. Moveable Interactive Projected Displays Using Projector Based Tracking. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 63–72. ACM Press, 2005. doi: 10.1145/1095034.1095045. (Cited on page 211.)

[229] J. C. Lee, S. E. Hudson, and E. Tse. Foldable Interactive Displays. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 287–290. ACM Press, 2008. doi: 10.1145/1449715.1449763. (Cited on pages 199, 201, and 211.)

[230] A. Lehmann, H. Schumann, O. Staadt, and C. Tominski. Physical Navigation to Support Graph Exploration on a Large High-Resolution Display. In G. Bebis, R. Boyle, B. Parvin, D. Koracin, S. Wang, K. Kyungnam, B. Benes, K. Moreland, C. Borst, S. DiVerdi, C. Yi-Jen, and J. Ming, editors, *Advances in Visual Computing*, volume 6938 of *Lecture Notes in Computer Science*, pages 496–

507. Springer, 2011. doi: 10.1007/978-3-642-24028-7_46. (Cited on pages 152 and 215.)

[231] M. Leissler, M. Hemmje, and E. J. Neuhold. Automatic Updates of Interactive Information Visualization User Interfaces through Database Triggers. In H. Arisawa and T. Catarci, editors, *Advances in Visual Information Management*, pages 341–366. Kluwer Academic Publishers, 2000. doi: 10.1007/978-0-387-35504-7_22. (Cited on page 233.)

[232] C. Letondal, S. Chatty, W. G. Phillips, F. André, and S. Conversy. Usability Requirements for Interaction-Oriented Development Tools. In *Proceedings of the 22nd Annual Workshop of the Psychology of Programming Interest Group (PPIG)*, pages 12–26, 2010. URL http://www.ppig.org/papers/22nd-UX-2.pdf. (Cited on page 17.)

[233] C.-C. Lin, Y.-Y. Lee, and H.-C. Yen. Mental Map Preserving Graph Drawing Using Simulated Annealing. *Information Sciences*, 181(19):4253–4272, 2011. doi: 10.1016/j.ins.2011.06.005. (Cited on page 176.)

[234] M. Lind, C. Forsell, and A. Allard. Effective Visualizations for Large Displays – The Role of Transsaccadic Memory. In *Proceedings of the IASTED International Conference on Visualization, Imaging, and Image Processing (VIIP)*, volume 396, pages 1028–1033. ACTA Press, 2003. (Cited on page 153.)

[235] L. D. Lins, J. T. Klosowski, and C. E. Scheidegger. Nanocubes for Real-Time Exploration of Spatiotemporal Datasets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2456–2465, 2013. doi: 10.1109/TVCG.2013.179. (Cited on page 29.)

[236] H. Liu, Y. Gao, L. Lu, S. Liu, H. Qu, and L. Ni. Visual Analysis of Route Diversity. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 171–180. IEEE Computer Society, 2011. doi: 10.1109/VAST.2011.6102455. (Cited on page 134.)

[237] Z. Liu and J. Stasko. Mental Models, Visual Reasoning and Interaction in Information Visualization: A Top-down Perspective. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):999–1008, 2010. doi: 10.1109/TVCG.2010.177. (Cited on page 12.)

[238] Z. Liu, N. Nersessian, and J. Stasko. Distributed Cognition as a Theoretical Framework for Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1173–1180, 2008. doi: 10.1109/TVCG.2008.121. (Cited on page 12.)

[239] Z. Liu, B. Jiang, and J. Heer. *imMens*: Real-time Visual Querying of Big Data. *Computer Graphics Forum*, 32(3):421–430, 2013. doi: 10.1111/cgf.12129. (Cited on page 29.)

[240] M. Luboschik, H. Schumann, and H. Cords. Particle-Based Labeling: Fast Point-Feature Labeling without Obscuring Other Visual Features. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1237–1244, 2008. doi: 10.1109/TVCG.2008.152. (Cited on pages 185 and 221.)

[241] M. Luboschik, A. Radloff, and H. Schumann. A New Weaving Technique for Handling Overlapping Regions. In *Proceedings of the Conference on Advanced Visual Interfaces (AVI)*, pages 25–32. ACM Press, 2010. doi: 10.1145/1842993.1842998. (Cited on page 159.)

[242] A. M. MacEachren. *Some Truth With Maps: A Primer on Symbolization and Design*. Association of American Geographers, 1994. (Cited on page 128.)

[243] J. Mackinlay. Automating the Design of Graphical Presentations of Relational Information. *ACM Transactions on Graphics*, 5(2): 110–141, 1986. doi: 10.1145/22949.22950. (Cited on pages 5, 21, and 24.)

[244] F. T. Marchese. The Origins and Rise of Medieval Information Visualization. In *Proceedings of the International Conference Information Visualisation (IV)*, pages 389–395. IEEE Computer Society, 2012. doi: 10.1109/IV.2012.71. (Cited on page 3.)

[245] B. Mathieu, S. Heymann, and M. Jacomy. Gephi: An Open Source Software for Exploring and Manipulating Networks. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, pages 361–362. Association for the Advancement of Artificial Intelligence, 2009. URL `http://www.aaai.org/ocs/index.php/ICWSM/09/paper/download/154/1009`. (Cited on page 258.)

[246] K. Matković, H. Hauser, R. Sainitzer, and E. Gröller. Process Visualization with Levels of Detail. In *Proceedings of the IEEE Symposium Information Visualization (InfoVis)*, pages 67–70. IEEE Computer Society, 2002. doi: 10.1109/INFVIS.2002.1173149. (Cited on pages 231 and 233.)

[247] K. Matković, D. Gracanin, Z. Konyha, and H. Hauser. Color Lines View: An Approach to Visualization of Families of Function Graphs. In *Proceedings of the International Conference Information Visualisation (IV)*, pages 59–64. IEEE Computer Society, 2007. doi: 10.1109/IV.2007.35. (Cited on page 126.)

[248] T. G. Mattson, B. A. Sanders, and B. L. Massingill. *Patterns for Parallel Programming*. Addison-Wesley, 2004. (Cited on page 65.)

[249] T. May, M. Steiger, J. Davey, and J. Kohlhammer. Using Signposts for Navigation in Large Graphs. *Computer Graphics Forum*, 31(3pt2):985–994, 2012. doi: 10.1111/j.1467-8659.2012.03091.x. (Cited on page 260.)

[250] B. H. McCormick, T. A. DeFanti, and M. D. Brown. Visualization in Scientific Computing. *ACM SIGRAPH Computer Graphics*, 21(6):3, 1987. doi: 10.1145/41997.41998. (Cited on page 3.)

[251] D. S. McCrickard, R. Catrambone, and J. T. Stasko. Evaluating Animation in the Periphery as a Mechanism for Maintaining Awareness. In *Proceedings of the TC13 IFIP International Conference on Human-Computer Interaction (INTERACT)*. IOS Press, 2001. (Cited on page 247.)

[252] M. J. McGuffin and I. Jurisica. Interaction Techniques for Selecting and Manipulating Subgraphs in Network Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6): 937–944, 2009. doi: 10.1109/TVCG.2009.151. (Cited on pages 30, 31, 151, and 175.)

[253] B. S. Michel and H. Zima. Bridging Multicore's Programmability Gap. Workshop at the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2008. (Cited on pages 63 and 65.)

[254] K. Miriyala, S. W. Hornick, and R. Tamassia. An Incremental Approach to Aesthetic Graph Layout. In *Proceedings of the International Workshop on Computed-Aided Software Engineering (CASE)*, pages 297–308. IEEE Computer Society, 1993. doi: 10.1109/CASE.1993.634832. (Cited on page 191.)

[255] D. Molyneaux and H. Gellersen. Projected Interfaces: Enabling Serendipitous Interaction with Smart Tangible Objects. In *Proceedings of the International Conference on Tangible and Embedded Interaction (TEI)*, pages 385–392. ACM Press, 2009. doi: 10.1145/1517664.1517741. (Cited on page 197.)

[256] T. Moscovich, F. Chevalier, N. Henry, E. Pietriga, and J.-D. Fekete. Topology-Aware Navigation in Large Networks. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 2319–2328. ACM Press, 2009. doi: 10.1145/1518701.1519056. (Cited on pages 85, 151, and 217.)

[257] P. Muigg, J. Kehrer, S. Oeltze, H. Piringer, H. Doleisch, B. Preim, and H. Hauser. A Four-level Focus+Context Approach to Interactive Visual Analysis of Temporal Features in Large Scien-

tific Data. *Computer Graphics Forum*, 27(3):775–782, 2008. doi: 10.1111/j.1467-8659.2008.01207.x. (Cited on page 81.)

[258] T. Munzner, F. Guimbretière, S. Tasiran, L. Zhang, and Y. Zhou. TreeJuxtaposer: Scalable Tree Comparison Using Focus+Context with Guaranteed Visibility. *ACM Transactions on Graphics*, 22(3):453–462, 2003. doi: 10.1145/882262.882291. (Cited on page 151.)

[259] T. Ni, G. S. Schmidt, O. G. Staadt, M. A. Livingston, R. Ball, and R. May. A Survey of Large High-Resolution Display Technologies, Techniques, and Applications. In *Proceedings of the IEEE Virtual Reality Conference (VR)*, pages 223–236, 2006. doi: 10.1109/VR.2006.20. (Cited on page 216.)

[260] J. Nielsen. *Usability Engineering*. Morgan Kaufmann, 1993. (Cited on pages 9 and 24.)

[261] A. Noack. Energy Models for Graph Clustering. *Journal of Graph Algorithms and Applications*, 11(2):453–480, 2007. doi: 10.7155/jgaa.00154. (Cited on page 91.)

[262] D. A. Norman. *The Design of Everyday Things*. Basic Books, 2002. (Cited on pages 9, 24, 25, 51, 57, 277, and 278.)

[263] C. North, N. Conklin, and V. Saini. Visualization Schemas for Flexible Information Visualization. In *Proceedings of the IEEE Symposium Information Visualization (InfoVis)*, pages 15–22. IEEE Computer Society, 2002. doi: 10.1109/INFVIS.2002.1173142. (Cited on page 228.)

[264] S. C. North. Incremental Layout in DynaDAG. In *Proceedings of the International Symposium on Graph Drawing (GD)*, pages 409–418. Springer, 1996. doi: 10.1007/BFb0021824. (Cited on page 176.)

[265] M. Novotny and H. Hauser. Outlier-Preserving Focus+Context Visualization in Parallel Coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):893–900, 2006. doi: 10.1109/TVCG.2006.170. (Cited on pages 64 and 72.)

[266] M. Okoe, S. S. Alam, and R. Jianu. A Gaze-enabled Graph Visualization to Improve Graph Reading Tasks. *Computer Graphics Forum*, 33(3):251–260, 2014. doi: 10.1111/cgf.12381. (Cited on page 56.)

[267] S. M. Peck, C. North, and D. Bowman. A Multiscale Interaction Technique for Large, High-Resolution Displays. In *Proceedings of the IEEE Symposium on 3D User Interfaces (3DUI)*, pages 31–38. IEEE Computer Society, 2009. doi: 10.1109/3DUI.2009.4811202. (Cited on page 218.)

[268] A. Perer and F. van Ham. Integrating Querying and Browsing in Partial Graph Visualizations. IBM Technical Report 12-01, IBM Research, 2012. (Cited on page 260.)

[269] W. A. Pike, J. T. Stasko, R. Chang, and T. A. O'Connell. The Science of Interaction. *Information Visualization*, 8(4):263–274, 2009. doi: 10.1057/ivs.2009.22. (Cited on pages 12 and 151.)

[270] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser. CRAWDAD data set epfl/mobility (v. 2009-02-24), 2009. URL `http://crawdad.cs.dartmouth.edu/epfl/mobility`. (Cited on page 140.)

[271] H. Piringer, W. Berger, and H. Hauser. Quantifying and Comparing Features in High-Dimensional Datasets. In *Proceedings of the International Conference Information Visualisation (IV)*, pages 240–245. IEEE Computer Society, 2008. doi: 10.1109/IV.2008.17. (Cited on pages 29, 63, and 76.)

[272] H. Piringer, C. Tominski, P. Muigg, and W. Berger. A Multi-Threading Architecture to Support Interactive Visual Exploration. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1113–1120, 2009. doi: 10.1109/TVCG.2009.110. (Cited on page 61.)

[273] P. Pirolli and S. Card. The Sensemaking Process and Leverage Points for Analyst Technology as Identified Through Cognitive Task Analysis. In *Proceedings of the International Conference on Intelligence Analysis*, 2005. (Cited on page 12.)

[274] C. Plaisant, J. Grosjean, and B. B. Bederson. SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation. In *Proceedings of the IEEE Symposium Information Visualization (InfoVis)*, pages 57–64. IEEE Computer Society, 2002. doi: 10.1109/INFVIS.2002.1173148. (Cited on page 260.)

[275] M. Plumlee and C. Ware. Zooming versus multiple window interfaces: Cognitive costs of visual comparisons. *ACM Transactions on Computer-Human Interaction*, 13(2):179–209, 2006. doi: 10.1145/1165734.1165736. (Cited on pages 155 and 162.)

[276] K. Pulo. Navani: Navigating Large-Scale Visualisations with Animated Transitions. In *Proceedings of the International Conference Information Visualisation (IV)*, pages 271–276. IEEE Computer Society, 2007. doi: 10.1109/IV.2007.82. (Cited on pages 17 and 103.)

[277] H. C. Purchase. Metrics for Graph Drawing Aesthetics. *Journal of Visual Languages & Computing*, 13(5):501–516, 2002. doi: 10.1006/jvlc.2002.0232. (Cited on page 180.)

[278] J. Raisamo, R. Raisamo, and P. Karkkainnen. A Method for Interactive Graph Manipulation. In *Proceedings of the International Conference Information Visualisation (IV)*, pages 581–587. IEEE Computer Society, 2004. doi: 10.1109/IV.2004.1320202. (Cited on page 175.)

[279] D. A. Randell, Z. Cui, and A. Cohn. A Spatial Logic Based on Regions and Connection. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 165–176. Morgan Kaufmann, 1992. (Cited on pages 238 and 239.)

[280] R. Rao and S. K. Card. The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus + Context Visualization for Tabular Information. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 318–322. ACM Press, 1994. doi: 10.1145/191666.191776. (Cited on pages 53 and 251.)

[281] F. Reinders. *Feature-Based Visualization of Time-Dependent Data*. PhD thesis, Delft University of Technology, 2001. URL http://resolver.tudelft.nl/uuid:cd6c7fe1-7331-4626-ba8f-a84e1f690ba4. (Cited on page 230.)

[282] F. Reinders, F. H. Post, and H. J. Spoelder. Visualization of Time-dependent Data with Feature Tracking and Event Detection. *The Visual Computer*, 17(1):55–71, 2001. doi: 10.1007/PL00013399. (Cited on pages 230, 233, 247, and 248.)

[283] E. M. Reingold and J. S. Tilford. Tidier Drawings of Trees. *IEEE Transactions on Software Engineering*, 7(2):223–228, 1981. doi: 10.1109/TSE.1981.234519. (Cited on page 93.)

[284] S. Rinzivillo, D. Pedreschi, M. Nanni, F. Giannotti, N. Andrienko, and G. Andrienko. Visually Driven Analysis of Movement Data by Progressive Clustering. *Information Visualization*, 7(3-4):225–239, 2008. doi: 10.1057/palgrave.ivs.9500183. (Cited on page 129.)

[285] A. H. Robinson. The Thematic Maps of Charles Joseph Minard. *Imago Mundi*, 21:95–108, 1967. URL http://www.jstor.org/stable/1150482. (Cited on page 126.)

[286] R. E. Roth. An Empirically-Derived Taxonomy of Interaction Primitives for Interactive Cartography and Geovisualization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2356–2365, 2013. doi: 10.1109/TVCG.2013.130. (Cited on page 278.)

[287] G. Rößling. *ANIMAL-FARM: An Extensible Framework for Algorithm Visualization*. PhD thesis, University of Siegen, 2002. URL http://dokumentix.ub.uni-siegen.de/opus/volltexte/2006/184/. (Cited on page 233.)

[288] M. R. Sadri. *Optimization of Sequence Queries in Database Systems*. PhD thesis, University of California, Los Angeles, 2001. (Cited on pages 238 and 244.)

[289] R. Sadri, C. Zaniolo, A. Zarkesh, and J. Adibi. Expressing and Optimizing Sequence Queries in Database Systems. *ACM Transactions on Database Systems*, 29(2):282–318, 2004. doi: 10.1145/1005566.1005568. (Cited on page 240.)

[290] T. Saito, H. N. Miyamura, M. Yamamoto, H. Saito, Y. Hoshiya, and T. Kaseda. Two-Tone Pseudo Coloring: Compact Visualization for One-Dimensional Data. In *Proceedings of the IEEE Symposium Information Visualization (InfoVis)*, pages 173–180. IEEE Computer Society, 2005. doi: 10.1109/INFOVIS.2005.35. (Cited on page 129.)

[291] J. Sanneblad and L. E. Holmquist. Ubiquitous Graphics: Combining Hand-held and Wall-size Displays to Interact with Large Images. In *Proceedings of the Conference on Advanced Visual Interfaces (AVI)*, pages 373–377. ACM Press, 2006. doi: 10.1145/1133265.1133343. (Cited on pages 197 and 211.)

[292] P. Saraiya, C. North, V. Lam, and K. Duca. An Insight-Based Longitudinal Study of Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1511–1522, Nov 2006. doi: 10.1109/TVCG.2006.85. (Cited on page 15.)

[293] M. Sarkar and M. H. Brown. Graphical Fisheye Views. *Communications of the ACM*, 37(12):73–83, 1994. doi: 10.1145/198366.198384. (Cited on page 196.)

[294] D. Schmidt, M. Stal, H. Rohnert, and F. Buschmann. *Pattern-Oriented Software Architecture Volume 2: Patterns for Concurrent and Networked Objects*. Wiley, 2000. (Cited on pages 65 and 68.)

[295] S. Schmidt, M. A. Nacenta, R. Dachselt, and M. S. T. Carpendale. A Set of Multi-Touch Graph Interaction Techniques. In *Proceedings of the International Conference on Interactive Tabletops and Surfaces (ITS)*, pages 113–116. ACM Press, 2010. doi: 10.1145/1936652.1936673. (Cited on page 185.)

[296] O. Schmitt, J. Modersitzki, S. Heldmann, S. Wirtz, and B. Fischer. Image Registration of Sectioned Brains. *International Journal of Computer Vision*, 73(1):5–39, 2007. doi: 10.1007/s11263-006-9780-x. (Cited on pages 114 and 116.)

[297] H.-J. Schulz, T. Nocke, M. Heitzler, and H. Schumann. A Design Space of Visualization Tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2366–2375, 2013. doi: 10.1109/TVCG.2013.120. (Cited on pages 22 and 37.)

[298] H.-J. Schulz, M. Streit, T. May, and C. Tominski. Towards a Characterization of Guidance in Visualization. Poster presentation, IEEE Conference on Information Visualization (InfoVis), 2013. URL http://www.informatik.uni-rostock.de/~ct/pub_files/Schulz13Guidance.pdf. (Cited on pages 57 and 277.)

[299] C. Schwesig, I. Poupyrev, and E. Mori. Gummi: A Bendable Computer. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 263–270. ACM Press, 2003. doi: 10.1145/985692.985726. (Cited on page 199.)

[300] K. Sedig and P. Parsons. Interaction Design for Complex Cognitive Activities with Visual Representations: A Pattern-Based Approach. *AIS Transactions on Human-Computer Interaction*, 5(2):84–133, 2013. (Cited on pages 11, 13, 23, 38, and 278.)

[301] K. Sedig, P. Parsons, and A. Babanski. Towards a Characterization of Interactivity in Visual Analytics. *Journal of Multimedia Processing and Technologies*, 3(1):12–28, 2012. (Cited on page 15.)

[302] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks. *Genome Research*, 13(11):2498–2504, 2003. doi: 10.1101/gr.1239303. (Cited on page 186.)

[303] B. Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. *IEEE Computer*, 16(8):57–69, 1983. doi: 10.1109/MC.1983.1654471. (Cited on pages 10, 11, 24, 45, 62, 84, 151, 152, and 196.)

[304] B. Shneiderman. Dynamic Queries for Visual Information Seeking. *IEEE Software*, 11(6):70–77, 1994. doi: 10.1109/52.329404. (Cited on pages 16, 23, 62, 63, 68, and 242.)

[305] B. Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proceedings of the IEEE Symposium on Visual Languages (VL)*, pages 336–343. IEEE Computer Society, 1996. (Cited on pages 10, 21, and 107.)

[306] B. Shneiderman. Why Not Make Interfaces Better Than 3D Reality? *Computer Graphics and Applications*, 23(6):12–15, 2003. doi: 10.1109/MCG.2003.1242376. (Cited on page 133.)

[307] T. A. Slocum, R. B. MacMaster, F. C. Kessler, and H. H. Howard. *Thematic Cartography and Geovisualization*. Pearson Education, 3rd edition, 2009. (Cited on page 128.)

[308] A. Som, C. Harder, B. Greber, M. Siatkowski, Y. Paudel, G. Warsow, C. Cap, H. Schöler, and G. Füllen. The PluriNetWork: An Electronic Representation of the Network Underlying Pluripotency in Mouse, and Its Applications. *PLoS One*, 5(12), 2010. doi: 10.1371/journal.pone.0015165. (Cited on pages 175 and 186.)

[309] J. Somervell, D. S. McCrickard, C. North, and M. Shukla. An Evaluation of Information Visualization in Attention-Limited Environments. In *Proceedings of the Joint Eurographics - IEEE TCVG Symposium on Visualization (VisSym)*, pages 211–216. Eurographics Association, 2002. URL `http://diglib.eg.org/EG/DL/WS/VisSym/VisSym02/211-216.pdf`. (Cited on page 247.)

[310] R. Spence. *Information Visualization: Design for Interaction*. Prentice-Hall, 2nd edition, 2007. (Cited on pages 10, 11, 15, 16, 31, 54, 62, 68, 151, 258, and 275.)

[311] M. Spindler and R. Dachselt. Towards Pen-based Annotation Techniques for Tangible Magic Lenses Above a Tabletop. In *Proceedings of the International Conference on Interactive Tabletops and Surfaces (ITS)*. ACM Press, 2009. Poster presentation. (Cited on pages 201 and 202.)

[312] M. Spindler, S. Stellmach, and R. Dachselt. PaperLens: Advanced Magic Lens Interaction Above the Tabletop. In *Proceedings of the International Conference on Interactive Tabletops and Surfaces (ITS)*, pages 69–76. ACM Press, 2009. doi: 10.1145/1731903.1731920. (Cited on pages 45, 197, 198, 201, 203, 210, 211, and 213.)

[313] M. Spindler, C. Tominski, H. Schumann, and R. Dachselt. Tangible Views for Information Visualization. In *Proceedings of the International Conference on Interactive Tabletops and Surfaces (ITS)*, pages 157–166. ACM Press, 2010. doi: 10.1145/1936652.1936684. (Cited on page 193.)

[314] M. Spindler, M. Martsch, and R. Dachselt. Going Beyond the Surface: Studying Multi-layer Interaction Above the Tabletop. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 1277–1286. ACM Press, 2012. doi: 10.1145/2207676.2208583. (Cited on page 47.)

[315] D. Spretke, P. Bak, H. Janetzko, B. Kranstauber, F. Mansmann, and S. Davidson. Exploration Through Enrichment: A Visual Analytics Approach for Animal Movement. In *Proceedings of the ACM SIGSPATIAL International Symposium on Advances in*

Geographic Information Systems (ACM GIS), pages 421–424. ACM Press, 2011. doi: 10.1145/2093973.2094038. (Cited on page 126.)

[316] A. S. Spritzer and C. M. D. S. Freitas. A Physics-Based Approach for Interactive Manipulation of Graph Visualizations. In *Proceedings of the Conference on Advanced Visual Interfaces (AVI)*, pages 271–278. ACM Press, 2008. doi: 10.1145/1385569.1385613. (Cited on page 175.)

[317] J. B. Strother, J. M. Ulijn, and Z. Fazal, editors. *Information Overload: An International Challenge for Professional Engineers and Technical Communicators*. Wiley, 2012. (Cited on page 3.)

[318] S. Subramanian, D. Aliakseyeu, and A. Lucero. Multi-Layer Interaction for Digital Tables. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 269–272. ACM Press, 2006. doi: 10.1145/1166253.1166295. (Cited on page 200.)

[319] R. Tamassia, editor. *Handbook of Graph Drawing and Visualization*. CRC Press, 2013. (Cited on page 30.)

[320] D. Tang, C. Stolte, and R. Bosch. Design Choices When Architecting Visualizations. *Information Visualization*, 3(2):65–79, 2004. doi: 10.1057/palgrave.ivs.9500067. (Cited on pages 88 and 241.)

[321] E. Tanin, R. Beigel, and B. Shneiderman. Incremental Data Structures and Algorithms for Dynamic Query Interfaces. *ACM SIGMOD Record*, 25(4):21–24, 1996. doi: 10.1145/245882.245891. (Cited on page 64.)

[322] A. Telea and D. Auber. Code Flows: Visualizing Structural Evolution of Source Code. *Computer Graphics Forum*, 27(3): 831–838, 2008. doi: 10.1111/j.1467-8659.2008.01214.x. (Cited on page 176.)

[323] M. Theus. Interactive Data Visualization using Mondrian. *Journal of Statistical Software*, 7(11):1–9, 11 2002. URL `http://www.jstatsoft.org/v07/i11`. (Cited on page 66.)

[324] C. Thiede, G. Fuchs, and H. Schumann. Smart Lenses. In *Proceedings of the Smart Graphics (SG)*, pages 178–189. Springer, 2008. doi: 10.1007/978-3-540-85412-8_16. (Cited on pages 100 and 196.)

[325] J. J. Thomas and K. A. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society, 2005. (Cited on pages 12, 15, 84, 151, and 254.)

[326] C. Tominski. CGV – Coordinated Graph Visualization. Interactive prototype, 2008. URL `http://www.informatik.`

uni-rostock.de/~ct/software/CGV/CGV.html. Retrieved: 14. Juli 2014. (Cited on page 114.)

[327] C. Tominski. Event-Based Concepts for User-Driven Visualization. *Information Visualization*, 10(1):65–81, 2011. doi: 10.1057/ivs.2009.32. (Cited on page 227.)

[328] C. Tominski. Foldable Visualization. Interactive prototype, 2012. URL http://goo.gl/LwREL. Retrieved: 14. Juli 2014. (Cited on page 165.)

[329] C. Tominski, J. Abello, and H. Schumann. Axes-Based Visualizations with Radial Layouts. In *Proceedings of the ACM Symposium on Applied Computing (SAC)*, pages 1242–1247. ACM, 2004. doi: 10.1145/967900.968153. (Cited on pages 29, 53, 63, 81, and 249.)

[330] C. Tominski, J. Abello, F. van Ham, and H. Schumann. Fisheye Tree Views and Lenses for Graph Visualization. In *Proceedings of the International Conference Information Visualisation (IV)*, pages 17–24. IEEE Computer Society, 2006. doi: 10.1109/IV.2006.54. (Cited on pages 85, 95, 100, 101, 175, and 196.)

[331] C. Tominski, G. Fuchs, and H. Schumann. Task-Driven Color Coding. In *Proceedings of the International Conference Information Visualisation (IV)*, pages 373–380. IEEE Computer Society, 2008. doi: 10.1109/IV.2008.24. (Cited on pages 22, 151, and 208.)

[332] C. Tominski, J. Abello, and H. Schumann. CGV – An Interactive Graph Visualization System. *Computers & Graphics*, 33(6):660–678, 2009. doi: 10.1016/j.cag.2009.06.002. (Cited on pages 29, 63, 81, 83, 156, 217, 242, 258, and 267.)

[333] C. Tominski, J. Abello, and H. Schumann. Two Novel Techniques for Interactive Navigation of Graph Layouts. Poster at Eurographics/IEEE Symposium on Visualization (EuroVis), 2009. (Cited on pages 85, 105, and 196.)

[334] C. Tominski, J. F. Donges, and T. Nocke. Information Visualization in Climate Research. In *Proceedings of the International Conference Information Visualisation (IV)*, pages 298–305. IEEE Computer Society, 2011. doi: 10.1109/IV.2011.12. (Cited on page 15.)

[335] C. Tominski, H. Schumann, M. Spindler, and R. Dachselt. Towards Utilizing Novel Interactive Displays for Information Visualization. In *Proceedings of theWorkshop on Data Exploration for Interactive Surfaces (DEXIS)*. HAL - Inria Open Archive, 2011. URL http://hal.inria.fr/hal-00659469. (Cited on pages 24, 44, and 277.)

[336] C. Tominski, C. Forsell, and J. Johansson. Interaction Support for Visual Comparison Inspired by Natural Behavior. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2719–2728, 2012. doi: 10.1109/TVCG.2012.237. (Cited on page 147.)

[337] C. Tominski, H. Schumann, G. Andrienko, and N. Andrienko. Stacking-Based Visualization of Trajectory Attribute Data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2565–2574, 2012. doi: 10.1109/TVCG.2012.265. (Cited on page 121.)

[338] C. Tominski, H. Schumann, G. Andrienko, and N. Andrienko. Stacking-Based Visualization of Trajectory Attribute Data. Interactive prototype, 2012. URL `http://goo.gl/wIC1k`. Retrieved: 14. Juli 2014. (Cited on page 145.)

[339] C. Tominski, S. Gladisch, U. Kister, R. Dachselt, and H. Schumann. A Survey on Interactive Lenses in Visualization. In *EuroVis State-of-the-Art Reports*, pages 43–62. Eurographics Association, 2014. doi: 10.2312/eurovisstar.20141172. (Cited on pages 37, 41, and 275.)

[340] M. Tory and T. Möller. Evaluating Visualizations: Do Expert Reviews Work? *Computer Graphics and Applications*, 25(5):8–11, 2005. doi: 10.1109/MCG.2005.102. (Cited on page 112.)

[341] A. M. Treisman and G. Gelade. A Feature-Integration Theory of Attention. *Cognitive Psychology*, 12(4):97–136, 1980. doi: 10.1016/0010-0285(80)90005-5. (Cited on page 247.)

[342] Y. Tu and H.-W. Shen. Visualizing Changes of Hierarchical Data Using Treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1286–1293, 2007. doi: 10.1109/TVCG.2007.70529. (Cited on page 176.)

[343] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 2nd edition, 2001. (Cited on page 247.)

[344] B. Tversky, J. B. Morrison, and M. Betrancourt. Animation: Can It Facilitate? *International Journal of Human-Computer Studies*, 57(4):247–262, 2002. doi: 10.1006/ijhc.2002.1017. (Cited on pages 103 and 247.)

[345] A. M. Uhrmacher, A. Rolfs, and J. Frahm. DFG Research Training Group 1387/1: dIEM oSiRiS - Integrative Development of Modelling and Simulation Methods for Regenerative Systems. *it - Information Technology*, 49(6):388–395, 2007. doi: 10.1524/itit.2007.49.6.388. (Cited on page 114.)

[346] B. Ullmer and H. Ishii. The MetaDESK: Models and Prototypes for Tangible User Interfaces. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages

223–232. ACM Press, 1997. doi: 10.1145/263407.263551. (Cited on pages 197 and 211.)

[347] B. Ullmer, H. Ishii, and R. J. K. Jacob. Tangible Query Interfaces: Physically Constrained Tokens for Manipulating Database Queries. In *Proceedings of the TC13 IFIP International Conference on Human-Computer Interaction (INTERACT)*, pages 279–286. IOS Press, 2003. (Cited on page 202.)

[348] A. Valli. The Design of Natural Interaction. *Multimedia Tools and Applications*, 38(3):295–305, 2008. doi: 10.1007/s11042-007-0190-z. (Cited on pages 10, 25, and 152.)

[349] A. van Dam. Post-WIMP User Interfaces. *Communications of the ACM*, 40(2):63–67, 1997. doi: 10.1145/253671.253708. (Cited on page 10.)

[350] F. van Ham and A. Perer. Search, Show Context, Expand on Demand: Supporting Large Graph Exploration with Degree-of-Interest. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):953–960, 2009. doi: 10.1109/TVCG.2009.108. (Cited on pages 55, 56, 260, and 263.)

[351] F. van Ham and J. J. van Wijk. Interactive Visualization of Small World Graphs. In *Proceedings of the IEEE Symposium Information Visualization (InfoVis)*, pages 199–206. IEEE Computer Society, 2004. doi: 10.1109/INFVIS.2004.43. (Cited on pages 205 and 217.)

[352] R. van Liere and W. C. de Leeuw. GraphSplatting: Visualizing Graphs as Continuous Fields. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):206–212, 2003. doi: 10.1109/TVCG.2003.1196007. (Cited on page 94.)

[353] J. J. van Wijk. The Value of Visualization. In *Proceedings of the IEEE Visualization Conference (Vis)*, pages 79–86. IEEE Computer Society, 2005. doi: 10.1109/VIS.2005.102. (Cited on pages 41 and 174.)

[354] J. J. van Wijk. Views on Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):421–433, 2006. doi: 10.1109/TVCG.2006.80. (Cited on page 5.)

[355] J. J. van Wijk and W. A. A. Nuij. A Model for Smooth Viewing and Navigation of Large 2D Information Spaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):447–458, 2004. doi: 10.1109/TVCG.2004.1. (Cited on pages 32, 103, 156, and 196.)

[356] B. Victor. Magic Ink – Information Software and the Graphical Interface, 2006. URL http://worrydream.com/MagicInk. Retrieved: 14. Juli 2014. (Cited on pages 15 and 52.)

[357] F. B. Viégas, M. Wattenberg, F. van Ham, J. Kriss, and M. McKeon. ManyEyes: A Site for Visualization at Internet Scale. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1121–1128, 2007. doi: 10.1109/TVCG.2007.70577. (Cited on page 90.)

[358] D. Vogel and R. Balakrishnan. Interactive Public Ambient Displays: Transitioning from Implicit to Explicit, Public to Personal, Interaction with Multiple Users. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 137–146. ACM Press, 2004. doi: 10.1145/1029632.1029656. (Cited on page 218.)

[359] D. Vogel and R. Balakrishnan. Distant Freehand Pointing and Clicking on Very Large, High Resolution Displays. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 33–42. ACM Press, 2005. doi: 10.1145/1095034.1095041. (Cited on page 152.)

[360] S. Voida, M. Tobiasz, J. Stromer, P. Isenberg, and M. S. T. Carpendale. Getting Practical with Interactive Tabletop Displays: Designing for Dense Data, "Fat Fingers", Diverse Interactions, and Face-To-Face Collaboration. In *Proceedings of the International Conference on Interactive Tabletops and Surfaces (ITS)*, pages 109–116. ACM Press, 2009. doi: 10.1145/1731903.1731926. (Cited on pages 24 and 197.)

[361] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J.-D. Fekete, and D. W. Fellner. Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011. doi: 10.1111/j.1467-8659.2011.01898.x. (Cited on pages 31 and 33.)

[362] B. Waldhauer. Touchinteratkion für die exploration und manipulation von graphen. Master's thesis, Institute for Computer Science, University of Rostock, 2013. (Cited on page 43.)

[363] M. Q. Wang Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for Using Multiple Views in Information Visualization. In *Proceedings of the Conference on Advanced Visual Interfaces (AVI)*, pages 110–119. ACM Press, 2000. doi: 10.1145/345513.345271. (Cited on pages 62, 87, 155, 194, 196, and 204.)

[364] M. Ward and J. Yang. Interaction Spaces in Data and Information Visualization. In *Proceedings of the Joint Eurographics - IEEE*

TCVG Symposium on Visualization (VisSym), pages 137–146. Eurographics Association, 2004. URL http://diglib.eg.org/EG/DL/WS/VisSym/VisSym04/137-146.pdf. (Cited on page 84.)

[365] M. O. Ward, G. Grinstein, and D. Keim. *Interactive Data Visualization: Foundations, Techniques, and Applications*. A K Peters/CRC Press, 2010. (Cited on pages 11 and 22.)

[366] C. Ware. *Visual Thinking for Design*. Morgan Kaufmann, 2008. (Cited on page 3.)

[367] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, 3rd edition, 2012. (Cited on pages 10, 11, 15, and 37.)

[368] C. Ware, R. Arsenault, M. Plumlee, and D. Wiley. Visualizing the Underwater Behavior of Humpback Whales. *Computer Graphics and Applications*, 26(4):14–18, 2006. doi: 10.1109/MCG.2006.93. (Cited on page 126.)

[369] C. Weaver. Building Highly-Coordinated Visualizations in Improvise. In *Proceedings of the IEEE Symposium Information Visualization (InfoVis)*, pages 159–166. IEEE Computer Society, 2004. doi: 10.1109/INFVIS.2004.12. (Cited on page 66.)

[370] C. E. Weaver. *Improvise: A User Interface for Interactive Construction of Highly-Coordinated Visualizations*. PhD thesis, University of Wisconsin–Madison, 2006. (Cited on pages 62, 66, 72, 87, and 108.)

[371] C. E. Weaver and M. Livny. Improving Visualization Interactivity in Java. In *Proceedings of the Conference on Visualization and Data Analysis (VDA)*, pages 62–72. SPIE, 2000. doi: 10.1117/12.378919. (Cited on page 66.)

[372] P. Wegner. Why Interaction Is More Powerful Than Algorithms. *Communications of the ACM*, 40(5):80–91, 1997. doi: 10.1145/253769.253801. (Cited on page 15.)

[373] M. Weiss, J. Wagner, Y. Jansen, R. Jennings, R. Khoshabeh, J. D. Hollan, and J. O. Borchers. SLAP Widgets: Bridging the Gap Between Virtual and Physical Controls on Tabletops. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 481–490. ACM Press, 2009. doi: 10.1145/1518701.1518779. (Cited on page 200.)

[374] S. White, D. Feng, and S. Feiner. Interaction and Presentation Techniques for Shake Menus in Tangible Augmented Reality. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 39–48. IEEE Computer Society, 2009. doi: 10.1109/ISMAR.2009.5336500. (Cited on page 201.)

[375] D. Wigdor and D. Wixon. *Brave NUI World: Designing Natural User Interfaces for Touch and Gesture*. Morgan Kaufmann, 2011. (Cited on page 25.)

[376] L. Wilkinson. *The Grammar of Graphics*. Springer, 2nd edition, 2005. doi: 10.1007/0-387-28695-0. (Cited on pages 255 and 278.)

[377] N. Willems, H. van de Wetering, and J. J. van Wijk. Visualization of Vessel Movements. *Computer Graphics Forum*, 28(3): 959–966, 2009. doi: 10.1111/j.1467-8659.2009.01440.x. (Cited on page 34.)

[378] G. Wills. Selection: 524,288 Ways to Say "This is Interesting". In *Proceedings of the IEEE Symposium Information Visualization (InfoVis)*, pages 54–60. IEEE Computer Society, 1996. doi: 10.1109/INFVIS.1996.559216. (Cited on page 96.)

[379] P. C. Wong, W. Cowley, H. Foote, E. Jurrus, and J. Thomas. Visualizing Sequential Patterns for Text Mining. In *Proceedings of the IEEE Symposium Information Visualization (InfoVis)*, pages 105–111. IEEE Computer Society, 2000. doi: 10.1109/INFVIS.2000.885097. (Cited on page 232.)

[380] M. Wybrow, K. Marriott, and P. J. Stuckey. Orthogonal Connector Routing. In *Proceedings of the International Symposium on Graph Drawing (GD)*, pages 219–231. Springer, 2009. doi: 10.1007/978-3-642-11805-0_22. (Cited on page 191.)

[381] B. Wylie and J. Baumes. A Unified Toolkit for Information and Scientific Visualization. In *Proceedings of the Conference on Visualization and Data Analysis (VDA)*, pages 72430H–1–72430H–16. SPIE, 2009. doi: 10.1117/12.805589. (Cited on page 108.)

[382] L. Xiao, J. Gerth, and P. Hanrahan. Enhancing Visual Analysis of Network Traffic Using Knowledge Representation. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 107–114. IEEE Computer Society, 2006. doi: 10.1109/VAST.2006.261436. (Cited on pages 232, 233, 242, and 254.)

[383] D. Yang, E. A. Rundensteiner, and M. O. Ward. Analysis Guided Visual Exploration to Multivariate Data. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 83–90. IEEE Computer Society, 2007. doi: 10.1109/VAST.2007.4389000. (Cited on page 254.)

[384] K.-P. Yee. Peephole Displays: Pen Interaction on Spatially Aware Handheld Computers. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 1–8. ACM Press, 2003. doi: 10.1145/642611.642613. (Cited on page 197.)

[385] J. S. Yi, Y. ah Kang, J. T. Stasko, and J. A. Jacko. Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, 2007. doi: 10.1109/TVCG.2007.70515. (Cited on pages 4, 11, 13, 15, 23, 38, 84, 109, 151, 194, 196, and 207.)

[386] F. Ying, P. Mooney, P. Corcoran, and A. C. Winstanley. Dynamic Visualization of Geospatial Data on Small Screen Mobile Devices. In G. Gartner and F. Ortag, editors, *Advances in Location-Based Services*, pages 77–90. Springer, 2012. doi: 10.1007/978-3-642-24198-7_5. (Cited on page 23.)

[387] B. Yost and C. North. The Perceptual Scalability of Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):837–844, 2006. doi: 10.1109/TVCG.2006.184. (Cited on page 23.)

[388] L. Yu, K. Efstathiou, P. Isenberg, and T. Isenberg. Efficient Structure-Aware Selection Techniques for 3D Point Cloud Visualizations with 2DOF Input. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2245–2254, 2012. doi: 10.1109/TVCG.2012.217. (Cited on page 24.)

[389] S. Zhai, J. Wright, T. Selker, and S.-A. Kelin. Graphical Means of Directing User's Attention in the Visual Interface. In *Proceedings of the TC13 IFIP International Conference on Human-Computer Interaction (INTERACT)*. Chapman & Hall, 1997. (Cited on page 247.)

[390] J. Zhao, F. Chevalier, and R. Balakrishnan. KronoMiner: Using Multi-foci Navigation for the Visual Exploration of Time-series Data. In *Proceedings of the SIGCHI Conference Human Factors in Computing Systems (CHI)*, pages 1737–1746. ACM Press, 2011. doi: 10.1145/1978942.1979195. (Cited on page 22.)

[391] Q. Zhao and P. Mitra. Event Detection and Visualization for Social Text Streams. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, 2007. URL `http://www.icwsm.org/papers/paper34.html`. (Cited on page 232.)

[392] M. Zinsmaier, U. Brandes, O. Deussen, and H. Strobelt. Interactive Level-of-Detail Rendering of Large Graphs. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2486–2495, 2012. doi: 10.1109/TVCG.2012.238. (Cited on page 29.)

[393] E. Zudilova-Seinstra, T. Adriaansen, and R. van Liere. *Trends in Interactive Visualization: State-of-the-Art Survey*. Springer, 2009. (Cited on page 13.)

FULL LIST OF PUBLICATIONS

STATISTICS

Books:	1
Journal articles:	13
Contributions to books:	7
Refereed papers at conferences and workshops:	29
Posters and demos:	12
Unrefereed contributions:	2
Theses:	1
	65

PUBLICATIONS

In progress

65. C. Tominski, H. Schumann, S. Miksch, and W. Aigner. Images of Time – Visual Representations of Time-Oriented Data. In *Gower Handbook of Information Design*. Gower Publishing. (submitted).

64. W. Aigner, S. Miksch, H. Schumann, and C. Tominski. Visualization Techniques for Time-Oriented Data. In *Interactive Data Visualization: Foundations, Techniques, and Applications*. AK Peters. (submitted).

63. S. Gladisch and C. Tominski. Toward Integrated Exploration and Manipulation of Data Attributes in Graphs. Poster at IEEE Conference on Information Visualization (InfoVis). (submitted).

62. A. Radloff, C. Tominski, T. Nocke, and H. Schumann. Supporting Presentation and Discussion of Visualization Results in Smart Meeting Rooms. *The Visual Computer*. (accepted).

2014

61. C. Tominski, S. Gladisch, U. Kister, R. Dachselt, and H. Schumann. A Survey on Interactive Lenses in Visualization. In *EuroVis State-of-the-Art Reports*. Eurographics Association, 2014.

60. S. Gladisch, H. Schumann, M. Ernst, G. Füllen, and C. Tominski. Semi-Automatic Editing of Graphs with Customized Layouts. *Computer Graphics Forum*, 33(3):381–390, 2014. doi:10.1111/cgf.12394.

59. G. Andrienko, N. Andrienko, H. Schumann, and C. Tominski. Visualization of Trajectory Attributes in Space–Time Cube and Trajectory Wall. In M. Buchroithner, N. Prechtel, and D. Burghardt, editors, *Cartography from Pole to Pole*, Lecture Notes in Geoinformation and Cartography, pages 157–163. Springer, 2014. ISBN 978-3-642-32617-2. doi:10.1007/978-3-642-32618-9_11.

58. C. Eichner, A. Bittig, H. Schumann, and C. Tominski. Analyzing Simulations of Biochemical Systems with Feature-Based Visual Analytics. *Computers & Graphics*, 38:18–26, 2014. doi:10.1016/j.cag.2013.09.001.

2013

57. H.-J. Schulz, M. Streit, T. May, and C. Tominski. Towards a Characterization of Guidance in Visualization. Poster at IEEE Conference on Information Visualization (InfoVis), 2013.

56. C. Collins, S. Attfield, F. Chevalier, M. Czerwinski, H. Lam, C. Plaisant, C. Tominski, and M. X. Zhou. Mixed Initiative Interaction. In D. S. Ebert, B. D. Fisher, and P. Isenberg, editors, *Interaction with Information for Visual Reasoning (Dagstuhl Seminar 13352)*, pages 162–164. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013. URL http://drops.dagstuhl.de/opus/volltexte/2013/4346/pdf/dagrep_v003_i008_p151_s13352.pdf#subsection.5.4.

55. D. Keefe, S. Carpendale, P. Cheng, F. Chevalier, C. Collins, T. Isenberg, D. Kirsh, H. Lam, C. North, K. Sedig, C. Tominski, and X. Yuan. Magic Interactions with Information for Visual Reasoning. In D. S. Ebert, B. D. Fisher, and P. Isenberg, editors, *Interaction with Information for Visual Reasoning (Dagstuhl Seminar 13352)*, page 165. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013. URL http://drops.dagstuhl.de/opus/volltexte/2013/4346/pdf/dagrep_v003_i008_p151_s13352.pdf#subsection.5.6.

54. S. Gladisch, H. Schumann, and C. Tominski. Navigation Recommendations for Exploring Hierarchical Graphs. In G. Bebis, R. Boyle, B. Parvin, D. Koracin, B. Li, F. Porikli, V. Zordan, J. Klosowski, S. Coquillart, X. Luo, M. Chen, and D. Gotz, editors, *Advances in Visual Computing*, volume 8034 of *Lecture Notes in Computer Science*, pages 36–47. Springer, 2013. ISBN 978-3-642-41938-6. doi:10.1007/978-3-642-41939-3_4.

53. C. Eichner, A. Bittig, H. Schumann, and C. Tominski. Feature-Based Visual Analytics for Studying Simulations of Dynamic Bi-Stable Spatial Systems. In *Proceedings of the EuroVis Workshop on Visual Analytics (EuroVA)*, pages 25–29. Eurographics Association, 2013. doi:10.2312/PE.EuroVAST.EuroVA13.025-029.

52. S. Buschmann, T. Nocke, and C. Tominski. Towards Visualizing Geo-Referenced Climate Networks. Workshop GeoViz Interactive Maps that Help People Think, 2013.

2012

51. M. Luboschik, C. Tominski, A. T. Bittig, A. M. Uhrmacher, and H. Schumann. Towards Interactive Visual Analysis of Microscopic-Level Simulation Data. In *Proceedings of the Annual SIGRAD Conference, Special Theme: Interactive Visual Analysis of Data*, pages 91–94. Linköping University Electronic Press, 2012. URL http://www.ep.liu.se/ecp_article/index.en.aspx?issue=081;article=013.

50. C. Tominski and H.-J. Schulz. The Great Wall of Space-Time. In *Proceedings of the Workshop on Vision, Modeling & Visualization (VMV)*, pages 199–206. Eurographics Association, 2012. doi:10.2312/PE/VMV/VMV12/199-206.

49. C. Tominski, C. Forsell, and J. Johansson. Interaction Support for Visual Comparison Inspired by Natural Behavior. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2719–2728, 2012. doi:10.1109/TVCG.2012.237.

48. C. Tominski, H. Schumann, G. Andrienko, and N. Andrienko. Stacking-Based Visualization of Trajectory Attribute Data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2565–2574, 2012. doi:10.1109/TVCG.2012.265.

47. C. Tominski and H.-J. Schulz. A Wall-Like Visualization for Spatio-Temporal Data. Workshop GeoVisual Analytics, Time to Focus on Time at the International Conference on Geographic Information Science (GIScience), 2012.

2011

46. C. Tominski, H. Schumann, M. Spindler, and R. Dachselt. Towards Utilizing Novel Interactive Displays for Information Visualization. Workshop on Data Exploration for Interactive Surfaces at the ACM International Conference on Interactive Tabletops and Surfaces (ITS), 2011.
URL http://hal.inria.fr/hal-00659469.

45. C. Tominski and F. Löffler. Novel Interaction Techniques for Visual Comparison. Poster at IEEE Conference on Information Visualization (InfoVis), 2011.

44. A. Lehmann, H. Schumann, O. Staadt, and C. Tominski. Physical Navigation to Support Graph Exploration on a Large High-Resolution Display. In G. Bebis, R. Boyle, B. Parvin, D. Koracin, S. Wang, K. Kyungnam, B. Benes, K. Moreland, C. Borst,

S. DiVerdi, C. Yi-Jen, and J. Ming, editors, *Advances in Visual Computing*, volume 6938 of *Lecture Notes in Computer Science*, pages 496–507. Springer, 2011. ISBN 978-3-642-24027-0. doi:10.1007/978-3-642-24028-7_46.

43. H. Schumann and C. Tominski. Analytical, Visual, and Interactive Concepts for Geo-Visual Analytics. *Journal of Visual Languages & Computing*, 22(4):257–267, 2011. doi:10.1016/j.jvlc.2011.03.002.

42. C. Tominski, J. F. Donges, and T. Nocke. Information Visualization in Climate Research. In *Proceedings of the International Conference Information Visualisation (IV)*, pages 298–305. IEEE Computer Society, 2011. doi:10.1109/IV.2011.12.

41. D. Q. Nguyen, C. Tominski, H. Schumann, and T. A. Ta. Visualizing Tags with Spatiotemporal References. In *Proceedings of the International Conference Information Visualisation (IV)*, pages 32–39. IEEE Computer Society, 2011. doi:10.1109/IV.2011.43.

40. W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of Time-Oriented Data*. Springer, 2011. ISBN 978-0-85729-078-6. doi:10.1007/978-0-85729-079-3.

39. C. Tominski. Event-Based Concepts for User-Driven Visualization. *Information Visualization*, 10(1):65–81, 2011. doi:10.1057/ivs.2009.32.

2010

38. G. Andrienko, N. Andrienko, U. Demšar, D. Dransch, J. Dykes, S. I. Fabrikant, M. Jern, M.-J. Kraak, H. Schumann, and C. Tominski. Space and Time. In Daniel Keim, Jörn Kohlhammer, Geoffrey Ellis, and Florian Mansmann, editors, *Mastering the Information Age – Solving Problems with Visual Analytics*. Eurographics Association, 2010. ISBN 978-3-905673-77-7.

37. M. Spindler, C. Tominski, H. Schumann, and R. Dachselt. Tangible Views for Information Visualization. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS)*, pages 157–166. ACM, 2010. doi:10.1145/1936652.1936684.

36. M. Spindler, C. Tominski, M. Hauschild, H. Schumann, and R. Dachselt. Novel Fields of Application for Tangible Displays above the Tabletop. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS)*, page 315. ACM, 2010. doi:10.1145/1936652.1936743.

35. M. Spindler, C. Tominski, H. Schumann, and R. Dachselt. Poster: Towards Making InfoVis Views Tangible. Poster at IEEE Conference on Information Visualization (InfoVis), 2010.

34. G. Andrienko, N. Andrienko, U. Demšar, D. Dransch, J. Dykes, S. I. Fabrikant, M. Jern, M.-J. Kraak, H. Schumann, and C. Tominski. Space, Time and Visual Analytics. *International Journal of Geographical Information Science*, 24(10):1577–1600, 2010. doi:10.1080/13658816.2010.508043.

33. S. Hadlak, C. Tominski, H.-J. Schulz, and H. Schumann. Visualization of Attributed Hierarchical Structures in a Spatio-Temporal Context. *International Journal of Geographical Information Science*, 24(10):1497–1513, 2010. doi:10.1080/13658816.2010.510840.

32. S. Hadlak, C. Tominski, H.-J. Schulz, and H. Schumann. Visualization of Hierarchies in Space and Time. Workshop GeoVA(t) - Geospatial Visual Analytics: Focus on Time at the AGILE International Conference on Geographic Information Science, 2010.

2009

31. C. Tominski, J. Abello, and H. Schumann. CGV – An Interactive Graph Visualization System. *Computers & Graphics*, 33(6):660–678, 2009. doi:10.1016/j.cag.2009.06.002.

30. H. Piringer, C. Tominski, P. Muigg, and W. Berger. A Multi-Threading Architecture to Support Interactive Visual Exploration. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1113–1120, 2009. doi:10.1109/TVCG.2009.110.

29. C. Thiede, C. Tominski, and H. Schumann. Service-Oriented Information Visualization for Smart Environments. In *Proceedings of the International Conference Information Visualisation (IV)*, pages 227–234. IEEE Computer Society, 2009. doi:10.1109/IV.2009.54.

28. C. Tominski, J. Abello, and H. Schumann. Two Novel Techniques for Interactive Navigation of Graph Layouts. Poster at Eurographics/IEEE Symposium on Visualization (EuroVis), 2009.

2008

27. C. Tominski and H. Schumann. Enhanced Interactive Spiral Display. In *Proceedings of the Annual SIGRAD Conference, Special Theme: Interactivity*, pages 53–56. Linköping University Electronic Press, 2008. URL http://www.ep.liu.se/ecp_article/index.en.aspx?issue=034;article=013.

26. C. Tominski and H. Schumann. Visualization of Gene Combinations. In *Proceedings of the International Conference Information Visualisation (IV)*, pages 120–126. IEEE Computer Society, 2008. doi:10.1109/IV.2008.23.

25. C. Tominski, G. Fuchs, and H. Schumann. Task-Driven Color Coding. In *Proceedings of the International Conference on Information Visualisation (IV)*, pages 373–380. IEEE Computer Society, 2008. doi:10.1109/IV.2008.24.

24. C. Tominski, P. Schulze-Wollgast, and H. Schumann. Visual Methods for Analyzing Human Health Data. In N. Wickramasinghe and E. Geisler, editors, *Encyclopedia of Healthcare Information Systems*, pages 1357–1364. Information Science Reference, 2008. ISBN 978-1-59904-889-5.

23. M. John, C. Tominski, and H. Schumann. Visual and Analytical Extensions for the Table Lens. In *Proceedings of Visualization and Data Analysis (VDA)*, pages 680907–1–680907–12. SPIE/IS&T, 2008. doi:10.1117/12.766440.

22. W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski. Visual Methods for Analyzing Time-Oriented Data. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):47–60, 2008. doi:10.1109/TVCG.2007.70415.

2007

21. W. Aigner, A. Bertone, S. Miksch, C. Tominski, and H. Schumann. Towards a Conceptual Framework for Visual Analytics of Time and Time-Oriented Data. In *Proceedings of the Winter Simulation Conference (WSC)*, pages 721–729. IEEE Press, 2007. doi:10.1145/1351542.1351675.

20. C. Tominski, C. Holzhüter, A. Unger, and H. Schumann. Interactive Poster: Visualization of Gene Combinations. Poster at IEEE Information Visualization Conference (InfoVis), 2007.

19. J. Abello, H.-J. Schulz, H. Schumann, and C. Tominski. Interactive Poster: CGV – Coordinated Graph Visualization. Poster at IEEE Information Visualization Conference (InfoVis), 2007.

18. J. Abello, H.-J. Schulz, B. Gaudin, and C. Tominski. Interactive Poster: Name That Cluster – Text vs. Graphics. Poster at IEEE Information Visualization Conference (InfoVis), 2007.

17. W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski. Visualizing Time-Oriented Data – A Systematic View. *Computers & Graphics*, 31(3):401–409, 2007. doi:10.1016/j.cag.2007.01.030.

16. G. Bieber, C. Tominski, and B. Urban. TiDi Browser: A Novel Photo Browsing Technique for Mobile Devices. In *Proceedings of Multimedia on Mobile Devices*, pages 65070O–1–65070O–8. SPIE/IS&T, 2007. doi:10.1117/12.703924.

2006

15. C. Tominski. *Event-Based Visualization for User-Centered Visual Analysis*. PhD thesis, University of Rostock, Germany, 2006.

14. M. John, C. Tominski, and H. Schumann. Interactive Poster: Two-Tone Pseudo Coloring for Multiple Variables. Poster at IEEE Information Visualization Conference (InfoVis), 2006.

13. C. Tominski, J. Abello, F. van Ham, and H. Schumann. Fisheye Tree Views and Lenses for Graph Visualization. In *Proceedings of the International Conference Information Visualisation (IV)*, pages 17–24. IEEE Computer Society, 2006. doi:10.1109/IV.2006.54.

12. R. Rosenbaum, C. Tominski, and H. Schumann. Graphical Content On Mobile Devices. In M. Khosrow-Pour, editor, *Encyclopedia of E-Commerce, E-Government, and Mobile Commerce*, pages 545–551. Idea Group Reference, 2006. ISBN 1-59140-799-0.

2005

11. C. Tominski, J. Abello, and H. Schumann. Interactive Poster: 3D Axes-Based Visualizations for Time Series Data. Poster at IEEE Symposium on Information Visualization (InfoVis), 2005.

10. C. Tominski, P. Schulze-Wollgast, and H. Schumann. 3D Information Visualization for Time Dependent Data on Maps. In *Proceedings of the International Conference Information Visualisation (IV)*, pages 175–181. IEEE Computer Society, 2005. doi:10.1109/IV.2005.3.

9. P. Schulze-Wollgast, C. Tominski, and H. Schumann. Enhancing Visual Exploration by Appropriate Color Coding. In *Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 203–210, 2005.

2004

8. C. Tominski and H. Schumann. An Event-Based Approach to Visualization. In *Proceedings of the International Conference Information Visualisation (IV)*, pages 101–107. IEEE Computer Society, 2004. doi:10.1109/IV.2004.1320131.

7. R. Rosenbaum, H. Schumann, and C. Tominski. Presenting Large Graphical Contents on Mobile Devices – Performance Issues. In *Proceedings of the Information Resources Management Association International Conference (IRMA)*, pages 371–374. Idea Group, Inc., 2004.

6. C. Tominski, J. Abello, and H. Schumann. Axes-Based Visualizations with Radial Layouts. In *Proceedings of the ACM Symposium on Applied Computing (SAC)*, pages 1242–1247. ACM, 2004. doi:10.1145/967900.968153.

2003

5. C. Tominski, J. Abello, and H. Schumann. Interactive Poster: Axes-Based Visualizations for Time Series Data. Poster at IEEE Symposium on Information Visualization (InfoVis), 2003.

4. G. Bieber and C. Tominski. Visualization Techniques for Personal Tasks on Mobile Computers. In *Proceedings of the HCI International Conference*, pages 659–663. Lawrence Erlbaum Associates, Inc., 2003.

3. R. Rosenbaum and C. Tominski. Pixels vs. Vectors: Presentation of Large Images on Mobile Devices. In *International Workshop on Mobile Computing (IMC)*, 2003.

2. P. Schulze-Wollgast, H. Schumann, and C. Tominski. Visual Analysis of Human Health Data. In *Proceedings of the Information Resources Management Association International Conference (IRMA)*, pages 580–583. Idea Group, Inc., 2003.

1. C. Tominski, P. Schulze-Wollgast, and H. Schumann. Visualisierung zeitlicher Verläufe auf geografischen Karten. In *Visualisierung und Erschließung von Geodaten*, volume 7 of *Kartographische Schriften*, pages 47–57. Kirschbaum Verlag, 2003. ISBN 3-7812-1577-6.

CURRICULUM VITAE

PERSÖNLICHE DATEN

Name: Christian Tominski
Akademischer Grad: Dr.-Ing.
Geburtsdatum: 14.07.1977
Geburtsort: Greifswald
Wohnort: Rostock

QUALIFIKATIONEN

2006, November	Promotion zum Doktor-Ingenieur Fakultät für Informatik und Elektrotechnik, Universität Rostock Prädikat: "magna cum laude"
2002, April	Graduierung zum Diplom Informatiker Fachbereich Informatik, Universität Rostock Prädikat: "sehr gut"

BERUFLICHE TÄTIGKEITEN

laufend 2009, September	Wissenschaftlicher Mitarbeiter (unbefristet) Lehrstuhl Computergraphik, Institut für Informatik, Universität Rostock
2009, August 2005, Juli	Wissenschaftlicher Mitarbeiter (befristet) Lehrstuhl Computergraphik, Institut für Informatik, Universität Rostock
2007, April 2006, April	Consultant (Nebentätigkeit) Ask.com, Piscataway, New Jersey, USA
2006, Februar 2002, November	Wissenschaftlicher Assistent (Nebentätigkeit) Fraunhofer Institut für Graphische Datenverarbeitung, Rostock
2002, September 2002, Juni	Wissenschaftlicher Mitarbeiter Fraunhofer Institut für Graphische Datenverarbeitung, Rostock
2002, Mai 1999, April	Softwareentwickler (Nebentätigkeit) systeon GmbH, Rostock

FÖRDERUNGEN

2005, Juni 2004, Oktober	Promotionsstipendiat im Graduiertenkolleg "Verarbeitung, Verwaltung, Darstellung und Transfer multimedialer Daten – technische Grundlagen und gesellschaftliche Implikationen" (GRK466) der Deutschen Forschungsgemeinschaft
2004, September 2002, Oktober	Promotionsstipendiat gefördert durch die Landesgraduiertenförderung des Landes Mecklenburg-Vorpommern

AUSLANDSAUFENTHALTE

2007, März	Gastwissenschaftler am DIMACS, Rutgers University, New Jersey, USA
2006, März	Gastwissenschaftler am DIMACS, Rutgers University, New Jersey, USA
2005, September 2005, August	Gastwissenschaftler am DIMACS, Rutgers University, New Jersey, USA

Christian Tominski Rostock, den 14. Juli 2014

ERKLÄRUNG

Ich erkläre, dass ich die eingereichte Habilitationsschrift selbständig und ohne fremde Hilfe verfasst, andere als die von mir angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Rostock, 14. Juli 2014

Christian Tominski

THESES

1. In visualization research, aspects of interaction are payed less attention than aspects of the graphical representation. There is a need to strengthen the interaction side of visualization.

2. Interaction in visualization involves specific requirements and constraints in addition to general interaction aspects. Therefore, it deserves special attention and dedicated research.

3. Interaction has to be considered at different levels, including low-level interaction basics, intermediate-level interaction techniques, and higher-level interactive problem solving.

4. A unified view based on the four cornerstones data, tasks, technology, and human helps in structuring, understanding, and discussing interaction in visualization.

5. Designing and engineering interaction in visualization requires taking into account all four cornerstones. Only then can useful and usable interaction solutions be obtained.

6. Interactive lenses crystallize as universally useful design across all cornerstones. Their lightweight and focused character suits exploratory visualization scenarios in particular.

7. A broad utilization of modern display and interaction technologies is beneficial. New technologies open up possibilities that visualization research is still to fully take advantage of.

8. Interaction is a powerful means to support the user in working with visual representations of data. Yet interaction should not be understood as a universal cure.

9. Interaction involves costs for executing it and evaluating its outcome. Automatic methods are a valuable addition to reduce interaction costs and support the human user.

10. Supporting the human engineer is important as well. New concepts and tools are needed to make modeling, implementing, testing, and evaluating interaction for visualization easier.

11. Bertin's visual variables are the building blocks of visual representations. We lack such building blocks for interaction. An interaction vocabulary should be established to close this gap.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/